repository.ub.ac.id

PENGEMBANGAN PERSONAL DATA ANALITIK MENGGUNAKAN PHP-ML DAN APACHE SPARK PADA *IOT CLOUD APPS*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Ahmad Riyadh Al Faathin
NIM: 155150207111052

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2019

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

PENGESAHAN

PENGEMBANGAN PERSONAL DATA ANALITIK MENGGUNAKAN PHP-ML DAN APACHE SPARK PADA *IOT CLOUD APPS*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Ahmad Riyadh Al Faathin
NIM: 155150207111052

Skripsi ini telah diuji dan dinyatakan lulus pada
2 Januari 2019
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Eko Sakti Pramukantoro, S.Kom, M.Kom	Fariz Andri Bakhtiar, S.T., M.Kom.
NIK: 2011028608051001	NIK: 2017098403141001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D

NIP. 19710518 200312 1 001

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata di dalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 22 Januari 2019

Ahmad Riyadh Al Faathin

NIM: 155150207111052

UNIVERSITAS BRAWIJAYA

PRAKATA

Assalamualaikum warahmatullahi wabarakatuh.

Puji syukur penulis panjatkan kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga laporan skripsi yang berjudul "PENGEMBANGAN PERSONAL DATA ANALITIK MENGGUNAKAN PHP-ML DAN APACHE SPARK PADA IOT CLOUD APPS" dapat terselesaikan.

Selama pengerjaan laporan skripsi penulis mendapat banyak pelajaran dan pengalaman baru. Penulis juga banyak mengaplikasikan ilmu yang didapat selama menduduki bangku perkuliahan ke dala laporan skripsi ini. Penulis menyadari bahwa laporan skripsi ini tidak akan berhasil tanpa dukungan, doa, dan bantuan dari beberapa pihak. Oleh karena itu, penulis ingin menyampaikan rasa erima kasih dan rasa hormat kepada :

1. Orang tua dan keluarga besar saya yang tidak pernah berhenti memberikan doa dan dukungan selama menempuh pendidikan di Malang termasuk saat pengerjaan skripsi ini.
2. Bapak Eko Sakti Pramukantoro, S.Kom.,M.Kom., dan bapak Fariz Andri Bakhtiar, S.T.,M.Kom., selaku dosen pembimbing skripsi yang telah membimbing dengan sabar dan juga memberikan arahan dengan sangat baik sehingga penulis dapat menyelesaikan skripsi ini.
3. Bapak Wayan Firdaus Mahmudy, S.Si.,M.T., Ph.D., selaku Dekan Fakultas Ilmu Komputer
4. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D., selaku Ketua Jurusan Teknik Informatika
5. Bapak Agus Wahyu Widodo, S.T., M.Cs., selaku Ketua Program Studi Teknik Informatika
6. Rezananda, Dias, Ulul, Nuel, Eric, Azzu, Naufal, Agus dkk teman setia saat nongkrong didepan ruang dosen
7. Muhammad, Adi, Agus, Samiyono, Hakim dan Pratama, yang telah memberikan banyak bantuan baik materi maupun non-materi selama satu hidup satu kontrakan
8. Aulia, Sulaiman, Jeva, Agung, Pindo, Teguh, Heru, Steven, Delarta, Azhar, Dicky, Febristya dan teman-teman yang lain yang tak dapat penulis sebutkan satu persatu, teman berbagi suka duka selama menjalani perkuliahan di Malang
9. Aik, El, Uyo, Thania, Breh, Digna, Hannah, Fenda, Gina, Kevin, Infi, Sam, Fidiya, Feni, Frilla, Ukik, dan Wina, yang telah memberi warna selama mengabdi di desa Tegalweru

10. Keluarga Besar Beswan Djarum Malang, yang telah memberikan banyak pengalaman dan dukungan moral
11. Keluarga UKM Mahasiswa Wirausaha Universitas Brawijaya yang selalu berbagi ilmu dan memberikan semangat, doa, dan banyak pengalaman menyenangkan
12. Seluruh civitas akademika Fakultas Ilmu Komputer yang selalu memberikan dukungan dan bantuan dalam proses penyelesaian skripsi ini
13. Teman-teman keminatan KBJ dan teman – teman Teknik Informatika Angkatan 2015 lainnya yang selalu berbagi ilmu dari awal perkuliahan sampai tahap penyelesaian skripsi
14. Microsoft Corporation sebagai pihak yang telah membantu pengerjaan skripsi dengan memberikan fasilitas layanan teknologi selama mengerjakan skripsi
15. Djarum Foundation yang telah memberikan pengalaman luar biasa selama menjadi bagian dari Beswan Djarum 33, kegiatan CB, LD, NB, WC dan pastinya CE benar-benar mengubah penulis menjadi pribadi yang lebih baik
16. Bill Gates dan Hartono bersaudara yang selalu menjadi *role model* penulis dalam menjalani kehidupan selama ini
17. Semua pihak yang telah banyak membantu, berbagi ilmu dan pengalaman, serta memberikan dukungan selama proses pengerjaan skripsi yang tidak dapat penulis sebutkan satu per satu

Penulis mengakui bahwa dalam penyusunan skripsi ini masih terdapat banyak kekurangan, sehingga kritik yang membangun dan juga saran sangat penulis butuhkan. Akhir kata penulis berharap agar skripsi ini bisa membawa manfaat bagi semua pihak yang menggunakannya.

Wassalamualaikum warahmatullahi wabarakatuh.

Malang, 22 Januari 2019

Penulis

riyadh@faathin.com

repository.ub.ac.id

ABSTRAK

Ahmad Riyadh Al Faathin, Pengembangan Personal Data Analitik Menggunakan PHP-ML dan Apache Spark pada *IoT Cloud Apps*

Pembimbing: Eko Sakti Pramukantoro dan Fariz Andri Bakhtiar

Internet of things mempunyai sistem dan perangkat untuk memonitor objek, mengumpulkan, dan meneruskan data. Data yang beragam akan diakuisisi oleh sensor, yang kemudian akan diteruskan hingga disimpan pada suatu media penyimpanan. Data yang dihasilkan oleh perangkat IoT umumnya merupakan data yang *low density,* sehingga tidak mempunyai banyak nilai kebermanfaatan jika tidak dilakukan analisis maupun visualisasi. Selain itu data yang dikumpulkan biasanya mempunyai maksud dan tujuan yang spesifik (personal) sesuai kebutuhan pengguna. Oleh karena itu diperlukan sebuah sistem yang dapat menyimpan data yang beragam, melakukan analisis sekaligus personalisasi terhadap data IoT. Peneliti mencoba memecahkan beberapa permasalahan tersebut, melalui sebuah *framework* yang terdiri dari sebuah *data storage*, *data analytic* dan juga *visualization webservice* yang di dalamnya akan dikembangkan menggunakan pendekatan personal. Pendekatan personal yang diimplementasikan adalah kemampuan dari sistem untuk menyesuaikan dengan kebutuhan pengguna meliputi topik, perangkat, data bahkan mekanisme analisis data. Selain itu dalam sistem *framework* yang dikembangkan juga akan diimplementasikan konsep *semantic interopability* untuk memperjelas makna dari data, sehingga data dapat diklasifikasikan secara lebih baik di dalam *framework*. Pengujian kinerja sistem dilakukan dari segi fungsional, skalabilitas, *response time* penyimpanan dan pengambilan data. Hasil dari pengujian fungsional didapatkan bahwa sistem *framework* yang dikembangkan sudah berjalan sesuai dengan fungsinya dalam menyimpan beragam data ke dalam *data storage*, melakukan analisis data dan juga melakukan personalisasi servis berdasarkan kebutuhan pengguna. Selain itu berdasarkan pengujian skalabilitas yang telah dilakukan menunjukkan bahwa sistem mempunyai kemampuan yang cukup baik dengan tingkat error mencapai 0% dan 0.7% pada pengujian pengambilan 300 data dan pengiriman 1000 data secara simultan. Selain itu berdasarkan pengujian skalabilitas data analitik dapat diketahui bahwa sistem dapat menangani hingga 100.000 data menggunakan PHP-ML dan 500.000 menggunakan Apache Spark. Berdasarkan hasil tersebut, sistem ini dapat menjadi solusi dari permasalahan IoT dalam *cloud*.

Kata kunci: IoT, Cloud Computing, Sistem, Personal, Data Analitik

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

ABSTRACT

Ahmad Riyadh Al Faathin, Development of Personal Data Analytic Using PHP-ML and Apache Spark on IoT Cloud Apps

Supervisor: Eko Sakti Pramukantoro dan Fariz Andri Bakhtiar

The internet of things has systems and devices to monitor objects, collect and forward data. Diverse data will be acquired by sensors, which will then be forwarded to be stored on a storage media. The data generated by IoT devices is generally low density data, so it does not have many usefulness values if neither analysis nor visualization is done. In addition, data collected usually has specific goals (personal) according to user needs. Therefore we need a system that can store diverse data, conduct analysis as well as personalization of IoT data. The researcher tried to solve some of these problems, through a framework consisting of a data storage, analytic data and also a visualization webservice which will be developed using a personal approach. The personal approach that is implemented is the ability of the system to adapt to user needs covering topics, devices, data and even data analysis mechanisms. In addition, the framework system developed will also implement the concept of semantic interopability to clarify the meaning of data, so that data can be better classified within the framework. System performance testing is done in terms of functional, scalability, response time for storing and retrieving data. The results of functional testing found that the framework system developed was running in accordance with its function in storing various data into data storage, performing data analysis and also personalizing service based on user needs. In addition, based on the scalability testing that has been done shows that the system has a pretty good ability with an error rate of 0% and 0.7% in testing the retrieval of 300 data and sending 1000 data simultaneously. In addition, based on scalability testing analytical data can be seen that the system can handle up to 100,000 data using PHP-ML and 500,000 using Apache Spark. Based on these results, this system can be a solution to IoT problems in the cloud.

Keywords: IoT, Cloud Computing, System, Personal, Data Analytics

UNIVERSITAS BRAWIJAYA

DAFTAR ISI

PENGESAHAN ii
PERNYATAAN ORISINALITAS iii
PRAKATA iv
ABSTRAK vi
ABSTRACT vii
DAFTAR ISI viii
DAFTAR TABEL xii
DAFTAR GAMBAR xv
DAFTAR KELUARAN xix
DAFTAR KONFIGURASI xxi
DAFTAR PSEUDOCODE xxiii
DAFTAR LAMPIRAN xxv
BAB 1 PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Tujuan 3
1.4 Manfaat 4
1.5 Batasan Masalah 4
1.6 Sistematika Pembahasan 4
BAB 2 LANDASAN KEPUSTAKAAN 7
2.1 Kajian Pustaka 7
2.2 Dasar Teori 8
2.2.1 *Internet of things* 8
2.2.2 IoT Application 9
2.2.3 Cloud Computing 10
2.2.4 Personal 10
2.2.5 Kajian Pustaka Metode Pengujian 11
BAB 3 METODOLOGI 12
3.1 Studi Literatur 12
3.2 Analisis Kebutuhan 12

3.2.1 Komponen Komunikasi IoT dan Komponen *Sensor Node* 13
3.2.2 Komponen *IoT Cloud Apps* 13
3.2.3 Komponen Arsitektur *Cloud* 13
3.3 Perancangan Sistem 13
3.3.1 Perancangan Lingkungan Sistem 14
3.3.2 Perancangan Infrastruktur 14
3.3.3 Perancangan Aplikasi 16
3.3.4 Perancangan Data 16
3.4 Perancangan Pengujian 16
3.5 Implementasi Sistem 17
3.6 Pengujian Sistem 17
3.6.1 Pengujian Fungsional 17
3.6.2 Pengujian Non Fungsional 17
3.7 Pembahasan 17
BAB 4 ANALISIS KEBUTUHAN DAN PERANCANGAN 18
4.1 Deskripsi Umum Sistem 18
4.2 Kebutuhan Sistem 19
4.2.1 Kebutuhan Fungsional 19
4.2.2 Kebutuhan Non Fungsional 21
4.3 Perancangan Sistem 23
4.3.1 Perancangan Lingkungan Sistem 23
4.3.2 Perancangan Infrastruktur 24
4.3.3 Perancangan Aplikasi 28
4.4 Perancangan Data 52
4.4.1 Perancangan Data Semi Terstruktur 53
4.4.2 Perancangan Data Tidak Terstruktur 54
4.5 Perancangan Pengujian 54
4.5.1 Pengujian Fungsional 54
4.5.2 Pengujian Skalabilitas 59
BAB 5 IMPLEMENTASI 60
5.1 Implementasi Infrastruktur 60
5.1.1 Implementasi *IoT Data Storage* 60

5.1.2 Implementasi *IoT Data Analytic* 67
5.1.3 Implementasi *IoT Visualization Webservice* 68
5.2 Implementasi Aplikasi 76
5.2.1 Implementasi *API Internal IoT Data Storage* 76
5.2.2 Implementasi *API Internal IoT data analytic* 78
5.2.3 Implementasi Modul *Data Analytic* 81
5.2.4 Implementasi Menyimpan Data IoT 84
5.2.5 Implementasi Menampilkan Daftar Topik Terautorisasi 86
5.2.6 Implementasi Menampilkan Data dari Topik 87
5.2.7 Implementasi Menampilkan Detail Data dari Topik 88
5.2.8 Implementasi Menampilkan Daftar Perangkat IoT 89
5.2.9 Implementasi Mendaftarkan Perangkat IoT 90
5.2.10 Implementasi Menampilkan Informasi Perangkat IoT 91
5.2.11 Implementasi Mengubah Perangkat IoT 91
5.2.12 Implementasi Menghapus Perangkat IoT 92
5.2.13 Implementasi Menampilkan Daftar Topik IoT 93
5.2.14 Implementasi Mendaftarkan Topik IoT 94
5.2.15 Implementasi Menampilkan Informasi Topik IoT 95
5.2.16 Implementasi Mengubah Topik IoT 96
5.2.17 Implementasi Menghapus Topik IoT 97
5.2.18 Implementasi *Data Explorer* 98
5.2.19 Implementasi *Data Viewer* 99
5.2.20 Implementasi *Data Downloader* 101
5.2.21 Implementasi Visualisasi Data 102
5.2.22 Implementasi Modul Prediksi Least Square 103
5.2.23 Implementasi Modul Klasifikasi Naïve Bayes 104
5.2.24 Implementasi Keluaran Analitik 106
5.2.25 Implementasi Menampilkan Daftar Pengguna 107
5.2.26 Implementasi Menampilkan Informasi Pengguna 108
5.2.27 Implementasi Mengubah Pengguna 109
5.2.28 Implementasi Menghapus Pengguna 110
BAB 6 PENGUJIAN 112

6.1 Pengujian Fungsional 112
6.1.1 Autentikasi Login Perangkat Menggunakan JWT 112
6.1.2 Menampilkan Daftar Topik Melalui Sebuah *API* 114
6.1.3 Menerima Data Terstruktur Dalam Bentuk JSON 116
6.1.4 Menerima Data Semi Terstruktur Dalam Bentuk XML 117
6.1.5 Menerima Data Tidak Terstruktur Dalam Bentuk Gambar JPG 119
6.1.6 Menampilkan Daftar Data yang Disimpan Dalam Suatu Topik Melalui Sebuah *API* 121
6.1.7 Menampilkan Data yang Disimpan Dalam Suatu Topik Melalui Sebuah *API* 124
6.1.8 Registrasi Pengguna Baru 126
6.1.9 Login Pengguna 128
6.1.10 CRUD (Manajemen) Perangkat IoT 129
6.1.11 CRUD (Manajemen) Topik IoT 133
6.1.12 CRUD (Manajemen) Pengguna 136
6.1.13 Menampilkan Daftar Data yang Disimpan Dalam Suatu Topik 140
6.1.14 Menampilkan Data yang Disimpan Dalam Suatu Topik 142
6.1.15 Prediksi Data Menggunakan Least Square (personalisasi) 144
6.1.16 Klasifikasi Data Menggunakan Naïve bayes (personalisasi) 145
6.1.17 Menampilkan Informasi Statistik dari Sebuah Topik 147
6.2 Pengujian Skalabilitas 148
6.2.1 Pengujian Pengambilan Data 148
6.2.2 Pengujian Pengiriman Data 151
6.2.3 Pengujian Analitik Data 154
BAB 7 PENUTUP 162
7.1 Kesimpulan 162
7.2 Saran 164
DAFTAR REFERENSI 165
LAMPIRAN A IMPLEMENTASI 167
LAMPIRAN B PENGUJIAN 176

DAFTAR TABEL

Tabel 2.1 Kajian pustaka (bagian 1) 7
Tabel 2.2 Kajian pustaka (bagian 2) 8
Tabel 2.3 Kajian pustaka metode pengujian 11
Tabel 4.1 Kebutuhan fungsional 20
Tabel 4.2 Kebutuhan non fungsional 21
Tabel 4.3 Kebutuhan perangkat keras 22
Tabel 4.4 Kebutuhan perangkat lunak (bagian 1) 22
Tabel 4.5 Kebutuhan perangkat lunak (bagian 2) 23
Tabel 4.6 *Use case scenario* menyimpan data dari Sensor 30
Tabel 4.7 *Use case scenario* menampilkan topik terautorisasi 32
Tabel 4.8 *Use case scenario* menampilkan data dari topik 34
Tabel 4.9 *Use case scenario* menampilkan detail data dari topik 36
Tabel 4.10 *Use case scenario* manajemen perangkat IoT (bagian 1) 38
Tabel 4.11 Use case scenario manajemen perangkat IoT (bagian 2) 39
Tabel 4.12 *Use case scenario* manajemen topik IoT (bagian 1) 42
Tabel 4.13 *Use case scenario* manajemen topik IoT (bagian 2) 43
Tabel 4.14 *Use case scenario* mengolah Data 47
Tabel 4.15 *Use case scenario* manajemen pengguna (bagian 1) 49
Tabel 4.16 *Use case scenario* manajemen pengguna (bagian 2) 50
Tabel 4.17 Perancangan data semi terstruktur JSON 53
Tabel 4.18 Perancangan data semi terstruktur XML 53
Tabel 4.19 Perancangan data tidak terstruktur RAW 54
Tabel 4.20 Pengujian fungsional (bagian 1) 55
Tabel 4.21 Pengujian fungsional (bagian 2) 56
Tabel 4.22 Pengujian fungsional (bagian 3) 57
Tabel 4.23 Pengujian fungsional (bagian 4) 58
Tabel 4.24 Pengujian fungsional (bagian 5) 59
Tabel 6.1 Hasil pengujian fungsional autentikasi login perangkat menggunakan JWT (bagian 1) 112

Tabel 6.2 Hasil pengujian fungsional autentikasi login perangkat menggunakan JWT (bagian 2) 113

Tabel 6.3 Hasil pengujian fungsional menampilkan daftar topik melalui sebuah *API* 114

Tabel 6.4 Hasil pengujian fungsional menerima data terstruktur dalam bentuk JSON (bagian 1) 116

Tabel 6.5 Hasil pengujian fungsional menerima data terstruktur dalam bentuk JSON (bagian 2) 117

Tabel 6.6 Hasil pengujian fungsional menerima data semi terstruktur dalam bentuk XML (bagian 1) 117

Tabel 6.7 pengujian fungsional menerima data semi terstruktur dalam bentuk XML (bagian 2) 118

Tabel 6.8 Hasil pengujian fungsional menerima data tidak terstruktur dalam bentuk Gambar JPG 119

Tabel 6.9 Hasil pengujian fungsional menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API* (bagian 1) 121

Tabel 6.10 Hasil pengujian fungsional menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API* (bagian 2) 122

Tabel 6.11 Hasil pengujian fungsional menampilkan data yang disimpan dalam suatu topik melalui sebuah *API* (bagian 1) 124

Tabel 6.12 Hasil pengujian fungsional menampilkan data yang disimpan dalam suatu topik melalui sebuah *API* (bagian 2) 125

Tabel 6.13 Hasil pengujian fungsional registrasi pengguna baru (bagian 1) 126

Tabel 6.14 Hasil pengujian fungsional registrasi pengguna baru (bagian 2) 127

Tabel 6.15 Hasil pengujian fungsional login pengguna 128

Tabel 6.16 Hasil pengujian fungsional CRUD (manajemen) perangkat IoT 129

Tabel 6.17 Hasil pengujian fungsional CRUD (manajemen) topik IoT 133

Tabel 6.18 Hasil pengujian fungsional CRUD (manajemen) pengguna 136

Tabel 6.19 Hasil pengujian fungsional menampilkan daftar data yang disimpan dalam suatu topik (bagian 1) 140

Tabel 6.20 Hasil pengujian fungsional menampilkan daftar data yang disimpan dalam suatu topik (bagian 2) 141

Tabel 6.21 Hasil pengujian fungsional menampilkan data yang disimpan dalam suatu topik (bagian 1) 142

Tabel 6.22 Hasil pengujian fungsional menampilkan data yang disimpan dalam suatu topik (bagian 2) 143

Tabel 6.23 Hasil pengujian fungsional prediksi data menggunakan least square 144
Tabel 6.24 Hasil pengujian fungsional klasifikasi data menggunakan Naïve bayes 145
Tabel 6.25 Hasil pengujian fungsional menampilkan informasi statistik dari sebuah topik 147

DAFTAR GAMBAR

Gambar 3.1 Langkah-langkah penelitian 12
Gambar 4.1 Deskripsi umum sistem 18
Gambar 4.2 Perancangan Lingkungan sistem 24
Gambar 4.3 *Use case* sistem *webservice* 28
Gambar 4.4 *Sequence diagram* dari menyimpan data dari sensor 31
Gambar 4.5 *Sequence diagram* menampilkan topik terautorisasi 33
Gambar 4.6 *Sequence diagram* menampilkan data dari topik 35
Gambar 4.7 *Sequence diagram* menampilkan detail data dari topik 37
Gambar 4.8 *Sequence diagram* manajemen perangkat IoT (bagian 1) 40
Gambar 4.9 *Sequence diagram* manajemen perangkat IoT (bagian 2) 41
Gambar 4.10 *Sequence diagram* manajemen topik IoT (bagian 1) 44
Gambar 4.11 *Sequence diagram* manajemen topik IoT (bagian 2) 45
Gambar 4.12 *Sequence diagram* manajemen topik IoT (bagian 3) 46
Gambar 4.13 *Sequence diagram* mengolah data IoT 48
Gambar 4.14 *Sequence diagram* manajemen pengguna (bagian 1) 51
Gambar 4.15 *Sequence diagram* manajemen pengguna (bagian 2) 52
Gambar 4.16 Perancangan data tidak terstruktur JPG 54
Gambar 5.1 Ringkasan Apache Hadoop 66
Gambar 5.2 *Datanode* Apache Hadoop 66
Gambar 5.3 Ringkasan Apache Spark 68
Gambar 5.4 Menyimpan data IoT 86
Gambar 5.5 Menampilkan daftar topik terautorisasi 87
Gambar 5.6 Menampilkan data dari topik 88
Gambar 5.7 Menampilkan detail data dari topik 89
Gambar 5.8 Menampilkan daftar perangkat IoT 90
Gambar 5.9 Mendaftarkan perangkat IoT 90
Gambar 5.10 Menampilkan informasi perangkat IoT 91
Gambar 5.11 Mengubah perangkat IoT 92
Gambar 5.12 Menghapus perangkat IoT 93
Gambar 5.13 Menampilkan daftar topik IoT 94

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

Gambar 5.14 Mendaftarkan topik IoT 95
Gambar 5.15 Menampilkan informasi topik IoT 96
Gambar 5.16 Mengubah topik IoT 97
Gambar 5.17 Menghapus topik IoT 98
Gambar 5.18 *Data explorer* 99
Gambar 5.19 *Data viewer* (JSON) 100
Gambar 5.20 *Data viewer* (binary) 100
Gambar 5.21 Data *downloader* JSON 101
Gambar 5.22 Data *downloader* Binary 101
Gambar 5.23 Visualisasi data 103
Gambar 5.24 Modul prediksi least square 104
Gambar 5.25 Modul klasifikasi naïve bayes 106
Gambar 5.26 Keluaran analitik least square 107
Gambar 5.27 Keluaran analitik naïve bayes 107
Gambar 5.28 Menampilkan daftar pengguna 108
Gambar 5.29 Menampilkan informasi pengguna 109
Gambar 5.30 Mengubah pengguna 110
Gambar 5.31 Menghapus Pengguna 111
Gambar 6.1 Pengujian fungsional menerima data tidak terstruktur dalam bentuk JPG (*webservice*) 121
Gambar 6.2 Pengujian fungsional registrasi pengguna baru (*webservice*) 127
Gambar 6.3 Pengujian fungsional registrasi pengguna baru (*database*) 128
Gambar 6.4 Pengujian fungsional login pengguna 129
Gambar 6.5 Pengujian fungsional CRUD perangkat IoT (*index*) 130
Gambar 6.6 Pengujian fungsional CRUD perangkat IoT (*index database*) 130
Gambar 6.7 Pengujian fungsional CRUD perangkat IoT (*create*) 130
Gambar 6.8 Pengujian fungsional CRUD perangkat IoT (*create database*) 131
Gambar 6.9 Pengujian fungsional CRUD perangkat IoT (*read*) 131
Gambar 6.10 Pengujian CRUD perangkat IoT (*update*) 131
Gambar 6.11 Pengujian CRUD perangkat IoT (*update database*) 132
Gambar 6.12 Pengujian CRUD perangkat IoT (*delete*) 132
Gambar 6.13 Pengujian CRUD perangkat IoT (*delete database*) 132

Gambar 6.14 Pengujian fungsional CRUD topik IoT (*index*)..... 133
Gambar 6.15 Pengujian fungsional CRUD topik IoT (*index database*)..... 134
Gambar 6.16 Pengujian fungsional CRUD topik IoT (*create*)..... 134
Gambar 6.17 Pengujian fungsional CRUD topik IoT (*create database*)..... 134
Gambar 6.18 Pengujian fungsional CRUD topik IoT (*read*)..... 135
Gambar 6.19 Pengujian fungsional CRUD topik IoT (*update*)..... 135
Gambar 6.20 Pengujian fungsional CRUD topik IoT (*update database*)..... 135
Gambar 6.21 Pengujian fungsional CRUD topik IoT (*delete*)..... 136
Gambar 6.22 Pengujian fungsional CRUD topik IoT (*delete database*)..... 136
Gambar 6.23 Pengujian fungsional CRUD pengguna (*index*)..... 137
Gambar 6.24 Pengujian fungsional CRUD pengguna (*index database*)..... 138
Gambar 6.25 Pengujian fungsional CRUD (*read*)..... 138
Gambar 6.26 Pengujian fungsional CRUD pengguna (*update*)..... 139
Gambar 6.27 Pengujian fungsional CRUD pengguna (*update database*)..... 139
Gambar 6.28 Pengujian fungsional CRUD pengguna (*delete*)..... 140
Gambar 6.29 Pengujian fungsional CRUD pengguna (*delete database*)..... 140
Gambar 6.30 Pengujian fungsional menampilkan daftar data yang disimpan dalam suatu topik..... 141
Gambar 6.31 Pengujian menampilkan daftar data yang disimpan dalam suatu topik (*database*)..... 142
Gambar 6.32 Pengujian fungsional menampilkan daftar data yang disimpan dalam suatu topik..... 143
Gambar 6.33 Pengujian fungsional menampilkan daftar data yang disimpan dalam suatu topik (*database*)..... 143
Gambar 6.34 Pengujian fungsional prediksi data menggunakan least square (masukkan pada PHP-ML)..... 144
Gambar 6.35 Pengujian fungsional prediksi data menggunakan least square (keluaran pada PHP-ML)..... 145
Gambar 6.36 Pengujian fungsional klasifikasi data menggunakan naïve bayes (masukkan pada PHP-ML)..... 146
Gambar 6.37 Pengujian fungsional klasifikasi data menggunakan naïve bayes (keluaran pada PHP-ML)..... 146
Gambar 6.38 Pengujian fungsional menampilkan informasi statistik dari sebuah topik..... 148

Gambar 6.39 Pengujian skalabilitas *request* 50 data 149
Gambar 6.40 Pengujian skalabilitas *request* 300 data 149
Gambar 6.41 Pengujian skalabilitas *request* 500 data 150
Gambar 6.42 Pengujian skalabilitas *request* 200 data 150
Gambar 6.43 Pengujian skalabilitas *request* 1000 data 150
Gambar 6.44 Pembahasan kinerja pengambilan data 151
Gambar 6.45 Pengujian skalabilitas *POST* 100 data 152
Gambar 6.46 Pengujian skalabilitas *POST* 300 data 152
Gambar 6.47 Pengujian skalabilitas *POST* 500 data 153
Gambar 6.48 Pengujian skalabilitas *POST* 700 data 153
Gambar 6.49 Pengujian skalabilitas *POST* 1000 data 153
Gambar 6.50 Pembahasan kinerja pengiriman data 154
Gambar 6.51 Pengujian skalabilitas analisis 1.000 Data (PHP-ML) 155
Gambar 6.52 Pengujian skalabilitas analisis 1.000 Data (Apache Spark) 155
Gambar 6.53 Pengujian skalabilitas analisis 10.000 Data (PHP-ML) 156
Gambar 6.54 Pengujian skalabilitas analisis 10.000 Data (Apache Spark) 156
Gambar 6.55 Pengujian skalabilitas analisis 100.000 Data (PHP-ML) 157
Gambar 6.56 Pengujian skalabilitas analisis 100.000 Data (Apache Spark) 157
Gambar 6.57 Pengujian skalabilitas analisis 500.000 Data (PHP-ML) 158
Gambar 6.58 Pengujian skalabilitas analisis 500.000 Data (Error PHP-ML) 158
Gambar 6.59 Pengujian skalabilitas analisis 500.000 Data (Apache Spark) 159
Gambar 6.60 Pengujian skalabilitas analisis 1.000.000 Data (PHP-ML) 159
Gambar 6.61 Pengujian skalabilitas analisis 500.000 Data (Error PHP-ML) 160
Gambar 6.62 Pengujian skalabilitas analisis 1.000.000 Data (Apache Spark) 160
Gambar 6.63 Pengujian skalabilitas analisis 1.000.000 Data (Error Apache Spark) 160
Gambar 6.64 Pembahasan kinerja analitik data 161

DAFTAR KELUARAN

Keluaran 6.1 Pengujian fungsional autentikasi login menggunakan JWT (Postman) 113

Keluaran 6.2 Pengujian fungsional autentikasi login menggunakan JWT (*database*) 113

Keluaran 6.3 Pengujian fungsional menampilkan daftar topik melalui sebuah *API* (Postman) 114

Keluaran 6.4 Pengujian fungsional menampilkan daftar topik melalui sebuah *API* (*database* bagian 1)..... 115

Keluaran 6.5 Pengujian fungsional menampilkan daftar topik melalui sebuah *API* (*database* bagian 2)..... 116

Keluaran 6.6 Pengujian fungsional menerima data terstruktur dalam bentuk JSON (Postman) 117

Keluaran 6.7 Pengujian fungsional menerima data terstruktur dalam bentuk JSON (database) 117

Keluaran 6.8 Pengujian fungsional menerima data semi terstruktur dalam bentuk XML (Postman)..... 118

Keluaran 6.9 Pengujian fungsional menerima data semi terstruktur dalam bentuk XML (*database*) 119

Keluaran 6.10 Pengujian fungsional menerima data tidak terstruktur dalam bentuk JPG..... 120

Keluaran 6.11 Pengujian fungsional menerima data tidak terstruktur dalam bentuk JPG (*chunks database*) 120

Keluaran 6.12 Pengujian fungsional menerima data tidak terstruktur dalam bentuk JPG (*metadata database*)..... 120

Keluaran 6.13 Pengujian menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API* (Postman bagian 1) 122

Keluaran 6.14 Pengujian menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API* (Postman bagian 2) 123

Keluaran 6.15 Pengujian menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API* (Postman bagian 3) 124

Keluaran 6.16 Pengujian menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API* (*database*) 124

Keluaran 6.17 Pengujian menampilkan data yang disimpan dalam suatu topik melalui sebuah *API* (Postman bagian 1)..... 125

Keluaran 6.18 Pengujian menampilkan data yang disimpan dalam suatu topik melalui sebuah *API* (Postman bagian 2)..... 126

Keluaran 6.19 Pengujian menampilkan data yang disimpan dalam suatu topik melalui sebuah *API* (*database*) 126

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

DAFTAR KONFIGURASI

Konfigurasi 5.1 Mongo 61
Konfigurasi 5.2 Core-Site.xml (bagian 1) 61
Konfigurasi 5.3 Core-Site.xml (bagian 2) 62
Konfigurasi 5.4 Hadoop-env.sh 62
Konfigurasi 5.5 Hdfs-site.xml 62
Konfigurasi 5.6 Mapred-site.xml (bagian 1) 62
Konfigurasi 5.7 Mapred-site.xml (bagian 2) 63
Konfigurasi 5.8 Yarn-site.xml (bagian 1) 63
Konfigurasi 5.9 Yarn-site.xml (bagian 2) 64
Konfigurasi 5.10 Apache Hadoop Masters 64
Konfigurasi 5.11 Apache Hadoop Slaves 64
Konfigurasi 5.12 Apache Hadoop 65
Konfigurasi 5.13 Apache Spark 67
Konfigurasi 5.14 Spark-env.sh 67
Konfigurasi 5.15 Masters 68
Konfigurasi 5.16 Slaves 68
Konfigurasi 5.17 Ptriot.com.conf 69
Konfigurasi 5.18 Api.ptriot.com.conf 69
Konfigurasi 5.19 Admin.ptriot.com.conf 70
Konfigurasi 5.20 Apache Webservice 70
Konfigurasi 5.21 Ptriot.com (bagian 1) 71
Konfigurasi 5.22 Ptriot.com (bagian 2) 72
Konfigurasi 5.23 Api.ptriot.com (bagian 1) 72
Konfigurasi 5.24 Api.ptriot.com (bagian 2) 73
Konfigurasi 5.25 Admin.ptriot.com (bagian 1) 73
Konfigurasi 5.26 Admin.ptriot.com (bagian 2) 74
Konfigurasi 5.27 Nginx 74
Konfigurasi 5.28 Redis 75
Konfigurasi 5.29 Laravel 75
Konfigurasi 5.30 .Env (bagian 1) 75

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

Konfigurasi 5.31 .Env (bagian 2)..... 76

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

DAFTAR PSEUDOCODE

Pseudocode 5.1 *Driver* MongoDB 77
Pseudocode 5.2 *Driver* Apache Hadoop 78
Pseudocode 5.3 *Driver analytic* (bagian 1) 78
Pseudocode 5.4 *Driver analytic* (bagian 2) 79
Pseudocode 5.5 *Driver analytic* (bagian 3) 80
Pseudocode 5.6 *Daemon* Apache Spark (bagian 1) 80
Pseudocode 5.7 *Daemon* Apache Spark (bagian 2) 81
Pseudocode 5.8 Least Square pada PHP-ML 82
Pseudocode 5.9 Least Square pada Apache Spark (bagian 1) 82
Pseudocode 5.10 Least Square pada Apache Spark (bagian 2) 83
Pseudocode 5.11 Naïve bayes pada PHP-ML 83
Pseudocode 5.12 Naïve bayes pada Apache Spark 84
Pseudocode 5.13 Menyimpan data IoT 85
Pseudocode 5.14 Menampilkan daftar topik terautorisasi 86
Pseudocode 5.15 Menampilkan data dari topik (bagian 1) 87
Pseudocode 5.16 Menampilkan data dari topik (bagian 1) 88
Pseudocode 5.17 Menampilkan detail data dari topik (bagian 1) 88
Pseudocode 5.18 Menampilkan detail data dari topik (bagian 2) 89
Pseudocode 5.19 Menampilkan daftar perangkat IoT 89
Pseudocode 5.20 Mendaftarkan perangkat IoT 90
Pseudocode 5.21 Menampilkan informasi perangkat IoT 91
Pseudocode 5.22 Mengubah perangkat IoT 92
Pseudocode 5.23 Menghapus perangkat IoT 93
Pseudocode 5.24 Menampilkan daftar topik IoT 93
Pseudocode 5.25 Mendaftarkan topik IoT 94
Pseudocode 5.26 Menampilkan informasi topik IoT 95
Pseudocode 5.27 Mengubah topik IoT 96
Pseudocode 5.28 Menghapus topik IoT 97
Pseudocode 5.29 *Data explorer* 98
Pseudocode 5.30 *Data viewer* 99

Pseudocode 5.31 *Data downloader* 101
Pseudocode 5.32 Visualisasi data 102
Pseudocode 5.33 Modul prediksi least square 104
Pseudocode 5.34 Modul klasifikasi naïve bayes 105
Pseudocode 5.35 Keluaran analitik 106
Pseudocode 5.36 Menampilkan daftar pengguna 108
Pseudocode 5.37 Menampilkan informasi pengguna 109
Pseudocode 5.38 Mengubah pengguna 110
Pseudocode 5.39 Menghapus pengguna 111

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

DAFTAR LAMPIRAN

LAMPIRAN A IMPLEMENTASI 167
A.1 Implementasi *IoT Data Storage* 167
A.1.1 Implementasi *Data Storage* Berbasis Apache Spark 167
A.2 Implementasi *IoT Data Analytic* 170
A.2.1 Implementasi *Data Analytic* Berbasis Apache Spark 170
A.3 Implementasi *IoT Visualization Webservice* 172
A.3.1 Implementasi *Webservice* Berbasis Apache Webserver 172
A.3.2 Implementasi *Webservice* Berbasis NGiNX 173
A.3.3 Implementasi Webservice Berbasis Laravel 175
LAMPIRAN B PENGUJIAN 176
B.1 Pengujian Fungsional 176
B.1.1 Autentikasi *Login* Perangkat Menggunakan JWT 176
B.1.2 Menampilkan Daftar Topik Melalui Sebuah *API* 177
B.1.3 Menerima Data Terstruktur Dalam Bentuk JSON 178
B.1.4 Menerima Data Semi Terstruktur Dalam Bentuk XML 179
B.1.5 Menerima Data Tidak Terstruktur Dalam Bentuk JPG 180
B.1.6 Menampilkan Daftar Data yang Disimpan Dalam Suatu Topik Melalui Sebuah *API* 181
B.1.7 Menampilkan Data yang Disimpan Dalam Suatu Topik Melalui Sebuah *API* 182
B.2 Pengujian Skalabilitas 183
B.2.1 Pengujian Pengambilan Data 183
B.2.2 Pengujian Pengiriman Data 185
B.2.3 Pengujian Analitik Data 186

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Kemajuan peradaban manusia yang sangat pesat dibeberapa dekade terakhir merupakan pengaruh dari perkembangan ilmu pengetahuan dan teknologi. Dalam setiap peradaban manusia teknologi selalu berubah bentuk dari revolusi industri pertama yang ditandai dengan munculnya mesin uap hingga revolusi industri ke empat yang dimotori oleh kemunculan *big data* (Dhoot, et al ., 2016). *Big data* sendiri muncul berkat semakin canggihnya teknologi saat ini seperti munculnya *Internet of things*, blockchain hingga web semantik. Salah satu teknologi yang berperan paling besar dalam perkembangan *big data* adalah penggunaan *Internet of thing* yang semakin massif dalam beberapa tahun terakhir.

Internet of things (IoT) sendiri sudah menjadi primadona sejak pertama kali diperkenalkan pada tahun 1982 silam, hal ini dikarenakan kemampuannya saat diimplementasikan yang sangatlah luar biasa. Kelebihanya dalam mengumpulkan, menggabungkan, menyimpan, memproses, dan menggunakan data menjadi hal yang sangat penting, apalagi data yang dihasilkan oleh perangkat IoT sangatlah berukuran besar (Jiang, et al., 2017). Salah satu fungsi perangkat IoT adalah berfungsi untuk mengumpulkan data maupun informasi dari beberapa lingkungan, yang selanjutnya akan diproses dengan cara yang bermacam-macam sesuai dengan kebutuhan. Umumnya, IoT mempunyai 3 (tiga) komponen utama yaitu benda yang dilengkapi modul IoT (sensor, aktuator), perangkat koneksi internet (*middleware*), dan *cloud* yang berfungsi sebagai tempat penyimpanan maupun pemrosesan data (Tuti Wardani, et al., 2017).

Pada dasarnya komponen *cloud computing* menangani pemrosesan dari data yang dihasilkan oleh infrastruktur IoT. Komponen *cloud computing* lazimnya memiliki *resource* yang sangat besar dan cenderung tidak terbatas untuk memproses data. Akan tetapi komponen *cloud computing* tidak serta merta bebas dari berbagai persoalan. Umumnya persoalan yang muncul pada *cloud* adalah di sisi heteregonitas dari data yang dihasilkan oleh perangkat IoT, sumberdaya dari *cloud computing* yang tidak dimanfaatkan secara maksimal, kemampuan manajemen dan orkestrasi pengguna, perangkat maupun data secara dinamis dan yang terakhir adalah kemampuan personalisasi dari *cloud computing* terhadap kebutuhan dari pengguna dan servis yang dibutuhkan pengguna (Abdur Rahim, et al., 2014).

Dalam penelitian yang dilakukan oleh Tuti Wardani berjudul "Pengembangan *Webservice* pada *IoT Data Storage* dengan Pendekatan Semantik dan Penambahan Mekanisme Autentikasi" telah dikembangkan sebuah sistem *IoT data storage* berbasis MongoDB dan GridFS yang digunakan untuk mengatasi permasalahan *semantic interopability* dan heterogenitas data pada *IoT data storage*, dengan melakukan *filtering* sebelum melakukan penyimpan data (Tuti Wardani, et al., 2017). Pada penelitian tersebut digunakan MongoDB sebagai lokasi penyimpanan data. Selain itu dalam penelitian tersebut juga dilakukan

sebuah mekanisme autentikasi dalam pertukaran data menggunakan JSON Web Token (JWT) untuk memberikan lapisan keamanan tambahan.

Dalam penelitian lainnya yang dilakukan oleh Kim Jaehoo dan Lee Jang-Woon pada tahun 2014 dikembangkan juga sebuah *IoT framework* yang dipublikasikan dalam sebuah jurnal dengan judul "*OpenIoT: An Open Service Framework for The Internet of things*" dengan tujuan untuk mengatasi permasalahan personalisasi servis pada cloud computing (Lee Jang-Woon, et al., 2014). Pada penelitian tersebut dikembangkan sebuah *IoT framework* yang terdiri dari *data storage* dan *webservice* yang dapat digunakan oleh pengguna pada sebagai lokasi penyimpanan data yang dihasilkan oleh perangkat IoT. Sistem *framework* yang dibangun sudah mampu menyimpan data yang dikirim dari pengguna ke *IoT framework*. Selain itu *framework* ini juga menyediakan sebuah kumpulan *API* yang dapat digunakan oleh pengguna untuk mengakses data yang sudah disimpan ke dalam sistem.

Penelitian yang dilakukan oleh Onal pada tahun 2017 dikembangkan juga sebuah *framework IoT* dalam jurnal berjudul "*Weather Data Analysis and Sensor Fault Detection Using an Extended IoT Framework With Semantics, Big data, and Machine Learning*" dengan tujuan utama untuk mencoba memaksimalkan penggunaan dari teknologi IoT*, big data analytic dan cloud computing* (Onal, et al., 2017). Pada penelitian tersebut dikembangkan sebuah *framework* IoT yang terdiri dari *data storage*, *data analytic* dan juga *webservice* yang dapat digunakan untuk menganalisis data cuaca dan juga mendeteksi kesalahan pada sensor IoT yang digunakan. Pada penelitian ini digunakan metode *semantic* untuk melakukan klasifikasi data sebelum melakukan penyimpanan data. Selain itu penelitian ini juga mengimplementasikan *machine learning* untuk melakukan analisis data.

Dengan hadirnya beberapa penelitian sebelumnya beberapa permasalahan pada *IoT cloud computing* sudah berhasil dijawab walaupun hanya secara parsial atau per komponen saja. Seperti permasalahan personalisasi *cloud computing* terhadap servis yang dibutuhkan, heterogenitas data, serta penggunaan sumberdaya cloud computing yang masih rendah. Pada penelitian yang dilakukan oleh Tuti Wardani dalam "Pengembangan *Webservice* pada *IoT Data Storage* dengan Pendekatan Semantik dan Penambahan Mekanisme Autentikasi" sudah dilakukan implementasi MongoDB sebagai lokasi penyimpanan data IoT (Tuti Wardani, et al., 2017). Sayangnya dalam penelitian tersebut belum memiliki kemampuan manajemen topik ataupun pengguna yang bisa disebut kemampuan personalisasi servis. Sedangkan pada penelitian "*OpenIoT: An Open Service Framework for The Internet of things"* yang dilakukan oleh Kim Jaehoo sudah diterapkan kemampuan personalisasi servis ditandai dengan adanya kemampuan manajemen topik, pengguna maupun perangkat (Kim Jaehoo, et al., 2014). Akan tetapi pada penelitian tersebut belum diimplementasikan sebuah data analitik, sehingga data yang masuk dalam *framework* tersebut hanya dapat disimpan saja, tanpa ada pengolahan lebih lanjut. Lalu pada penelitian yang dilakukan Onal dalam "*Weather Data Analysis and Sensor Fault Detection Using an Extended IoT Framework With Semantics, Big data, and Machine Learning"* sudah

diimplementasikan penggunaan data analitik berupa sebuah *machine learning* yang digunakan untuk meningkatkan penggunaan dari sumber daya *cloud* (Onal, et al.,2017). Namun pada penelitian tersebut *framework* yang dihasilkan hanya mampu menerima dan memproses satu jenis data saja, yaitu data cuaca saja. Sehingga masih belum ada kemampuan untuk melakukan manajemen perangkat, topik, maupun data dari pengguna yang bisa disebut sebagai kemampuan personalisasi servis.

Berangkat dari penelitian sebelumnya diusulkan sebuah personal data analitik menggunakan PHP-ML dan Apache Spark pada *IoT cloud apps*. Penelitian ini menggabungkan beberapa teknologi dari penelitian sebelumnya seperti kemampuan personalisasi servis yang dimiliki oleh *Open IoT* (Kim Jaehoo, et al., 2014) dan juga kemampuan untuk memanfaatkan sumberdaya pada *cloud* secara maksimal yang terdapat pada penelitian *Extended IoT Framewok* (Onal, et al., 2017). Sehingga diharapkan akan menghasilkan sebuah *IoT cloud apps* yang terdiri dari *IoT data storage* dan *IoT data analytic* yang mampu menjawab beberapa permasalahan pada *cloud computing* secara utuh. Fokus utama pada penelitian ini sebenarnya ada pada sisi pengimplementasian personal data analitik menggunakan PHP-ML dan Apache Spark, akan tetapi pada penelitian ini juga akan diimplementasikan kemampuan *semantic filtering* penelitian yang dibangun oleh Tuti Wardani untuk mengatasi permasalahan *semantic interopability* dan juga lapisan keamanan tambahan menggunakan JWT. Pendekatan modular juga akan diimplementasikan dalam bentuk kemampuan dari sistem untuk menerima tambahan modul analitik, sehingga sistem dapat mengatasi permasalahan personalisasi servis secara maksimal. Pada penelitian ini juga akan dikembangkan sebuah *IoT visualization webservice* yang dapat menampilkan informasi atau data sesuai yang diinginkan oleh pengguna, sehingga dapat memudahkan dalam melakukan monitoring dan *decision making* dalam sistem IoT yang akhirnya akan memberikan kemudahan dalam pengelolaan perangkat maupun data dari sistem IoT.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dibahas sebelumnya, maka rumusan masalah yang menjadi pokok penelitian yaitu:

1. Bagaimana mengembangkan sebuah personal data analitik berbasis PHP-ML dan Apache Spark pada *IoT cloud apps* ?
2. Bagaimana personal data analitik berbasis PHP-ML dan Apache Spark pada *IoT cloud apps* ditinjau dari segi fungsional dan skalabilitas dari sistem?

1.3 Tujuan

Tujuan dari penelitian ini:

1. Mengembangkan sebuah personal data analitik berbasis PHP-ML dan Apache Spark pada *IoT cloud apps* yang dapat diakses secara *realtime* dan menghasilkan hasil yang baik.

2. Setelah tahap implementasi dan integrasi selesai dilakukan, berikutnya dilakukan pengujian untuk mengetahui fungsional dari sistem dan skalabilitas dari sistem.

1.4 Manfaat

Manfaat dari penelitian ini :

1. Memudahkan *developer* dalam mengembangkan sistem IoT tanpa perlu memikirkan aspek manajemen perangkat, *data storage* dan *data analytic*.
2. Hasil penelitian ini juga dapat digunakan sebagai dasar dari penelitian-penelitian selanjutnya karena masih banyak aspek dalam penelitian ini yang dapat ditingkatkan.
3. Mampu menghasilkan jurnal dan prototipe *IoT cloud apps*.
4. Mampu menghasilkan laporan akhir, buku log, dan juga laporan keuangan.

1.5 Batasan Masalah

Pada penelitian ini, fokus permasalahan yang akan dibahas yaitu :

1. Pengujian berfokus pada kinerja sistem diantaranya proses pengambilan data, pengiriman data dan juga analisis data.
2. Data yang digunakan untuk pengujian adalah data yang semi terstuktur seperti JSON, XML dan data tidak terstruktur seperti JPG,PNG, MP4, dan MP3.
3. Aplikasi analitik data yang digunakan yaitu Apache Spark dan PHP-ML.
4. Aplikasi visualisasi data dibangun menggunakan *framework webservice* dalam bahasa pemrograman PHP.
5. Tempat penyimpanan data yang digunakan yaitu MongoDB, GridFS dan Apache Hadoop.
6. Kapasitas *data storage* yaitu 100 GB.

1.6 Sistematika Pembahasan

Sistematika skripsi ini dibagi menjadi tujuh bab dengan masing – masing bab diuraikan sebagai berikut:

1. BAB I : Pendahuluan

   Bab pendahuluan mencakup latar belakang yang merupakan alasan penulisan, rumusan masalah sebagai inti permasalahan, batasan masalah sebagai batas cakupan bahasan, tujuan dari penelitian, manfaat yang diperoleh dari penelitian, dan sistematika penulisan laporan penelitian.

2. BAB II : Landasan Kepustakaan

   Bab landasan kepustakaan menjelaskan tentang teori – teori yang di pakai dalam penelitian, temuan dan bahan penelitian yang mungkin sebelumnya diperoleh dari beberapa referensi yang menunjang penelitian dalam penulisan skripsi. Ada juga teori – teori yang tercakup dalam bab ini adalah mengenai *Internet of things (*IoT*), IoT data storage, IoT data analytic ,* dan *IoT visualization webservices.*

3. BAB III : Metodologi

   Bab Metodologi tahapan-tahapan langkah-langkah dalam melakukan penelitian. Langkah-langkah seperti perancangan, implementasi, pengujian, serta analisis hasil dibahas secara umum. Selain itu pada bab metodologi juga dijelaskan secara singkat mengenai perangkat keras yang dibutuhkan, arsitektur sistem.

4. BAB IV : Perancangan

   Bab Perancangan menjelaskan tentang bagaimana arsitektur aplikasi IoT yang sesuai dengan objek penelitian. Dibuat pula rancangan sistem yang mencakup rancangan arsitektur, skema penyimpanan data, skema analitik data dan juga skema visualisasi data. Skenario pengujian yang mencakup tujuan penelitian juga disampaikan dalam bab ini.

5. BAB V : Implementasi

   Bab Implementasi merupakan bab yang menjelaskan tentang bagaimana sistem akan diterapkan berdasarkan sistematika yang telah dibuat sebelumnya secara rinci beserta dengan langkah-langkah pengerjaan yang dilakukan serta menampilkan Gambar-Gambar implementasi yang dilakukan.

6. BAB VI: Pengujian

   Pada Bab Pengujian akan menjelaskan mengenai hasil dari sistem setelah sistem berhasil diimplementasikan, pengujian akan dilakukan guna mengetahui kemampuan sistem yang telah diimplementasikan dengan skenario pengujian sebagai patokan.

7. BAB VII : Pembahasan

   Bab Pembahasan menyajikan data hasil pengujian beserta analisis dari implementasi sistem yang telah dibangun. Hasil pengujian didapatkan dari pengujian yang telah dilakukan sesuai dengan skenario pengujian yang telah ditentukan. Dari hasil tersebut dapat dilakukan analisis terhadap kinerja sistem, serta mengetahui bagaimana sistem dapat mencapai tujuan.

8. BAB VIII : Penutup

   Pada Bab Penutup dipaparkan kesimpulan dari pelaksanaan penelitian. Kesimpulan ini dibuat dengan hasil pengujian dan analisis terhadap perancangan dan implementasi arsitektur sistem sebagai dasarnya. Saran-saran juga diberikan agar hasil dari penelitian ini dapat diperbaiki dan disempurnakan apabila penelitian ini dikembangkan kemudian.

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

BAB 2 LANDASAN KEPUSTAKAAN

Untuk mendukung penelitian ini, peneliti mengumpulkan studi literatur dan dasar dasar teori yang dibutuhkan untuk menyelesaikan penelitan ini. Studi literatur yang digunakan untuk mengerjakan berasal dari beberapa penelitian atau publikasi sebelumnya seperti skripsi, jurnal maupun buku.

2.1 Kajian Pustaka

Pada penelitian ini, penulis menggunakan penelitian yang berkaitan dengan topik yang diambil penulis saat ini, serta memaparkan hasil implementasi topik dari beberapa referensi yang berkaitan dengan topik. Penelitian yang digunakan sebagai referensi yaitu penelitian mengenai *Internet of things* (IoT)*, big data analytic* dan juga *IoT applications*. Untuk mempermudah pembaca dalam memahami dan membandingkan kajian pustaka dengan penelitian akan di sajikan dalam bentuk Tabel. Berikut penulis lampirkan Tabel 2.1 dan Tabel 2.2:

Tabel 2.1 Kajian pustaka (bagian 1)

Judul Jurnal	Metode	Kesimpulan	Perbedaan
Pengembangan *Webservice* pada *IoT Data storage* dengan Pendekatan Semantik dan Penambahan Mekanisme Autentikasi. (Hamid TW, et al. 2017)	Penggunaan metode semantik dalam memisahkan data sensor.	Sistem dapat memisahkan data antara yang terstruktur dan yang tidak terstruktur.	Pada penelitian yang akan dilakukan akan dilakukan penambahan sistem analitik data dan juga visualisasi data, selain itu sistem juga akan diterapkan pada lingkungan *cloud computing*.
Topic Based IoT data storage framework For Heterogenous Sensor Data. (Pramukantoro E.S , et al. 2017)	Menggunakan nama topik untuk menyeleksi data sensor yang masuk dan menempatkannya berdasarkan nama topik.	Metode dapat digunakan untuk menyimpan data yang beragam dan volume yang besar.	Penelitian yang peneliti ajukan akan menggunakan pendekatan semantik dalam menyeleksi data yang masuk, kriteria data yang diterima akan bersifat dinamis

Tabel 2.2 Kajian pustaka (bagian 2)

			dan dapat berubah sesuai keinginan pengguna, tanpa harus mengubah kode program.
Weather Data Analysis and Sensor Fault Detection using an Extended IoT Framework with Semantics, Big data, and Machine Learning. (Onal A.C, et al. 2017)	Mengintegrasikan teknologi IoT, *big data analytic* dan *cloud computing*.	Sistem dapat berjalan dan menghasilkan hasil yang cukup baik.	Pada penelitian yang akan dilakukan akan menambahkan komponen personalisasi servis.
OpenIoT: An Open Service Framework for the Internet of things. (Kim J. et al. 2014)	Membuat sistem IoT untuk menjembatani antara pengguna dengan sistem IoT menggunakan sebuah *API* terbuka.	Sistem dapat digunakan, dan sistem dapat digunakan untuk penggunaan komersil.	Pada penelitian ini akan ditambahkan sebuah sistem analitik dan visualisasi tidak hanya *API* saja.

2.2 Dasar Teori

2.2.1 *Internet of things*

Internet of things (IoT) merupakan sebuah model baru yang berkembang sangat pesat dalam lingkup Teknologi telekomunikasi nirkabel yang modern. Gagasan dasar dari ide ini adalah persebaran berbagai jenis objek atau benda seperti sensor, aktuator, mobile phone, *radio-frequency identification* (RFID) dan lain-lain yang melalui skema penyamaan pengalamatan yang unik yang mampu berinteraksi satu sama lain dan bekerja sama dalam mencapai tujuan (Atzori, et al., 2010). Akan tetapi saat ini*, Internet of things* dapat dikatakan tidak lagi sebagai teknologi yang baru dan aneh karena tiap hari sudah sangat banyak benda-benda yang tekoneksi dengan internet seperti gedung maupun kendaraan. Berdasarkan penelitian terbaru diproyeksikan terdapat 20 milliar benda-benda yang akan terhubung satu sama lain dengan internet pada tahun 2020 (Grgic, et al., 2016).

Internet of things menandai evolusi internet generasi selanjutnya dan memperluas kemampuannya untuk menggabungkan, memproses dan mendistribusikan data yang akan diolah menjadi informasi. *Internet of things* dapat ditemukan hampir di seluruh lingkungan kehidupan manusia seperti kehidupan sosial, pendidikan, pariwisata, kesehatan, manajemen distribusi suplai, militer, industri, dan lain-lain. Perangkat IoT dapat memproduksi data sensor dalam jumlah yang sangat besar untuk mencerminkan lingkungan fisik dari suatu benda atau manusia. Seperti yang sering dibawa oleh perangkat *Internet of things* yaitu sumber dan kapasitas penyimpanan data yang terbatas, sensor data butuh untuk ditransmisikan dan disimpan pada kerangka yang kaya akan *resource* seperti *cloud*. Di sisi lain, dalam proses analisis riwayat sensor data merupakan hal yang sangat terpenting dalam pembuatan keputusan pada berbagai macam aplikasi berbasis IoT (Li, et al., 2017).

2.2.2 IoT Application

Internet of things application atau dapat disingkat *IoT apps* adalah sebuah sistem IoT yang dapat terdiri dari sebuah sistem akuisi data, pengolahan data *(IoT data analytic)* dan juga visualisasi data *(IoT visualization webservices)*. Beberapa gabungan dari sistem tersebut yang bekerja secara terpadu untuk suatu tujuan bersama yang sudah ditetapkan juga dapat diklasifikasikan sebagai *IoT apps* (Kim J, et al. 2014).

Dengan adanya *IoT apps* maka data yang dihasilkan dari perangkat IoT akan lebih bernilai, daripada data mentah yang dihasilkan dari sistem. Karena data yang dihasilkan dari perangkat IoT itu memiliki karakteristik *volume* atau jumlah yang besar, akan tetapi *density* atau densitas yang sangat rendah, sehingga tidak akan memberikan nilai apapun selama data yang dihasilkan ttersebut idak diproses.

2.2.2.1 IoT Data Analytic

IoT data analytic adalah sebuah komponen dari *IoT apps* yang berfungsi untuk mengolah data yang dihasilkan daripada perangkat IoT. Sistem *IoT data analytic* sendiri biasanya terdapat berdampingan dengan *IoT data storage* atau tempat penyimpanan data.

Pada dasarnya *IoT data analytic* diklasifikasikan menjadi 2 (dua) jenis yang pertama adalah *realtime data analytic* yang berarti data akan langsung diolah sesaat sesudah data diterima oleh sistem, sistem ini biasanya digunakan pada aplikasi-aplikasi kritikal seperti sistem monitoring peluncuran satelit ataupun sistem monitoring reactor nuklir yang memerlukan pengambilan keputusan secara *realtime*. Kemudian ada juga *batch processing data analytic* yang berarti data tidak akan langsung diolah saat data diterima dan alih-alih akan menunggu sesuai kondisi yang ditentukan, sistem ini biasanya diadopsi pada sistem yang tidak kritikal seperti sistem survey tv dan iklan yang hasilnya tidak diperlukan secara *realtime* (Bashir, M.R., et al. 2016).

2.2.2.2 IoT Visualization Webservices

IoT visualization webservice adalah komponen penting yang terdapat di dalam *IoT apps* dikarenakan hasil dari akuisisi data hingga pemrosesan tidak akan berguna jika tidak dapat ditampilkan kepada pengguna atau sistem.

Dalam membangun *IoT visualization webservices* terdapat beberapa komponen yang harus dicermati untuk menghasilkan sistem IoT yang optimal diantaranya adalah (1) sistem *IoT visualization webservice* harus mampu mengolah *metadata* yang terdapat pada data yang dihasilkan oleh perangkat IoT. (2) sistem *IoT visualization webservices* harus bersifat interaktif dan harus membuat pengguna terlibat penuh di dalam sistem. (3) sistem *IoT visualization webservices* harus mempu menampilkan hasil yang natural sesuai dengan keadaan yang ada pada perangkat IoT (Marjani, M., et al. 2017).

2.2.3 Cloud Computing

Cloud computing adalah gabungan dari penggunaan teknologi komputer dan pengembangan berbasis internet *(cloud)*. *Cloud* atau awan merupakan metafora dari internet, *cloud* dalam istilah *cloud computing* juga merupakan abstraksi dari infrastruktur kompleks yang disembunyikan di dalamnya.

Cloud computing adalah suatu metoda komputasi di mana kapabilitas terkait teknologi informasi disajikan sebagai suatu layanan *(as a service)*, sehingga pengguna dapat mengaksesnya lewat Internet tanpa mengetahui apa yang ada di dalamnya, ahli dengannya, atau memiliki kendali terhadap infrastruktur teknologi yang membantunya. Menurut sebuah makalah tahun 2008 yang dipublikasi *IEEE Internet Computing* "*Cloud computing* adalah suatu paradigma di mana informasi secara permanen tersimpan di server di internet dan tersimpan secara sementara di komputer pengguna *(client)* termasuk di dalamnya adalah desktop, komputer tablet, *notebook*, *handheld*, sensor-sensor, monitor dan lain-lain."(Hewwit Carl. 2008).

2.2.4 Personal

Personal adalah sebuah konsep tentang individu sebagai suatu entitas atau suatu unit. Personal juga berarti bahwa suatu hal tersebut merujuk atau mengikuti dari entitas yang menaunginya, atau dapat diartikan personal adalah suatu konsep tentang kepemilikan suatu hal. Personalisasi adalah penyesuaian suatu informasi sistem dengan preferensi atau kebutuhan pengguna. Personalisasi bisa berkontribusi pada peningkatan hubungan pelanggan atau efisiensi organisasi. (Reis , J.L., et al. 2014)

Dalam penelitian ini yang dimaksud personalisasi adalah kemampuan dari sistem untuk mengikuti atau menyesuaikan dengan kebutuhan dari pengguna. Hal yang dapat dilakukan personalisasi dalam sistem ini melingkupi beberapa hal seperti topik, perangkat, data yang dihasilkan oleh pengguna dan juga kemampuan untuk menyesuaikan sistem analitik terhadap metode algoritme, mesin analitik dan juga data yang dimiliki oleh pengguna.

2.2.5 Kajian Pustaka Metode Pengujian

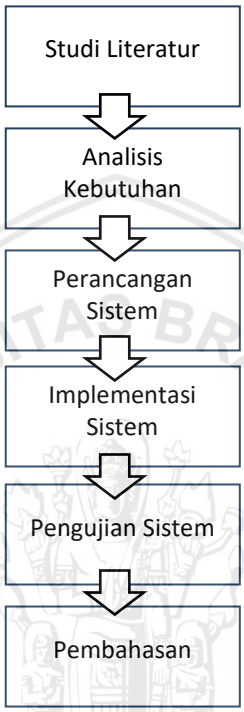
Peneliti mengumpulkan kajian-kajian mengenai pengujian yang memiliki model penelitian sejenis yang telah dilakukan oleh peneliti sebelumnya. Peneliti merangkumnya dalam sebuah Tabel 2.3 berikut :

Tabel 2.3 Kajian pustaka metode pengujian

Parameter	Metode Pengujian	Referensi
Fungsional	Di dalam penelitian tersebut digunakan pengujian secara fungsional dari fitur-fitur yang ada di dalam sistem.	*A Data-Centric Framework for Development and Deployment of Internet of things Applications In Clouds*. (Khodadadi F, et al. 2015)
Skalabilitas	Di dalam penelitian tersebut digunakan pengujian untuk menguji tingkat skalabilitas dari sistem menggunakan stress test. Parameter yang akan diuji antara lain waktu dalam pengambilan data penyimpanan data, serta analisis data.	Pengembangan *Webservice* pada *IoT Data Storage* Dengan Pendekatan Semantik dan Penambahan Mekanisme Autentikasi. (Hamid TW, et al. 2017)

BAB 3 METODOLOGI

Pada bab ini akan dijelaskan langkah-langkah yang dilakukan untuk menyelesaikan penelitian ini. Langkah -langkah tersebut meliputi studi literatur, perancangan, implementasi, pengujian, analisis hasil pengujian dan pengambilan kesimpulan. Metodologi penelitian yang akan digunakan pada penelitian ini ditunjukkan pada Gambar 3.1 :



Gambar 3.1 Langkah-langkah penelitian

3.1 Studi Literatur

Studi literatur merupakan kegiatan yang dilakukan untuk menentukan objek penelitian yang sesuai dengan topik yang diambil. Studi literatur juga berfungsi sebagai referensi untuk penelitian yang dilakukan. Studi literatur digunakan untuk mendukung penelitian dalam penyelesaian masalah agar tercapainya tujuan penelitian. Teori pendukung dapat diperoleh dari jurnal, buku, dan penelitian sebelumnya. Studi literatur dalam penelitian ini antara lain berkenaan dengan *Internet of things* (IoT), *IoT data storage – data analytic*, tantangan *data storage – data analytic* dalam IoT, *integration testing*, serta metodologi pengujian.

3.2 Analisis Kebutuhan

Rancangan analisis kebutuhan sistem dituliskan berdasarkan kerangka kerja rekayasa kebutuhan. Analisis kebutuhan sistem di tahap awal dijelaskan dengan menggunakan *system context* yang terdiri dari aspek subyek, usage, sistem

teknologi informasi, dan development. Adapun analisis kebutuhan sistem dilihat dari aspek subyek adalah sebagai berikut.

3.2.1 Komponen Komunikasi IoT dan Komponen *Sensor Node*

1. Terdapat dua program emulasi *sensor node* yang akan bertindak seolah-olah sebagai alat yang melakukan akuisisi data lingkungan.
2. *Sensor node* mengirimkan data ke *IoT cloud apps* menggunakan internet.

3.2.2 Komponen *IoT Cloud Apps*

1. *IoT cloud apps* terdiri dari panel manajemen pengguna, *IoT data storage*, *IoT data analytic* dan *IoT data visualization webservice*.
2. Pengguna mampu mendaftar pada sistem *IoT cloud apps*.
3. Pengguna mampu login pada sistem *IoT cloud apps*.
4. Pengguna mampu mendaftarkan topik mereka sendiri pada sistem *IoT cloud apps*.
5. Pengguna mampu mengelola data yang masuk pada topik yang didaftarkan pada sistem *IoT cloud apps*.
6. Pengguna mampu meminta sistem untuk melakukan analisis data beserta menampilkan hasil analisis data.
7. *IoT data storage* bertugas menampung data
8. Data yang berada di *data storage* kemudian dianalisis menggunakan *IoT data analytic* yang kemudian hasilnya akan kembali disimpan di *IoT data storage*.
9. Data hasil analisis kemudian ditampilkan menggunakan *IoT visualization webservice*.

3.2.3 Komponen Arsitektur *Cloud*

1. *Virtual Private Server (VPS)* yang memiliki *IP* publik sehingga dapat diakses dari luar jaringan.
2. Tiga buah *VPS* untuk mengimplementasikan arsitektur terdistribusi dari sistem.
3. Beberapa domain yang digunakan untuk memudahkan pengembangan dan pengaksesan.
4. Aplikasi yang dapat digunakan untuk membangun arsitektur *cloud* mulai dari *webservice*, *data storage*, hingga *data analytic.*

3.3 Perancangan Sistem

Perancangan sistem dibuat dengan menggambarkan arsitektur sistem berdasarkan hasil analisis kebutuhan. Perancangan ini juga didasarkan pada teori yang diperoleh pada kajian pustaka berupa perancangan lingkungan sistem, perancangan infrastruktur, hingga perancangan aplikasi. Perancangan sistem ini dibuat untuk mempermudah implementasi, pengujian, dan analisis.

3.3.1 Perancangan Lingkungan Sistem

Pada sistem yang dikembangkan dalam penelitian ini terdapat tiga komponen utama, yaitu *IoT data storage*, *IoT data analytic* dan *IoT visualization webservice* app. Lingkungan sistem dalam penelitian ini akan dirancang mengikuti seluruh kebutuhan dari setiap komponen yang sudah ditentukan. Perancangan ini dibutuhkan agar penelitian menjadi lebih terarah dan gambaran umum dari sistem dapat dengan mudah dipahami, sehingga proses penelitian menjadi lebih cepat dan mudah.

3.3.2 Perancangan Infrastruktur

Pada penelitian ini terdapat terdapat bagian infrastruktur yang digunakan untuk meletakkan semua aplikasi beserta konfigurasi dari aplikasi yang dikembangkan. Infrastruktur ini terdiri dari *IoT data storage* yang digunakan untuk menyimpan data, *IoT data analytic* yang digunakan untuk melakukan analitik pada data yang disimpan, dan yang terakhir adalah *IoT visualization webservice app* yang digunakan untuk menampilkan data yang ada pada sistem. Pada perancangan ini akan berisi semua konfigurasi dan instalasi untuk melalukan implementasi sistem.

3.3.2.1 Perancangan IoT Data Storage

Pada penelitian ini akan terdapat banyak sekali perubahan dari pada penelitian sebelumnya, yang pertama MongoDB sebagai sistem *database* untuk menyimpan *data analytic* beserta data sensor. Kemudian ada adalah digunakannya Apache Hadoop untuk menyimpan data yang akan ataupun telah diakses oleh Apache Spark dalam proses analitik.

(a) Perancangan Data Storage Berbasis MongoDB dan GridFS

MongoDB sendiri akan digunakan sebagai sistem storage utama yang digunakan untuk menyimpan data dan juga sebagai penghubung utama ke aplikasi Visualisasi data, MongoDB digunakan karena kinerja *realtime* aplikasi lebih baik daripada Apache Hadoop dan juga lebih banyak sup*port* ke aplikasi *webservice*, sehingga memudahkan dalam proses implementasi.

GridFS juga akan digunakan bersamaan dengan MongoDB, jika pada MongoDB akan menyimpan data terstruktur maka dalam GridFS akan digunakan untuk menyimpan data tidak terstruktur.

(b) Perancangan Data Storage Berbasis Apache Hadoop

Untuk sistem storage sendiri akan menggunakan Apache Hadoop, aplikasi Apache Hadoop akan digunakan untuk menyimpan data analisis yang akan diproses dan sudah diproses oleh Apache Spark.

Peneliti merancang penggunaan Apache Hadoop untuk data analitik dari Apache Spark, dikarenakan karakteristik Apache Hadoop sendiri sangat cocok untuk secara terdistribusi, sehingga akan memudahkan dalam memproses secara *parallel* menggunakan Apache Spark.

3.3.2.2 *Perancangan* IoT Data Analytic

Pada penelitian kali ini akan dikembangkan sebuah IoT *data Analytic*, komponen ini merupakan komponen baru yang ditambahkan dari pada penelitian sebelumnya. Komponen ini berguna untuk melakukan pengolahan analisis data yang berasal dari MongoDB dan hasil analisis sebelumnya.

(a) Perancangan Data Analytic Berbasis Apache Spark

Apache Spark akan peneliti gunakan sebagai aplikasi analitik data utama, data yang dianalisis berasal dari MongoDB yang merupakan data dari sensor. Apache Spark sendiri dapat digunakan untuk melakukan analisis data dari yang berukuran kecil hingga berukuran sangat besar atau *big data*, tergantung pada lingkungan yang digunakan saat mengimplementasikan Apache Spark.

(b) Perancangan Data Analytic Berbasis PHP-ML

PHP-ML atau PHP-Machine Learning adalah sebuah library analitik yang dikembangkan untuk bahasa pemrograman PHP, library ini bersifat *lightweight* atau ringan. Sehingga cocok untuk melakukan analisis data yang berukuran kecil atau dibawah 100.000 *entry*.

3.3.2.3 Perancangan IoT Visualization Webservice *App*

Pada penelitian ini akan dirancang juga sebuah aplikasi visualisasi data hasil analitik IoT. Aplikasi ini sendiri akan dibangun menggunakan bahasa pemrograman PHP menggunakan *framework* Laravel.

Pada aplikasi visualisasi ini dilakukan pengimplementasian beberapa fitur. Beberapa fitur yang tersedia diantaranya adalah melakukan penggunaan modul analitik dan visualisasi kemudian ada juga komponen *API* yang digunakan untuk mengambil data pada sistem.

Penelitian ini juga akan memberikan sebuah tambahin komponen untuk melakukan manajemen pengguna, fungsi dari komponen ini adalah memberikan kebebasan kepada pengguna untuk melakukan kustomisasi sistem seperti manajemen perangkat IoT, manajemen data dan juga manajemen analitik data.

(a) Perancangan Webservice Berbasis Apache Webserver

Apache Webserver akan digunakan sebagai *webserver* dari sistem ini. Fungsinya adalah untuk menangani *request* dari client dan kemudian menjalankan kode program yang tersedia di sistem.

(b) Perancangan Webservice Berbasis NGINX

Nginx akan digunakan sebagai *reverse proxy* dari sistem yang digunakan. Kelebihan menggunakan *reverse proxy* diantaranya dapat memisah aliran data yang masuk dari internet antara terhadap *file* dan program.

(c) Perancangan Webservice Berbasis Redis

Redis akan digunakan sebagai lokasi penyimpanan sementara dari sistem. Data yang akan disimpan dalam redis antara lain sessions dan queue dari proses yang berjalan di *webservice*.

(d) Perancangan Webservice Berbasis Laravel

Laravel akan digunakan sebagai *framework* dalam pengembangan sistem *webservice*. *Framework* Laravel menawarkan penggunaan kembali terhadap beberapa komponen sistem sehingga dapat mempercepat proses pengembangan dari sistem itu sendiri.

3.3.3 Perancangan Aplikasi

Perancangan aplikasi adalah tahap untuk membuat rancangan dari aplikasi yang digunakan. Bagian ini sendiri adalah perancangan untuk aplikasi yang digunakan pada infrastruktur yang dikembangkan. Aplikasi yang dikembangkan sendiri terdiri dari beberapa komponen yaitu *IoT data storage, IoT data analytic dan IoT visualization webservice app.*

3.3.4 Perancangan Data

Perancangan data adalah tahap untuk mendefinisikan data apa saja yang dapat diterima dalam sistem. Dalam tahap ini akan di definisikan limitasi dari sistem berupa data apa saja yang dapat diterima, disimpan ataupun diolah dari sistem.

3.3.4.1 Perancangan Data Semi Terstruktur

Data semi terstruktur adalah salah satu data yang dapat diterima oleh sistem. Dalam data ini akan diklasifikan lagi menjadi data JSON dan data XML. Data yang masuk dalam kategori ini akan mampu disistem dalam semua topik data yang terstruktur, semi terstruktur maupun tidak terstruktur.

3.3.4.2 Perancangan Data Tidak Terstruktur

Data tidak terstruktur adalah salah satu data yang dapat diterima oleh sistem. Dalam data ini akan diklasifikan lagi menjadi data raw dan binary. Data yang masuk dalam kategori ini akan hanya mampu disimpan dalam topik yang tidak terstruktur

3.4 Perancangan Pengujian

Perancangan pengujian diperlukan agar diketahui batas minimal pengujian dan melakukan penelitian berdasarkan skenario pengujian yang telah dibuat. Dalam perancangan ini akan di definisikan beberapa hal yang akan diuji pada sistem.

3.5 Implementasi Sistem

Pada tahap ini akan dilakukan integrasi dari sistem berbasis penelitian sebelumnya. Berikutnya akan dilakukan pemindahan *data storage* dari server di lingkungan kampus ke *VPS* di Microsoft Azure, yang umum dikenal dengan *cloud computing*. Alasan penggunaan *VPS* di Microsoft Azure sendiri adalah karena peneliti memiliki fasilitas berupa beberapa *VPS* yang sudah berjalan pada Microsoft Azure. Selain itu pada tahap ini juga dilakukan proses *coding* berdasarkan perancangan sistem yang telah dilakukan, proses ini akan membuat sistem menggunakan bahasa pemrograman yang berbeda dari penelitian sebelumnya yaitu *hybrid* antara Python dan PHP.

3.6 Pengujian Sistem

Tahap pengujian dilakukan agar diketahui tingkat keberhasilan serta kekurangan yang diperoleh oleh sistem yang diusulkan.

3.6.1 Pengujian Fungsional

Pengujian fungsional, untuk mengetahui apakah sistem sudah dapat berjalan sesuai perancangan. Pengujian ini dilakukan berdasarkan parameter fungsionalitas yang telah ditentukan pada saat perancangan pengujian. Hasil dari tahap ini adalah untuk mengetahui sejauh mana sistem dapat berjalan sesuai kebutuhan fungsional yang telah didefinisikan.

3.6.2 Pengujian Non Fungsional

Pengujian skalabilitas sistem, yang nantinya juga akan dilakukan untuk mengetahui kinerja dari parameter yang akan diuji antara lain waktu dalam pengambilan, penyimpanan dan analisis data. Pengujian ini dilakukan berdasarkan tahap-tahap yang telah ditentukan pada saat perancangan pengujian. Hasil dari tahap ini adalah untuk mengetahui sejauh mana sistem dapat berjalan atau untuk melihat kemampuan sistem yang dibangun dalam menangani pengguna.

3.7 Pembahasan

Setelah melakukan perancangan dan pengujian maka ditahap ini akan dilakukan pembahasan dari pengujian yang sudah dilakukan, ditahap ini juga akan dihasilkan sebuah kesimpulan dan juga saran untuk penelitian selanjutnya.

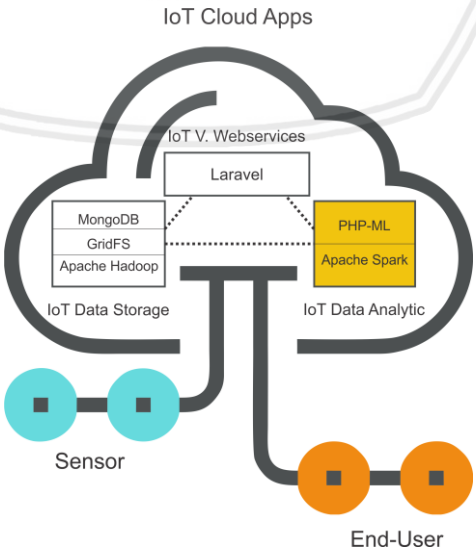
BAB 4 ANALISIS KEBUTUHAN DAN PERANCANGAN

Bagian ini membahas daftar kebutuhan sistem meliputi kebutuhan perangkat lunak, kebutuhan perangkat keras, kebutuhan fungsional dan kebutuhan non fungsional. Kebutuhan ini digambarkan dalam bentuk tabel ataupun gambaran umum sistem. Selanjutnya dijelaskan perancangan sistem meliputi perancangan lingkungan sistem, perancangan aplikasi serta perancangan pengujian.

4.1 Deskripsi Umum Sistem

Tujuan utama pada sistem yang dikembangkan adalah menyediakan sebuah sistem yang terintegrasi yang terdiri dari *data storage* yang berguna untuk menyimpan data yang dihasilkan dari perangkat IoT yang dikirimkan pada *framework IoT application.* Tujuan selanjutnya adalah menyediakan sebuah sistem *data analytic* berbasis *machine learning* yang akan mengolah data yang dihasilkan oleh perangkat IoT dan sudah disimpan dalam sistem *data storage.* Kemudian juga dikembangkan sebuah *webservice* yang menyediakan fitur panel manajemen pengguna dan juga aplikasi visualisasi data hasil analisis, dalam *webservice* ini pengguna dapat dengan mudah melakukan kustomisasi sistem dengan mengelola jumlah perangkat IoT. Fokus pada penelitian adalah pada sisi *data analytics,* akan tetapi diperlukan pengembangan komponen lainnya sebagai *supporting system*. *Framework* ini sendiri dikembangkan agar pada developer hanya perlu fokus saja di bagian perangkat IoT dan modul pengolahan datanya saja, tidak perlu menampilkan infrastruktur cloud computing yang digunakan untuk menerima, mengolah dan menampilkan datanya.

Secara keseluruhan gambaran umum dari sistem yang dikembangkan dapat dilihat pada Gambar 4.1.



Gambar 4.1 Deskripsi umum sistem

Keterangan dari Gambar 4.1. diatas adalah :

1. Sensor mengirimkan data ke *IoT cloud apps* menggunakan protokol HTTP.
2. Setelah data diterima oleh sistem maka akan data akan disimpan di dalam *IoT data storage*.
3. Data yang telah disimpan maka data dapat dianalitik menggunakan IoT *data analytic*.
4. *Webservice* dapat mengambil data yang sudah disimpan di *IoT data storage.*
5. Pengguna dapat mengambil data pada sistem, menggunakan protokol HTTP.
6. Pengguna dapat memberikan perintah ke *framework* untuk melakukan sesuatu, menggunakan protokol HTTP.

Pada penelitian ini dibangun sebuah sistem sebagai lingkungan pengujian pada sistem yang telah dikembangkan. Sistem terdiri dari dua perangkat sensor *dummy* yang mengirimkan data ke *framework* yang dibangun, kemudian setelah data masuk maka dalam *framework* akan diarahkan ke sistem *sata storage* yang akan melakukan *filtering data* beserta penyimpanan data, setelah disimpan data akan dianalitik dalam sistem *data analytic* berbasis *machine learning*, dan setelah dianalitik maka akan ditampilkan dalam sebuah *webservice*.

4.2 Kebutuhan Sistem

Kebutuhan sistem bertujuan untuk mengetahui kebutuhan yang diperlukan dalam membangun sistem dan mengimplementasikan *framework*. Kebutuhan ini dapat mempermudah proses perancangan dan implementasi. Kebutuhan dalam penelitian ini terbagi menjadi dua yakni kebutuhan fungsional dan kebutuhan non fungsional.

4.2.1 Kebutuhan Fungsional

Kebutuhan fungsional adalah kebutuhan-kebutuhan yang harus dipenuhi dan proses-proses yang dapat dilakukan oleh sistem, dalam kebutuhan fungsional ini peneliti mengklasifikasikan kebutuhan fungsional ke dalam 3 (tiga) bagian besar yaitu *data storage*, *data analytic*, dan *webservice*.

Klasifikasi ini digunakan untuk mempermudah dalam tahap implementasi dari sistem yang dikembangkan, klasifikasi *data storage* berarti kebutuhan fungsional tersebut akan diterapkan pada infrastruktur *data storage.* Begitu juga untuk seterusnya.

Untuk memperjelas penelitian ini maka kebutuhan fungsional sistem dalam penelitian ini dijelaskan pada Tabel 4.1 berikut ini.

Tabel 4.1 Kebutuhan fungsional

No	Kebutuhan Fungsional	Klasifikasi
1	Sistem dapat melakukan fungsi autentikasi login perangkat menggunakan JWT melalui sebuah *API*	*Webservice*
2	Sistem dapat melakukan fungsi menampilkan daftar topik melalui sebuah *API*	
3	Sistem dapat menerima data terstruktur dalam bentuk JSON	*Data storage*
4	Sistem dapat menerima data semi terstruktur dalam bentuk XML	
5	Sistem dapat menerima data tidak terstruktur dalam bentuk gambar JPG	
6	Sistem dapat menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API*	*Webservice*
7	Sistem dapat menampilkan data yang disimpan dalam suatu topik melalui sebuah *API*	
8	Sistem dapat melakukan registrasi pengguna baru	
9	Sistem dapat melakukan login pengguna	
10	Sistem dapat melakukan CRUD (manajemen) perangkat IoT	
11	Sistem dapat melakukan CRUD (manajemen) topik IoT	
12	Sistem dapat melakukan CRUD (manajemen) pengguna	
13	Sistem dapat menampilkan daftar data yang disimpan dalam suatu topik	
14	Sistem dapat menampilkan data yang disimpan dalam suatu topik	
15	Sistem dapat melakukan prediksi data menggunakan least square	*Data analytic*
16	Sistem dapat melakukan klasifikasi data menggunakan Naïve bayes	
17	Sistem dapat menampilkan informasi statistik dari sebuah topik	

4.2.2 Kebutuhan Non Fungsional

Kebutuhan non fungsional adalah kebutuhan yang menitik beratkan pada perilaku dan batasan fungsi pada sistem. Dalam penelitian ini kebutuhan nonfungsional terdiri dari kebutuhan sistem termasuk software dan hardware.

Kebutuhan non fungsional dalam sistem ini penulis klasifikasikan menjadi 3 (tiga) kelompok besar yaitu keamanan *(security),* ketersediaan *(availability)* dan batasan *(constraint)*, seluruh kebutuhan non fungsional tersebut dapat dilihat dalam Tabel 4.2 dibawah ini.

Tabel 4.2 Kebutuhan non fungsional

No	Kebutuhan Non Fungsional	Klasifikasi
1	Sistem *framework* menyediakan fungsi daftar dan login	Keamanan
2	Sistem *framework* menyediakan fungsi lupa password	
3	Sistem *framework* mampu membedakan role user dari admin dan pengguna biasa	
4	Sistem dapat memblokir pengguna yang login salah selama 3 kali dalam waktu 15 menit	
5	Sistem dapat memblokir JWT yang sudah tidak terpakai lagi	
6	Sistem dapat 24 jam selama 7 hari dengan presentase 80%.	Ketersediaan
7	Sistem harus mampu menerima data dalam bentuk JSON, XML dan JPG	Batasan
8	Sistem harus mampu menerima data melalui protokol HTTP	
9	Sistem harus mampu menyimpan data dengan Batasan ukuran dibawah 1 MB.	

4.2.2.1 Kebutuhan Perangkat Keras

Perangkat keras merupakan komponen utama dari sistem yang dibangun. Perangkat keras disini juga meliputi lingkungan dari sistem yang nantinya sistem akan diimplementasikan dalam perangkat keras ini.

Kebutuhan perangkat keras yang dibutuhkan untuk mengembangkan *framework* dan sistem dalam penelitian ini dijelaskan dalam Tabel 4.3 yang terdapat pada bagian dibawah ini.

Tabel 4.3 Kebutuhan perangkat keras

Perangkat	Keterangan
Server	*Server* akan digunakan sebagai lokasi peletakan dari sistem *framework* yang dibangun.

4.2.2.2 Kebutuhan Perangkat Lunak

Perangkat lunak juga merupakan komponen vital dalam sistem ini. Dikarenakan tanpa perangkat lunak perangkat keras tidak akan dapat melaksanakan fungsinya. Perangkat yang disebutkan dalam penelitian kali ini tidak hanya terbatas pada perangkat lunak yang berhubungan langsung dalam proses membangun sistem, akan tetapi juga termasuk perangkat lunak yang akan digunakan dalam proses pengujian juga.

Kebutuhan perangkat lunak yang dibutuhkan untuk mengembangkan *framework* dan sistem dalam penelitian ini dijelaskan dalam Tabel 4.4 dan Tabel 4.5 yang terdapat pada bagian dibawah ini.

Tabel 4.4 Kebutuhan perangkat lunak (bagian 1)

Perangkat	Keterangan
Sistem Operasi	Sistem operasi berbasis linux yang akan berjalan pada sisi server.
SSH	SSH digunakan sebagai media komunikasi dalam melakukan konfigurasi sistem di server.
Text Editor	Text Editor digunakan untuk melakukan proses pemrograman.
Postman	Postman digunakan untuk melakukan testing *webservice*.
Laravel	*Framework* Laravel digunakan untuk membuat sistem *webservice*.
Apache Hadoop	Apache Hadoop digunakan sebagai software distributed *file* system yang diatasnya akan dijalankan Apache Spark.
Apache Spark	Apache Spark digunakan sebagai sistem untuk melakukan analitik data.
NGINX	Nginx digunakan sebagai *reverse proxy* untuk mengakses *webserver*

Tabel 4.5 Kebutuhan perangkat lunak (bagian 2)

Apache Webserver	Apache webserver digunakan sebagai *webserver* dalam sistem.
MongoDB	MongoDB digunakan untuk menyimpan data terstruktur dari sistem seperti JSON atau XML.
GridFS	GridFS bekerja bebarengan dengan MongoDB, yang membedakan adalah GridFS akan menyimpan data yang tidak terstruktur.
Redis	Redis digunakan untuk menyimpan data sementara dari sistem seperti session dan juga untuk menyimpan antrian *(queue)* yang terdapat dalam sistem.
JMeter	JMeter digunakan sebagai perangkat lunak pengujian dari sistem yang dibangun, dalam JMeter dapat dibuat beberapa scenario pengujian fungsional maupun skalabilitas.

4.3 Perancangan Sistem

4.3.1 Perancangan Lingkungan Sistem

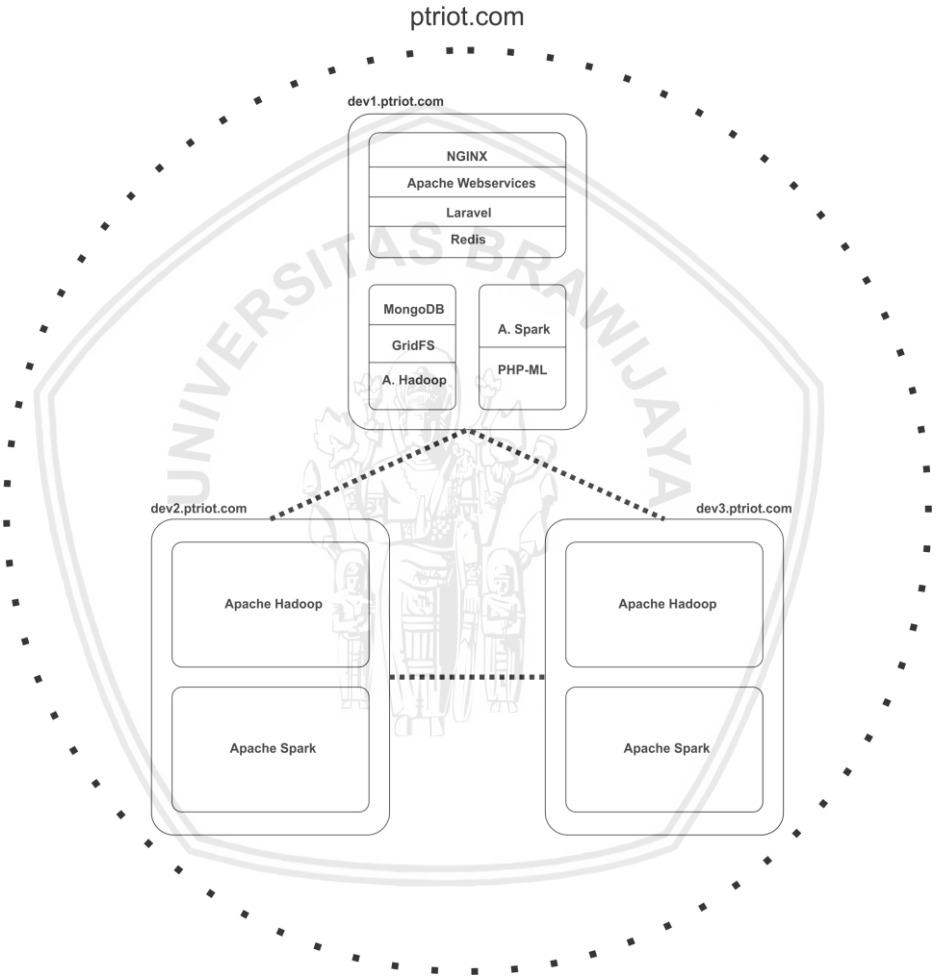
Dalam sistem ini terdapat tiga komponen besar yang saling berkesinambungan yakni perangkat *sensor emulator* sebagai publisher yang mengirimkan data ke *IoT cloud apps.* setelah data disimpan dan diolah maka data dapat diakses oleh *end-user* atau pengguna melalui berbagai perangkat yang mendukung protokol HTTP. Dalam *IoT cloud apps* terdapat beberapa bagian yang diantaranya adalah *IoT data storage, IoT data analytic,* dan *IoT visualization webservices.* Masing-masing bagian memiliki tujuan dan fungsinya sendiri-sendiri, yang setelah di integrasikan akan menghasilkan sebuah IoT cloud apps yang utuh dan lengkap.

Secara umum *IoT data storage* akan bertugas menyimpan data yang masuk kedalam sistem. Data yang dapat ditangani oleh komponen ini adalah data terstruktur, semi terstruktur dan tidak terstruktur. Untuk mendapatkan kemampuan ini akan digunakan berbagai aplikasi seperti MongoDB, GridFS dan Apache Hadoop.

IoT data analytic yang memiliki peran dalam melakukan analisis data yang tersimpan didalam *IoT cloud apps*. Data yang dapat ditangani oleh komponen ini hanyalah data terstruktur dan semi terstruktur saja. Analitik yang dapat dilakukan adalah prediksi data dan klasifikasi data. Untuk mendapatkan kemampuan ini akan digunakan modul aplikasi PHP-ML dan Apache Spark.

Komponen *IoT visualization webservices* yang dapat diibaratkan sebagai gerbang untuk mengakses keseluruhan sistem, pada penelitian ini juga akan menitikberatkan pada bagian ini. Pada komponen ini akan disediakan kemampuan personalisasi data dan personalisasi sistem yang memungkinkan pengguna untuk membuat atau mendaftarkan topik dan perangkat baru sesuai kebutuhan. Untuk mengimplementasikan bagian ini akan digunakan *framework* PHP yaitu Laravel, beserta aplikasi lainnya seperti NGiNX, Apache Webservice, dan Redis.

Untuk mempermudah memahami, lingkungan sistem akan ditunjukkan pada Gambar 4.2 dibawah ini.



Gambar 4.2 Perancangan Lingkungan sistem

4.3.2 Perancangan Infrastruktur

Pada penelitian ini terdapat terdapat bagian infrastruktur yang digunakan untuk meletakkan semua aplikasi beserta konfigurasi dari aplikasi yang dikembangkan. Infrastruktur ini terdiri dari IoT data storage yang digunakan untuk menyimpan data, IoT data analytic yang digunakan untuk melakukan analitik pada data yang disimpan, dan yang terakhir adalah IoT visualization webservice app yang digunakan untuk menampilkan data yang ada pada sistem. Pada

perancangan ini akan berisi semua konfigurasi dan instalasi untuk melalukan implementasi sistem.

4.3.2.1 Perancangan IoT Data Storage

IoT data storage adalah sebuah komponen yang berfungsi untuk melakukan manajemen penyimpanan data dari sistem *IoT cloud apps*. Sebuah *IoT data storage* harus memiliki kemampuan untuk melakukan manajemen penyimpanan data. Manajemen penyimpanan data yang dimaksud meliputi pembuatan data, penyisipan data, pengubahan data, penghapusan data. Selain itu *IoT data storage* juga harus memiliki kemampuan untuk melakukan manajemen data terstruktur, semi terstruktur dan juga data yang tidak terstruktur.

IoT data storage akan dibangun menggunakan beberapa aplikasi. Aplikasi yang digunakan antaralain adalah MongoDB, GridFS dan juga Apache Hadoop. MongoDB dan GridFS akan digunakan sebagai aplikasi penyimpanan data primer yang akan menyimpan data yang bisa diakses oleh *webservice* dan *data analytic*. Sedangkan pada Apache Hadoop akan digunakan sebagai aplikasi penyimpanan data sekunder yang akan menyimpan data pada saat dan sesudah proses analitik menggunakan Apache Spark.

Komponen *IoT data storage* akan dilakukan sebuah *classification* sederhana menggunakan pendekatan *semantic* untuk menentukan lokasi penyimpanan *file* berdasarkan topik data dan juga berdasarkan struktur dari data tersebut, jika data adalah terstruktur atau semi terstruktur dan berukuran dibawah 100KB maka akan disimpan di MongoDB, sedangkan jika tidak terstruktur atau diatas 100KB akan disimpan di dalam GridFS. Untuk pendekatan *semantic* akan lebih dijelaskan secara lebih jelas pada poin menyimpan data dari sensor yang dapat dilihat pada poin 4.3.3.3.

Untuk melakukan komunikasi antar komponen yang ada dalam *IoT cloud apps* maka *IoT data storage* akan mengimplementasikan sebuah *API* yang sudah ada sebelumnya seperti penggunaan pymongo dan pydoop yang berbasis bahasa pemrograman Python.

(a) Perancangan Data Storage Berbasis MongoDB dan GridFS

MongoDB sendiri akan digunakan sebagai sistem storage utama yang digunakan untuk menyimpan data dan juga sebagai penghubung utama ke aplikasi Visualisasi data, MongoDB digunakan karena kinerja *realtime* aplikasi lebih baik daripada Apache Hadoop dan juga lebih banyak sup*port* ke aplikasi *webservice*, sehingga memudahkan dalam proses implementasi.

GridFS juga akan digunakan bersamaan dengan MongoDB, jika pada MongoDB akan menyimpan data terstruktur maka dalam GridFS akan digunakan untuk menyimpan data tidak terstruktur.

(b) Perancangan Data Storage Berbasis Apache Hadoop

Untuk sistem storage sendiri akan menggunakan Apache Hadoop, aplikasi Apache Hadoop akan digunakan untuk menyimpan data analisis yang akan diproses dan sudah diproses oleh Apache Spark.

Peneliti merancang penggunaan Apache Hadoop untuk data analitik dari Apache Spark, dikarenakan karakteristik Apache Hadoop sendiri sangat cocok untuk secara terdistribusi, sehingga akan memudahkan dalam memproses secara *parallel* menggunakan Apache Spark.

4.3.2.2 Perancangan IoT Data Analytic

IoT data analytic adalah komponen yang berfungsi untuk melakukan pemrosesan data. Pemrosesan yang dilakukan adalah proses analitik pada data yang masuk ataupun tersimpan dalam sistem *framework* yang dibangun.

IoT data analytic sendiri akan dibangun menggunakan program Apache Spark yang dibantu oleh Apache Hadoop sebagai dasar dari aplikasi. Selain itu juga akan digunakan *library* PHP-ML yang akan difungsikan untuk menganalisis data ringan atau dibawah 100.000 *entry* data.

Data hasil analitik yang dihasilkan dalam proses *IoT data analytic* akan ditampilkan melalui *IoT visualization webservice* dan juga data tersebut akan disimpan dalam *IoT data storage* khususnya di MongoDB. Hal ini dikarenakan MongoDB memungkinkan penyimpanan data yang beragam yang sangat sesuai dengan kebutuhan dalam sistem ini.

Dalam sistem *IoT data analytic* ini dibangun beberapa modul analitik yang dapat disesuaikan sesuai data yang masuk. Modul analitik yang dibangun sendiri akan menggunakan beberapa bahasa pemrograman yang berbeda seperti Python dan PHP. Cara berkomunikasi antara modul dengan *framework* sendiri akan menggunakan *API* yang dibangun sendiri dan juga menggunakan *API call* yang ada dalam sistem operasi.

(a) Perancangan Data Analytic Berbasis Apache Spark

Apache Spark akan peneliti gunakan sebagai aplikasi analitik data utama. Data yang dianalisis berasal dari MongoDB yang merupakan data dari sensor. Apache Spark sendiri dapat digunakan untuk melakukan analisis data dari yang berukuran kecil hingga berukuran sangat besar atau *big data*, tergantung pada lingkungan yang digunakan saat mengimplementasikan Apache Spark.

(b) Perancangan Data Analytic Berbasis PHP-ML

PHP-ML atau PHP-Machine Learning adalah sebuah library analitik yang dikembangkan untuk bahasa pemrograman PHP. Library ini bersifat *lightweight* atau ringan. Sehingga cocok untuk melakukan analisis data yang berukuran kecil atau dibawah 100.000 *entry*.

4.3.2.3 Perancangan IoT Visualization Webservice

IoT visualization webservice adalah sebuah komponen yang paling sentral dalam *framework* ini. Hal ini dikarenakan bagian ini adalah bagian yang langsung

berinteraksi dengan pengguna dan juga sistem lain. *IoT visualization webservice* akan memiliki beberapa sub komponen yang terdiri dari bagian yang menampilkan data hasil analitik secara langsung ke pengguna dalam bentuk table dan grafik. Komponen kumpulan *API* yang memungkinkan pengguna untuk mengambil data hasil analitik pada sistem untuk diakses sistem lainnya. Pada *IoT visualization webservice* akan dibangun menggunakan *framework* Laravel yang dipadukan dengan *framework* javascript untuk fungsi menampilkan data hasil analitik.

IoT visualization webservice sendiri pada dasarnya akan memiliki kemampuan layaknya aplikasi *telemetry* yang mampu menampilkan data, selain itu *API* yang dibangun juga memiliki kemampuan yang serupa dengan penelitian sebelumnya pada Open IoT (Kim Jaehoo, et al., 2014).

(a) Perancangan Webservice Berbasis Apache Webserver

Apache Webserver akan digunakan sebagai *webserver* dari sistem ini. Fungsinya adalah untuk menangani *request* dari client dan kemudian menjalankan kode program yang tersedia di sistem.

(b) Perancangan Webservice Berbasis NGINX

Nginx akan digunakan sebagai *reverse proxy* dari sistem yang digunakan. Kelebihan menggunakan *reverse proxy* diantaranya dapat memisah aliran data yang masuk dari internet antara terhadap *file* dan program.

(c) Perancangan Webservice Berbasis Redis

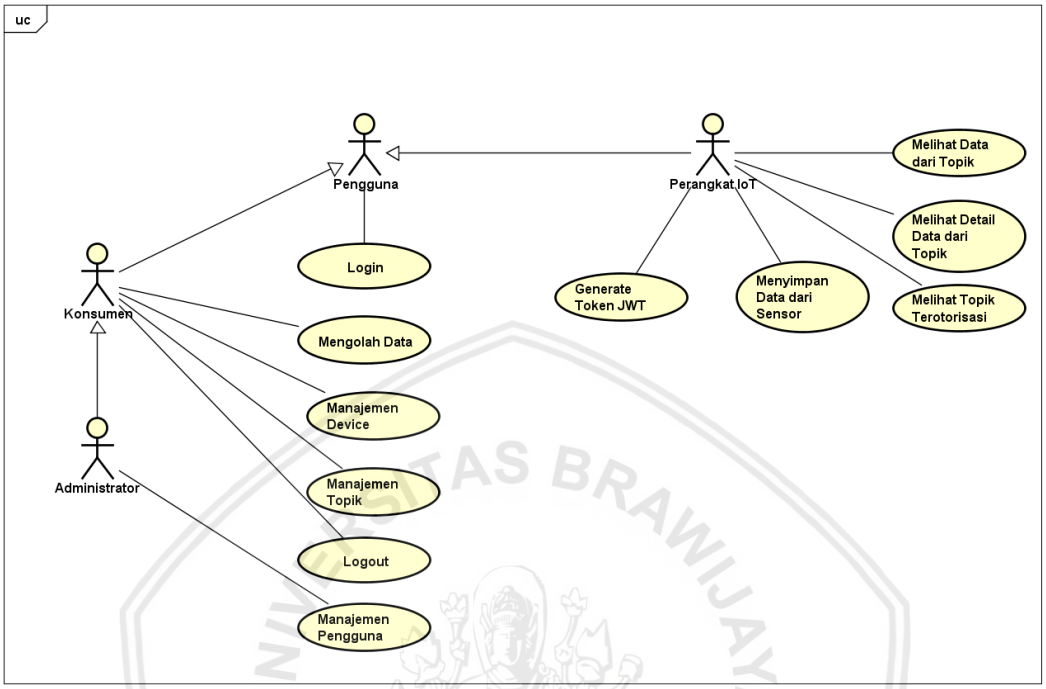
Redis akan digunakan sebagai lokasi penyimpanan sementara dari sistem. Data yang akan disimpan dalam redis antara lain sessions dan queue dari proses yang berjalan di *webservice*.

(d) Perancangan Webservice Berbasis Laravel

Laravel akan digunakan sebagai *framework* dalam pengembangan sistem *webservice*. *Framework* Laravel menawarkan penggunaan kembali terhadap beberapa komponen sistem sehingga dapat mempercepat proses pengembangan dari sistem itu sendiri.

4.3.3 Perancangan Aplikasi

Aplikasi yang dibangun dalam sistem ini akan mempunyai beberapa kemampuan yang digambarkan dalam Use case 4.3 dibawah ini.



Gambar 4.3 *Use case* sistem *webservice*

Sesuai pada *use case* diagram dari sistem yang terdapat pada Gambar 4.3 dalam aplikasi ini terdiri dari bagian autentikasi. Selain itu akan ada juga bagian untuk melakukan manajemen topik dan perangkat IoT beserta metode autentikasi yang digunakan agar tidak terjadi kesalahan pengelolaan dalam sistem.

Metode autentikasi yang digunakan dalam hal ini adalah menggunakan model JWT (JSON Web Token) yang akan diselipkan dalam *header* dari setiap pengaksesan *API* dari sistem. *Token* yang digunakan sendiri akan dihasilkan secara otomatis saat perangkat IoT melakukan *login* sistem.

Bagian-bagian lain seperti *IoT data storage* dan *IoT data analytic* pada dasarnya akan dimanejemen atau dikelola melalui komponen ini. Oleh karena itu untuk melakukannya dalam komponen panel manajemen pengguna juga diharuskan untuk mampu melakukan komunikasi kepada komponen lainnya. Komunikasi dapat dilakukan menggunakan *API* yang sudah ada dalam *library* setiap komponen ataupun menggunakan *API* khusus yang dibuat oleh peneliti sesuai kebutuhan sistem.

4.3.3.1 *Generate* JWT

Pada penelitian kali ini metode *generate* JWT yang digunakan adalah berdasar dari penelitian sebelumnya yang dilakukan oleh Tuti Wardani. Pada penelitian yang dilakukan dia menggunakan *email* dan *password* untuk menghasilkan JWT, akan tetapi pada penelitian kali ini akan terdapat sedikit

perbedaan yaitu menggunakan device *token* dan *secret key* untuk menghasilkan sebuah JWT.

Pada penelitian kali ini akan terdapat sedikit perbedaan pada implementasinya dikarenakan penelitian sebelumnya berjalan pada sistem yang berbasis Python, sedangkan pada penelitian kali ini peneliti akan menggunakan sistem berbasis PHP.

4.3.3.2 Login dan Logout

Pada penelitian kali ini metode autentikasi yang digunakan adalah berdasar dari penelitian sebelumnya yang dilakukan oleh Tuti Wardani yang menggunakan *email* dan *password* beserta penggunaan JWT pada *header* setiap *request* yang dilakukan oleh perangkat IoT.

JWT akan dihasilkan setiap perangkat melakukan login ke sistem. Perangkat akan melakukan *login* menggunakan *method POST* berstandar protokol Rest dengan menyertakan *token* dan *secret key* dari setiap perangkat. JWT sendiri akan valid selamanya, tanpa ada batasan waktu, kecuali jika dilakukan login ulang atau logout yang akan secara otomatis membuat JWT sebelumnya menjadi tidak valid.

Pada penelitian kali ini akan terdapat sedikit perbedaan pada implementasinya dikarenakan penelitian sebelumnya berjalan pada sistem yang berbasis Python, sedangkan pada penelitian kali ini peneliti akan menggunakan sistem berbasis PHP.

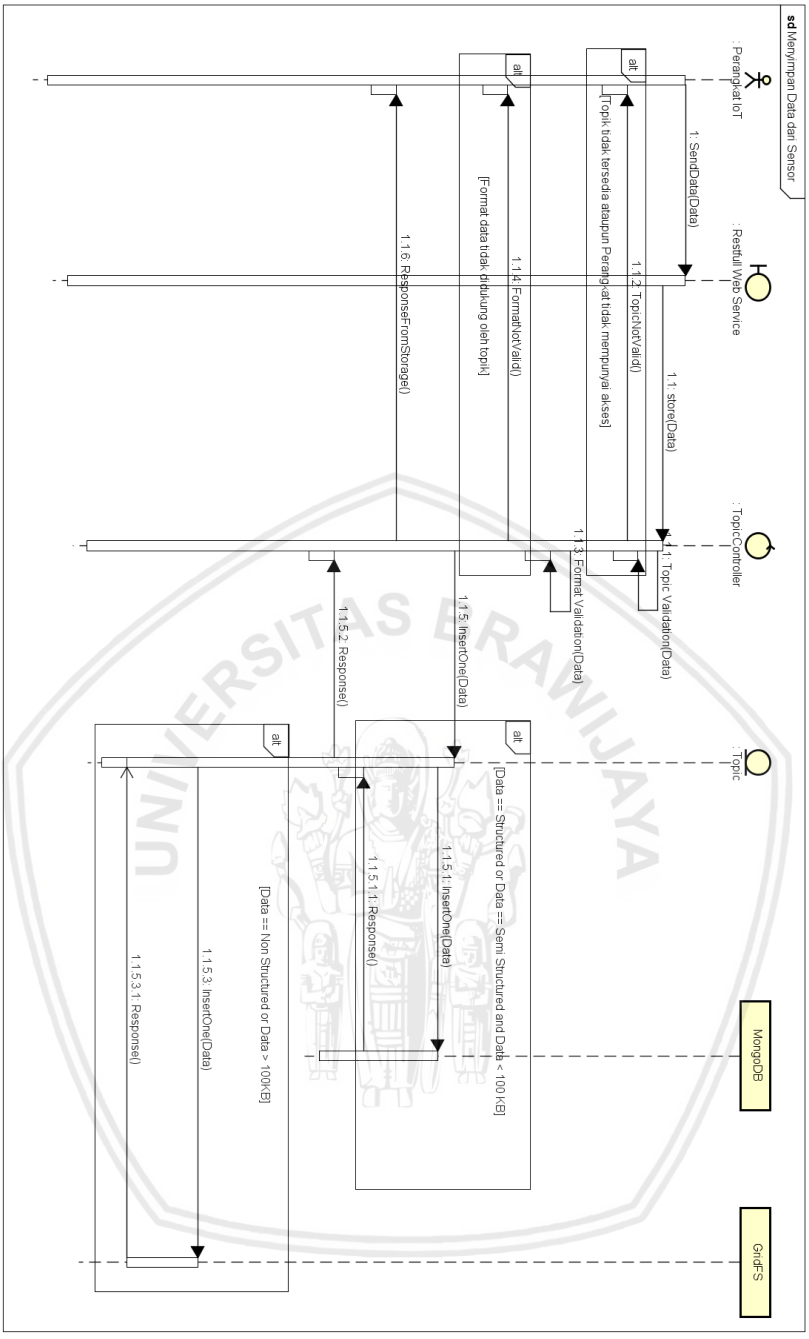
4.3.3.3 Menyimpan Data dari Sensor

Pada penelitian kali ini sistem akan menyediakan sebuah metode atau fitur untuk melakukan penyimpanan data dari sensor. Fitur ini akan diimplementasikan dalam bentuk sebuah *API* yang dapat diakses menggunakan protokol REST. Pada fitur kali ini akan bersumber daripada penelitian sebelumnya yang juga akan mengimplementasikan JWT untuk proses autentikasi penyimpanan data dan juga *semantic filtering*.

Saat menyimpan data dari sensor, sistem akan melakukan *filtering data* menggunakan beberapa pendekatan dan parameter untuk menentukan lokasi penyimpanan data yang paling tepat untuk data tersebut dimulai dengan mendeteksi topik yang masuk. Kemudian mencocokan dengan kesesuaian data, selanjutnya melakukan pendeteksian struktur data dan yang terakhir mendeteksi ukuran dan *form*at dari data. Jika data yang masuk adalah dalam bentuk JSON, XML atau dalam bentuk terstruktur maupun semi terstruktur dan berukuran dibawah 100 KB maka data akan disimpan dalam MongoDB, selain itu akan disimpan dalam GridFS. Untuk lebih jelasnya dapat dipahami melalui u*se case scenario* yang ada pada Tabel 4.6 dan juga *sequence diagram* pada Gambar 4.4 dibawah.

Tabel 4.6 *Use case scenario* menyimpan data dari Sensor

Use case	Menyimpan data dari sensor
Objective	Dapat menyimpan data dari sensor
Actors	Perangkat IoT
Pre - Condition	Autentikasi dan mempunyai JWT
Main Flow	1. Perangkat IoT mengirim data sensor dalam bentuk yang beragam 2. *Webserver* akan melakukan *filtering data* menggunakan pendekatan semantic 3. Jika data dalam bentuk JSON, XML atau dalam bentuk terstruktur maupun semi terstruktur dan berukuran dibawah 100 KB maka data akan disimpan dalam MongoDB 4. Jika data dalam bentuk selain itu maka akan disimpan dalam GridFS 5. *Webserver* akan mengembalikan respon berisi p*anjang* data beserta *timestamp*
Alternatif Flow	1. Jika data gagal disimpan maka *webserver* akan mengembalikan respon sesuai pesan kegagalan.
Post-Condition	Data sensor dari perangkat IoT dapat disimpan sesuai topik dan struktur data



Gambar 4.4 *Sequence diagram* dari menyimpan data dari sensor

4.3.3.4 Menampilkan Topik Terautorisasi

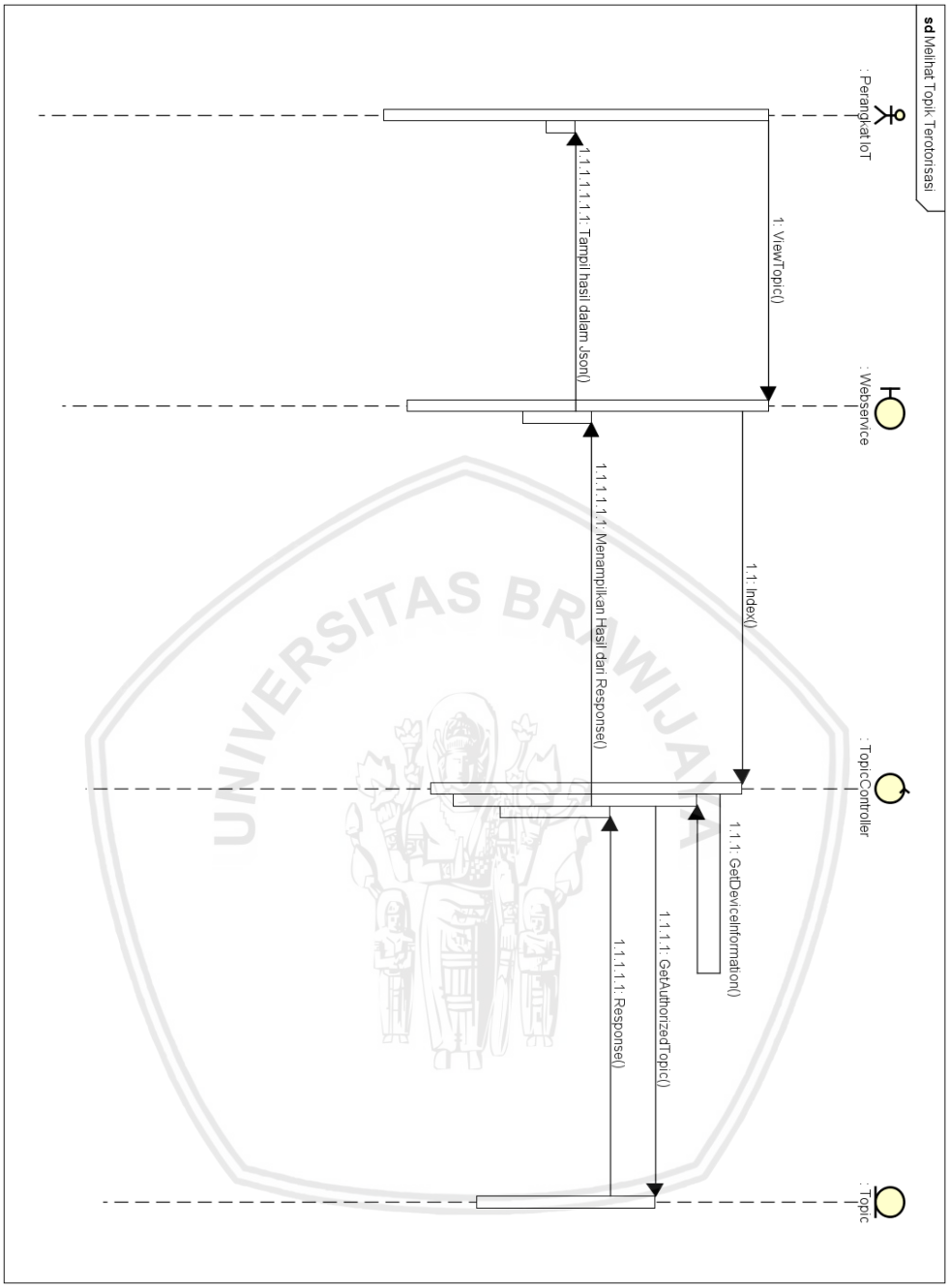
Menampilkan topik terautorisasi merupakan sebuah fitur yang disediakan oleh sistem yang peneliti kembangkan. Dengan fitur ini perangkat IoT dapat mengetahui topik apa saja yang diperbolehkan untuk diisi oleh perangkat tersebut. Data yang akan ditampilkan dari fitur ini adalah sebuah daftar *(list)* yang berisi daftar topik beserta informasi singkat dari topik tersebut.

Dalam proses mengambil informasi topik yang terautorisasi, sistem juga akan melakukan autentikasi terhadap JWT yang disertakan oleh perangkat.

Dengan JWT yang dihasilkan oleh sistem dan digunakan oleh perangkat akan memberikan sebuah informasi mengenai perangkat pemilik JWT tersebut. Sehingga sistem dapat mengenali perangkat hanya menggunakan JWT yang dikirimkan. Untuk proses menampilkan topik terautorisasi sendiri diketahui lebih lanjut dalam *use case scenario* dalam Tabel 4.7 dan *sequence diagram* pada Gambar 4.5 dibawah ini.

Tabel 4.7 *Use case scenario* menampilkan topik terautorisasi

Use case	Menampilkan topik teruatorisasi
Objective	Dapat menampilkan daftar topik yang terautorisasi dengan perangkat
Actors	Perangkat IoT
Pre - Condition	Autentikasi dan mempunyai JWT
Main Flow	1. Mengakses halaman daftar topik 2. Halaman daftar topik akan menampilkan daftar topik beserta informasi dari topik.
Alternatif Flow	-
Post-Condition	Informasi daftar topik akan ditampilkan



Gambar 4.5 *Sequence diagram* menampilkan topik terautorisasi

4.3.3.5 Menampilkan Data dari Topik

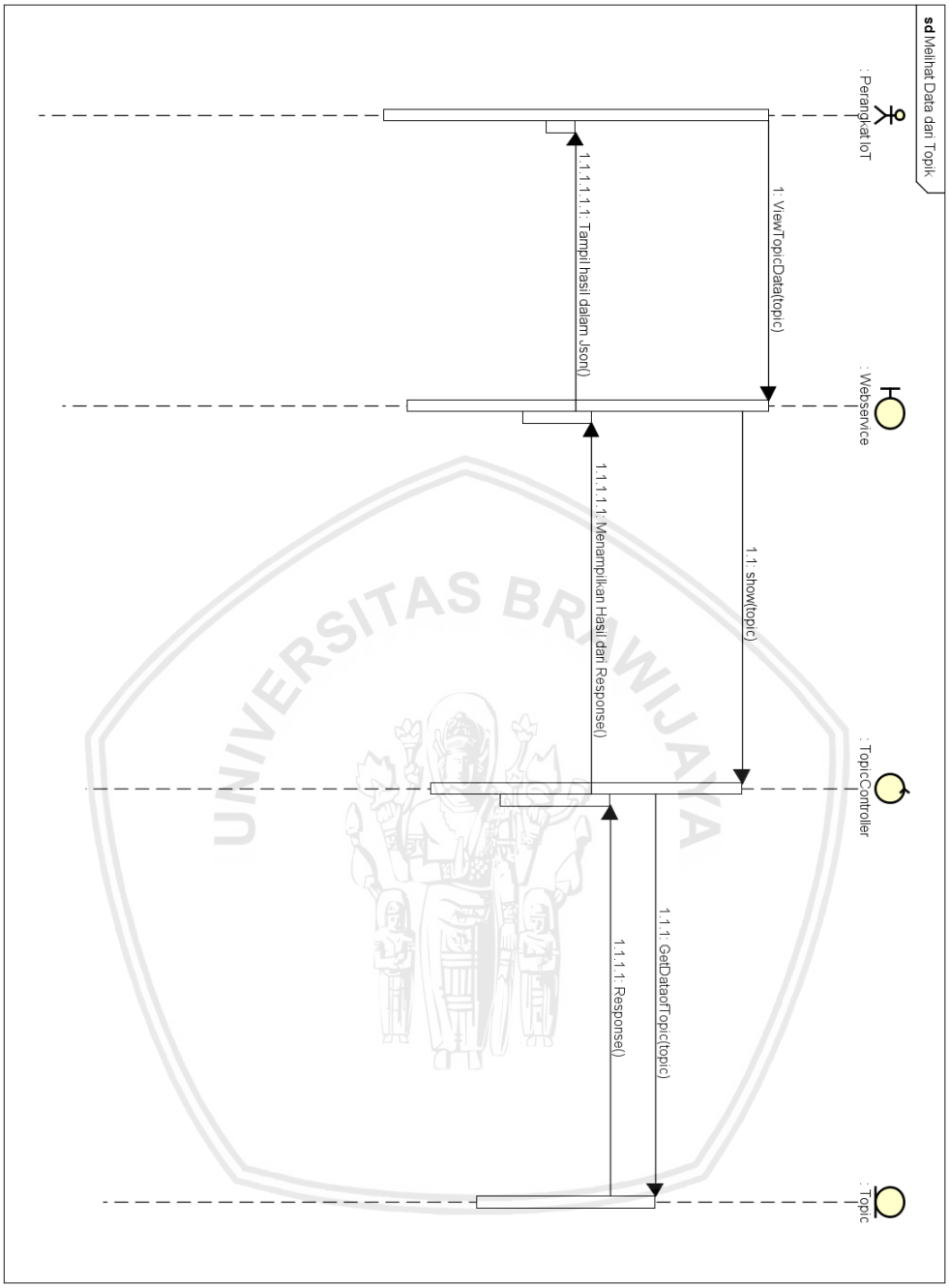
Menampilkan data dari topik merupakan sebuah fitur yang disediakan oleh sistem yang dikembangkan oleh peneliti. Dengan fitur ini perangkat IoT dapat mengetahui informasi dari data yang terdapat dalam topik ini. Data yang akan ditampilkan dari fitur ini adalah sebuah daftar *(list)* yang berisi kumpulan data yang sudah ada dan tersimpan dalam sistem terhadap topik tersebut.

Dalam proses mengambil informasi data dari topik, sistem juga akan melakukan autentikasi terhadap JWT yang disertakan oleh perangkat. setelah JWT

diketahui valid maka sistem akan mulai melakukan pengambilan data yang ada dari topik tersebut proses pengambilan data akan menyesuaikan kriteria dari yang dibutuhkan oleh perangkat IoT yang disertakan dalam melakukan *request*. Untuk proses menampilkan data dari topik sendiri dapat diketahui lebih lanjut dalam *use case scenario* dalam Tabel 4.8 dan *sequence diagram* di Gambar 4.5 dibawah ini.

Tabel 4.8 *Use case scenario* menampilkan data dari topik

Use case	Menampilkan data dari topik
Objective	Menampilkan data dari Topik
Actors	Perangkat IoT
Pre - Condition	Autentikasi dan mempunyai JWT
Main Flow	1. Mengakses halaman data topik 2. Halaman data topik akan mengembalikan sejumlah data topik berdasarkan kriteria yang dimasukkan oleh perangkat IoT
Alternatif Flow	-
Post-Condition	Informasi data dari topik akan ditampilkan



Gambar 4.6 *Sequence diagram* menampilkan data dari topik

4.3.3.6 Menampilkan Detail Data dari Topik

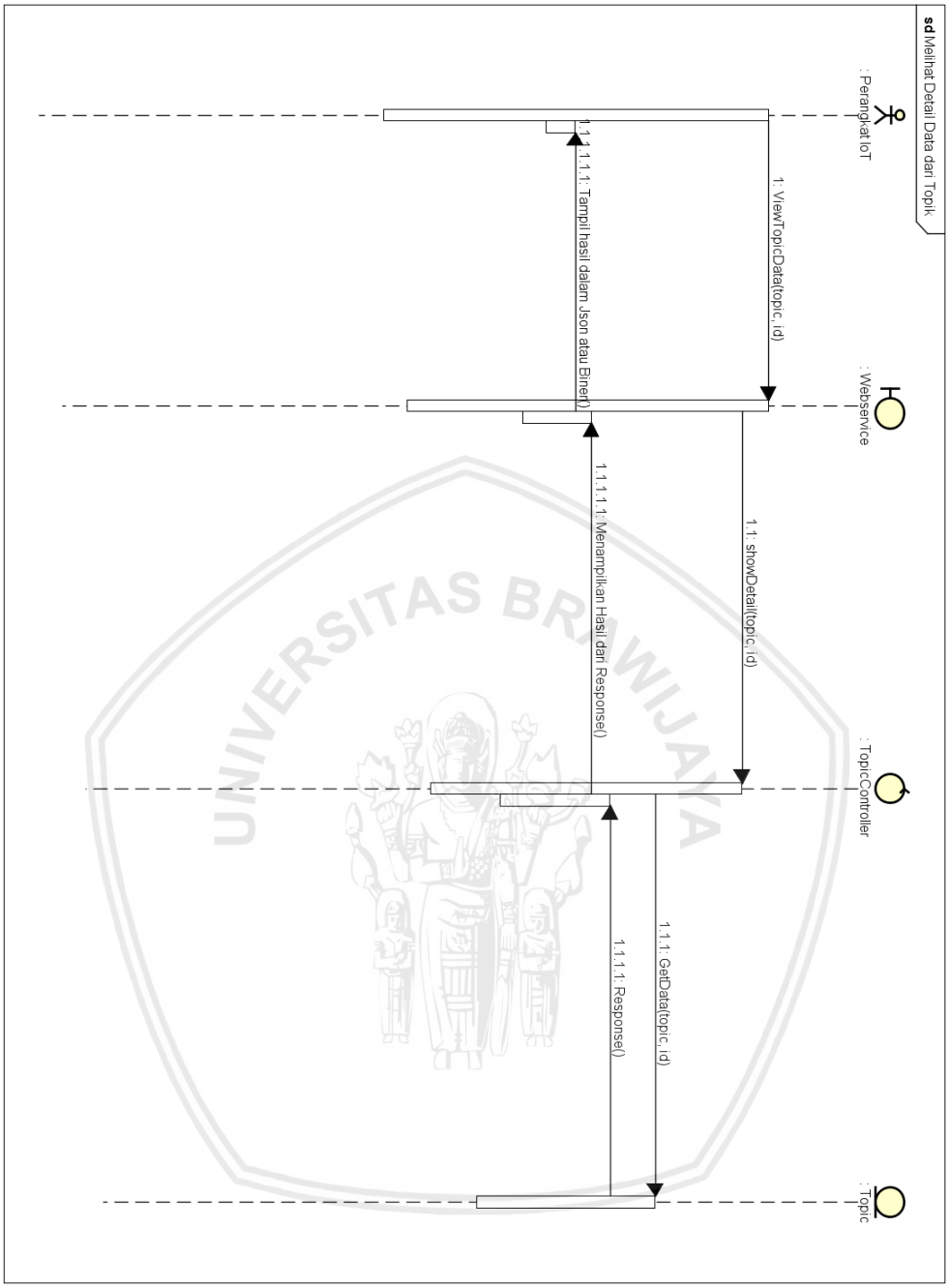
Menampilkan detail data dari topik merupakan sebuah fitur yang disediakan oleh sistem yang peneliti kembangkan. Dengan fitur ini perangkat IoT dapat mengetahui informasi detail dari sebuah data yang terdapat dalam topik ini. Data yang akan ditampilkan dari fitur ini adalah sebuah informasi lengkap dari sebuah data yang dipilih oleh perangkat IoT.

Dalam proses mengambil informasi data dari topik, sistem juga akan melakukan autentikasi terhadap JWT yang disertakan oleh perangkat. setelah JWT

diketahui valid maka sistem akan mulai melakukan pengambilan data yang ada, dalam mengembalikan data bentuk keluaran, sistem akan menyesuaikan dengan data yang ada. Jika dalam bentuk JSON maka juga akan dikembalikan dalam bentuk JSON, jika dalam bentuk binary maka juga akan dikembalikan dalam bentuk binary. Untuk proses menampilkan data dari topik sendiri dapat diketahui lebih lanjut dalam *use case scenario* dalam Tabel 4.9 dan *sequence diagram* pada Gambar 4.7 dibawah ini.

Tabel 4.9 *Use case scenario* menampilkan detail data dari topik

Use case	Menampilkan detail data dari topik
Objective	Dapat menampilkan detail dari sebuah data yang dimiliki oleh topik
Actors	Perangkat IoT
Pre - Condition	Autentikasi dan mempunyai JWT
Main Flow	1. Mengakses Halaman detail data topik 2. Halaman detail data topik akan mengembalikan sebuah detail data yang ada pada topik.
Alternatif Flow	-
Post-Condition	Informasi detail dari sebuah data akan ditampilkan



Gambar 4.7 *Sequence diagram* menampilkan detail data dari topik

4.3.3.7 Manajemen Perangkat IoT

Pada penelitian ini akan dikembangkan sebuah fitur manajemen perangkat IoT, fitur ini akan melakukan manajemen perangkat IoT mulai dari pendaftaran perangkat IoT, pengubahan data perangkat IoT, menampilkan data dari perangkat IoT dan juga melakukan penghapusan perangkat IoT.

Saat melakukan pendaftaran perangkat IoT akan dilakukan pengisian data dari perangkat IoT yang terdiri dari nama perangkat, deskripsi perangkat dan juga lokasi dari perangkat. setelah diisi maka data akan diverifikasi oleh sistem apakah

data dapat diterima oleh sistem, sistem verifikasi akan mengecek apakah penamaan perangkat sudah benar dan unik untuk setiap pengguna, dan juga jumlah huruf dari masukkan pengguna tidak kurang daripada 5 karakter dan lebih dari 50 karakter. Kemudian pada saat melakukan pengubahan data perangkat IoT juga akan dilakukan proses pengisian data yang mirip saat melakukan pendaftaran perangkat, dalam proses ini juga tetap ada proses verifikasi dari sistem terhadap masukkan dari pengguna. Saat melakukan penghapusan data perangkat IoT akan dicek juga apakah perangkat merupakan milik dari pengguna, dan pengguna memiliki akses terhadap perangkat tersebut. Jika memiliki maka sistem dapat menghapus perangkat. Dan yang terakhir saat menampilkan data dari perangkat juga akan dicek apakah perangkat merupakan milik dari pengguna, dan pengguna memiliki akses terhadap perangkat tersebut, jika memiliki maka sistem dapat menampilkan data dari perangkat.

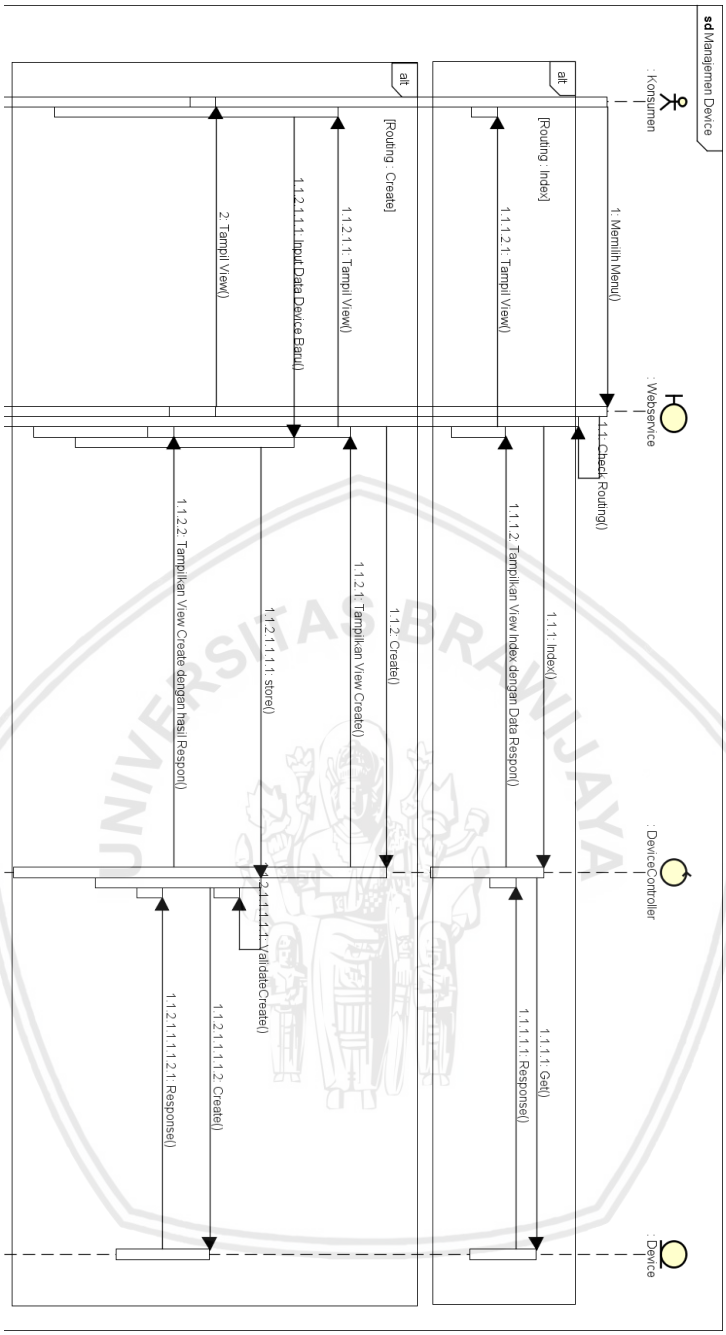
Untuk lebih jelasnya dapat dipahami melalui *use case scenario* yang ada pada Tabel 4.10 dan Tabel 4.11 dan juga *sequence diagram* pada Gambar 4.8 dan Gambar 4.9 dibawah.

Tabel 4.10 *Use case scenario* manajemen perangkat IoT (bagian 1)

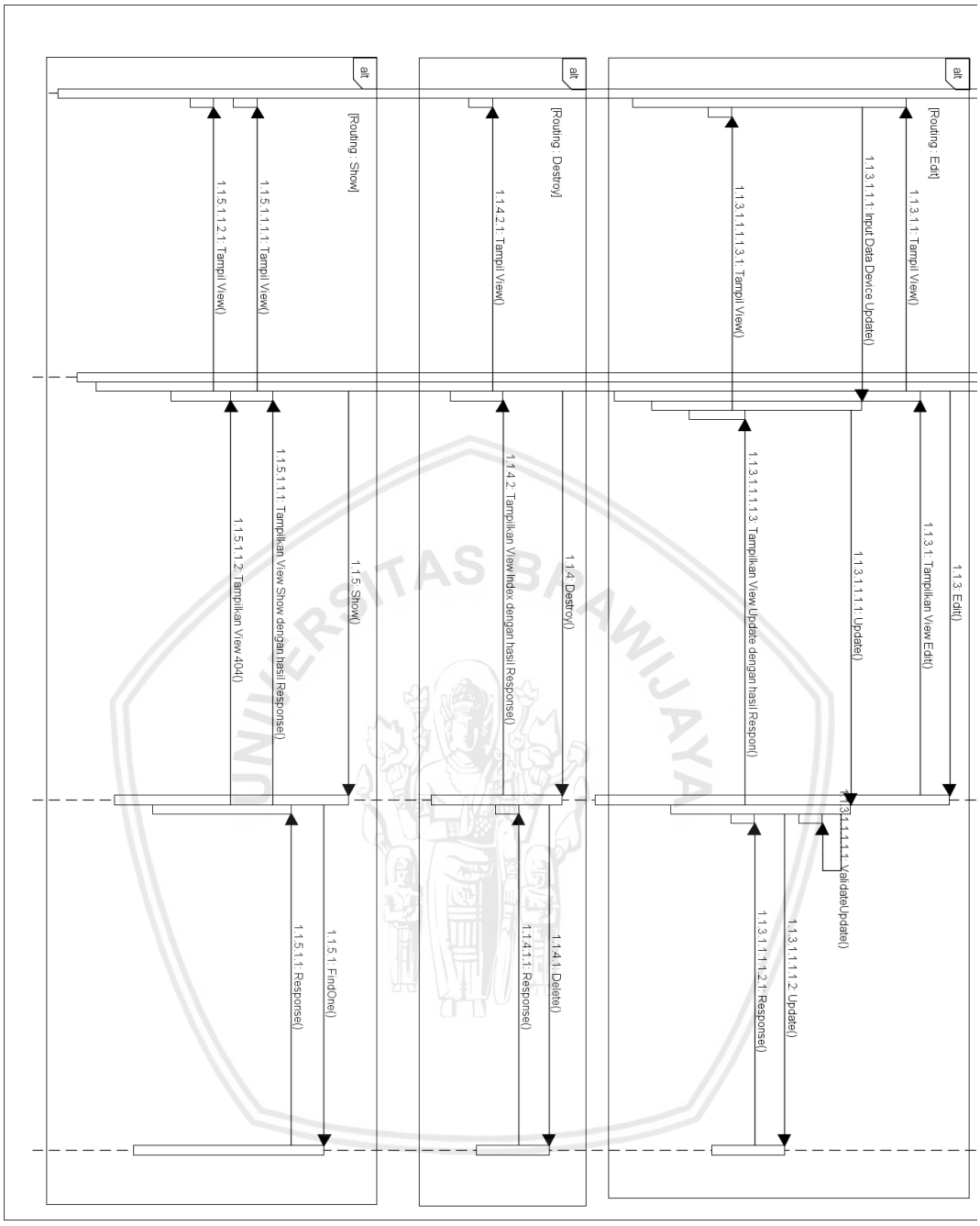
Use case	Manajemen perangkat IoT
Objective	Dapat melakukan manajemen perangkat IoT
Actors	Konsumen
Pre - Condition	Login web ptriot
Main Flow	1. Membuka Halaman manajemen perangkat IoT 2. Halaman manajemen perangkat IoT akan menampilkan daftar perangkat yang sudah pernah didaftarkan oleh konsumen beserta pilihan menu untuk lihat, ubah dan hapus perangkat beserta pilihan untuk menambahkan perangkat IoT
Alternatif Flow	1. Konsumen memilih menu menambahkan perangkat IoT 1.1. halaman tambah perangkat IoT akan muncul 1.2. Konsumen akan mengisi *form* yang ada pada halaman perangkat IoT 1.3. Konsumen melakukan *submit* 1.4. Jika data berhasil disimpan maka akan muncul notifikasi berhasil, begitu juga jika data gagal disimpan 2. Konsumen memilih menu menampilkan perangkat IoT

Tabel 4.11 Use case scenario manajemen perangkat IoT (bagian 2)

Alternatif Flow	2.1. Jika perangkat ditemukan maka informasi perangkat akan ditampilkan secara penuh 2.2. Jika perangkat tidak ditemukan maka akan ditampilkan halaman kesalahan 3. Konsumen memilih menu ubah perangkat IoT 3.1. Jika perangkat ditemukan maka Halaman ubah perangkat akan ditampilkan 3.2. Jika ditemukan maka akan diminta mengisi *form* perubahan data 3.3. Jika sudah diisi maka konsumen akan melakukan *submit* data 3.4. Jika data berhasil disimpan maka akan muncul notifikasi berhasil, begitu juga jika data gagal disimpan 3.5. Jika perangkat tidak ditemukan maka akan ditampilkan halaman kesalahan 4. Konsumen memilih menu hapus perangkat IoT 4.1. Jika perangkat ditemukan maka data perangkat akan dihapus Jika perangkat tidak ditemukan maka akan ditampilkan halaman kesalahan.
Post-Condition	Informasi daftar perangkat akan ditampilkan



Gambar 4.8 *Sequence diagram* manajemen perangkat IoT (bagian 1)



Gambar 4.9 *Sequence diagram* manajemen perangkat IoT (bagian 2)

4.3.3.8 Manajemen Topik IoT

Pada penelitian ini akan dikembangkan juga sebuah fitur manajemen topik IoT. Fitur ini akan melakukan manajemen topik IoT mulai dari pendaftaran topik IoT, pengubahan data topik IoT, menampilkan data dari topik IoT dan juga melakukan penghapusan topik IoT.

Saat melakukan pendaftaran topik IoT akan dilakukan pengisian data dari topik IoT yang terdiri dari nama topik, deskripsi topik dan juga perangkat yang diperbolehkan mengkases dari topik. setelah diisi maka data akan diverifikasi oleh

sistem apakah data dapat diterima oleh sistem. Sistem verifikasi akan mengecek apakah penamaan topik sudah benar dan unik untuk setiap topik, dan juga jumlah huruf dari masukkan pengguna tidak kurang daripada 5 karakter dan lebih dari 50 karakter. Kemudian pada saat melakukan pengubahan data topik IoT juga akan dilakukan pengisian data yang mirip dengan saat melakukan pendaftaran topik, dalam proses ini juga tetap ada tahap verifikasi dari sistem terhadap masukkan dari pengguna. Saat melakukan penghapusan data topik IoT akan dicek juga apakah topik merupakan milik dari pengguna, dan pengguna memiliki akses terhadap topik tersebut, jika memiliki maka sistem dapat menghapus topik. Dan yang terakhir saat menampilkan data dari topik juga akan dicek apakah topik merupakan milik dari pengguna, dan pengguna memiliki akses terhadap topik tersebut. Jika pengguna memiliki maka sistem dapat menampilkan data dari topik.

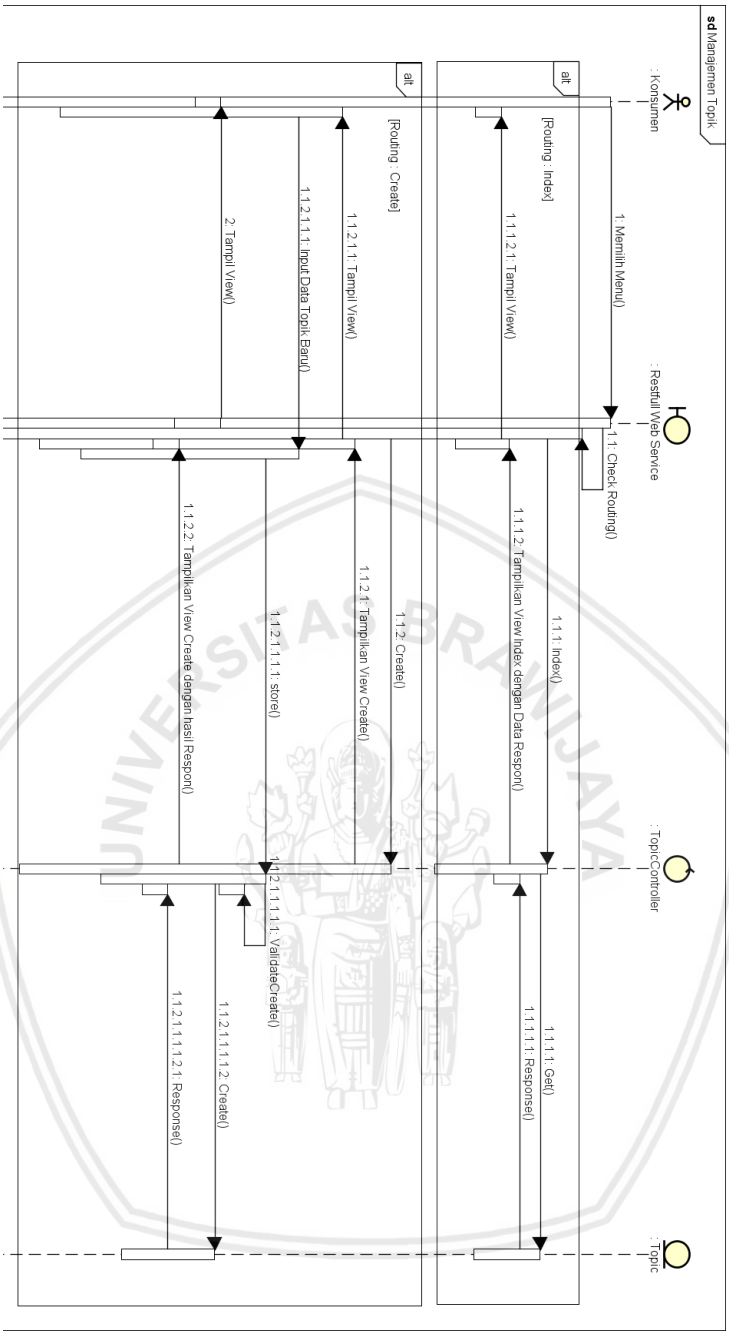
Untuk penjelasan lebih rinci dan sistematis dapat dipahami melalui *use case scenario* pada Tabel 4.12 dan Tabel 4.13 dan *sequence diagram* pada Gambar 4.10, 4.11 dan 4.12 dibawah ini.

Tabel 4.12 *Use case scenario* manajemen topik IoT (bagian 1)

Use case	Manajemen topik IoT
Objective	Dapat melakukan manajemen topik IoT
Actors	Konsumen
Pre - Condition	Login
Main Flow	1. Membuka Halaman manajemen Topik IoT 2. Halaman manajemen topik IoT akan menampilkan daftar Topik yang sudah pernah didaftarkan oleh konsumen beserta pilihan menu untuk lihat, ubah dan hapus topik beserta pilihan untuk menambahkan topik IoT
Alternatif Flow	1. Konsumen memilih menu menambahkan Topik IoT 1.1. Halaman tambah topik IoT akan muncul 1.2. Konsumen akan mengisi *form* yang ada pada Halaman topik IoT 1.3. Konsumen melakukan *submit* 1.4. Jika data berhasil disimpan maka akan muncul notifikasi berhasil, begitu juga jika data gagal disimpan 2. Konsumen memilih menu menampilkan topik IoT 2.1. Jika topik ditemukan maka informasi topik akan ditampilkan secara penuh

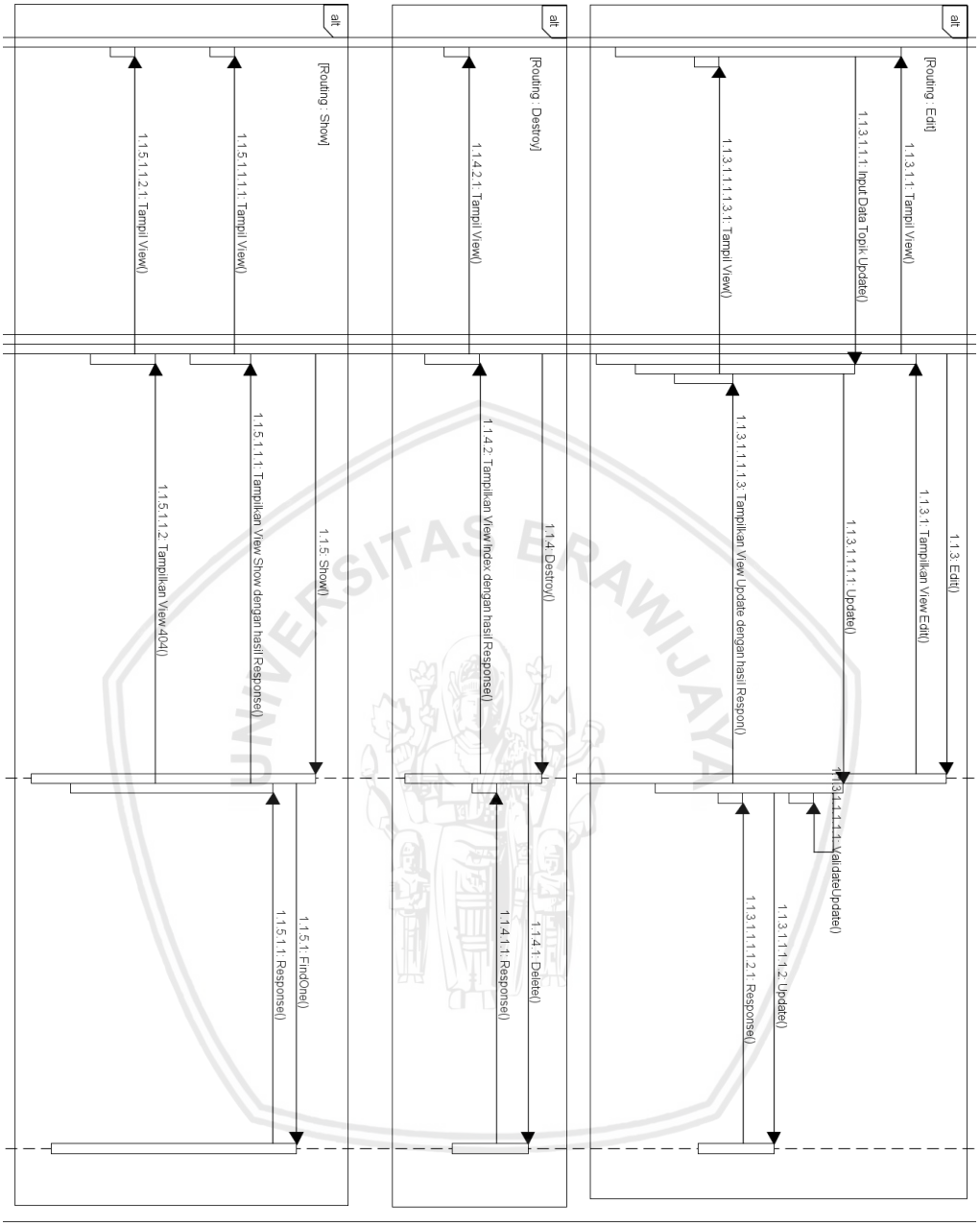
Tabel 4.13 *Use case scenario* manajemen topik IoT (bagian 2)

Alternatif Flow	2.2. Jika topik tidak ditemukan maka akan ditampilkan halaman kesalahan 3. Konsumen memilih menu ubah topik IoT 3.1. Jika Topik ditemukan maka halaman ubah Topik akan ditampilkan 3.2. Jika ditemukan maka akan diminta mengisi *form* perubahan data 3.3. Jika sudah diisi maka konsumen akan melakukan *submit* data 3.4. Jika data berhasil disimpan maka akan muncul notifikasi berhasil, begitu juga jika data gagal disimpan 3.5. Jika topik tidak ditemukan maka akan ditampilkan halaman kesalahan 4. Konsumen memilih menu hapus topik IoT 4.1. Jika Topik ditemukan maka data topik akan dihapus 4.2. Jika topik tidak ditemukan maka akan ditampilkan halaman kesalahan.
Post-Condition	Informasi daftar topik akan ditampilkan

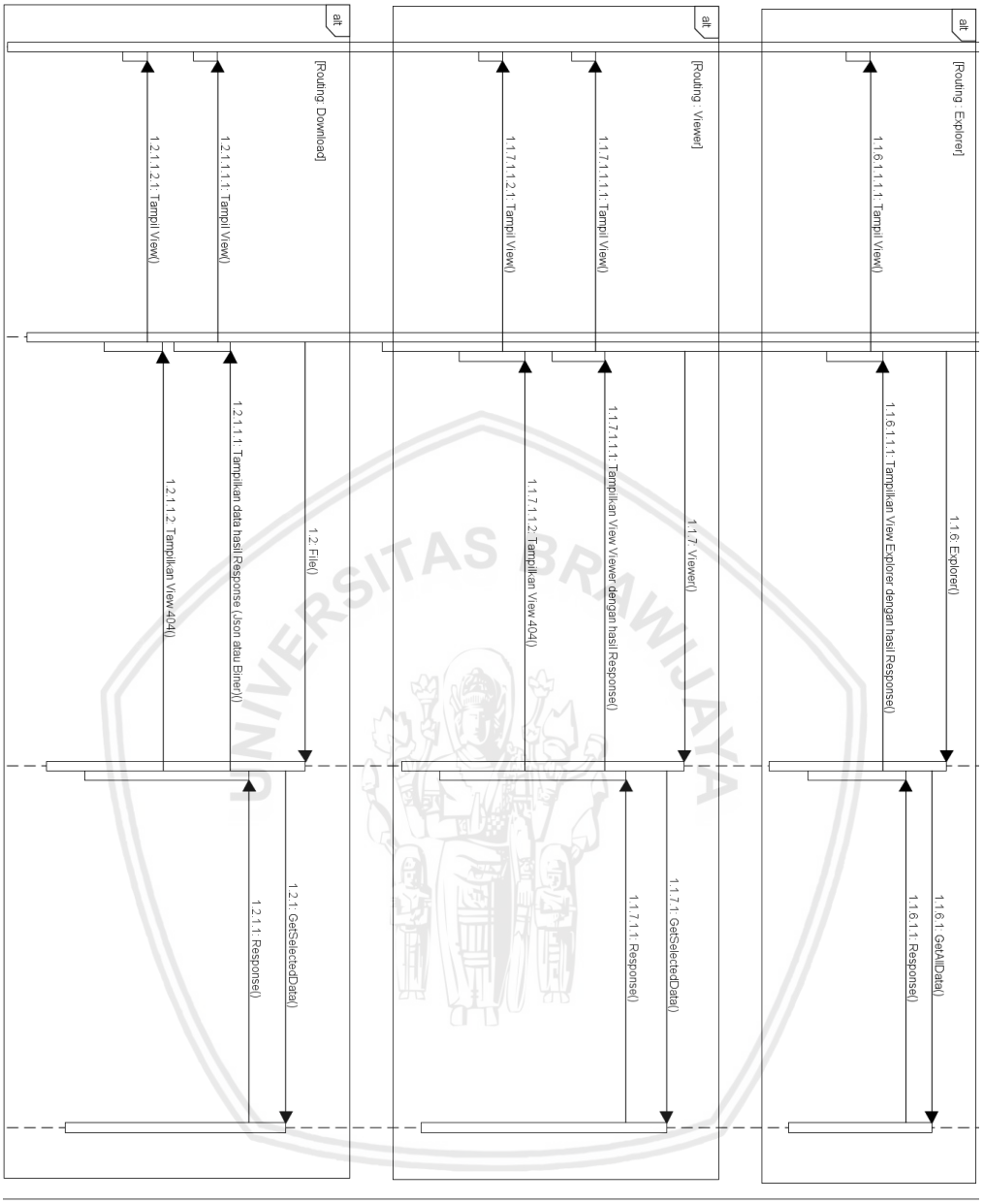


Gambar 4.10 *Sequence diagram* manajemen topik IoT (bagian 1)

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA



Gambar 4.11 *Sequence diagram* manajemen topik IoT (bagian 2)



Gambar 4.12 *Sequence diagram* manajemen topik IoT (bagian 3)

4.3.3.9 Mengolah Data

Pada penelitian ini akan dikembangkan juga sebuah fitur mengolah data IoT. Fitur ini akan melakukan pengolahan data IoT mulai dari menampilkan data IoT, analitik data.

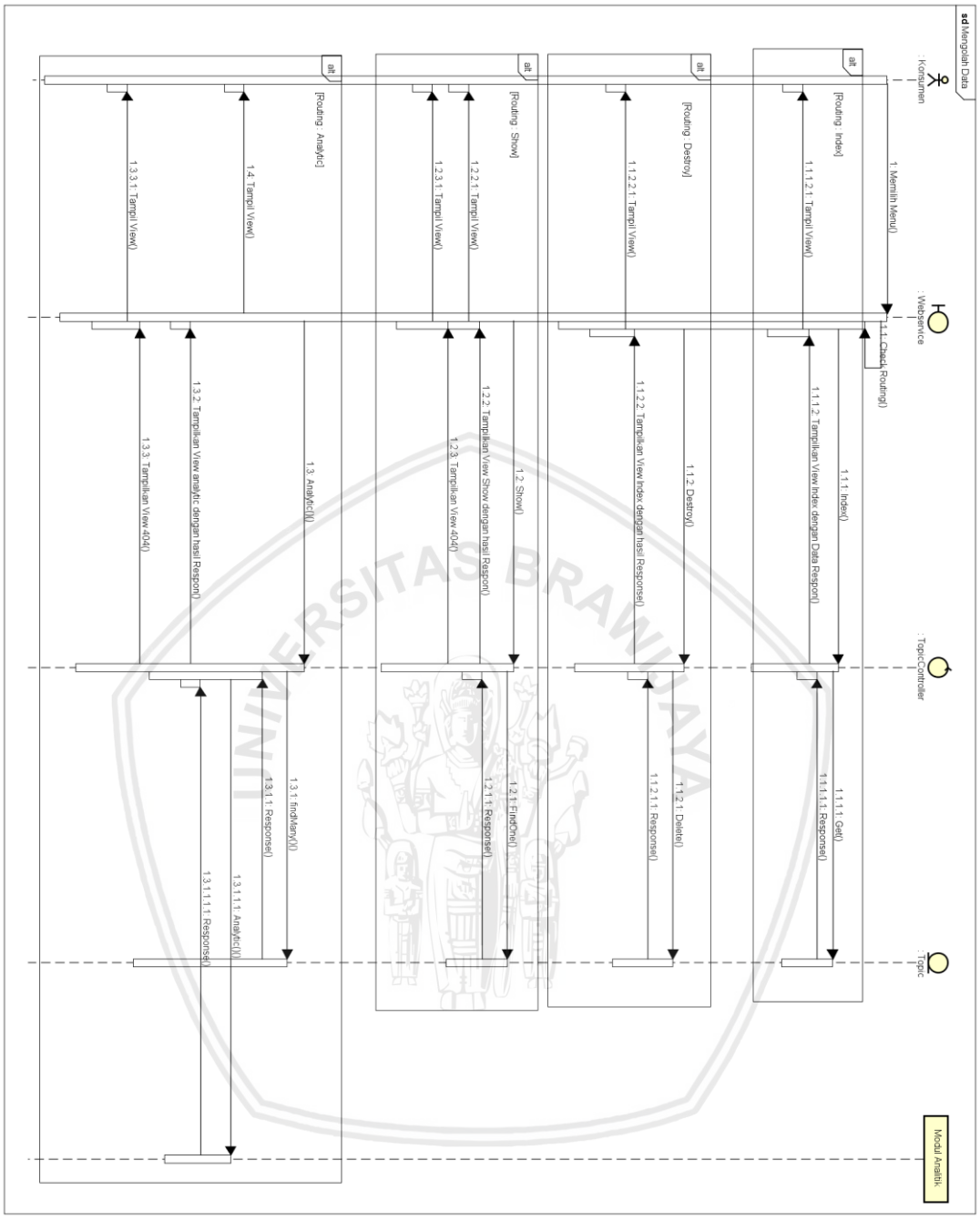
Saat melakukan analitik data IoT akan dicek terlebih dahulu apakah data merupakan milik pengguna, dan pengguna memiliki akses terhadap data tersebut, jika memiliki maka sistem akan memulai menganalisis data menggunakan modul analitik yang tersedia. Dan yang terakhir saat menampilkan data dari data juga

akan dicek apakah data merupakan milik dari pengguna, dan pengguna memiliki akses terhadap data tersebut, jika memiliki maka sistem dapat menampilkan data.

Untuk penjelasan yang lebih sistematis dapat dipahami melalui penjelasan yang terdapat pada *use case scenario* Tabel 4.14 dan *sequence diagram* pada Gambar 4.13.

Tabel 4.14 *Use case scenario* mengolah Data

Use case	Mengolah data
Objective	Dapat melakukan pengolahan data IoT
Actors	Konsumen
Pre - Condition	Login
Main Flow	1. Membuka halaman mengolah data IoT 2. Halaman pengolahan data IoT akan menampilkan daftar topik yang ada disistem beserta pilihan menu untuk melakukan analisis data.
Alternatif Flow	1. Konsumen memilih menu menampilkan proses analitik IoT 1.1. Halaman proses analitik akan muncul beserta informasi proses analitik yang sudah pernah dilakukan 1.2. Konsumen dapat menampilkan status dan keluaran dari data analitik yang sudah dilakukan 2. Konsumen memilih menu analitik data 2.1. Halaman analitik data akan muncul beserta *form* yang perlu diisi konsumen, *form* yang akan muncul disesuaikan dengan kebutuhan dari proses 2.2. Konsumen melakukan *submit* proses analitik 2.3. Jika proses analitik yang dilakukan oleh sistem berhasil maka akan dikembalikan Halaman yang berisi keluaran dari proses 2.4. Jika proses analitik yang dilakukan oleh sistem gagal maka akan dikembalikan Halaman yang berisi pesan kegagalan
Post-Condition	Informasi daftar topik beserta pilihan analitik akan muncul



Gambar 4.13 *Sequence diagram* mengolah data IoT

4.3.3.10 Manajemen Pengguna

Pada penelitian ini akan dikembangkan sebuah fitur manajemen pengguna. Fitur ini akan melakukan manajemen pengguna mulai dari pengubahan data pengguna, menampilkan data dari pengguna dan juga melakukan penghapusan pengguna.

Saat melakukan pengubahan data pengguna akan dilakukan pengisian data dari pengguna yang terdiri dari nama pengguna, alamat email pengguna, dan *username*. setelah diisi maka data akan diverifikasi oleh sistem apakah data dapat

diterima oleh sistem. Sistem verifikasi akan mengecek apakah penamaan pengguna sudah benar, dan juga jumlah huruf dari masukkan pengguna tidak kurang daripada 5 karakter dan lebih dari 50 karakter. Saat melakukan penghapusan data pengguna IoT akan dicek juga apakah pengguna merupakan milik dari pengguna, dan pengguna memiliki akses terhadap pengguna tersebut, jika memiliki maka sistem dapat menghapus pengguna. Dan yang terakhir saat menampilkan data dari pengguna juga akan dicek apakah pengguna merupakan milik dari pengguna, dan pengguna memiliki akses terhadap pengguna tersebut, jika memiliki maka sistem dapat menampilkan data dari pengguna.

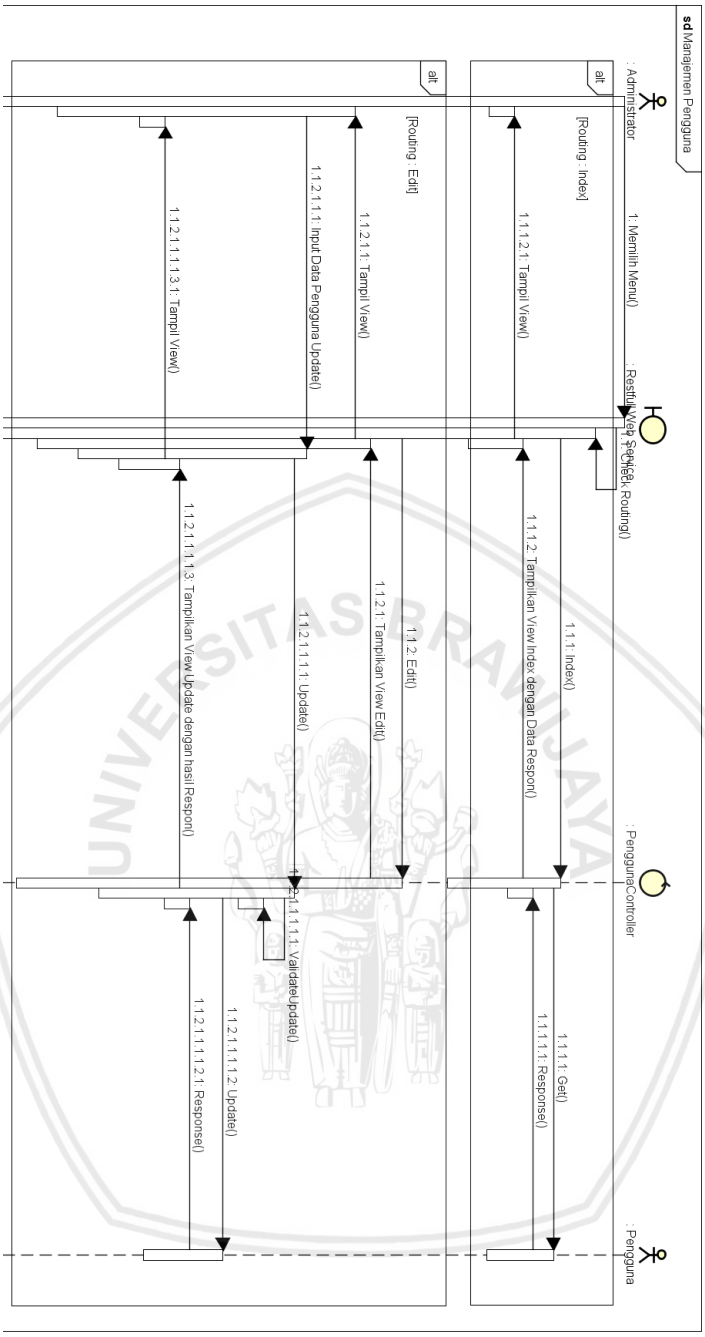
Untuk mempermudah pemahaman penulis sertakan *use case scenario* yang dapat dilihat dalam Tabel 4.15 dan Tabel 4.16, kemudian pada Gambar *sequence diagram* 4.14 dan 4.15 dibawah ini.

Tabel 4.15 *Use case scenario* manajemen pengguna (bagian 1)

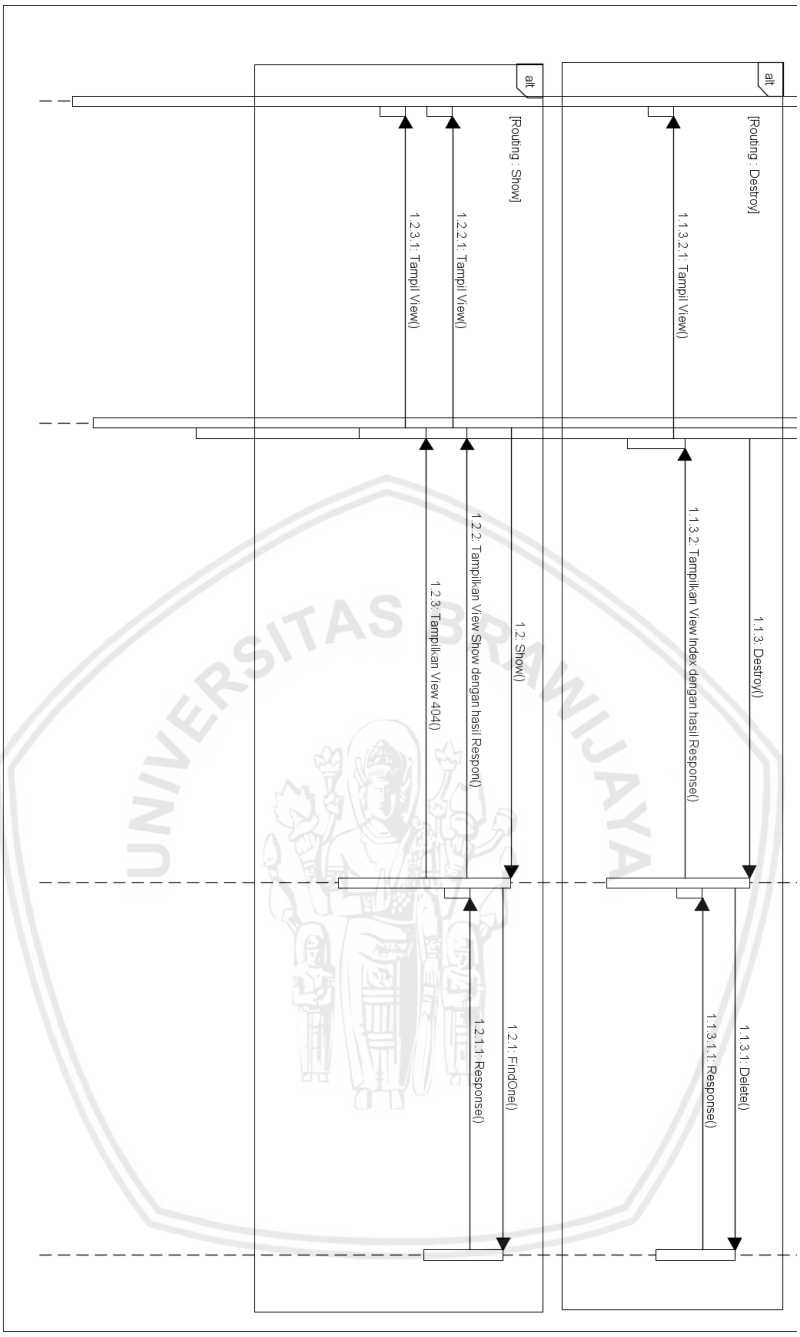
Use case	Manajemen pengguna IoT
Objective	Dapat melakukan manajemen pengguna IoT
Actors	Administrator
Pre - Condition	Login
Main Flow	1. Membuka halaman manajemen pengguna IoT 2. Halaman manajemen pengguna IoT akan menampilkan daftar pengguna yang sudah pernah didaftarkan oleh administrator beserta pilihan menu untuk lihat, ubah dan hapus pengguna beserta pilihan untuk menambahkan pengguna
Alternatif Flow	1. Administrator memilih menu menambahkan pengguna IoT 1.1. Halaman tambah pengguna IoT akan muncul 1.2. Administrator akan mengisi *form* yang ada pada halaman pengguna IoT 1.3. Administrator melakukan *submit* 1.4. Jika Pengguna berhasil disimpan maka akan muncul notifikasi berhasil, begitu juga jika pengguna gagal disimpan 2. Administrator memilih menu menampilkan pengguna IoT 2.1. Jika pengguna ditemukan maka informasi Pengguna akan ditampilkan secara penuh

Tabel 4.16 *Use case scenario* manajemen pengguna (bagian 2)

Alternatif Flow	2.2. Jika pengguna tidak ditemukan maka akan ditampilkan halaman kesalahan 3. Administrator memilih menu ubah pengguna IoT 3.1. Jika pengguna ditemukan maka halaman ubah pengguna akan ditampilkan 3.2. Jika ditemukan maka akan diminta mengisi *form* perubahan pengguna 3.3. Jika sudah diisi maka administrator akan melakukan *submit* pengguna 3.4. Jika pengguna berhasil disimpan maka akan muncul notifikasi berhasil, begitu juga jika pengguna gagal disimpan 3.5. Jika pengguna tidak ditemukan maka akan ditampilkan halaman kesalahan 4. Administrator memilih menu hapus pengguna IoT 4.1. Jika pengguna ditemukan maka pengguna pengguna akan dihapus 4.2. Jika pengguna tidak ditemukan maka akan ditampilkan halaman kesalahan.
Post-Condition	Informasi daftar pengguna akan ditampilkan



Gambar 4.14 *Sequence diagram* manajemen pengguna (bagian 1)



Gambar 4.15 *Sequence diagram* manajemen pengguna (bagian 2)

4.4 Perancangan Data

Perancangan data diperlukan untuk mendefinisikan data apasaja yang dapat diterima, disimpan, dan diproses oleh sistem yang peneliti kembangkan. Perancangan ini juga termasuk untuk mempermudah pengguna dalam memahami sistem yang telah dikembangkan.

4.4.1 Perancangan Data Semi Terstruktur

Data semi terstruktur adalah data yang memiliki struktur namun belum sepenuhnya terstruktur. Tipe data ini berisi komponen tidak terstruktur yang tersusun pada data terstruktur.

Data semiterstruktur diatur dalam bentuk entitas. Entitas yang sama dijadikan satu kelompok , namun dapat berbeda atributnya. Contoh dari data semitruktur adalah daftar riwayat hidup (CV). Individu pertama dapat memiliki pengalaman kerja , berbeda dengan individu kedua yang memiliki pengalaman dalam melakukan penelitian dan berbeda pula dengan orang ketiga yang memiliki pengalaman dalam hal pengajaran. Sering dijumpai pula perbedaan dalam penulisan CV lebih dari satu macam.

Standar yang biasa digunakan pada data semiterstruktur adalah penggunaan teknologi XML. XML mendukung pengembangan dokumen semiterstruktur yang berisi metadata dan teks. Metadata menggunakan tag XML dan dapat menjelaskan struktur dokumen. Tanpa metadata aplikasi tidak mampu memahami dan menguraikan isi dari dokumen XML.

Dalam sistem ini terdapat dua standar yang dapat dipahami oleh sistem yaitu yang pertama standar XML dan berikutnya adalah JSON. Contoh dari data yang berbentuk JSON dapat dilihat dalam Tabel 4.17, sedangkan contoh data dengan standar XML dapat dilihat dalam Tabel 4.18.

Tabel 4.17 Perancangan data semi terstruktur JSON

Contoh data JSON	
1	[{
2	"humidity": 74,
3	"temperature": 30,
4	"class" : "sangat baik"
5	},
6	]

Tabel 4.18 Perancangan data semi terstruktur XML

Contoh data XML	
1	<?xml version="1.0" encoding="UTF-8"?>
2	<Data>
3	<element>
4	<class>cukup baik</class>
5	<humidity>71</humidity>
6	<temperature>29</temperature>
7	</element>
8	</Data>

Data yang masuk dalam kategori ini akan mampu disimpan dalam semua topik, termasuk topik terstruktur (dengan tambahan pengecekan), topik semi terstruktur hingga topik tidak terstruktur. Selain itu data yang masuk dalam kategori ini akan mampu dianalitik ataupun diproses oleh sistem menggunakan PHP-ML ataupun Apache Spark.

4.4.2 Perancangan Data Tidak Terstruktur

Data tidak terstruktur biasanya ada pada teks, file, video, email, laporan (report), presentasi power point, pesan suara (voice mail), memo, dan citra. Data dapat berbentuk dalam tipe apapun tanpa perlu mengikuti format, aturan atau alur tertentu. Sebagai contoh data yang ditampilkan pada halaman web termasuk data tidak testruktur dan tidak beraturan.

Dalam sistem semua data tidak terstruktur dapat disimpan. Contoh dari data yang tidak terstruktur adalah gambar berformat JPG dapat dilihat dalam Gambar 4.16, selain itu juga data raw yang dapat dilihat dalam Tabel 4.19.

Gambar 4.16 Perancangan data tidak terstruktur JPG

Tabel 4.19 Perancangan data tidak terstruktur RAW

Contoh data RAW	
1	CANADA 1CNDA
2	15 14312 2140633.6 2.06.3
3	BAHAMAS 1BHMS
4	35999999 17099.099.09.0
5	CUBA 1CUBA
6	27 5033 856599.0 6.17.5

Data yang masuk dalam kategori ini akan mampu disimpan dalam satu topik saja yaitu topik tidak terstruktur. Selain itu data yang masuk dalam kategori ini akan tidak akan mampu dianalitik ataupun diproses oleh sistem.

4.5 Perancangan Pengujian

Perancangan pengujian diperlukan agar diketahui batas minimal pengujian dan melakukan penelitian berdasarkan skenario pengujian yang telah dibuat.

4.5.1 Pengujian Fungsional

Pengujian fungsional bertujuan agar diketahui apakah sistem yang dibuat telah memenuhi kebutuhan fungsional atau tidak. Jika seluruh kebutuhan fungsional telah terpenuhi maka dilanjutkan ke pengujian berikutnya. Namun jika kebutuhan fungsional belum memenuhi kebutuhan fungsional secara keseluruhan, maka dilakukan pengecekan kembali rancangan sistem dan juga

kode sistem yang digunakan, lalu diadakan pengujian sekali lagi. Hal itu dilakukan terus menerus sampai semua kebutuhan fungsional tecapai. Untuk skneario pengujian, bisa dilihat pada Tabel 4.20, 4.21, 4.22, 4.23 dan 4.24. Pada Tabel tersebut terdapat kolom kode yang berawalan "FR" yang menandakan bahwa pengujian yang dilakukan adalah pengujian fungsional pada sistem diikuti dengan angka yang menunjukkan urutan pengujian. Kolom fungsi berisikan fungsi-fungsi yang harus dijalankan oleh *webservice*. Fungsi tersebut dijalankan sesuai dengan langkah pengujian yang telah dituliskan sesuai dengan fungsinya masing-masing.

Tabel 4.20 Pengujian fungsional (bagian 1)

Kode	Fungsi	Langkah Uji
FR_001	Autentikasi login perangkat menggunakan JWT	- Sistem dijalankan - Pengguna menjalankan aplikasi Postman - Pengguna memasukkan *Token* dan *Secret key* dari perangkat - Pengguna melihat hasil respon dari aplikasi Postman
FR_002	Menampilkan daftar topik melalui sebuah *API*	- Sistem dijalankan - Pengguna menjalankan aplikasi Postman - Pengguna memasukkan JWT dari perangkat - Pengguna melihat hasil respon dari aplikasi Postman
FR_003	Menerima data terstruktur dalam bentuk JSON	- Sistem dijalankan - Pengguna menjalankan aplikasi Postman - Pengguna memasukkan JWT dari perangkat - Pengguna memasukkan *file* JSON yang mengandung data terstruktur

Tabel 4.21 Pengujian fungsional (bagian 2)

FR_003	Menerima data terstruktur dalam bentuk JSON	- Pengguna melihat hasil respon dari aplikasi Postman
FR_004	Menerima data semi terstruktur dalam bentuk XML	- Sistem dijalankan - Pengguna menjalankan aplikasi Postman - Pengguna memasukkan JWT dari perangkat - Pengguna memasukkan *file* XML yang mengandung data semi terstruktur - Pengguna melihat hasil respon dari aplikasi Postman
FR_005	Menerima data tidak terstruktur dalam bentuk Gambar JPG	- Sistem dijalankan - Pengguna menjalankan aplikasi Postman - Pengguna memasukkan JWT dari perangkat - Pengguna memasukkan *file* Gambar - Pengguna melihat hasil respon dari aplikasi Postman
FR_006	Menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API*	- Sistem dijalankan - Pengguna menjalankan aplikasi Postman - Pengguna memasukkan JWT dari perangkat beserta topik yang ingin ditampikan - Pengguna melihat hasil respon dari aplikasi Postman

Tabel 4.22 Pengujian fungsional (bagian 3)

FR_007	Menampilkan data yang disimpan dalam suatu topik melalui sebuah *API*	- Sistem dijalankan - Pengguna menjalankan aplikasi Postman - Pengguna memasukkan *Token* dan *Secret key* dari perangkat beserta topik dan id data yang ingin ditampikan - Pengguna melihat hasil respon dari aplikasi Postman
FR_008	Registrasi pengguna baru	- Sistem dijalankan - Pengguna melakukan input data pengguna baru - Pengguna mencocokan data dengan *database*
FR_009	Login pengguna	- Sistem dijalankan - Pengguna melakukan input data pengguna - Pengguna mencocokan respon
FR_010	CRUD (manajemen) perangkat IoT	- Sistem dijalankan - Pengguna melakukan CRUD dalam perangkat IoT - Pengguna mencocokan data dengan *database*
FR_011	CRUD (manajemen) topik IoT	- Sistem dijalankan - Pengguna melakukan CRUD dalam topik IoT - Pengguna mencocokan data dengan *database*

Tabel 4.23 Pengujian fungsional (bagian 4)

FR_012	CRUD (manajemen) pengguna	- Sistem dijalankan - Administrator melakukan CRUD pengguna - Administrator mencocokan data dengan *database*
FR_013	Menampilkan daftar data yang disimpan dalam suatu topik	- Sistem dijalankan - Pengguna memilih topik - Pengguna mencocokan hasil keluaran dari sistem dengan *database*
FR_014	Menampilkan data yang disimpan dalam suatu topik	- Sistem dijalankan - Pengguna memilih topik dan id data yang dipilh - Pengguna mencocokan hasil keluaran dari sistem dengan *database*
FR_015	Prediksi data menggunakan least square	- Sistem dijalankan - Pengguna memilih topik dan memilih modul analitik least square - Pengguna memasukkan data yang ingin diprediksi - Pengguna melihat hasil keluaran dari sistem

Tabel 4.24 Pengujian fungsional (bagian 5)

FR_016	Klasifikasi data menggunakan Naïve bayes	- Sistem dijalankan - Pengguna memilih topik dan memilih modul analitik naïve - Pengguna memasukkan data yang ingin diklasifikasikan - Pengguna melihat hasil keluaran dari sistem
FR_017	Menampilkan informasi statistik dari sebuah topik	- Sistem dijalankan - Pengguna memilih topik - Pengguna melihat hasil keluaran dari sistem

4.5.2 Pengujian Skalabilitas

Pengujian skalabilitas bertujuan untuk mengetahui batas kemampuan suatu sistem dalam menangani beban sistem sehingga sistem tersebut akan digunakan sesuai dengan kemampuannya dalam menangani beban sistem (Pramukantoro, 2017). Pengujian skalabilitas dilakukan dengan cara melakukan pengambilan data dengan jumlah *request* data yaitu 100 *request*, 300 *request*, 500 *request*, 700 *request*, dan 1000 *request*. Selain pengambilan data, dilakukan pula uji coba penyimpanan data sebanyak 100 kali, 300 kali, 500 kali, 700 kali, dan 1000 kali. Dilakukan pula pengujian analisis data yang sebanyak 1000 data, 10.000 data, 100.000 data, 500.000 data dan 1.000.000 data menggunakan modul PHP-ML dan Apache Spark.

Untuk penentuan angka dalam pengujian adalah dengan mempertimbangkan tingkat penurunan kinerja dari sistem. Jika sudah ditemukan *error* pada saat pengujian di salah satu skenario pengujian maka angka tersebut akan dijadikan batas atas.

Data yang digunakan untuk melakukan pengujian pengiriman data dan pengambilan data ke sistem adalah sebuah data JSON yang memiliki 3 (tiga) parameter yaitu ph, nitrogen*,* dan *humidity.* Contoh data dapat dilihat di Lampiran B.2.2.1 untuk data yang dikirim dan Lampiran B.2.1.1 untuk data yang diambil. Selain itu data yang dilakukan analisis adalah data DHT atau *temperature* dan *humidity*, data dilakukan analisis menggunakan Naïve bayes untuk menentukan kelas dari sebuah data. Data yang dianalisis terdapat di Lampiran B.2.3.1, sedangkan data yang digunakan sebagai data latih terdapat pada Lampiran B.2.3.2.

repository.ub.ac.id

UNIVERSITAS BRAWIJAYA

BAB 5 IMPLEMENTASI

Implementasi merupakan hal yang paling penting dari sebuah penelitian. Implementasi dilakukan untuk memenuhi tujuan penelitian yang berdasarkan pada perancangan yang telah dilakukan sebelumnya. Adapun implementasi yang dilakukan yaitu implementasi Infrastruktur yang meliputi *IoT data storage*, *IoT data analytic* dan juga IoT *Visualization*s *webservice* beserta implementasi aplikasi. Hasil pengimplementasian tersebut diujikan pada bab berikutnya yaitu bab pengujian.

Dikarenakan ini merupakan penelitian pengembangan lanjutan, maka beberapa komponen akan ada yang tidak disertakan secara eksplisit, dan hanya komponen tambahan atau komponen baru saja yang akan dituliskan. Pada implementasi aplikasi sendiri bahasa pemrograman yang digunakan akan berbeda dari penelitian sebelumnya, sehingga diperlukan proses *coding* ulang menggunakan bahasa pemrograman PHP dan Python.

5.1 Implementasi Infrastruktur

Sebelum mengimplementasikan aplikasi, maka sebelumnya akan dilakukan implementasi infrastruktur. Infrastruktur sendiri adalah *backbone* dari sistem yang dibangun, dikarenakan tanpa infrastruktur ini maka aplikasi tidak akan berjalan.

Dalam tahap implementasi infrastruktur akan digunakan 3 (tiga) buah *Virtual Private Server*, yang semuanya terletak pada layanan *cloud* Microsoft Azure. Ketiga *Virtual Private Server* sendiri akan diberi *identifier* berupa *domain* dimulai dengan dev1.ptriot.com, dev2.ptriot.com dan yang terakhir dev3.ptriot.com.

5.1.1 Implementasi *IoT Data Storage*

Untuk mengimplementasikan sebuah *IoT data storage* diperlukan berbagai macam hal yang perlu dilakukan, dimulai dari instalasi perangkat lunak yang dibutuhkan seperti MongoDB, GridFS dan Apache Hadoop, selain proses instalasi maka setelah itu akan dilakukan proses konfigurasi perangkat lunak agar dapat digunakan dalam sistem.

Dikarenakan server yang digunakan untuk implementasi sistem adalah sebuah linux dengan distro Ubuntu server 17.10 maka pada penjelasan implementasi kali ini akan banyak sekali perintah *(syntax)* dari Ubuntu yang digunakan.

5.1.1.1 Implementasi Data Storage Berbasis MongoDB dan GridFS

Ada beberapa opsi untuk menginstal MongoDB beserta GridFS, akan tetapi pada penelitian kali ini penulis menginstal MongoDB beserta GridFS pada mesin yang sama dengan webserver. Hal ini dilakukan agar proses pertukaran data dari dan menuju webserver semakin cepat.

Untuk mengintall MongoDB beserta GridFS diperlukan Server dengan OS yang memenuhi spesifikasi MongoDB, dalam hal ini adalah Ubuntu Server 17.10. setelah melakukan instalasi MongoDB maka gridfs juga akan langsung secara otomatis terinstal juga, hal ini dikarenakan MongoDB dan gridfs terdapat pada satu *package* aplikasi yang sama. Oleh karena itu selanjutnya maka diperlukan konfigurasi lanjutan untuk memulai menggunakan MongoDB dengan perintah yang terdapat pada Konfigurasi 5.1.

Konfigurasi 5.1 Mongo

ConfigureMongo.sh

```
1   #Perintah untuk allow port 27017
2   sudo ufw allow 27017
3   sudo sed -i "s/bindIp: .*/bindIp: 0.0.0.0/" /etc/mongod.conf
4
5   #Perintah untuk membuat database dengan nama ptriot
6   Mongo   ptriot   --eval   "db.test.insert({name:'db   creation'})"
7   2>/dev/null
8
9   #Menginstal driver untuk MongoDB pada Python
10  Pip instal pymongo
```

5.1.1.2 Implementasi Data Storage Berbasis Apache Hadoop

Apache Hadoop sendiri akan digunakan oleh sistem sebagai *file storage* sekunder setelah MongoDB, Apache Hadoop sendiri akan digunakan untuk menyimpan *file* input, keluaran dan *file* tambahan yang digunakan dalam proses analitik yang oleh Apache Spark. Apache Hadoop sendiri nantinya akan diinstal dalam 3 (tiga) buah server, sebuah server akan bertindak sebagai *master* dan *slave*, sedangkan 2 (dua) buah lainnya hanya akan bertindak sebagai slave saja.

Setelah melakukan instalasi pada 3 (tiga) buah server, maka sistem belum bisa langsung digunakan, masih diperlukan beberapa konfigurasi tambahan agar sistem dapat berjalan dengan baik. Konfigurasi yang perlu dilakukan adalah melakukan penimpaan *(overwriting)* terhadap beberapa *file* yaitu core-site.xml sesuai dengan isi pada Konfigurasi 5.2 dan 5.3, kemudian Apache Hadoop-env.sh yang sesuai dengan isi pada Konfigurasi 5.4, kemudian Hdfs-site.xml yang akan ditimpa dengan isi dari Konfigurasi 5.5, kemudian mapred-site.xml yang akan digantikan dengan isi dari Konfigurasi 5.6 dan 5.7, setelah itu juga ada yarn-site.xml yang perlu digantikan dengan isi dari Konfigurasi 5.8 dan 5.9.

Kemudian juga diperlukan konfigurasi *master* dan *slave* yang perlu diinisiasi melalui sebuah *file* khusus bernama *masters* dan *slaves* yang isi dari *file* tersebut adalah informasi dari *masters* ataupun *slaves*, yang dalam penelitian ini dapat dilihat dalam Konfigurasi 5.10 dan 5.11 Seluruh bukti implementasi dari konfigurasi ini dapat ditemukan pada Lampiran A.1.1.1 hingga A.1.1.7.

Konfigurasi 5.2 Core-Site.xml (bagian 1)

Core-site.xml

```
1   <!-- Generated By Carbonifero for Apache Hadoop Core Site -->
2   <configuration>
3       <property>
4           <name>fs.defaultFS</name>
```

Konfigurasi 5.3 Core-Site.xml (bagian 2)

```
5           <value>hdfs://dev1.ptriot.com:9000</value>
6       </property>
7
8       <property>
9           <name>hadoop.tmp.dir</name>
10          <value>/usr/local/bin/hadoop2/hadoop2_data/temp</value>
11      </property>
12  </configuration>
```

Konfigurasi 5.4 Hadoop-env.sh

Apache Hadoop-env.sh

```
1   export JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

Konfigurasi 5.5 Hdfs-site.xml

Hdfs-site.xml

```
1   <!-- Generated By Carbonifero for Apache Hadoop Hdfs Site -->
2   <configuration>
3       <property>
4           <name>dfs.replication</name>
5           <value>2</value>
6       </property>
7       <property>
8           <name>dfs.permissions</name>
9           <value>false</value>
10      </property>
11
12      <property>
13          <name>dfs.namenode.name.dir</name>
14      <value>/usr/local/bin/hadoop/hadoop2_data/hdfs/
15  namenode</value>
16      </property>
17
18      <property>
19          <name>dfs.datanode.data.dir</name>
20          <value>/usr/local/bin/hadoop/hadoop2_data/hdfs/
21  datanode</value>
22      </property>
23  </configuration>
```

Konfigurasi 5.6 Mapred-site.xml (bagian 1)

Mapred-site.xml

```
1   <!-- Generated By Carbonifero for Hadoop Mapred Site -->
2   <configuration>
3       <property>
4           <name>mapreduce.framework.name</name>
5           <value>yarn</value>
6       </property>
7
8       <property>
9           <name>mapreduce.map.memory.mb</name>
10          <value>1536</value>
11      </property>
12
```

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

Konfigurasi 5.7 Mapred-site.xml (bagian 2)

```
13      <property>
14          <name>mapreduce.reduce.memory.mb</name>
15          <value>3072</value>
16      </property>
17
18      <property>
19          <name>mapreduce.map.java.opts</name>
20          <value>-Xmx1228m</value>
21      </property>
22
23      <property>
24          <name>mapreduce.reduce.java.opts</name>
25          <value>-Xmx2457m</value>
26      </property>
27
28      <property>
29          <name>mapreduce.task.io.sort.mb</name>
30          <value>614</value>
31      </property>
32
33  </configuration>
```

Konfigurasi 5.8 Yarn-site.xml (bagian 1)

```
Yarn-site.xml
1   <!-- Generated By Carbonifero for Hadoop Yarn Site -->
2   <configuration>
3       <property>
4           <name>yarn.nodemanager.aux-services</name>
5           <value>mapreduce_shuffle</value>
6       </property>
7
8       <property>
9           <name>yarn.nodemanager.aux-
10  services.mapreduce.shuffle.class</name>
11          <value>org.apache.hadoop.mapred.ShuffleHandler</value>
12      </property>
13
14      <property>
15          <name>yarn.resourcemanager.resource-tracker.address</name>
16          <value>dev1.ptriot.com:8025</value>
17      </property>
18
19      <property>
20          <name>yarn.resourcemanager.scheduler.address</name>
21          <value>dev1.ptriot.com:8030</value>
22      </property>
23
24      <property>
25          <name>yarn.resourcemanager.address</name>
26          <value>dev1.ptriot.com:8050</value>
27      </property>
28
29      <property>
30          <name>yarn.nodemanager.resource.memory-mb</name>
31          <value>4608</value>
32      </property>
33
34      <property>
35          <name>yarn.scheduler.maximum-allocation-mb</name>
36          <value>4608</value>
37      </property>
```

Konfigurasi 5.9 Yarn-site.xml (bagian 2)

```
38
39      <property>
40          <name>yarn.scheduler.minimum-allocation-mb</name>
41          <value>1536</value>
42      </property>
43
44      <property>
45          <name>yarn.nodemanager.vmem-check-enabled</name>
46          <value>false</value>
47      </property>
48
49      <property>
50          <name>yarn.app.mapreduce.am.resource.mb</name>
51          <value>3072</value>
52      </property>
53
54      <property>
55          <name>yarn.app.mapreduce.am.command-opts</name>
56          <value>-Xmx2457m</value>
57      </property>
58
59      <property>
60         <name>yarn.application.classpath</name>
61         <value>
62              %HADOOP_HOME%\etc\hadoop,
63              %HADOOP_HOME%\share\hadoop\common\*,
64              %HADOOP_HOME%\share\hadoop\common\lib\*,
65              %HADOOP_HOME%\share\hadoop\hdfs\*,
66              %HADOOP_HOME%\share\hadoop\hdfs\lib\*,
67              %HADOOP_HOME%\share\hadoop\mapreduce\*,
68              %HADOOP_HOME%\share\hadoop\mapreduce\lib\*,
69              %HADOOP_HOME%\share\hadoop\yarn\*,
70              %HADOOP_HOME%\share\hadoop\yarn\lib\*
71         </value>
72      </property>
73
74      <property>
75          <name>yarn.log-aggregation-enable</name>
76          <value>true</value>
77      </property>
78
79      <property>
80          <name>yarn.nodemanager.log-aggregation.roll-monitoring-
81  interval-seconds</name>
82          <value>3600</value>
83      </property>
84
85  </configuration>
```

Konfigurasi 5.10 Apache Hadoop Masters

```
masters
1   Dev1.ptriot.com
```

Konfigurasi 5.11 Apache Hadoop Slaves

```
slaves
1   Dev1.ptriot.com
2   Dev2.ptriot.com
3   Dev3.ptriot.com
```

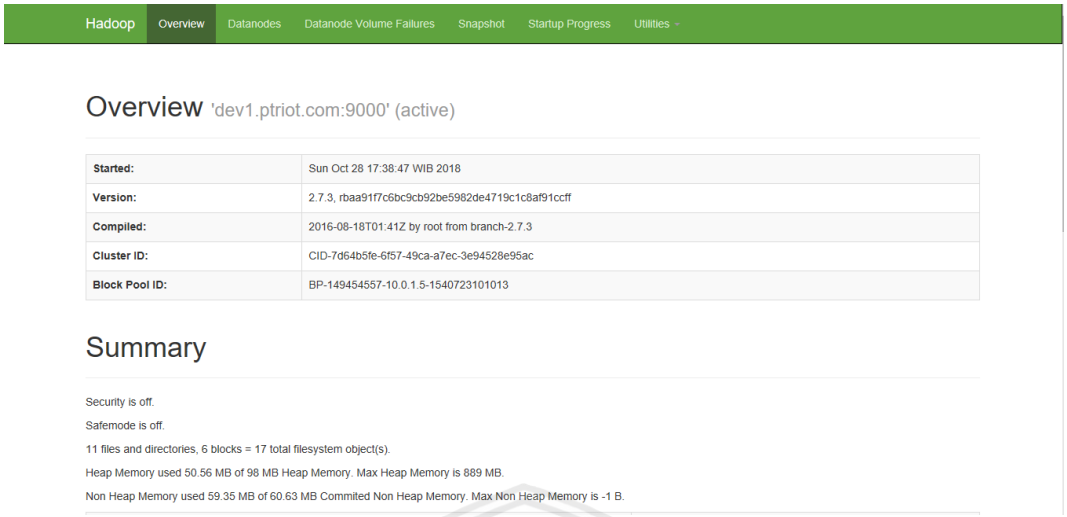
Setelah melakukan konfigurasi seperti pengaturan diatas, maka masih ada beberapa hal tambahan yang perlu dilakukan seperti pembuatan folder lokasi *file* akan disimpan. Langkah tersebut dapat dilakukan menggunakan perintah seperti berikut yang terdapat pada Konfigurasi 5.9.

Konfigurasi 5.12 Apache Hadoop

ConfigureApache Hadoop.sh

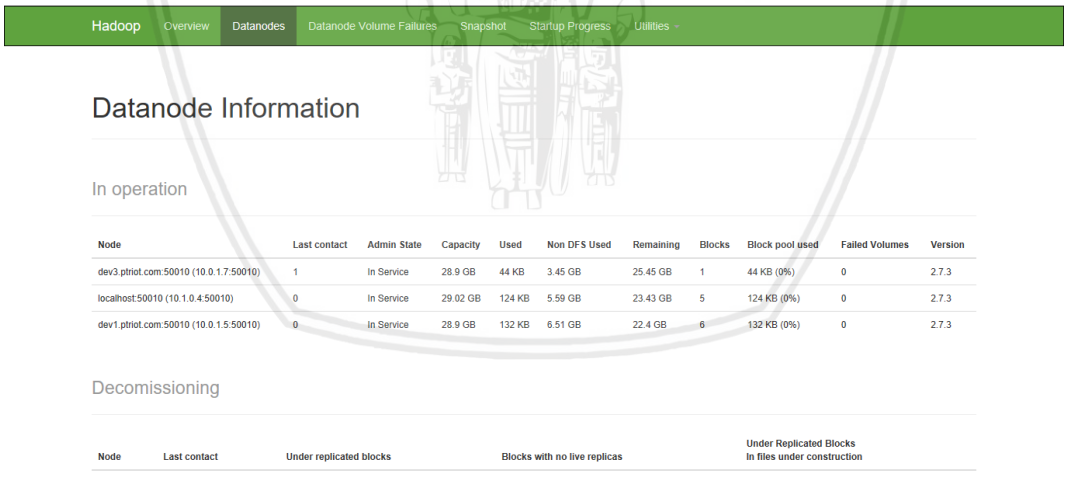
```
1   # Add Path
2   APACHE HADOOP_HOME=/usr/local/bin/hadoop
3   block="## Generate By Carbofinero for Apache Hadoop
4   export APACHE HADOOP_HOME='$APACHE HADOOP_HOME'
5   export PATH=$PATH:$APACHE HADOOP_HOME/bin
6   export PATH=$PATH:$APACHE HADOOP_HOME/sbin
7   export APACHE HADOOP_MAPRED_HOME=$APACHE HADOOP_HOME
8   export APACHE HADOOP_COMMON_HOME=$APACHE HADOOP_HOME
9   export APACHE HADOOP_HDFS_HOME=$APACHE HADOOP_HOME
10  export YARN_HOME=$APACHE HADOOP_HOME"
11
12  cd ~
13  sudo echo "$block" >> ~/.bashrc
14  source .bashrc
15
16  cd /usr/local/bin/Apache Hadoop
17  # Make Directory for Namenode and DataNode
18  sudo rm hadoop2_data/hdfs -rf
19  sudo mkdir -p hadoop2_data/hdfs/namenode
20  sudo mkdir -p hadoop2_data/hdfs/datanode
21  sudo mkdir -p hadoop2_data/hdfs/temp
22  sudo mkdir -p logs
23  sudo chown faathin -R /usr/local/bin/ hadoop2
24  sudo chmod 777 -R logs Apache Hadoop2_data
25
26  sudo start-dfs.sh
27  sudo start-yarn.sh
```

Setelah menjalankan perintah diatas maka Apache Hadoop sudah dapat berjalan dengan sempurna, seperti pada bukti tangkapan layar yang menunjukan ringkasan *(overview)* dari Apache Hadoop yang ada pada Gambar 5.1, kemudian juga ada informasi mengenai Datanode yang ada pada Gambar 5.2.



Gambar 5.1 Ringkasan Apache Hadoop

Pada Gambar 5.1 terlihat bahwa infrastruktur Apache Hadoop sudah berhasil diimplementasikan, ditandai dengan tampilan *web-ui* yang dapat diakses, dalam gambar tersebut terlihat kapan waktu dari Apache Hadoop berjalan yaitu Sun, Oct 28 17:38:47 WIB 2018. Dan juga terlihat versi dari Hadoop yang digunakan yaitu versi 2.7.3 yang dilakukan *compile* pada 2016-08-18T01:41Z, selain itu juga ada *summary* dari Apache Hadoop yang menandakan bahwa *safemode* dan *security* dimatikan, selain itu terdapat 11 *file* dan *directories* yang menggunakan sejumlah 6 blocks.



Gambar 5.2 *Datanode* Apache Hadoop

Selain itu berdasarkan Gambar 5.2 diketahui juga bahwa *datanode* dari Apache Hadoop yang beroperasi sejumlah 3 (tiga) *nodes*. Yaitu dev3.ptriot.com, localhost dan dev1.ptriot.com. Beserta alamat IP dari *nodes* tersebut, kapasitas penyimpanan, dan juga penggunaan penyimpanan dari *nodes* tersebut.

5.1.2 Implementasi *IoT Data Analytic*

Untuk mengimplementasikan sebuah *IoT data analytic* diperlukan berbagai macam hal yang perlu dilakukan, dimulai dari instalasi perangkat lunak yang dibutuhkan seperti Apache Spark dan juga PHP-ML *(PHP Machine Learning),* selain proses instalasi maka setelah itu akan dilakukan proses konfigurasi perangkat lunak agar dapat digunakan dalam sistem.

5.1.2.1 Implementasi Data Analytic Berbasis Apache Spark

Untuk melakukan instalasi Apache Spark dapat digunakan berbagai macam cara, cara yang paling gampang adalah dengan menginstal melalui repository Debian menggunakan *apt*, akan tetapi jika menggunakan cara seperti itu maka proses konfigurasi Apache Spark akan sedikit lebih susah.

Oleh karena itu karena akan ada beberapa konfigurasi tambahan maka untuk proses instalasi akan menggunakan cara manual menggunakan source code Apache Spark, yang dapat dilihat pada peritah berikut yang terdapat pada Konfigurasi 5.13.

Konfigurasi 5.13 Apache Spark

ConfigApache Spark.sh

```
1  # Menambahkan Path
2  APACHE SPARK_HOME=/usr/local/bin/spark
3
4  block="## Generate By Carbofinero for Apache Spark
5  export APACHE SPARK_HOME='$APACHE SPARK_HOME'
6  export PATH=$PATH:$APACHE
7  SPARK_HOME/bin:/usr/local/bin/hadoop/bin:/usr/
8  local/bin/hadoop/sbin"
9
10 cd ~
11 sudo echo "$block" >> ~/.bashrc
12 source .bashrc
13
14 pip instal pyspark --no-cache-dir
```

Setelah melakukan instalasi seperti diatas maka akan diperlukan konfigurasi tambahan, konfigurasi ini akan menentukan apakah Apache Spark yang ada pada node tersebut bertindak sebagai *master* atau sebagai *slave*, oleh karena itu perlu konfigurasi pada *file Spark-env.sh,* selain itu juga perlu mendefinisikan jumlah *masters* dan *slaves* melalui dua *file masters* dan *slaves,* untuk konfigurasi sendiri dapat dilihat dalam Konfigurasi 5.14 untuk Spark-env.sh, kemudian 5.15 untuk konfigurasi *masters*, dan yang terakhir adalah 5.16 untuk konfigurasi *slaves*. Seluruh bukti implementasi dari konfigurasi ini dapat ditemukan pada Lampiran A.2.1.1 hingga A.2.1.3.

Konfigurasi 5.14 Spark-env.sh

Spark-env.sh

```
1  #!/usr/bin/env bash
2  HADOOP_CONF_DIR=/usr/local/bin/ Hadoop/etc/Hadoop
3  SPARK_LOCAL_IP= NODE_IP #Isi dengan IP dari Node (10.0.1.4)
4  SPARK_MASTER_HOST=MASTER_IP #Isi dengan IP dari Master(10.0.1.5)
5  SPARK_MASTER_PORT=7077
6  SPARK_WORKER_MEMORY=1G #Isi dengan Memory yang digunakan
```

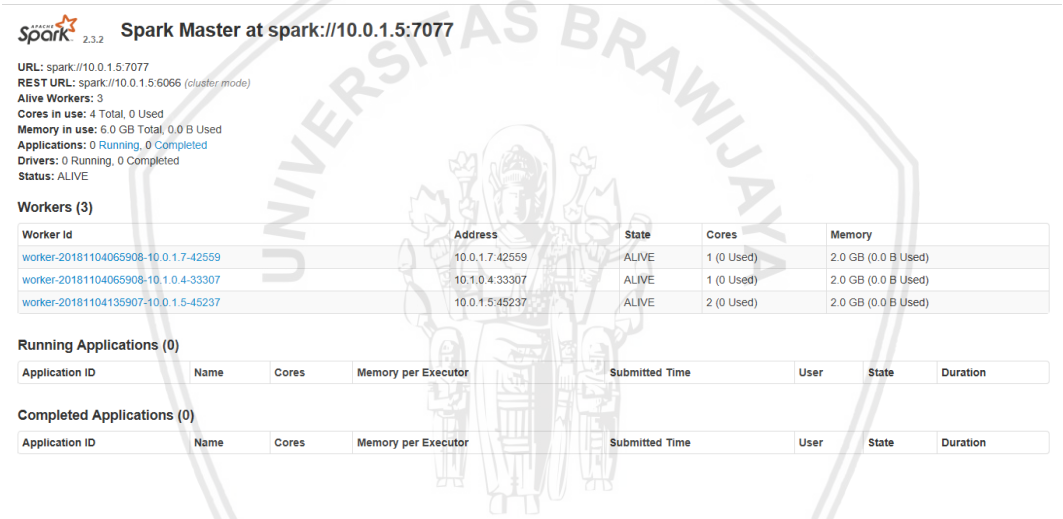
Konfigurasi 5.15 Masters

```
masters
1   Dev1.ptriot.com
```

Konfigurasi 5.16 Slaves

```
slaves
1   Dev1.ptriot.com
2   Dev2.ptriot.com
3   Dev3.ptriot.com
```

Setelah konfigurasi dilakukan maka Apache Spark sudah dapat dijalankan, dan dibawah ini pada Gambar 5.3 akan ditampilkan tampilan setelah Apache Spark master dijalankan pada node master *dev1.ptriot.com (10.0.1.5)*



Gambar 5.3 Ringkasan Apache Spark

Berdasarkan Gambar 5.3 diketahui bahwa implementasi Infrastruktur Apache Spark sudah berhasil dilakukan, ditandai dengan *web-ui* yang dapat diakses. Dalam *web-ui* yang dapat diakses terlihat beberapa informasi berupa jumlah *worker, memory,* dan *applications* yang berjalan ataupun selesai. Selain itu juga ditunjukan informasi detail dari tiap-tiap *workers* dari *master* tersebut.

5.1.2.2 Implementasi Data Analytic Berbasis PHP-ML

Setelah melakukan instalasi dan konfigurasi Apache Spark maka diperlukan instalasi lagi terhadap satu buah *library* yang digunakan yaitu PHP-ML (PHP Machine Learning), untuk proses instalasinya dapat melalui program *dependency* manager seperti *composer*.

5.1.3 Implementasi *IoT Visualization Webservice*

Untuk mengimplementasikan sebuah *IoT visualization webservice* diperlukan berbagai macam hal yang perlu dilakukan, dimulai dari instalasi

perangkat lunak yang dibutuhkan seperti Apache Webserver, NGiNX, Redis dan juga Laravel, selain proses instalasi maka setelah itu akan dilakukan proses konfigurasi perangkat lunak agar dapat digunakan dalam sistem.

5.1.3.1 Implementasi Webservice Berbasis Apache Webserver

Proses pertama yang diakukan dalam implementasi *webservice* adalah melakukan instalasi webserver yang digunakan untuk menjalankan kode program. *Webserver* yang digunakan sendiri adalah Apache Webserver dikarenakan kemampuan Apache Webserver sendiri sudah tidak diragukan lagi. Selain itu Apache Webserver juga merupakan program yang *opensource* sehingga tidak diperlukan lisensi untuk menggunakanya.

Untuk proses instalasi sendiri dapat dilakukan melalui berbagai cara, yang paling mudah untuk sistem operasi Ubuntu adalah dengan menggunakan repository *apt*. setelah melakukan proses instalasi maka akan dilakukan lagi proses konfigurasi *webserver*, dikarenakan dalam sistem ini akan digunakan tiga buah *subdomain* maka konfigurasi webserver juga akan dilakukan tiga kali, selain itu dikarenakan dalam sistem ini akan di implementasikan *reverse proxy* maka ketiga *webserver* ini akan dikonfigurasikan untuk berjalan di *port* 8080, dari yang biasanya 80, *file* konfigurasi dapat dilihat seperti potongan kode yang terdapat pada Konfigurasi 5.17 untuk konfigurasi ptriot.com, kemudian Konfigurasi 5.18 untuk konfigurasi api.ptriot.com dan yang terakhir pada Konfigurasi 5.19 untuk konfigurasi admin.ptriot.com. Seluruh bukti implementasi dari konfigurasi ini dapat ditemukan pada Lampiran A.3.1.1 hingga A.3.1.3.

Konfigurasi 5.17 Ptriot.com.conf

ptriot.com.conf

```
1  ## AUTOMATICALY GENERATED BY CARBONIFERO
2  ## SITE : ptriot.com
3  ## IP   : IP_ADDRESS : 8080
4  <VirtualHost *:8080>
5      ServerAdmin sysadmin@ptriot.com
6      ServerName ptriot.com
7      DocumentRoot /var/www/ptriot/public
8      ErrorLog ${_LOG_DIR}/ptriot_error.log
9      CustomLog ${_LOG_DIR}/ptriot_access.log combined
10 </VirtualHost>
11 # vim: syntax= ts=4 sw=4 sts=4 sr noet
```

Konfigurasi 5.18 Api.ptriot.com.conf

Api.ptriot.com.conf

```
1  ## AUTOMATICALY GENERATED BY CARBONIFERO
2  ## SITE : API.ptriot.com
3  ## IP   : IP_ADDRESS : 8080
4  <VirtualHost *:8080>
5      ServerAdmin sysadmin@ptriot.com
6      ServerName API.ptriot.com
7      DocumentRoot /var/www/ptriot/public
8      ErrorLog ${_LOG_DIR}/ptriot_error.log
9      CustomLog ${_LOG_DIR}/ptriot_access.log combined
10 </VirtualHost>
11 # vim: syntax= ts=4 sw=4 sts=4 sr noet
```

Konfigurasi 5.19 Admin.ptriot.com.conf

Admin.ptriot.com.conf

```
1  ## AUTOMATICALY GENERATED BY CARBONIFERO
2  ## SITE : admin.ptriot.com
3  ## IP   : IP_ADDRESS : 8080
4  <VirtualHost *:8080>
5      ServerAdmin sysadmin@ptriot.com
6      ServerName admin.ptriot.com
7      DocumentRoot /var/www/ptriot/public
8      ErrorLog ${_LOG_DIR}/ptriot_error.log
9      CustomLog ${_LOG_DIR}/ptriot_access.log combined
10 </VirtualHost>
11 # vim: syntax= ts=4 sw=4 sts=4 sr noet
```

Setelah *file* konfigurasi dibuat maka langkah selanjutnya adalah mengaktifkan tiga buah konfigurasi yang sudah dibuat, proses aktivasi dapat dilihat seperti pada perintah yang terdapat pada Konfigurasi 5.20 dibawah ini.

Konfigurasi 5.20 Apache Webservice

Configure.sh

```
1  sudo sed -i -e s/None/All/ /etc/2/2.conf
2  sudo a2dissite 000-default.conf
3  sudo a2enmod rewrite
4  sudo a2ensite ptriot.com
5  sudo a2ensite API.ptriot.com
6  sudo a2ensite admin.ptriot.com
7  sudo systemctl reload apache2
```

5.1.3.2 Implementasi Webservice Berbasis NGiNX

Setelah melakukan instalasi *webserver* maka langkah selanjutnya adalah dengan melakukan instalasi *reverse proxy*, *reverse proxy* sendiri akan digunakan untuk memisahkan koneksi antara *file* statis dan proses PHP, untuk *reverse proxy* yang digunakan adalah NGiNX.

Untuk melakukan instalasi NGiNX dapat dilakukan dengan berbagai cara salah satunya melalui repository *apt*. Setelah proses instalasi selesai dilakukan maka diperlukan konfigurasi tambahan yaitu melakukan konfigurasi pemisahan akses yang diatur dalam sebuah *file*. Dikarenakan jumlah *webserver* ada tiga maka *file* konfigurasi yang perlu dikonfigurasi juga berjumlah sama dengan *webserver*, dalam konfigurasi dibawah dilakukan pemisahan akses. Pemisahan akses antara lain jika *request* adalah *file* statis seperti JPG, PNG dll maka akan langsung diproses oleh NGiNX, jika selain itu maka akan diteruskan ke Apache Webserver melalui *port* 8080, *file* konfigurasi dapat dilihat pada potongan kode yang terdapat pada Konfigurasi 5.21 dan 5.22 untuk ptriot.com, kemudian Konfigurasi 5.23 dan 5.24 untuk api.ptriot.com dan yang terakhir pada Konfigurasi 5.25 dan 5.26 untuk admin.ptriot.com. Seluruh bukti implementasi dari konfigurasi ini dapat ditemukan pada Lampiran A.3.2.1 hingga A.3.2.3.

Konfigurasi 5.21 Ptriot.com (bagian 1)

ptriot.com

```
## AUTOMATICALY GENERATED BY CARBONIFERO
## SITE : ptriot.com
## IP   : IP_ADDRESS : 443
server {

        server_name ptriot.com;

        root /var/www/ptriot/public;
        index index.php index.html index.htm;

        location ~*^.+\.(jpg|jpeg|gif|PNG|ico|css|zip
|tgz|gz|rar|bz2|pdf|txt|tar|wav|bmp|rtf|js|flv|swf|html|htm)$ {
                root /var/www/ptriot/public;
        }

        location ~*^!.+\.(jpg|jpeg|gif|PNG|ico|css|zip
|tgz|gz|rar|bz2|pdf|txt|tar|wav|bmp|rtf|js|flv|swf|html|htm)$ {
                if (!-f $request_filename) {
                        return 404;
                }
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header Host $host;
                proxy_set_header X-Forwarded-For
                $proxy_add_x_forwarded_for;
                proxy_pass http://127.0.0.1:8080;
                include /etc/nginx/proxy_params;
        }


        location / {
                proxy_pass http://127.0.0.1:8080;
                include /etc/nginx/proxy_params;
        }

        # deny access to .htaccess files, if 's document root
        # concurs with nginx's one
        #

        location ~ /\.ht {
                deny all;
        }


    listen [::]:443 http2 ssl ipv6only=on; # managed by Certbot
    listen 443 http2 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/ptriot.com/fullchain.pem;
# managed by Certbot
    ssl_certificate_key
/etc/letsencrypt/live/ptriot.com/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by
Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by
Certbot

}

server {
    if ($host = ptriot.com) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


```

Konfigurasi 5.22 Ptriot.com (bagian 2)

```
63          listen 80;
64          listen [::]:80;
65
66          server_name ptriot.com www.ptriot.com;
67      return 404; # managed by Certbot
68
68
70  }
```

Konfigurasi 5.23 Api.ptriot.com (bagian 1)

Api.ptriot.com

```
1   ## AUTOMATICALY GENERATED BY CARBONIFERO
2   ## SITE : API.ptriot.com
3   ## IP   : IP_ADDRESS : 443
4   server {
5
6           server_name API.ptriot.com;
7
8           root /var/www/ptriot/public;
9           index index.php index.html index.htm;
10
11          location ~*^.+\.(jpg|jpeg|gif|PNG|ico|css|zip
12  |tgz|gz|rar|bz2|pdf|txt|tar|wav|bmp|rtf|js|flv|swf|html|htm)$ {
13                  root /var/www/ptriot/public;
14          }
15
16          location ~*^!.+\.(jpg|jpeg|gif|PNG|ico|css|zip
17  |tgz|gz|rar|bz2|pdf|txt|tar|wav|bmp|rtf|js|flv|swf|html|htm)$ {
18                  if (!-f $request_filename) {
19                          return 404;
20                  }
21                  proxy_set_header X-Real-IP $remote_addr;
22                  proxy_set_header Host $host;
23                  proxy_set_header X-Forwarded-For
24                  $proxy_add_x_forwarded_for;
25                  proxy_pass http://127.0.0.1:8080;
26                  include /etc/nginx/proxy_params;
27          }
28
29
30          location / {
31                  proxy_pass http://127.0.0.1:8080;
32                  include /etc/nginx/proxy_params;
33          }
34
35          # deny access to .htaccess files, if 's document root
36          # concurs with nginx's one
37          #
38
39          location ~ /\.ht {
40                  deny all;
41          }
42
43
44      listen [::]:443 http2 ssl ipv6only=on; # managed by Certbot
45      listen 443 http2 ssl; # managed by Certbot
46      ssl_certificate
47  /etc/letsencrypt/live/api.ptriot.com/fullchain.pem; # managed by
48  Certbot
```

Konfigurasi 5.24 Api.ptriot.com (bagian 2)

```
49      ssl_certificate_key
50  /etc/letsencrypt/live/ptriot.com/privkey.pem; # managed by Certbot
51      include /etc/letsencrypt/options-ssl-nginx.conf; # managed by
52  Certbot
53      ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by
54  Certbot
55
56  }
57
58  server {
59      if ($host = API.ptriot.com) {
60          return 301 https://$host$request_uri;
61      } # managed by Certbot
62
63
64          listen 80;
65          listen [::]:80;
66
67          server_name API.ptriot.com www.api.ptriot.com;
68      return 404; # managed by Certbot
68
70
71  }
```

Konfigurasi 5.25 Admin.ptriot.com (bagian 1)

Admin.ptriot.com

```
1   ## AUTOMATICALY GENERATED BY CARBONIFERO
2   ## SITE : admin.ptriot.com
3   ## IP   : IP_ADDRESS : 443
4   server {
5
6           server_name admin.ptriot.com;
7
8           root /var/www/ptriot/public;
9           index index.php index.html index.htm;
10
11          location ~*^.+\.(jpg|jpeg|gif|PNG|ico|css|zip
12  |tgz|gz|rar|bz2|pdf|txt|tar|wav|bmp|rtf|js|flv|swf|html|htm)$ {
13                  root /var/www/ptriot/public;
14          }
15
16          location ~*^!.+\.(jpg|jpeg|gif|PNG|ico|css|zip
17  |tgz|gz|rar|bz2|pdf|txt|tar|wav|bmp|rtf|js|flv|swf|html|htm)$ {
18                  if (!-f $request_filename) {
19                          return 404;
20                  }
21                  proxy_set_header X-Real-IP $remote_addr;
22                  proxy_set_header Host $host;
23                  proxy_set_header X-Forwarded-For
24                  $proxy_add_x_forwarded_for;
25                  proxy_pass http://127.0.0.1:8080;
26                  include /etc/nginx/proxy_params;
27          }
28
29
30          location / {
31                  proxy_pass http://127.0.0.1:8080;
32                  include /etc/nginx/proxy_params;
33          }
34
35          # deny access to .htaccess files, if 's document root
```

Konfigurasi 5.26 Admin.ptriot.com (bagian 2)

```
36          # concurs with nginx's one
37          #
38
39          location ~ /\.ht {
40                  deny all;
41          }
42
43
44      listen [::]:443 http2 ssl ipv6only=on; # managed by Certbot
45      listen 443 http2 ssl; # managed by Certbot
46      ssl_certificate
47  /etc/letsencrypt/live/admin.ptriot.com/fullchain.pem; # managed by
48  Certbot
49      ssl_certificate_key
50  /etc/letsencrypt/live/admin.ptriot.com/privkey.pem; # managed by
51  Certbot
52      include /etc/letsencrypt/options-ssl-nginx.conf; # managed by
53  Certbot
54      ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by
55  Certbot
56
57  }
58
59  server {
60      if ($host = admin.ptriot.com) {
61          return 301 https://$host$request_uri;
62      } # managed by Certbot
63
64
65          listen 80;
66          listen [::]:80;
67
68          server_name admin.ptriot.com www.admin.ptriot.com;
68      return 404; # managed by Certbot
70
71
72  }
```

Setelah *file* selesai dibuat maka proses selanjutnya adalah melakukan aktivasi konfigurasi yang dibuat sebelumnya. Proses tersebut dapat dilakukan melalui perintah yang tercuplik dalam Konfigurasi 5.27 dibawah ini.

Konfigurasi 5.27 Nginx

```
ConfigureNginx.sh
1   Sudo rm /etc/nginx/sites-enabled/default
2   Sudo ln -s /etc/nginx/site-available/ptriot.com /etc/nginx/sites-
3   enabled/ptriot.com
4   Sudo ln -s /etc/nginx/site-available/api.ptriot.com
5   /etc/nginx/sites-enabled/api.ptriot.com
6   Sudo ln -s /etc/nginx/site-available/admin.ptriot.com
7   /etc/nginx/sites-enabled/admin.ptriot.com
8   Sudo systemctl nginx enable
9   Sudo systemctl nginx reload
```

5.1.3.3 Implementasi Webservice Berbasis Redis

Redis akan peneliti gunakan untuk menyimpan *key-value* dari sebuah data temporary dalam sistem *webservice*, data tersebut antara lain data *queue* dan *sessions* yang ada dalam sistem *webservice*.

Proses instalasi redis dapat dilakukan dalam berbagai cara, akan tetapi yang paling *secure* adalah dengan melakukan compile sendiri kode yang disediakan oleh developer. setelah melakukan proses instalasi redis maka perlu dilakukan konfigurasi tambahan yaitu melakukan setting firewall dan menjadikan aplikasi redis sebagai aplikasi startup saat sistem operasi dijalankan, untuk konfigurasi redis dapat menggunakan cuplikan perintah yang terdapat pada Konfigurasi 5.28 dibawah ini.

Konfigurasi 5.28 Redis

```
ConfigureRedis.sh
1  sudo ufw allow 6379
2  sudo systemctl enable redis-server
3  sudo systemctl restart redis-server
```

5.1.3.4 Implementasi Webservice Berbasis Laravel

Laravel akan digunakan dalam sistem ini sebagai *framework* dalam pembangunan *webservice*, hal ini dikarenakan dalam Laravel sudah banyak fitur builtin yang dapat mempermudah proses pengembangan sistem.

Langkah berikutnya adalah melakukan instalasi Laravel, proses instalasi Laravel dapat dilakukan dengan cara melakukan *clone* repository Laravel melalui github yang disediakan oleh Laravel. Setelah instalasi Laravel maka diperlukan konfigurasi tambahan agar Laravel mulai dapat digunakan, yang pertama adalah melakukan konfigurasi lingkungan Laravel *(environment)* dalam *file .env*. Proses tersebut dapat dilakukan seperti pada perintah dalam cuplikan Konfigurasi 5.29, kemudian isi dari konfigurasi *.env* dapat dilihat pada Konfigurasi 5.30 dan 5.31.

Konfigurasi 5.29 Laravel

```
ConfigureLaravel.sh
1  cd /var/www/ptriot
2  sudo -u www-data cp .env.example .env
3  sudo -u www-data PHP artisan key:generate
4  sudo -u www-data nano .env
```

Konfigurasi 5.30 .Env (bagian 1)

```
.env
1  APP_NAME=Ptriot
2  APP_ENV=production
3  APP_KEY=base64:thisTokenIsSecRetSoThisISONlyADummy
4  APP_DEBUG=false
5  APP_URL=https://ptriot.com
6  APP_DOMAIN=ptriot.com
7  FIREWALL_ENABLED=false
8
9  LOG_CHANNEL=daily
10
11 DB_CONNECTION=mongo
12 DB_HOST=127.0.0.1
13 DB_PORT=27017
14 DB_DATABASE=database_Name_Remain_Secret
15 DB_USERNAME=OffCourse_ThisIs_Secret
16 DB_PASSWORD=ThisIsAbsoluteLy_SEcret
17
```

Konfigurasi 5.31 .Env (bagian 2)

```
18  BROADCAST_DRIVER=redis
19  CACHE_DRIVER=redis
20  QUEUE_CONNECTION=redis
21  SESSION_DRIVER=redis
22  SESSION_LIFETIME=120
23
24  REDIS_HOST=127.0.0.1
25  REDIS_PASSWORD=null
26  REDIS_PORT=6379
27
28  MAIL_DRIVER=smtp
29  MAIL_HOST=mail.ptriot.com
30  MAIL_PORT=8888
31  MAIL_USERNAME=no-reply@ptriot.com
32  MAIL_FROM_ADDRESS=no-reply@ptriot.com
33  MAIL_FROM_NAME=Ptriot
33  MAIL_PASSWORD=password_secret_nothing_here_but_example
34
45  PUSHER_APP_ID=
45  PUSHER_APP_KEY=
46  PUSHER_APP_SECRET=
47  PUSHER_APP_CLUSTER=mt1
48
49  MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
50  MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
51
52  JWT_SECRET=thisTokenIsSecRetSoThisISONlyADummy
```

Bukti dari implementasi konfigurasi ini dapat dilihat dalam Lampiran A.3.3.1.

5.2 Implementasi Aplikasi

Komponen aplikasi adalah komponen paling vital dalam penelitian ini, dikarenakan melalui aplikasi inilah penelitian ini di dasarkan. Aplikasi ini akan berjalan diatas komponen infrastruktur yang telah diimplementasikan sebelumnya. Melalui aplikasi inilah infrastruktur yang telah dibangun akan dimaksimalkan penggunaannya. Aplikasi yang dibangun sendiri akan berjalan dalam bahasa pemrograman Python dan PHP, sehingga perlu dilakukan *coding* ulang dari penelitian sebelumnya.

5.2.1 Implementasi *API Internal IoT Data Storage*

Api Internal IoT data storage sendiri adalah komponen yang bertanggung jawab pada infrastruktur IoT data storage yang telah dijelaskan sebelumnya pada poin 5.1.1 *API* ini dibangun menggunakan bahasa pemrograman Python. Hal ini dikarenakan *API* ini sendiri akan sering digunakan oleh komponen *IoT data analytic* yang juga dibangun menggunakan bahasa pemrograman Python. Untuk implementasinya peneliti membuat dua buah *API* yang terdiri Driver untuk MongoDB dan Driver untuk Apache Hadoop.

API pertama yaitu driver MongoDB yang akan digunakan untuk melakukan komunikasi antara aplikasi MongoDB dengan sistem yang dibangun. Driver ini akan diimplementasikan dalam bahasa pemrograman Python, untuk pseudocode sendiri dapat dilihat pada Pseudocode 5.1.

Pseudocode 5.1 *Driver* MongoDB

Driver Mongo

```
1   class MongoController
2
3       var client
4       var method
5
6       method constructor(str host, int port, str database)
7           create connection with property (host, port, database)
8       end method
9
10      method GETCollection(str collection)
11          return data_from(collection)
12      end method
13
14      method insertOneDocument(str collection, object data)
15          return insert_data(collection, data)
16      end method
17
18      method findOneDocument(str collection, arr find, arr select)
19          return insert_find_data(collection, data, select)
20      end method
21
22      method updateOneDocument(str collection, arr find, object
23  update)
24          return update_one_data(collection, find, update)
25      end method
26
27      method deleteOneDocument(str collection, arr find)
28          return delete_data(collection, data)
29      end method
30
31      method insertBulkDocument(str collection, object data)
32          return insert_many_data(collection, data)
33      end method
34
35      method findBulkDocument(str collection, arr find, arr select,
36  arr sort, int limit)
38          return find_many_data(collection, find, select, sort, limit)
39      end method
40
41      method updateBulkDocument(str collection, arr find, object
42  update)
43          return update_many_data(collection, find, update)
44      end method
45
46      method deleteBulkDocument(str collection, arr find)
47          return delete_many_data(collection, find)
48      end method
49  end class
```

API kedua adalah *driver* Apache Hadoop yang akan digunakan untuk melakukan komunikasi antara aplikasi Apache Hadoop dengan sistem yang dibangun. *Driver* ini dibangun dengan dasar agar memudahkan proses berjalannya aplikasi, sehingga perlu dilakukan implementasi lanjutan daripada *library* pydoop yang telah ada sebelumnya. *Driver* ini akan diimplementasikan dalam bahasa pemrograman Python, untuk pseudocode sendiri dapat dilihat pada Pseudocode 5.2.

Pseudocode 5.2 *Driver* Apache Hadoop

Driver Apache Hadoop

```
1   class Apache HadoopController
2
3       var hdfsobj
4
5       method constructor(str host, int port)
6           create connection with property(host, port)
7       end method
8
9       method pwd()
10          return working_directory()
11      end method
12
13      method listDirectory(str path)
14          return list_directory(path)
15      end method
16
17      method delete(str path)
18          delete_file(path)
19      end method
20
21      method put(str source, str destination)
22          put_file(source, destination)
23      end method
24
25      method copyFile(str source, str destination)
26          copy_file(source, destination)
27      end method
28
29      method write(str path, object data)
30          write data to path
31      end method
32  end class
```

5.2.2 Implementasi *API Internal IoT data analytic*

Selain infrastruktur IoT data analytic seperti Apache Spark dan PHP-ML yang telah dikonfigurasikan dalam poin 5.1.2, maka diperlukan sebuah *API* tambahan yang perlu dibuat sendiri oleh peneliti, fungsi dari *API* tambahan ini adalah untuk mempermudah proses analitik data yang akan dijalankan, pada *API* yang dibangun ini juga akan menggunakan *API* yang dibuat sebelumnya seperti driver MongoDB dan driver Apache Hadoop.

Untuk *API* tambahan ini penulis kodekan menggunakan bahasa pemrograman Python, hal ini dikarenakan *API* ini akan sering digunakan oleh program yang menggunakan bahasa pemrograman Python, sehingga akan lebih efisien jika dibuat menggunakan bahasa pemrograman ini, untuk pseudocode dari *driver analytic* terdapat pada Pseudocode 5.3, 5.4 dan 5.5.

Pseudocode 5.3 *Driver analytic* (bagian 1)

Driver Analytic

```
1   class AnalyticController
2
3       var DATABASE_IP
4       var DATABASE_PORT
5
6       var STORAGE_IP
```

Pseudocode 5.4 *Driver analytic* (bagian 2)

```
7       var STORAGE_PORT
8
9       var DATABASE_NAME
10      var COLLECTION_NAME
11      var USER_COLLECTION_NAME
12      var TOPIC_COLLECTION_NAME
13
14      var id
15      var user_database
16      var user_topic
17
18      method constructor(int id)
19          create connection with property (DATABASE_IP,
20  DATABASE_PORT, DATABASE_NAME)
21          initialize id with data from User variable _id
22          initialize user_database with data from user variable
23  database_name
24          initialize user _topic with data from topic variable
25  nicename
26      end method
27
28      method GETInfo()
29          return info from connection
30      end method
31
32      method GETUser()
33          return user info from connection
34      end method
35
36      method GETTopic()
38          return topic info from connection
39      end method
40
41      method update()
42          return update analytic data through connection
43      end method
44
45      method remove()
46          return delete analytic data through connection
47      end method
48
49      method updateStatus(str status)
50          return update analytic status with 'status' value through
51  connection
52      end method
53
54      method updateOutput(object output)
55          return update analytic output with 'output' value through
56  connection
57      end method
58
59      method dumpSample(str database, arr find, arr select, int
60  limit)
61          create connection using property (database)
62          GET topic data and dump in list through connection
63          convert list to file
64          save dump file to Hadoop
65          return filename from file
66      end method
67
68      method dumpInput(object data)
69          convert data to JSON
70          convert JSON to file
```

Pseudocode 5.5 *Driver analytic* (bagian 3)

```
71          save dump file to Hadoop
72          return filename from file
73      end method
74
75      method dumpTrain(str database, arr find, arr select, int
76  limit)
77          create connection using property (database)
78          GET training data and dump in list through connection
79          convert list to file
80          save dump file to Hadoop
81          return filename from file
82      end method
83
84      method dumpOutput(object data)
85          convert data to JSON
86          convert JSON to file
87          save dump file to Hadoop
88          return filename from file
89      end method
90  end class
```

Selain *API* tersebut diperlukan juga sebuah daemon program yang akan menjalankan atau men-*trigger* Apache Spark untuk melakukan pekerjaanya, program ini akan mengecek di *database* MongoDB apakah ada pekerjaan yang belum dilakukan, dan jika ada maka pekerjaan tersebut akan diantrikan.

Untuk daemon ini juga dikodekan menggunakan bahasa pemrograman Python, hal ini dikarenakan *daemon* ini akan menggunakan kembali beberapa fungsi dari *API* yang telah dibuat sebelumnya, pseudocode dari daemon sendiri dapat ditemukan dalam Pseudocode 5.6 dan 5.7 dibawah ini.

Pseudocode 5.6 *Daemon* Apache Spark (bagian 1)

```
Daemon Apache Spark
1   if program == "__main__"
2
3       if length of argument < 1
4           output "Usage : Python SparkDaemon.py"
5           exit the process
6
7       output [Info] System Running and ready to start Apache Spark
8   data"
9       while true
10          DATABASE_IP = "127.0.0.1"
11          DATABASE_PORT = 27017
12
13          DATABASE_NAME = 'ptriot'
14          COLLECTION_NAME = 'analytic'
15          NUMBER_OF_DATA = 1
16
17          try
18              connect to Database and save connection to Database
19              GET data from database and save to logs
20              convert logs to JSON
21
22              initialize program and id with ""
23              if length of logs == 1
24                  if class in logs "Linear Regression"
25                      program = "LinearRegression.py"
```

Pseudocode 5.7 *Daemon* Apache Spark (bagian 2)

```
26                      GET id from logs and save to id
27                      output"[Info] Detecting process under id #”
28                  else if class in logs “Naïve bayes”
29                      program = "NaiveBayes.py"
30                      GET id from logs and save to id
31                      output"[Info] Detecting process under id #”
32                  else
33                      pass
34
35                  sysCall="Spark-submit "+program+" "+id
36                  call program using parameter from sysCall
37              else
38                  output "[Info] Queue is Empty"
39
40              sleep the program for 3 seconds
41          exception Keyboard
42              output '[Info] Program is killed by user'
43              try
44                  exit the process
46              exception SystemExit
47                  force exit
48          exception
49              output "[Info] Crash"
```

5.2.3 Implementasi Modul *Data Analytic*

Selain mengkonfigurasikan dan membuat *API* maka diperlukan juga sebuah modul analitik yang akan menjalankan proses analitik dalam sistem, dalam penelitian kali ini akan terdapat dua buah Machine Learning yang masing-masing akan diimplementasikan dalam PHP-ML untuk proses analitik yang ringan *(lightweight)* dan juga dalam Apache Spark untuk proses analitik yang berat *(intense)*

5.2.3.1 Implementasi Prediksi Data menggunakan Least Square

Machine learning pertama yang diimplementasikan adalah Least Square yang merupakan metode dalam *Linear Regression* yang dapat digunakan untuk melakukan proses prediksi sebuah data berdasarkan sebuah set data yang tersedia.

Least Square atau lebih tepatnya *Linear Regression* digunakan untuk proses prediksi data dikarenakan metode ini sangat cocok untuk data-data yang independent antar variabelnya, sehingga dalam sistem yang heterogen ini maka pendekatan ini sangatlah cocok dan mampu memberikan hasil yang lebih baik daripada metode lainnya, selain itu metode ini juga sangat ringan untuk dijalankan, sehingga sangat cocok untuk diimplementasikan dalam proses penelitian ini yang masih terkendala sumberdaya dari sistem.

Least Square sendiri akan peneliti implementasikan dalam dua bentuk yaitu dalam PHP-ML untuk data yang berukuran kecil atau dibawah satu juta (1.000.000) *entry* data yang pseudocode dapat dilihat dalam Pseudocode 5.8, dan Apache Spark untuk data diatas itu yang pseudocode dapat dilihat dalam Pseudocode 5.9 dan 5.10.

Pseudocode 5.8 Least Square pada PHP-ML

```
Least Square PHP-ML
1   method leastSquare(str topic, str targetCol, arr sampCol, arr
2   predict)
3           Create entry in analytic collections
4           If structure of topic != "non-structured"
5               Validate the user input
6               If error in validation
7                  Update entry with done status
8               else
9                   try
10                      if targetCol is valid with schema of topic
11                              if targetCol available in sampCol
12                                  update entry with done status
13                              else
14                                  populate sample from database
15                                  populate targets from database
16                                  train the least square models
17                                  predict the data using model
18                                  save predict to entry with done
19  status
20
21
22                          else
23                             update entry with done status
24                      exception
25                              update entry with error status
26          else
27              update entry with done status
28
29          save the entry
30          return entry
31  end method
```

Pseudocode 5.9 Least Square pada Apache Spark (bagian 1)

```
Least Square Apache Spark
1   if program == "__main__"
2
3       if length of argument < 2
4           output "Usage : Python LinearRegression.py <id>"
5           exit the process
6       try
7
8           APACHE HADOOP_USER = "faathin"
9           GET username information and save to USERNAME
10          GET topic information and save to TOPIC
11          GET target information and save to TARGET
12          GET header information and save to HEADER
13
14          Populate selected header from header and save to select
15          Dump sample information to Apache Hadoop and save filename
16  to filename
17          GET sample information and save to HEADER
18          Update process status to "queue"
19
20          Initialize Apache Spark process
21          Update process status to "process"
22          Create train dataframe for Apache Spark from sample data
23          Create predict dataframe for Apache Spark from predicted
24  data
25          Train Apache Spark Least Square with train dataframe
```

Pseudocode 5.10 Least Square pada Apache Spark (bagian 2)

```
26          Fit (Predict) the predict dataframe with training model
27
28          Save the predict output to database
29          Update process output from result of predict data
30          Update process status to "done"
31
32      exception
33          Update process status to "error"
```

5.2.3.2 Implementasi Classifier Data menggunakan Naïve Bayes

Machine learning berikutnya yang diimplementasikan adalah *Naïve Bayes* yang merupakan metode yang dapat digunakan untuk melakukan proses klasifikasi sebuah data berdasarkan sebuah set data yang tersedia.

Naïve Bayes digunakan untuk proses klasifikasi data dikarenakan metode ini sangat cocok untuk data-data yang independent antar variabelnya. Sehingga dalam sistem yang heterogen ini maka pendekatan ini sangatlah cocok dan mampu memberikan hasil yang lebih baik daripada metode lainnya. Selain itu metode ini juga sangat ringan untuk dijalankan, sehingga sangat cocok untuk diimplementasikan dalam proses penelitian ini yang masih terkendala sumberdaya dari sistem.

Naïve Bayes sendiri akan peneliti implementasikan dalam dua bentuk yaitu dalam PHP-ML untuk data yang berukuran kecil atau dibawah serratus ribu (100.000) *entry* data yang pseudocode dapat dilihat dalam Pseudocode 5.11, dan Apache Spark untuk data diatas itu yang pseudocode dapat dilihat dalam Pseudocode 5.12.

Pseudocode 5.11 Naïve bayes pada PHP-ML

```
Pseudocode Naïve bayes PHP-ML
1   method naiveBayes(topic topic, arr predicts)
2           Create entry in analytic collections
3           If structure of topic != "non-structured"
4              try
5                   initialize array samples
6                   initialize array targets
7
8                   populate samples from database
9                   populate targets from database
10
11                  train the Naïve bayes model
12                  predicts class of targets using Naïve bayes model
13                  save predict to entry with done status
14
15              exception
16                    update entry with error status
17
18          else
19              update entry with done status
20
21          save the entry
22          return entry
23  end method
```

Pseudocode 5.12 Naïve bayes pada Apache Spark

```
Pseudocode Naïve bayes Apache Spark
1   if program == "__main__"
2
3       if length of argument < 2
4           output "Usage : Python NaiveBayes.py <id>"
5           exit the process
6       try
7           APACHE HADOOP_USER = "faathin"
8           GET username information and save to USERNAME
9           GET topic information and save to TOPIC
10          GET target information and save to TARGET
11          GET header information and save to HEADER
12          GET predict information and save to PREDICT
13
14          Dump train information to Apache Hadoop and save filename to
    FILENAME
15          Update process status to "queue"
16
17          Initialize Apache Spark process
18          Update process status to "process"
19          Create train dataframe for Apache Spark from sample data
20
21          Create predict dataframe for Apache Spark from predicted
22  data
23          Create numerical class index from actual class
24
25          Train Apache Spark NaiveBayes with train dataframe
26          Fit (Predict) the predict dataframe with training model
27          Change numericall class from predict result to actual class
28
29          Save the predict output to database
30          Update process output from result of predict data
31          Update process status to "done"
32
33      exception
34          Update process status to "error"
```

5.2.4 Implementasi Menyimpan Data IoT

Untuk melakukan penyimpanan data IoT diperlukan sebuah *API* endpoint yang bertugas sebagai jalur masuk dari data ke dalam sistem IoT. Sebagai layaknya jalur masuk, di dalam penyimpanan data IoT ini juga terdapat beberapa filter atau penapis yang berfungsi untuk menyeleksi data yang masuk sesuai topik yang didaftarkan dan juga parameter lainnya.

Dalam melakukan penapisan atau seleksi beberapa parameter yang dilihat adalah ukuran data, *form*at data, struktur data dan kesesuaian data dengan data sebelumnya, sehingga dipastikan bahwa data yang masuk harus sesuai dengan standar yang telah ditetapkan oleh sistem. Selain itu setelah diseleksi juga akan dilakukan penentuan lokasi penyimpanan data. Penentuan ini hanya dilakukan saat data pertama kali masuk, setelah itu maka data akan otomatis dimasukkan ke lokasi yang telah ditetapkan sebelumnya. Untuk lebih lengkapnya dapat dilihat dalam Pseudocode 5.13.

repository.ub.ac.id

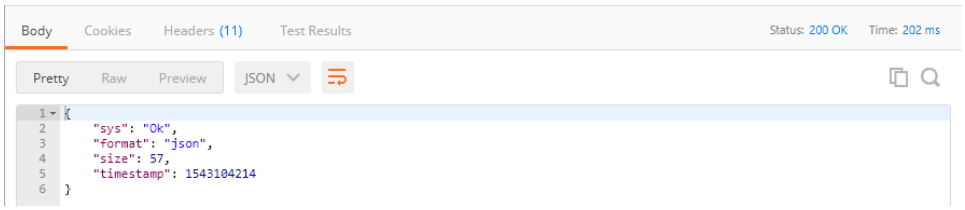
Pseudocode 5.13 Menyimpan data IoT

Menyimpan data IoT (TopicController.php)

```
1  method store(req request, str topic)
2       if topic available in system
3            if topic structure is not non-structured and topic is  not
4  authorize to add data
5                 return JSON with package limit info (entry)
6            else if topic is not authorize to add data
7                 return JSON with package limit info (MB)
8
9            if input data not valid
10                return JSON with validity info
11           else if device not authorized to write in topic
12                return JSON with not authorized info
13           else
14                GET data format info and save to format
15                GET data size info and save to size
16
17                if format and topic not corresponding
18                     return JSON with not corresponding info
19
20                if topic structure is structured
21                     if(data is not fit with topic structure)
22                          return JSON with not fit info
23
24                if format is binary
25                     if(data is the first entry of topic)
26                          set topic storage to gridfs
27
28                     if topic storage is gridfs
29                          save data to gridfs collection under topic
30 name
31                     else
32                          save data to MongoDB collection under topic
33 name
34                else if format is raw
35                     if data is the first entry of topic and size >
36 100 Kb
37                          set topic storage to gridfs
38                     else if data is the first entry of topic
39                          set topic storage to mongo
40
41                     if size>100 Kb and topic structure is "non-
42 structured" or topic storage is gridfs"
43                          save data to gridfs collection under topic
44 name
45                     else
46                          save data to MongoDB collection under topic
47 name
48                else
49                     if data is the first entry of topic
50                          set topic storage to mongo
51
52                     if format is JSON
53                         save data to MongoDB collection under topic
54 name
55                     else if format is XML
56                         convert data from XML to JSON
57                         save data to MongoDB collection under topic
58 name
59                return JSON with info that sucessfuly saved
60       else
61            return JSON with info that not sucessfuly saved
62  end method
```

UNIVERSITAS BRAWIJAYA

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan aplikasi Postman untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.4 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.4 Menyimpan data IoT

5.2.5 Implementasi Menampilkan Daftar Topik Terautorisasi

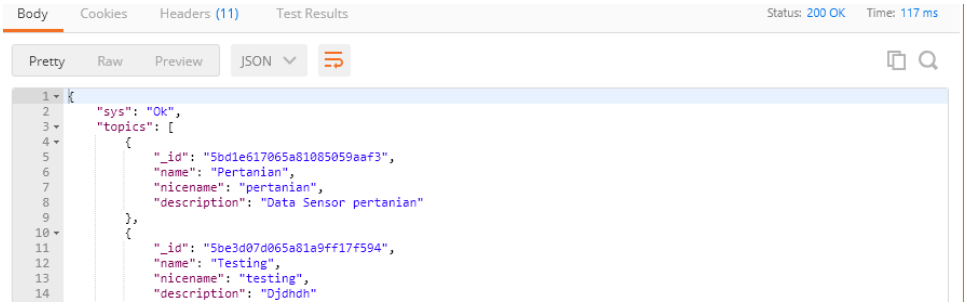
Untuk mempermudah perangkat IoT mengetahui topik yang dapat dia akses maka sistem menyediakan sebuah fitur untuk mengetahui daftar topik beserta informasi dasar dari topik. Dengan kemampuan ini maka perangkat tidak akan melakukan kesalahan pengiriman data.

Sistem pertama-tama akan mengecek apakah pengguna sudah terautentikasi menggunakan JWT, jika sudah maka sistem akan mencoba mengekstrak informasi JWT guna mendapatkan informasi dari perangkat. setelah informasi didapat maka sistem akan mengambil informasi daftar topik terautorisasi beserta informasi dari perangkat. Untuk lebih jelasnya dijelaskan dalam Pseudocode 5.10 yang ada dibawah ini.

Pseudocode 5.14 Menampilkan daftar topik terautorisasi

Menampilkan Daftar Topik Terautorisasi (TopicController.php)	
1	**method** index()
2	**try**
3	**if** user is authenticated with JWT
4	GET device information with authorized topics
5	**return** JSON with topics info
6	**exception**
7	**return** JSON with error info
8	**end method**

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan aplikasi Postman untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.5 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.5 Menampilkan daftar topik terautorisasi

5.2.6 Implementasi Menampilkan Data dari Topik

Menampilkan data dari topik adalah sebuah fitur untuk perangkat IoT. Fitur ini memungkinkan perangkat IoT untuk mendapatkan data dari sistem dalam bentuk yang sederhana dan universal seperti JSON.

Dalam fitur ini pertama-tama sistem akan mengecek apakah dalam *request* tersebut pengguna memiliki informasi *limit, skip* dan jika ada apakah informasi tersebut adalah bernilai numerik. Jika bukan maka akan di inisiasi oleh sistem menjadi bernilai 100. Fungsi dari informasi ini adalah untuk menentukan Batasan dalam proses query terhadap database data, kemudian sistem juka akan mengambil informasi tentang *filter, projection dan sort* dari data. Fungsi dari informasi ini adalah untuk membentuk hasil keluaran menjadi sesuai diinginkan oleh pengguna. Untuk lebih jelasnya bisa dipahami dalam pseudocode yang berada dalam Pseudocode 5.15 dan 5.16.

Pseudocode 5.15 Menampilkan data dari topik (bagian 1)

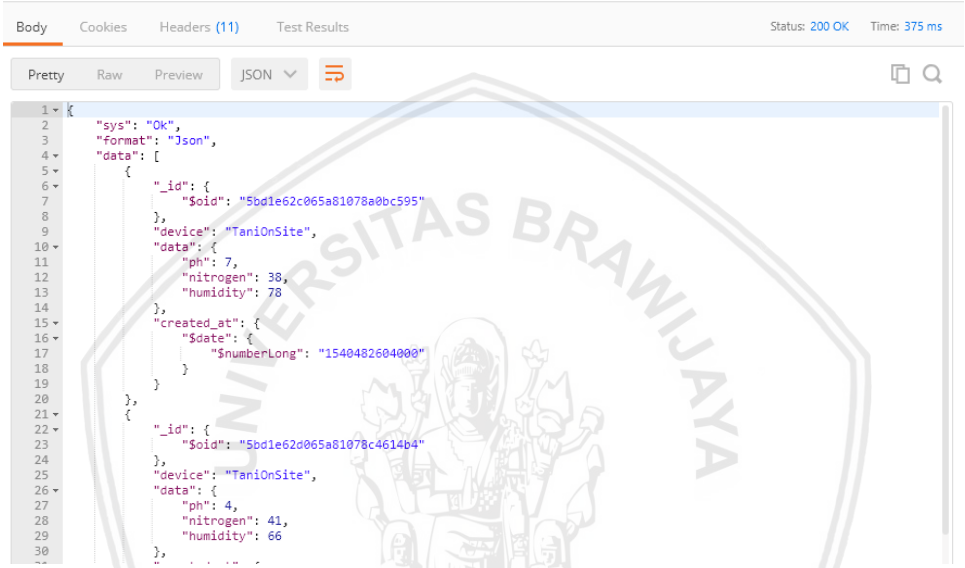
Menampilkan Data dari Topik (TopicController.php)

```
1   method show(Request request, str topic, int limit default 100, int
2   skip default 0)
3
4         if limit is not numeric
5               set limit value to 100
6
7         if skip is not numeric
8               set skip value to 0
9
10        if request with filter
11              save filter to array filter
12        else
13              initialize filter to array with no entry
14
15        if request with projection
16              save projection to array projection
17        else
18              initialize projection to array with no entry
19
20        if request with sort
21              save projection to array sort
22        else
23              initialize sort to array with no entry
24
25        if input is not valid
26              return JSON with validity info
27        else if device not authorized to write in topic
28              return JSON with not authorized info
```

Pseudocode 5.16 Menampilkan data dari topik (bagian 1)

```
29        else
30              GET topic data with option of limit, skip, filter,
31  projection and sort
32              return JSON with topic data
33  end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan aplikasi Postman untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.6 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.6 Menampilkan data dari topik

5.2.7 Implementasi Menampilkan Detail Data dari Topik

Menampilkan detail data dari topik merupakan kelanjutan dari fitur sebelumnya menampilkan data dari topik. Jika pada fitur sebelumnya data akan ditunjukkan secara keseluruhan di dalam topik, maka dalam fitur ini hanyalah satu data saja yang ditampilkan yaitu data yang dipilih.

Dalam pengimplementasian fitur ini pertama kali sistem akan mengecek apakah *input* dari pengguna valid, input yang dimaksud adalah nama dari topik dan id dari data yang dicari. Kemudian setelah itu akan dicek apakah perangkat tersebut memiliki otorisasi untuk mengakses topik tersebut. Jika iya maka detail dari data akan ditampilkan. Untuk lebih jelasnya dapat menampilkan pada pseudocode yang disediakan dalam Pseudocode 5.17 dan 5.18.

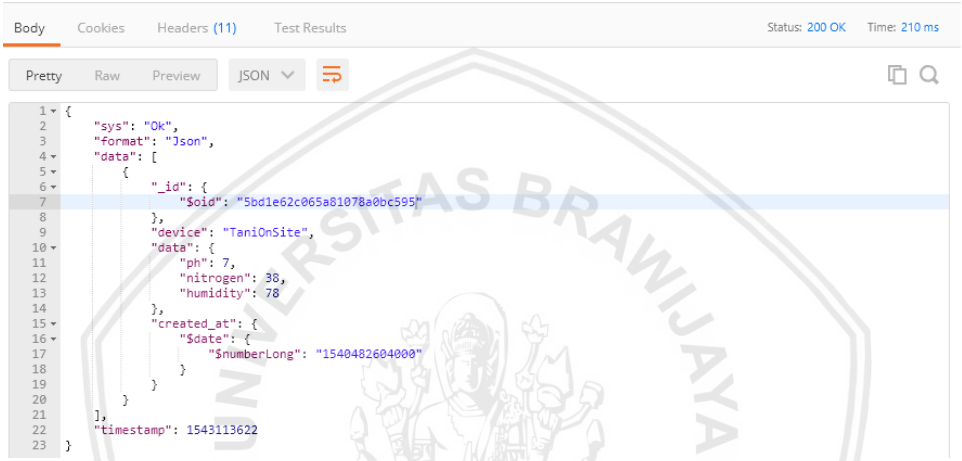
Pseudocode 5.17 Menampilkan detail data dari topik (bagian 1)

```
Menampilkan Detail Data Dari Topik (TopicController.php)
1   method showDetail(str topic,str id)
2         if input is not valid
3               return JSON with validity info
4         else if device not authorized to read in topic
5               return JSON with not authorized info
6         else
```

Pseudocode 5.18 Menampilkan detail data dari topik (bagian 2)

```
7            GET topic data with option of id
8            if topic data is not null
9                 return JSON with topic data
10           else
11                return JSON with not found info
12 end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan aplikasi Postman untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.7 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.7 Menampilkan detail data dari topik

5.2.8 Implementasi Menampilkan Daftar Perangkat IoT

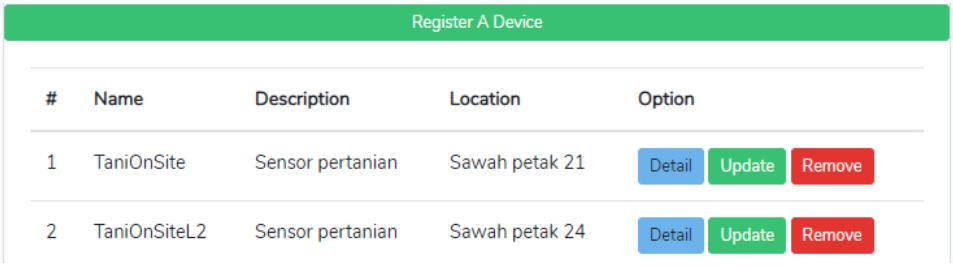
Menampilkan daftar perangkat IoT merupakan sebuah komponen dalam fitur manajemen perangkat IoT. Dalam komponen ini dapat dilihat perangkat IoT yang sudah didaftarkan oleh pengguna sistem di dalam sistem.

Dalam pengimplementasiannya komponen ini pertama kali hanya akan mengambil informasi dari pengguna yang masuk ke sistem, kemudian setelah itu sistem akan mencoba mengambil semua data dari perangkat yang didaftarkan oleh pengguna sistem tersebut. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.19 dibawah ini.

Pseudocode 5.19 Menampilkan daftar perangkat IoT

```
Implementasi Menampilkan Daftar Perangkat (DeviceController.php)
1  method show(req request){
2       GET datas of device from authorized user
3       return render view with all datas from device
4  end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.8 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.8 Menampilkan daftar perangkat IoT

5.2.9 Implementasi Mendaftarkan Perangkat IoT

Mendaftarkan perangkat IoT merupakan sebuah komponen dalam fitur manajemen perangkat IoT. Dalam komponen ini pengguna dapat mendaftarkan perangkat baru ke dalam sebuah sistem.

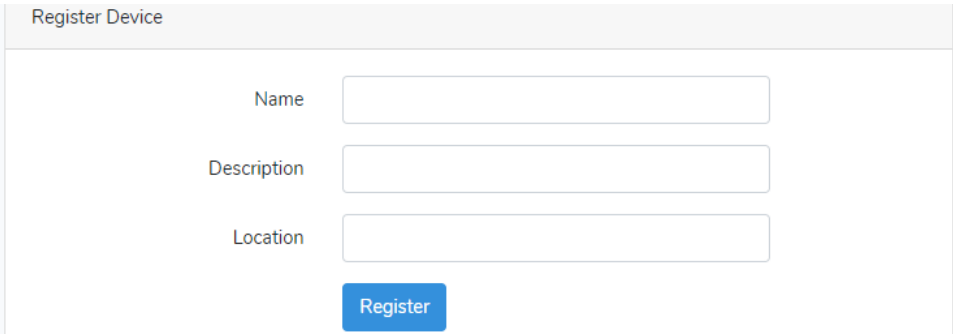
Dalam pengimplementasiannya komponen ini pertama kali akan mengecek apakah *input* yang dimasukkan oleh pengguna valid. Kemudian jika diketahui valid maka proses pendaftaran perangkat akan dilanjutkan. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.14 dibawah ini.

Pseudocode 5.20 Mendaftarkan perangkat IoT

Mendaftarkan Perangkat (DeviceController.php)

```
1  method store(req request){
2       if input request not valid
3            return render back with notification
4       else
5            if user package allowed to register more devices
6                 GET input from request
7                 save input to database
8                 return render view with all datas from device
9            else
10                return render back with notification
11 end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.9 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.9 Mendaftarkan perangkat IoT

5.2.10 Implementasi Menampilkan Informasi Perangkat IoT

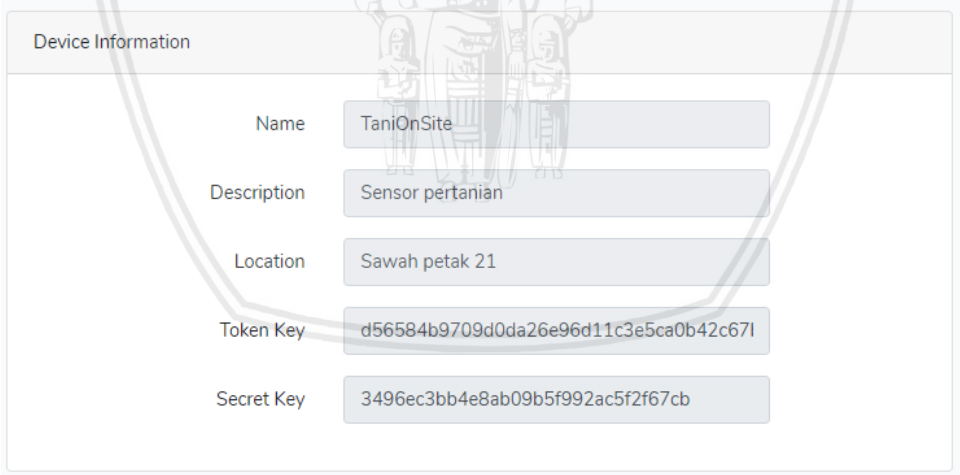
Menampilkan informasi perangkat IoT merupakan sebuah komponen dalam fitur manajemen perangkat IoT. Dalam komponen ini pengguna dapat menampilkan informasi detail dari setiap perangkat termasuk *device token dan device secret key.*

Dalam pengimplementasiannya komponen ini pertama kali akan mengecek apakah perangkat yang dipilih tersedia, kemudian jika diketahui tersedia maka proses menampilkan informasi perangkat akan dilanjutkan. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.21 dibawah ini.

Pseudocode 5.21 Menampilkan informasi perangkat IoT

Menampilkan Informasi Perangkat (DeviceController.php)	
1	**method** view(**req** *request*, **str** nicename){
2	**if** device with selected nicename available
3	**return** render view with all datas from device
4	**else**
5	**return** render view 404 page
6	**end method**

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.10 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.10 Menampilkan informasi perangkat IoT

5.2.11 Implementasi Mengubah Perangkat IoT

Mengubah perangkat IoT merupakan sebuah komponen dalam fitur manajemen perangkat IoT. Dalam komponen ini pengguna dapat mengubah informasi dari setiap perangkat termasuk deskripsi dan lokasi dari perangkat.

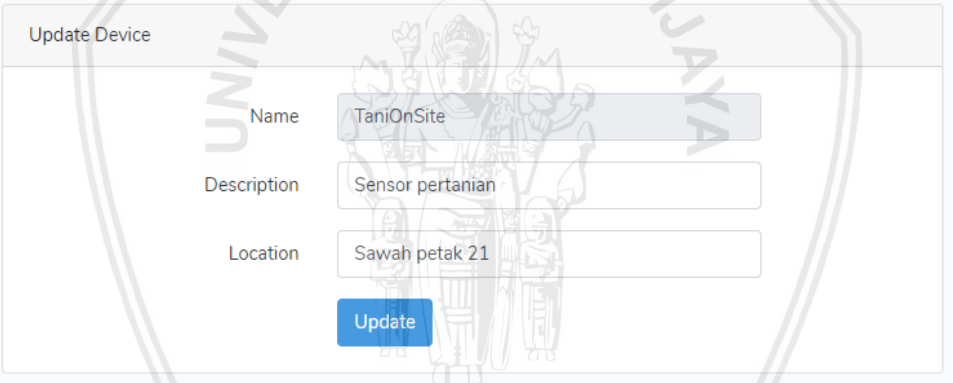
Dalam pengimplementasiannya komponen ini pertama kali akan mengecek apakah perangkat yang dipilih tersedia, kemudian jika diketahui

tersedia maka proses mengubah perangkat akan dilanjutkan. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.22 dibawah ini.

Pseudocode 5.22 Mengubah perangkat IoT

```
Mengubah Perangkat (DeviceController.php)
1   method update(req request, str nicename){
2       if device with selected nicename available
3           if input request not valid
4               return render back with notification
5           else
6               GET input from request
7               GET selected device from database
8               update selected device with input
9               return render view with all datas from device
10      else
11          return render view 404 page
12  end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.11 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.11 Mengubah perangkat IoT

5.2.12 Implementasi Menghapus Perangkat IoT

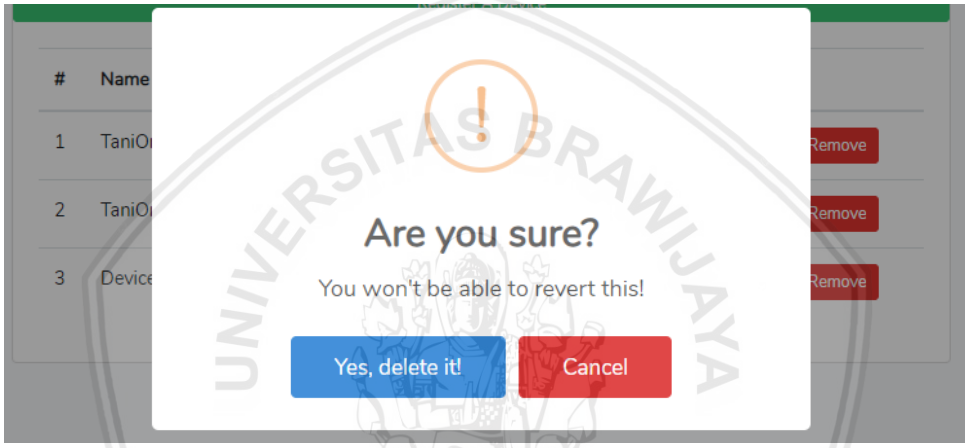
Menghapus perangkat IoT merupakan sebuah komponen dalam fitur manajemen perangkat IoT. Dalam komponen ini pengguna dapat menghapus perangkat yang pengguna miliki dari dalam sistem.

Dalam pengimplementasiannya komponen ini pertama kali akan mengecek apakah perangkat yang dipilih tersedia, kemudian jika diketahui tersedia maka proses menghapus perangkat akan dilanjutkan. Perangkat yang dihapus hanyalah perangkat yang terautorisasi atau dimiliki oleh pengguna tersebut. Jika terdapat perangkat yang memiliki nama sama, akan tetapi berbeda pemilik atau tidak terautorisasi, maka perangkat tidak akan dihapus oleh sistem. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.23 dibawah ini.

Pseudocode 5.23 Menghapus perangkat IoT

```
Menghapus Perangkat (DeviceController.php)
1   method destroy(req request, str nicename){
2       if device with selected nicename available
3           delele selected device from database
4           return render view to device list
5       else
6           return render view 404 page
7   end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.12 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.12 Menghapus perangkat IoT

5.2.13 Implementasi Menampilkan Daftar Topik IoT

Menampilkan daftar topik IoT merupakan sebuah komponen dalam fitur manajemen topik IoT. Dalam komponen ini dapat dilihat perangkat IoT yang sudah didaftarkan oleh pengguna sistem di dalam sistem.

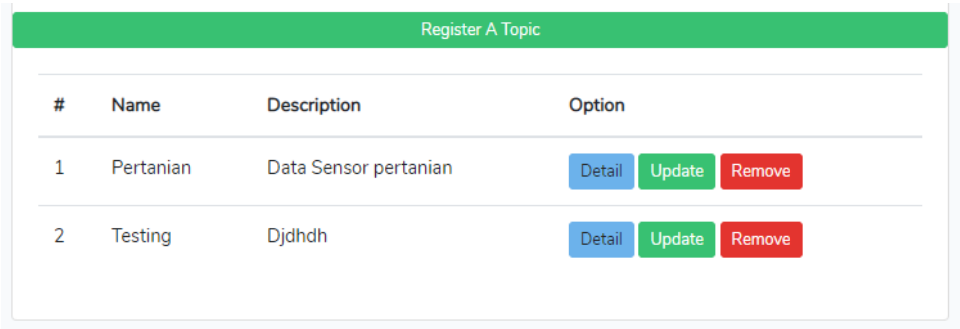
Dalam pengimplementasiannya komponen ini pertama kali hanya akan mengambil informasi dari pengguna yang masuk ke sistem, kemudian setelah itu sistem akan mencoba mengambil semua data dari topik yang didaftarkan oleh pengguna sistem tersebut. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.18 dibawah ini.

Pseudocode 5.24 Menampilkan daftar topik IoT

```
Menampilkan Daftar Topik (TopicController.php)
1   method show(req request){
2       GET datas of topic from authorized user
3       return render view with all datas from topic
4   end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik.

Dan seperti pada Gambar 5.13 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.13 Menampilkan daftar topik IoT

5.2.14 Implementasi Mendaftarkan Topik IoT

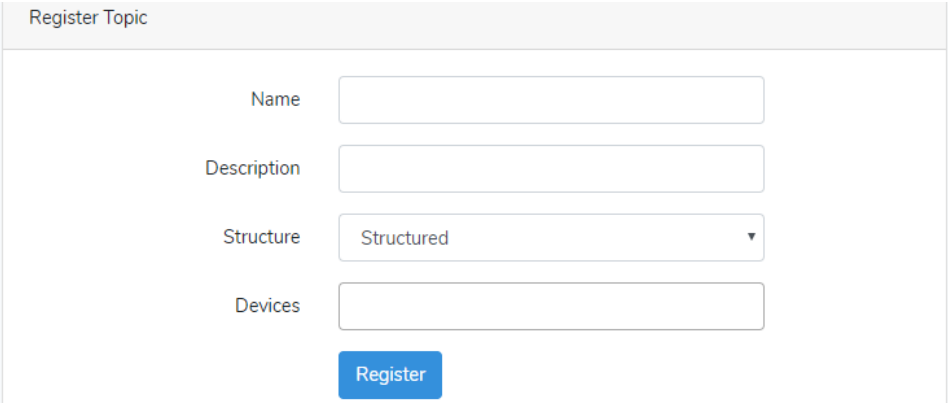
Mendaftarkan topik IoT merupakan sebuah komponen dalam fitur manajemen perangkat IoT. Dalam komponen ini pengguna dapat mendaftarkan topik baru ke dalam sebuah sistem.

Dalam pengimplementasiannya komponen ini pertama kali akan mengecek apakah *input* yang dimasukkan oleh pengguna valid, kemudian jika diketahui valid maka proses pendaftaran topik akan dilanjutkan. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.25 dibawah ini.

Pseudocode 5.25 Mendaftarkan topik IoT

```
Mendaftarkan Topik (TopicController.php)
1   method store(req request){
2       if input request not valid
3           return render back with notification
4       else
5           if user package allowed to register more topics
6               GET input from request
7               save input to database
8               return render view with all datas from topic
9           else
10              return render back with notification
11  end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.14 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.14 Mendaftarkan topik IoT

5.2.15 Implementasi Menampilkan Informasi Topik IoT

Menampilkan informasi topik IoT merupakan sebuah komponen dalam fitur manajemen topik IoT. Dalam komponen ini pengguna dapat menampilkan informasi dari setiap perangkat yang tertaut pada topik ini.

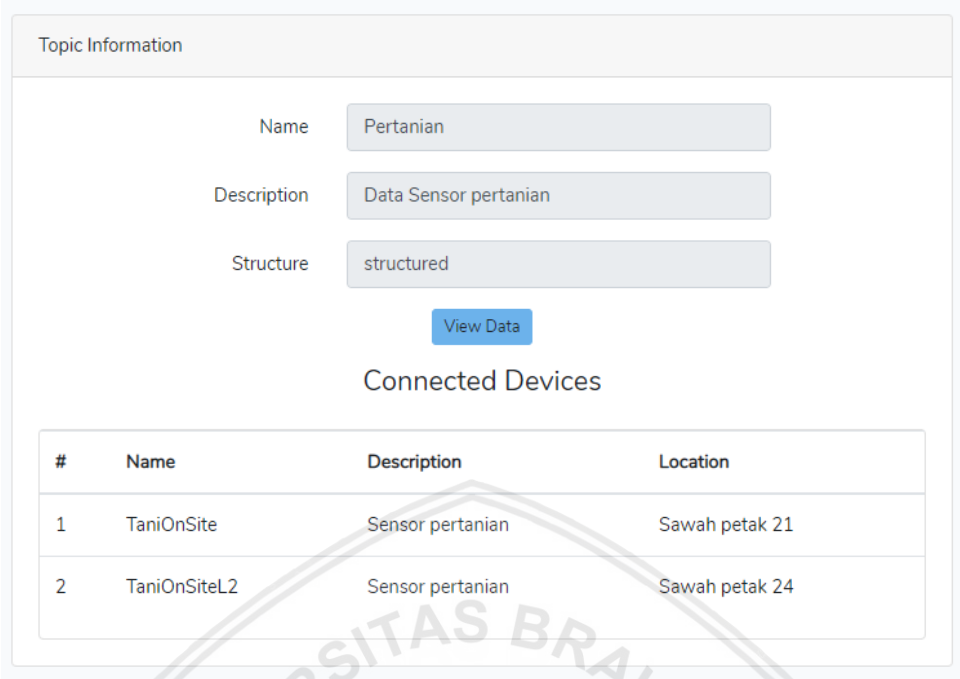
Dalam pengimplementasiannya komponen ini pertama kali akan mengecek apakah topik yang dipilih tersedia, kemudian jika diketahui tersedia maka proses menampilkan informasi topik akan dilanjutkan. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.26 dibawah ini.

Pseudocode 5.26 Menampilkan informasi topik IoT

Menampilkan Informasi Topik (TopicController.php)

```
1  method view(req request, str nicename){
2       if topic with selected nicename available
3            return render view with all datas from topic
4       else
5            return render view 404 page
6  end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.15 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.15 Menampilkan informasi topik IoT

5.2.16 Implementasi Mengubah Topik IoT

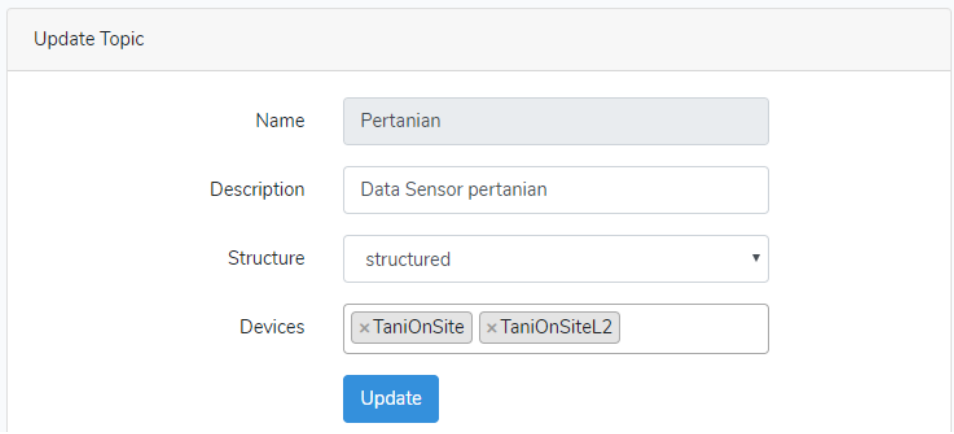
Mengubah topik IoT merupakan sebuah komponen dalam fitur manajemen topik IoT. Dalam komponen ini pengguna dapat mengubah informasi dari setiap topik termasuk deskripsi dari topik dan juga perangkat yang tertaut terhadap topik tersebut.

Dalam pengimplementasiannya komponen ini pertama kali akan mengecek apakah topik yang dipilih tersedia, kemudian jika diketahui tersedia maka proses mengubah topik akan dilanjutkan. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.27 dibawah ini.

Pseudocode 5.27 Mengubah topik IoT

```
Mengubah Topik (TopicController.php)
1   method update(req request, str nicename){
2       if topic with selected nicename available
3           if input request not valid
4               return render back with notification
5           else
6               GET input from request
7               GET selected topic from database
8               update selected topic with input
9               return render view with all datas from topic
10      else
11          return render view 404 page
12  end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.16 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.16 Mengubah topik IoT

5.2.17 Implementasi Menghapus Topik IoT

Menghapus topik IoT merupakan sebuah komponen dalam fitur manajemen topik IoT. Dalam komponen ini pengguna dapat menghapus topik beserta data yang dimiliki oleh topik dari dalam sistem.

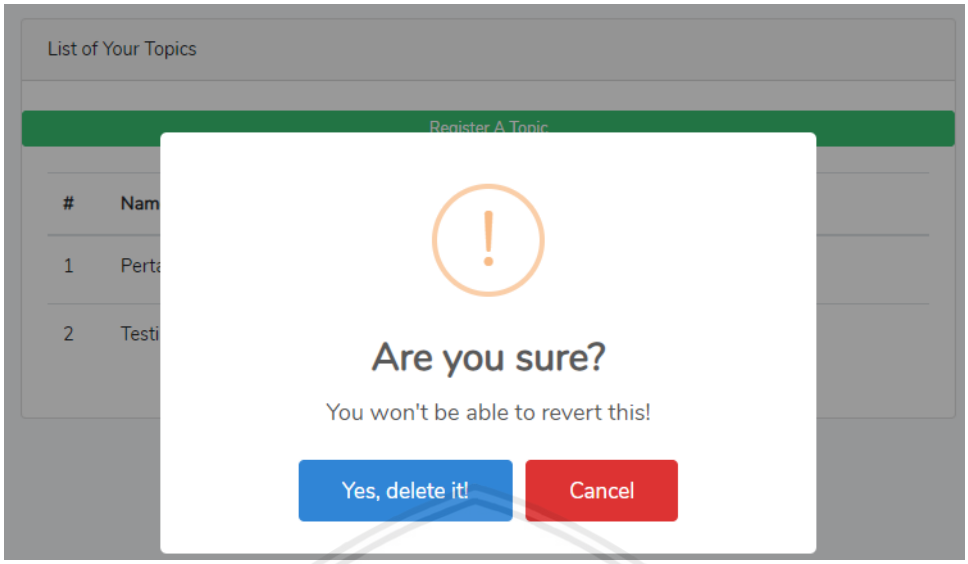
Dalam pengimplementasiannya komponen ini pertama kali akan mengecek apakah topik yang dipilih tersedia, kemudian jika diketahui tersedia maka proses menghapus topik akan dilanjutkan. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.28 dibawah ini.

Pseudocode 5.28 Menghapus topik IoT

Menghapus Topik (TopicController.php)

```
1  method destroy(req request, str nicename){
2       if topic with selected nicename available
3            delele selected topic from database
4            return render view to topic list
5       else
6            return render view 404 page
7  end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.17 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.17 Menghapus topik IoT

5.2.18 Implementasi *Data Explorer*

Data Explorer merupakan sebuah komponen dalam fitur manajemen topik IoT. Dalam komponen ini pengguna dapat menampilkan daftar data yang masuk pada topik ini.

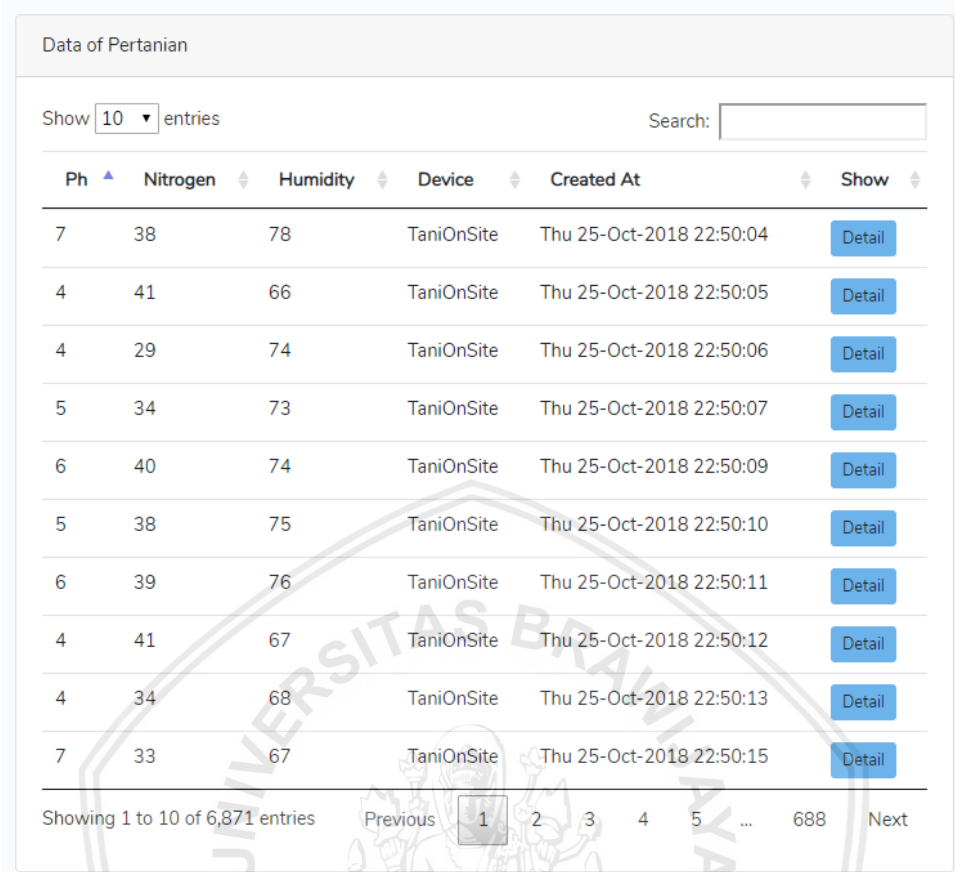
Dalam pengimplementasiannya komponen ini pertama kali akan mengecek apakah topik yang dipilih tersedia, kemudian jika diketahui tersedia maka proses menampilkan daftar data dari topik akan dilanjutkan. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.29 dibawah ini.

Pseudocode 5.29 *Data explorer*

Data Explorer (TopicController.php)

```
1  method explorer(req request, str nicename){
2       if topic with nicename selected is available
3            GET datas from topic
4            return render view with all datas from topic
5       else
6            return render view 404 page
7  end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.18 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.18 *Data explorer*

5.2.19 Implementasi *Data Viewer*

Data Viewer merupakan sebuah komponen dalam fitur manajemen topik IoT. Dalam komponen ini pengguna dapat menampilkan informasi detail dari setiap data yang masuk ke dalam topik yang telah terdaftar dalam sistem.

Dalam pengimplementasiannya komponen ini pertama kali akan mengecek apakah topik yang dipilih beserta data yang dipilih tersedia, kemudian jika diketahui tersedia maka proses menampilkan data dari topik akan dilanjutkan. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.30 dibawah ini.

Pseudocode 5.30 *Data viewer*

Data Viewer (TopicController.php)

```
1  method viewer(req request, str nicename, str id)
2      if topic with selected nicename available and id is valid
3          GET topic data with selected id
4          if data is available
5              return render view with topic data
6      return render view with 404 page
7  end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik.

Seperti terlihat pada Gambar 5.19 yaitu tampilan *data viewer* untuk data berbentuk JSON dan 5.20 yaitu tampilan *data viewer* untuk data berbentuk binary dapat dilihat bahwa sistem dapat berjalan dengan lancar.

Gambar 5.19 *Data viewer* (JSON)

Gambar 5.20 *Data viewer* (binary)

5.2.20 Implementasi *Data Downloader*

Data Downloader merupakan sebuah komponen dalam fitur manajemen topik IoT. Dalam komponen ini pengguna dapat mengunduh informasi detail dari setiap data yang masuk ke dalam topik yang telah terdaftar dalam sistem.

Dalam pengimplementasiannya komponen ini pertama kali akan mengecek apakah topik yang dipilih beserta data yang dipilih tersedia, kemudian jika diketahui tersedia maka proses menampilkan mengunduh data dari topik akan dilanjutkan. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.25 dibawah ini.

Pseudocode 5.31 *Data downloader*

```
Data Downloader (TopicController.php)
1  method file(req request, str nicename, str id)
2      if topic with selected nicename available and id is valid
3           GET topic data with selected id
4           if data is available
5                return render page with topic data (binary or JSON)
6      return render view with 404 page
7  end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.21 yaitu Gambar data downloader untuk data berbentuk JSON dan 5.21 yaitu Gambar data downloader untuk data berbentuk binary dapat dilihat bahwa sistem dapat berjalan dengan lancar.

{"_id":{"$oid":"5bd1e62c065a81078a0bc595"},"device":"TaniOnSite","data":{"ph":7,"nitrogen":38,"humidity":78},"created_at":{"$date":{"$numberLong":"1540482604000"}}}

Gambar 5.21 Data *downloader* JSON

Gambar 5.22 Data *downloader* Binary

5.2.21 Implementasi Visualisasi Data

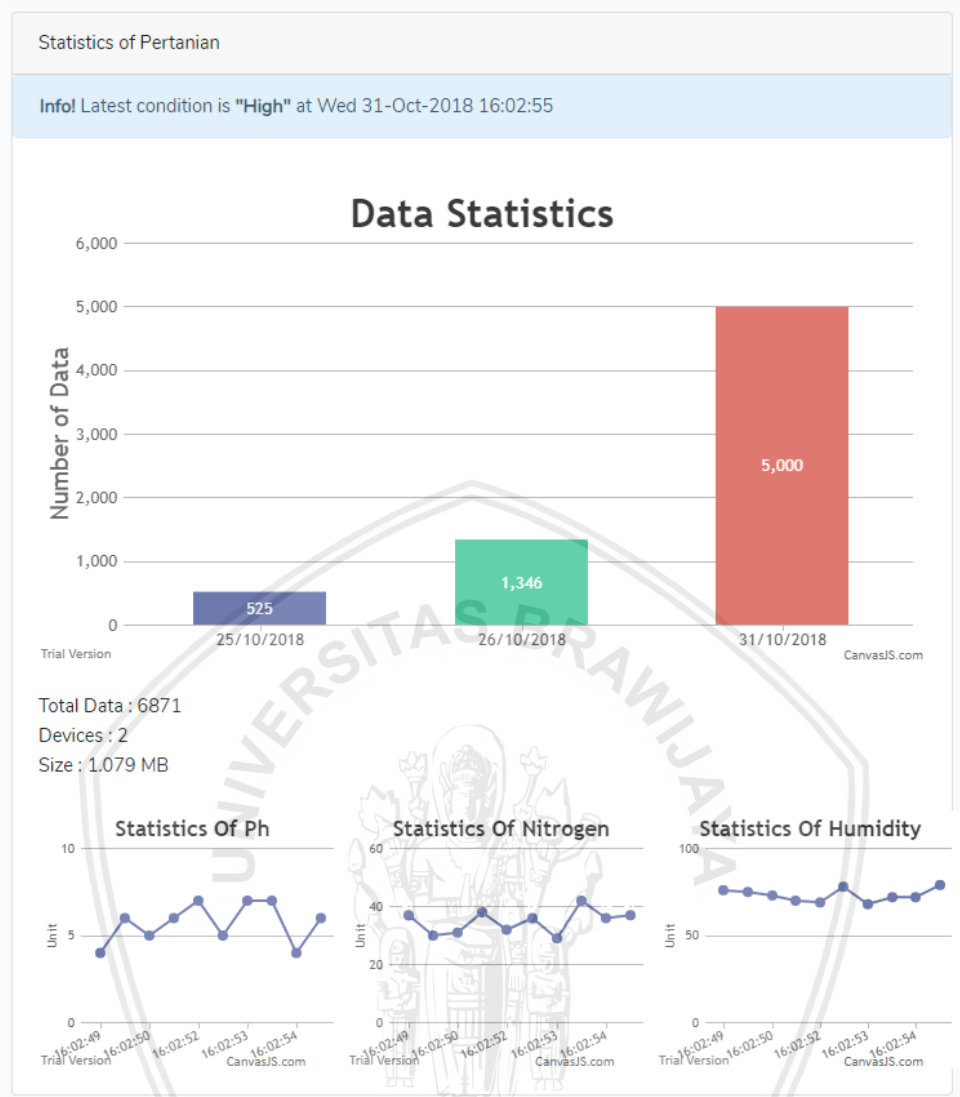
Visualisasi Data merupakan sebuah komponen dalam fitur mengolah data. Dalam komponen ini pengguna dapat menampilkan statistik dari topik seperti statistik pengiriman data, statistik dari setiap variable yang ada dalam topik data, dan juga terdapat informasi mengenai klasifikasi dari data yang terakhir masuk, untuk fitur terakhir ini hanya dapat berjalan jika topik data mempunyai data training dan juga package diatas bronze.

Dalam pengimplementasiannya komponen ini pertama kali akan mengecek apakah topik yang dipilih tersedia, kemudian jika diketahui tersedia maka proses visualisasi data dari topik akan dilanjutkan. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.32 dibawah ini.

Pseudocode 5.32 Visualisasi data

```
Implementasi Method Visualisasi Data (DataAnalyticController.php)
1   method show(str nicename)
2        if topic with nicename selected is available
3             count topic datas
4             GET authorized devices for this topic
5
6             GET topic last active time based on latest topic data
7             populate weekly statistiks
8
9             populate statistiks for each key in data
10           return render view with count, authorized device, last
11  active date, weekly statistiks and statistiks for each key
12       else
13            return render 404 page
14  end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.23 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.23 Visualisasi data

5.2.22 Implementasi Modul Prediksi Least Square

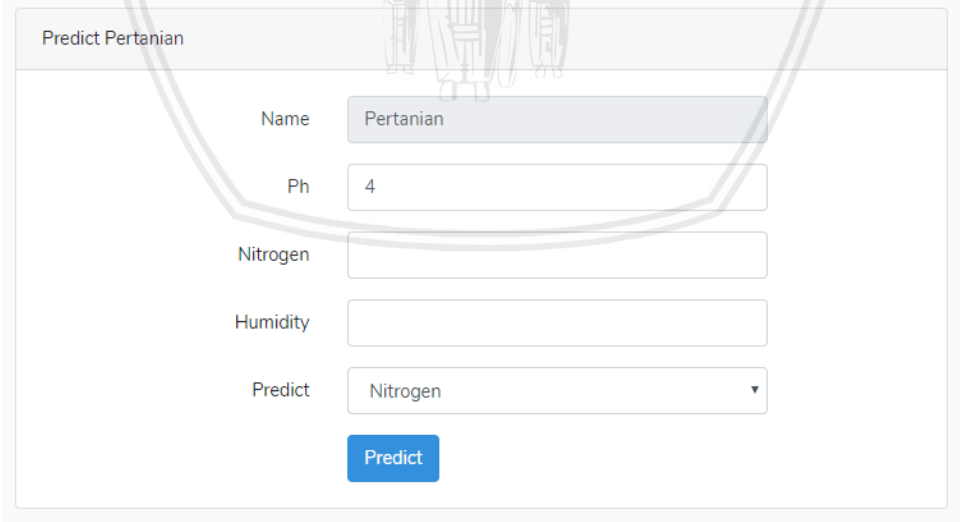
Modul prediksi least square merupakan sebuah komponen dalam fitur mengolah data. Dalam komponen ini pengguna dapat melakukan prediksi sebuah data menggunakan algoritma least square.

Dalam pengimplementasiannya komponen ini pertama kali akan mengecek apakah *input* yang dimasukkan oleh pengguna valid, kemudian jika diketahui valid maka proses prediksi dari topik akan dilanjutkan. Proses berikutnya Setelah melakukan validasi *input* adalah dengan melakukan penentuan *machine learning* apa yang akan digunakan, apakah Apache Spark atau PHP-ML, Setelah diketahui maka proses berikutnya akan diserahkan pada modul-modul yang ada. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.33 dibawah ini.

Pseudocode 5.33 Modul prediksi least square

```
Modul Prediksi Least Square (DataAnalyticController.php)
method least(req request, str nicename)
     if input request not valid
          return render back with notification

     if user package is allow doing analytic
          GET topic information with selected nicename
          if topic is available
               if topic structure is "non-structured"
                    return render view coming soon page
               else
                    populate predict from input request
                    populate targetCol from input request
                    populate sampCol from input request

                    if user package is bronze or machine is "php-ml"
                        do analytic with PHP-ML
                    else if machine is "spark"
                        do analytic with  Apache Spark

               return render analytic work page
          else
               return render 404 page
     else
          return render back with notification
end method

```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.24 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.24 Modul prediksi least square

5.2.23 Implementasi Modul Klasifikasi Naïve Bayes

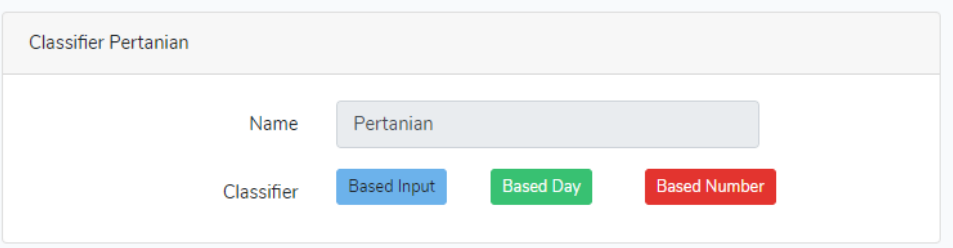
Modul klasifikasi Naïve bayes merupakan sebuah komponen dalam fitur mengolah data. Dalam komponen ini pengguna dapat melakukan klasifikasi sebuah data menggunakan algoritma naïve bayes.

Dalam pengimplementasiannya komponen ini pertama kali akan mengecek apakah *input* yang dimasukkan oleh pengguna valid, kemudian jika diketahui valid maka proses klasifikasi dari topik akan dilanjutkan. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.34 dibawah ini.

Pseudocode 5.34 Modul klasifikasi naïve bayes

```
Modul Klasifikasi NaiveBayes (DataAnalyticController.php)
1   method naive(req request, str nicename)
2        if input request not valid
3             return render back with notification
4
5        if user package is allow doing analytic
6             GET topic information with selected nicename
7             if topic is available
8                  if topic structure is "non-structured"
9                       return render view coming soon page
10                 else
11                      if topic does not have trainedData
12                           return render back with notification
13                      else
14                           if request method is "input"
15                                populate predict from input request
16                           else if request method is "day"
17                                populate predict from data with
18  selected date from database
19                           else if request method is "number"
20                                populate predict from first n-number
21  data from database
22                           else
23                                return render back with notification
24
25                      if predict length is 0
26                          return render back with notification
27                      else if user package is bronze or machine is
28  "php-ml"
29                          do analytic with PHP-ML
30                      else if machine is "spark"
31                          do analytic with  Apache Spark
32
33                 return render analytic work page
34            else
35                 return render 404 page
36       else
37            return render back with notification
38  end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.25 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.25 Modul klasifikasi naïve bayes

5.2.24 Implementasi Keluaran Analitik

Keluaran analitik merupakan sebuah komponen dalam fitur mengolah data. Dalam komponen ini pengguna dapat menampilkan hasil proses analitik yang dilakukan oleh modul-modul yang tersedia di dalam sistem.

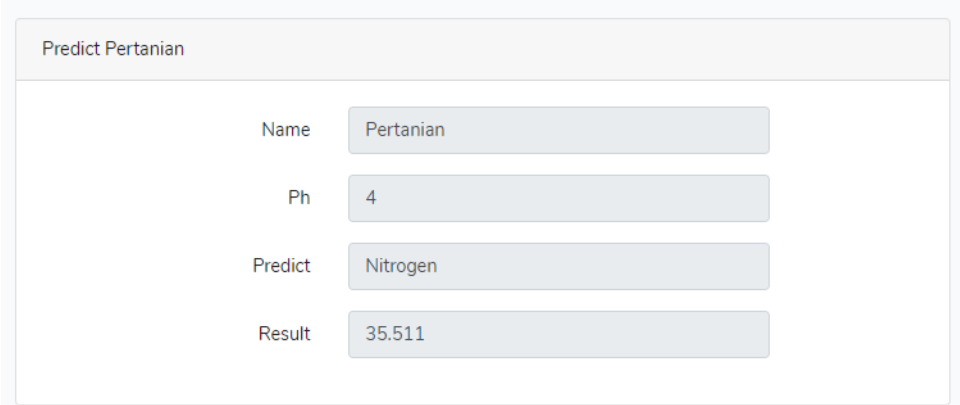
Dalam pengimplementasiannya komponen ini pertama kali akan mengecek apakah proses analitik yang dipilih oleh pengguna tersedia, kemudian jika diketahui tersedia maka proses keluaran analitik dari topik akan dilanjutkan. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseuodocode 5.35 dibawah ini.

Pseudocode 5.35 Keluaran analitik

Keluaran Analitik (DataAnalyticController.php)

```
1  method showWork(str id)
2       GET analytic process with selected id
3       if analytic is available
4            if analytic status is "done"
5                 switch class of analytic
6                      case "app\MultiAnalytic@leastSquare"
7                           return render least square 2 page with
8  analytic data
9                      case "app\SingleAnalytic@leastSquare"
10                          return render least square page with
11 analytic data
12                     case "app\SingleAnalytic@naiveBayes"
13                          return render Naïve bayes page with analytic
14 data
15                     case "app\MultiAnalytic@naiveBayes"
16                          return render Naïve bayes 2 page with
17 analytic data
18                     default
19                          return render 500 page
20           else if analytic status is "error"
21                return render error page
22           else
23                return render process page
24      else
25           return render 404 page
26 end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.26 yaitu Gambar keluaran analitik untuk proses least square, kemudian ada Gambar 5.27 yaitu Gambar keluaran analitik untuk proses Naïve bayes dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.26 Keluaran analitik least square

Classifier of Pertanian

Ph	Nitrogen	Humidity	Class
7	41	69	High
7	40	66	High
7	38	76	High
7	30	73	High
6	42	74	High
7	35	66	High
6	34	71	High
4	39	74	High
5	30	72	High
4	36	66	High
6	41	67	High

Gambar 5.27 Keluaran analitik naïve bayes

5.2.25 Implementasi Menampilkan Daftar Pengguna

Menampilkan daftar pengguna merupakan sebuah komponen dalam fitur manajemen pengguna. Dalam komponen ini dapat dilihat seluruh pengguna yang sudah terdaftar di dalam sistem.

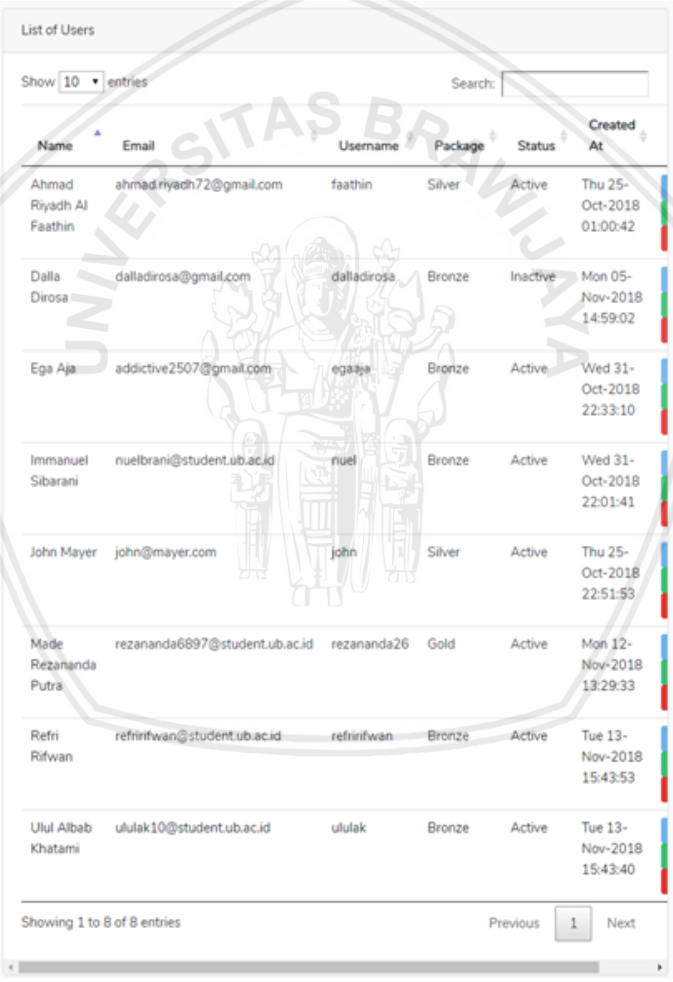
Dalam pengimplementasiannya komponen ini pertama kali hanya akan mengambil informasi dari seluruh pengguna yang masuk ke sistem, yang kemudian setelah itu sistem akan mencoba menampilkan di dalam sebuah view.

Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.36 dibawah ini.

Pseudocode 5.36 Menampilkan daftar pengguna

Menampilkan Daftar Pengguna (UserController.php)	
1	**method** show(**req** request){
2	GET datas of all user
3	**return** render view with all datas from user
4	**end method**

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.28 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.28 Menampilkan daftar pengguna

5.2.26 Implementasi Menampilkan Informasi Pengguna

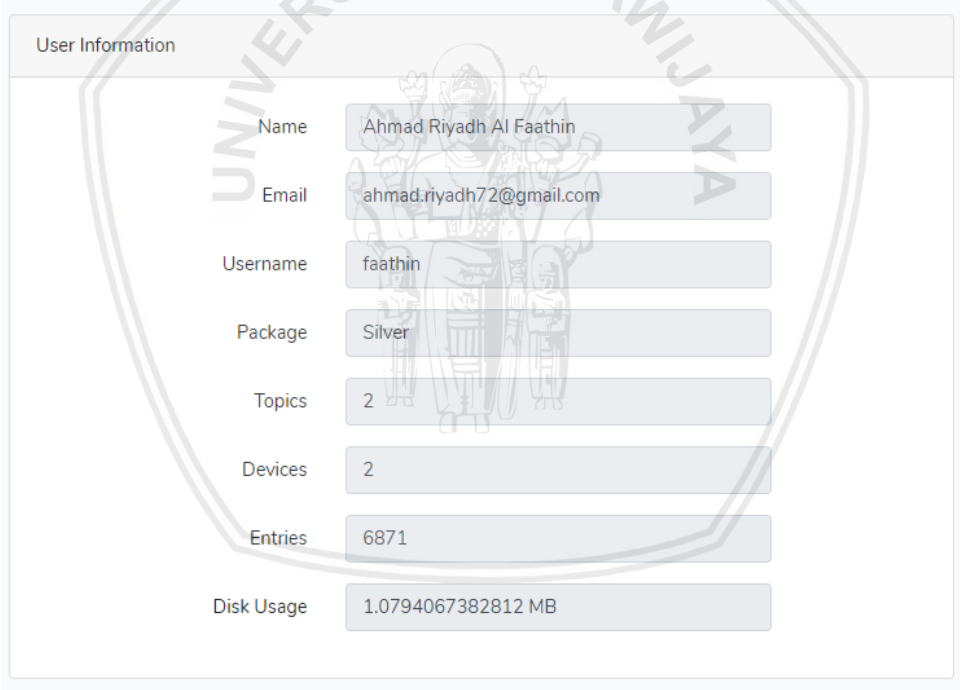
Menampilkan informasi pengguna merupakan sebuah komponen dalam fitur manajemen pengguna. Dalam komponen ini pengguna dapat menampilkan informasi detail dari setiap pengguna termasuk *username*, package, jumlah topik, jumlah perangkat dsb.

Dalam pengimplementasiannya komponen ini pertama kali akan mengecek apakah pengguna yang dipilih tersedia, kemudian jika diketahui tersedia maka proses menampilkan informasi pengguna akan dilanjutkan. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.37 dibawah ini.

Pseudocode 5.37 Menampilkan informasi pengguna

```
Menampilkan Informasi Pengguna (UserController.php)
1  method view(req request, str id){
2       if user with selected id available
3            return render view with all datas from user
4       else
5            return render view 404 page
6  end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.29 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.29 Menampilkan informasi pengguna

5.2.27 Implementasi Mengubah Pengguna

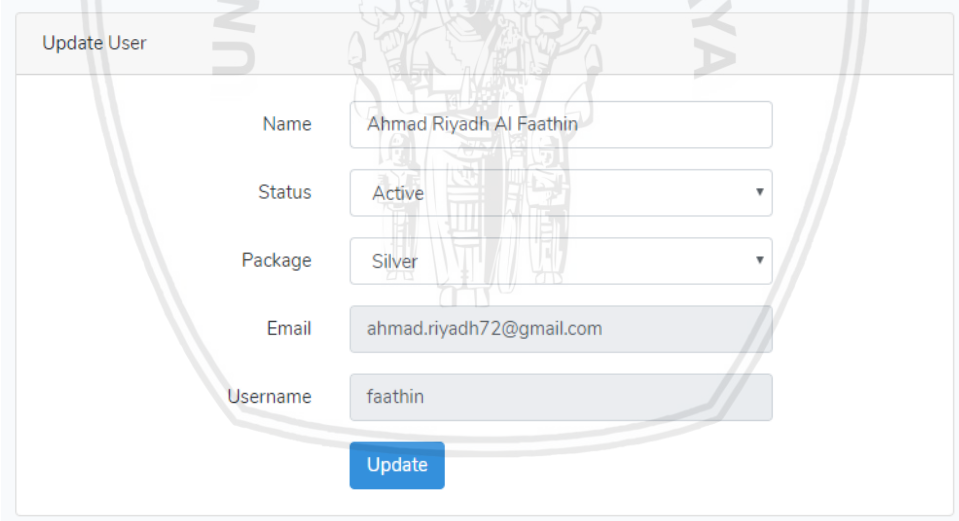
Mengubah pengguna merupakan sebuah komponen dalam fitur manajemen pengguna. Dalam komponen ini administrator dapat mengubah informasi dari setiap pengguna termasuk status, package dan alamat email.

Dalam pengimplementasiannya komponen ini pertama kali akan mengecek apakah pengguna yang dipilih tersedia, kemudian jika diketahui tersedia maka proses mengubah pengguna akan dilanjutkan. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.38 dibawah ini.

Pseudocode 5.38 Mengubah pengguna

```
Implementasi Mengubah Pengguna (UserController.php)
1   method update(req request, str id){
2       if user with selected id available
3           if input request not valid
4               return render back with notification
5           else
6               GET input from request
7               GET selected user from database
8               update selected user with input
9               return render view with all datas from topic
10      else
11          return render view 404 page
12  end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.30 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar.



Gambar 5.30 Mengubah pengguna

5.2.28 Implementasi Menghapus Pengguna

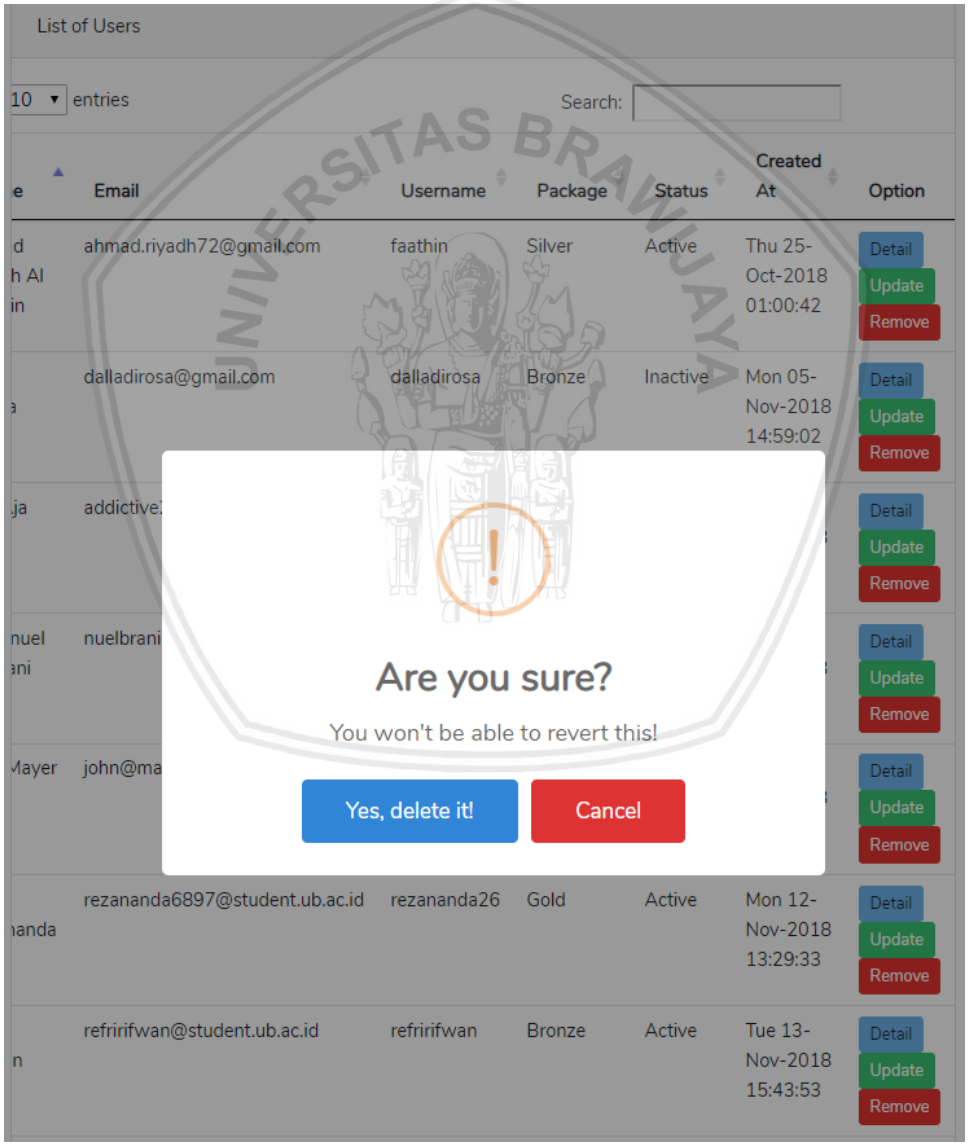
Menghapus pengguna merupakan sebuah komponen dalam fitur manajemen pengguna. Dalam komponen ini administrator dapat menghapus pengguna beserta data yang dimiliki oleh pengguna dari dalam sistem.

Dalam pengimplementasiannya komponen ini pertama kali akan mengecek apakah pengguna yang dipilih tersedia, kemudian jika diketahui tersedia maka proses menghapus pengguna akan dilanjutkan. Untuk lebih jelasnya dapat dipahami melalui pseudocode yang ada pada Pseudocode 5.33 dibawah ini.

Pseudocode 5.39 Menghapus pengguna

```
Implementasi Menghapus Pengguna (UserController.php)
1  method destroy(req request, str id){
2       if user with selected id available
3            delele selected user from database
4            return render view to user list
5       else
6            return render view 404 page
7  end method
```

Setelah pseudocode tersebut di implementasikan dalam bahasa pemrograman PHP menggunakan *framework* Laravel maka fitur dapat dicoba menggunakan sebuah browser untuk menentukan apakah fitur dapat berjalan dengan baik. Dan seperti pada Gambar 5.31 sistem dapat dilihat bahwa sistem dapat berjalan dengan lancar



Gambar 5.31 Menghapus Pengguna

BAB 6 PENGUJIAN

Pada bab ini dilakukan beberapa uji coba terhadap apa yang telah diimplementasikan. Pengujian ini dilakukan sesuai dengan perancangan pengujian yang telah dijelaskan pada bab-bab sebelumnya. Adapun pengujian yang dilakukan adalah pengujian fungsional dan pengujian skalabilitas.

6.1 Pengujian Fungsional

Pengujian fungsional dilakukan agar dapat mengetahui apakah fungsi-fungsi hasil implementasi dengan perancangan sesuai atau tidak. Dari hasil pengujian fungsional, diketahui apakah sistem itu berjalan dengan benar atau tidak. Pengujian akan dimulai dengan menguji autentikasi login menggunakan *JSON Web Token* (JWT), hingga percobaan *create, read, update, delete* (CRUD) pada beberapa data.

6.1.1 Autentikasi Login Perangkat Menggunakan JWT

Kasus uji proses autentikasi login perangkat menggunakan JWT ditunjukkan pada Tabel 6.1 dan Tabel 6.2.

Tabel 6.1 Hasil pengujian fungsional autentikasi login perangkat menggunakan JWT (bagian 1)

Kode	FR_001
Nama Kasus Uji	Autentikasi login perangkat menggunakan JWT
Tujuan Pengujian	Mengetahui apakah sistem dapat melakukan fungsi autentikasi login perangkat menggunakan JWT
Prosedur Pengujian	1. Sistem dijalankan 2. Pengguna menjalankan aplikasi Postman 3. Pengguna memasukkan JWT dari perangkat 4. Pengguna melihat hasil respon dari aplikasi Postman 5. Pengguna mencocokan hasil respon dengan data dari *database*

Tabel 6.2 Hasil pengujian fungsional autentikasi login perangkat menggunakan JWT (bagian 2)

Hasil yang Diharapkan	Sistem dapat melakukan autentikasi login perangkat menggunakan JWT
Hasil Pengujian	Sistem dapat melakukan autentikasi login perangkat menggunakan JWT

Setelah dilakukan pengujian berdasarkan prosedur pengujian yang telah ditentukan maka diketahui bahwa sistem dapat melakukan autentikasi login menggunakan JWT, seperti ditunjukkan pada Keluaran 6.1. Terlihat jika autentikasi berhasil maka akan ditunjukkan informasi perangkat beserta informasi dari pengguna, untuk bukti Gambar dapat dilihat pada Lampiran B.1.1.1.

Keluaran 6.1 Pengujian fungsional autentikasi login menggunakan JWT (Postman)

```
Keluaran Postman
1  {
2      "sys": "ok",
3      "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodH
4  RwczpcL1wvYXBpLnB0cmlvdC5jb21cL3VzZXJcL2xvZ2luIiwiaWF0IjoxNTQ0Nzk
5  1OTUxLCJuYmYiOjE1NDQ3OTU5NTEsImp0aSI6Im9jT1kwS3JQbjEwZ0VDZE4iLCJz
6  dWIiOiI1YmQwYjM0YTA2NWE4MTMwNjMxMzNhMzMiLCJwcnYiOiI4N2UwYWYxZWY5Z
7  mQxNTgxMmZkZWM5NzE1M2ExNGUwYjA0NzU0NmFhIiwiZGV2aWNlIjoiNWJkMWUzOD
8  MwNjVhODEwNzhhMGJjNTkyIn0.EC0oadWeHewBc3ca-pR5H7_KH9rk29dUPMvJnaj
9  TSQA"
10 }
```

Hal ini sesuai dengan hasil pengecekan pada data di *database*, data pengguna yang terdapat pada *database* sesuai dengan respon yang dikeluarkan oleh sistem, seperti yang ditunjukkan pada Keluaran 6.2, untuk bukti Gambar dapat dilihat pada Lampiran B.1.1.2.

Keluaran 6.2 Pengujian fungsional autentikasi login menggunakan JWT (*database*)

```
Keluaran database
1  { "_id" : ObjectId("5bd1e353065a81078c4614b3"), "name" :
2  "TaniOnSite",
3   "nicename" : "tanionsite", "description" : "Sensor pertanian",
4   "location" : "Sawah petak 21", "token" :
5  "d56584b9709d0da26e96d11c3e5
6  ca0b42c67b71a", "secret" : "3496ec3bb4e8ab09b5f992ac5f2f67cb",
7  "user_id" : "5bd0b34a065a813063133a33",
8  "updated_at" : ISODate("2018-12-04T18:25:11Z"),
9  "created_at" : ISODate("2018-10-25T15:37:55Z"),
10 "JWT_active" : "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodH
11 RwczpcL1wvYXBpLnB0cmlvdC5jb21cL3VzZXJcL2xvZ2luIiwiaWF0IjoxNTQzMTA0M
12 TQ2LCJuYmYiOjE1NDMxMDQxNDYsImp0aSI6IjM5Vk4zbEVQaEVRaHBSTFkiLCJzdWIi
13 OiI1YmQwYjM0YTA2NWE4MTMwNjMxMzNhMzMiLCJwcnYiOiI4N2UwYWYxZWY5ZmQxNTg
14 xMmZkZWM5NzE1M2ExNGUwYjA0NzU0NmFhIiwiZGV2aWNlIjoiNWJkMWUzNTMwNjVhOD
15 EwNzhjNDYxNGIzIn0.xTRWy7gybY9saRjL0C-Zz9cw8F2-o26P3DNSTLU2NcI",
16 "status" : "active" }
```

6.1.2 Menampilkan Daftar Topik Melalui Sebuah *API*

Kasus uji proses menampilkan daftar topik melalui sebuah *API* ditunjukkan pada Tabel 6.3.

Tabel 6.3 Hasil pengujian fungsional menampilkan daftar topik melalui sebuah *API*

Kode	FR_002
Nama Kasus Uji	Menampilkan daftar topik melalui sebuah *API*
Tujuan Pengujian	Mengetahui apakah sistem dapat melakukan fungsi menampilkan daftar topik melalui sebuah *API*
Prosedur Pengujian	1. Sistem dijalankan 2. Pengguna menjalankan aplikasi Postman 3. Pengguna memasukkan JWT dari perangkat 4. Pengguna melihat hasil respon dari aplikasi Postman 5. Pengguna mencocokan hasil respon dengan data dari *database*
Hasil yang Diharapkan	Sistem dapat menampilkan daftar topik melalui sebuah *API*
Hasil Pengujian	Sistem dapat menampilkan daftar topik melalui sebuah *API*

Setelah dilakukan pengujian berdasarkan prosedur pengujian yang telah ditentukan maka diketahui bahwa sistem dapat menampilkan daftar topik melalui sebuah *API*, seperti ditunjukkan pada Keluaran 6.3. Terlihat jika proses menampilkan daftar topik berhasil dengan ditunjukkan daftar topik yang terautorisasi pada perangkat tersebut yaitu “Pertanian” dan “Testing”, bukti dari keluaran Postman dapat dilihat pada Lampiran B.1.2.1.

Keluaran 6.3 Pengujian fungsional menampilkan daftar topik melalui sebuah *API* (Postman)

Keluaran Postman

```
1  {
2      "sys": "Ok",
3      "topics": [
4          {
```

```
5               "_id": "5bd1e617065a81085059aaf3",
6               "name": "Pertanian",
7               "nicename": "pertanian",
8               "description": "Data Sensor pertanian"
9           },
10          {
11              "_id": "5c03452e065a81b7f413d8f2",
12              "name": "Gambar CCTV",
13              "nicename": "Gambar_cctv",
14              "description": "Gambar cctv"
15          }
16      ]
17  }
```

Hal ini juga sesuai dengan hasil pengecekan data yang terdapat pada *database*. Dalam *database* data topik yang terautorisasi untuk perangkat tersebut menunjukkan kesamaan dengan data yang dikeluarkan oleh sistem. Seperti yang ditunjukkan pada Keluaran 6.4 dan 6.5 bukti dari keluaran Postman dapat dilihat pada Lampiran B.1.2.2.

Keluaran 6.4 Pengujian fungsional menampilkan daftar topik melalui sebuah *API* (*database* bagian 1)

Keluaran *database*

```
1   { "_id" : ObjectId("5bd1e617065a81085059aaf3"),
2   "name" : "Pertanian", "nicename" : "pertanian",
3   "description" : "Data Sensor pertanian", "structure" :
4   "structured",
5   "user_id" : "5bd0b34a065a813063133a33",
6   "updated_at" : ISODate("2018-12-10T12:36:47Z"),
7   "created_at" : ISODate("2018-10-25T15:49:43Z"),
8   "devices" : [ { "_id" : ObjectId("5bd1e353065a81078c4614b3"),
9   "name" : "TaniOnSite", "nicename" : "tanionsite",
10  "description" : "Sensor pertanian", "location" : "Sawah petak 21",
11  "token" : "d56584b9709d0da26e96d11c3e5ca0b42c67b71a",
12  "secret" : "3496ec3bb4e8ab09b5f992ac5f2f67cb",
13  "user_id" : "5bd0b34a065a813063133a33",
14  "updated_at" : ISODate("2018-10-25T15:46:18Z"),
15  "created_at" : ISODate("2018-10-25T15:37:55Z"),
16  "JWT_active" : "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJod
17  HRwczpcL1wvYXBpLnB0cmlvdC5mYWF0aGluLmNvbVwvbG9naW4iLCJpYXQiOjE1NDA
18  0ODIzNzgsIm5iZiI6MTU0MDQ4MjM3OCwianRpIjoiQTg4M21QV3JrdHpBbDFyMiIsI
19  nN1YiI6IjViZDBiMzRhMDY1YTgxMzA2MzEzM2EzMyIsInBydiI6Ijg3ZTBhZjFlZjl
20  mZDE1ODEyZmRlYzk3MTUzYTE0ZTBiMDQ3NTQ2YWEiLCJkZXZpY2UiOiI1YmQxZTM1M
21  zA2NWE4MTA3OGM0NjE0YjMifQ.f7TaGku8hsqlQXjZe8_ixhQIZl_GYRtO5SSghPOj
22  dg" },
23  { "_id" : ObjectId("5bd1e383065a81078a0bc592"), "name" :
24  "TaniOnSiteL2",
25  "nicename" : "tanionsitel2", "description" : "Sensor pertanian",
26  "location" : "Sawah petak 24",
27  "token" : "a366eba729db35726805f67b86b3d9613ac3581e",
28  "secret" : "9f7b4024279a44bb944a7ab256fbaf4a",
29  "user_id" : "5bd0b34a065a813063133a33",
30  "updated_at" : ISODate("2018-10-25T15:38:43Z"),
31  "created_at" : ISODate("2018-10-25T15:38:43Z") } ],
32  "storage" : "mongodb", "status" : "active", "entry" : 6875 }
33  { "_id" : ObjectId("5c03452e065a81b7f413d8f2"),
34  "name" : "Gambar CCTV", "nicename" : "Gambar_cctv",
35  "description" : "Gambar cctv", "structure" : "non-structured",
36  "user_id" : "5bd0b34a065a813063133a33",
37  "updated_at" : ISODate("2018-12-14T14:27:35Z"),
38  "created_at" : ISODate("2018-12-02T02:36:30Z"),
39  "devices" : [ { "_id" : ObjectId("5bd1e353065a81078c4614b3"),
40  "name" : "TaniOnSite", "nicename" : "tanionsite",
```

Keluaran 6.5 Pengujian fungsional menampilkan daftar topik melalui sebuah *API* (*database* bagian 2)

```
41   "description" : "Sensor pertanian", "location" : "Sawah petak
42  21",
43  "token" : "d56584b9709d0da26e96d11c3e5ca0b42c67b71a",
44  "secret" : "3496ec3bb4e8ab09b5f992ac5f2f67cb",
45  "user_id" : "5bd0b34a065a813063133a33",
46  "updated_at" : ISODate("2018-12-04T18:25:11Z"),
47  "created_at" : ISODate("2018-10-25T15:37:55Z"),
48  "JWT_active" : "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJod
49  HRwczpcL1wvYXBpLnB0cmlvdC5jb21cL3VzZXJcL2xvZ2luIiwiaWF0IjoxNTQzMTA
50  0MTQ2LCJuYmYiOjE1NDMxMDQxNDYsImp0aSI6IjM5Vk4zbEVQaEVRaHBSTFkiLCJzd
51  WIiOiI1YmQwYjM0YTA2NWE4MTMwNjMxMzNhMzMiLCJwcnYiOiI4N2UwYWYxZWY5ZmQ
52  xNTgxMmZkZWM5NzE1M2ExNGUwYjA0NzU0NmFhIiwiZGV2aWNlIjoiNWJkMWUzNTMwN
53  jVhODEwNzhjNDYxNGIzIn0.xTRWy7gybY9saRjL0C-Zz9cw8F2-
54  o26P3DNSTLU2NcI", "status" : "active" } ], "storage" : "gridfs",
55  "status" : "active", "size" : 83291, "entry" : 1 }
```

6.1.3 Menerima Data Terstruktur Dalam Bentuk JSON

Kasus uji proses menerima data terstruktur dalam bentuk JSON ditunjukkan pada Tabel 6.4 dan 6.5.

Tabel 6.4 Hasil pengujian fungsional menerima data terstruktur dalam bentuk JSON (bagian 1)

Kode	FR_003
Nama Kasus Uji	Menerima data terstruktur dalam bentuk JSON
Tujuan Pengujian	Mengetahui apakah sistem dapat menerima data terstruktur dalam bentuk JSON
Prosedur Pengujian	1. Sistem dijalankan 2. Pengguna menjalankan aplikasi Postman 3. Pengguna memasukkan JWT dari perangkat 4. Pengguna memasukkan *file* JSON yang mengandung data terstruktur 5. Pengguna melihat hasil respon dari aplikasi Postman
Hasil yang Diharapkan	Sistem dapat menerima data terstruktur dalam bentuk JSON

Tabel 6.5 Hasil pengujian fungsional menerima data terstruktur dalam bentuk JSON (bagian 2)

Hasil Pengujian	Sistem dapat menerima data terstruktur dalam bentuk JSON

Setelah dilakukan pengujian berdasarkan prosedur pengujian yang telah ditentukan maka diketahui bahwa sistem dapat menerima data terstruktur dalam bentuk JSON. Seperti ditunjukkan pada Keluaran 6.5 dan Lampiran B.1.3.1. Terlihat jika proses pengiriman data JSON oleh perangkat berhasil dan diterima oleh *framework* maka akan ditunjukkan informasi *response* pada sistem yaitu *form*at data yaitu JSON, ukuran data yaitu 57 *bytes*, dan *timestamp* dari data diterima oleh *framework*.

Keluaran 6.6 Pengujian fungsional menerima data terstruktur dalam bentuk JSON (Postman)

Keluaran Postman

```
1  {
2      "sys": "Ok",
3      "format": "json",
4      "size": 57,
5      "remark": "No Remark",
6      "timestamp": 1544799701
7  }
```

Langkah berikutnya adalah mencocokan dengan data di dalam *database*. Untuk melakukannya maka diperlukan login ke sistem vps untuk mendapatkan data dari dalam *database* MongoDB. setelah dilakukan *query* terhadap *database* maka dipastikan data telah tersimpan. Tersimpan dalam bentuk JSON dan cocok dengan hasil keluaran dari proses sebelumnya seperti pada Keluaran 6.6 dan Lampiran B.1.3.2.

Keluaran 6.7 Pengujian fungsional menerima data terstruktur dalam bentuk JSON (database)

Keluaran *database*

```
1  { "_id" : ObjectId("5c13c5d5065a819c0d3fa123"), "device" :
2  "TaniOnSite", "data" : { "ph" : 4, "nitrogen" : 41, "humidity" : 66
3  }, "created_at" : ISODate("2018-12-14T15:01:41Z") }
```

6.1.4 Menerima Data Semi Terstruktur Dalam Bentuk XML

Kasus uji proses menerima data semi terstruktur dalam bentuk XML ditunjukkan pada Tabel 6.6 dan 6.7.

Tabel 6.6 Hasil pengujian fungsional menerima data semi terstruktur dalam bentuk XML (bagian 1)

Kode	FR_004
Nama Kasus Uji	Menerima data semi terstruktur dalam bentuk XML

Tabel 6.7 pengujian fungsional menerima data semi terstruktur dalam bentuk XML (bagian 2)

Tujuan Pengujian	Mengetahui apakah sistem dapat menerima data semi terstruktur dalam bentuk XML
Prosedur Pengujian	1. Sistem dijalankan 2. Pengguna menjalankan aplikasi Postman 3. Pengguna memasukkan JWT dari perangkat 4. Pengguna memasukkan *file* XML yang mengandung data semi terstruktur 5. Pengguna melihat hasil respon dari aplikasi Postman
Hasil yang Diharapkan	Sistem dapat menerima data semi terstruktur dalam bentuk XML
Hasil Pengujian	Sistem dapat menerima data semi terstruktur dalam bentuk XML

Setelah dilakukan pengujian berdasarkan prosedur pengujian yang telah ditentukan maka diketahui bahwa *framework* menerima data semi terstruktur dalam bentuk XML, seperti ditunjukkan pada Keluaran 6.8 dan Lampiran B.1.4.1. Terlihat jika proses pengiriman data XML oleh perangkat berhasil dan diterima oleh *framework* maka akan ditunjukkan informasi *response* pada sistem yaitu *form*at data yaitu XML, ukuran data yaitu 119 *bytes*, dan *timestamp* dari data diterima oleh *framework*.

Keluaran 6.8 Pengujian fungsional menerima data semi terstruktur dalam bentuk XML (Postman)

```
Keluaran Postman
1  {
2      "sys": "Ok",
3      "format": "xml",
4      "size": 119,
5      "remark": "No Remark",
6      "timestamp": 1544800355
7  }
```

Langkah berikutnya adalah mencocokan dengan data di dalam *database*. Untuk melakukannya maka diperlukan login ke sistem vps untuk mendapatkan data dari dalam *database* MongoDB. setelah dilakukan *query* terhadap *database* maka dipastikan data telah tersimpan. Tersimpan dalam bentuk JSON dan cocok dengan hasil keluaran dari proses sebelumnya seperti pada Keluaran 6.9 dan Lampiran B.1.4.2.

Keluaran 6.9 Pengujian fungsional menerima data semi terstruktur dalam bentuk XML (*database*)

Keluaran *database*	
1	{ "_id" : ObjectId("5c13c863065a819c0d3fa124"), "device" :
2	"TaniOnSite", "data" : { "ph" : "4", "nitrogen" : "41", "humidity" :
3	"66" }, "created_at" : ISODate("2018-12-14T15:12:35Z") }

6.1.5 Menerima Data Tidak Terstruktur Dalam Bentuk Gambar JPG

Kasus uji proses menerima data tidak terstruktur dalam bentuk JPG ditunjukkan pada Tabel 6.8.

Tabel 6.8 Hasil pengujian fungsional menerima data tidak terstruktur dalam bentuk Gambar JPG

Kode	FR_005
Nama Kasus Uji	Menerima data tidak terstruktur dalam bentuk JPG
Tujuan Pengujian	Mengetahui apakah sistem dapat menerima data tidak terstruktur dalam bentuk JPG
Prosedur Pengujian	1. Sistem dijalankan 2. Pengguna menjalankan aplikasi Postman 3. Pengguna memasukkan JWT dari perangkat 4. Pengguna memasukkan *file* JPG yang mengandung data tidak terstruktur 5. Pengguna melihat hasil respon dari aplikasi Postman
Hasil yang Diharapkan	Sistem dapat menerima data tidak-terstruktur dalam bentuk JPG
Hasil Pengujian	Sistem dapat menerima data tidak-terstruktur dalam bentuk JPG

Setelah dilakukan pengujian berdasarkan prosedur pengujian yang telah ditentukan maka diketahui bahwa *framework* menerima data tidak-terstruktur dalam bentuk JPG, seperti ditunjukkan pada Keluaran 6.10. Terlihat jika proses pengiriman data JPG oleh perangkat berhasil dan diterima oleh *framework* maka akan ditunjukkan informasi *response* pada sistem yaitu *form*at data yaitu binary, ukuran data yaitu 325750 *bytes*, dan *timestamp* dari data diterima oleh

framework, bukti dari percobaan Postman ini dapat ditemui pada Lampiran B.1.5.1.

Keluaran 6.10 Pengujian fungsional menerima data tidak terstruktur dalam bentuk JPG

Keluaran Postman

```
{
    "sys": "Ok",
    "format": "binary",
    "size": 325750,
    "remark": "No Remark",
    "timestamp": 1544800884
}
```

Langkah berikutnya adalah mencocokan dengan data di dalam *database*. Untuk melakukannya maka diperlukan login ke sistem vps untuk mendapatkan data dari dalam *database* MongoDB. setelah dilakukan *query* terhadap *database* maka dipastikan data dapat tersimpan. Tersimpan sebagai *chunk* seperti pada Keluaran 6.11 maupun metadata pada Keluaran 6.12 dan juga cocok dengan hasil keluaran dari *webservice* pada Gambar 6.1, detail dari Keluaran 6.11 dapat ditemukan di Lampiran B.1.5.2 sedangkan Keluaran 6.12 dapat ditemukan dalam Lampiran B.1.5.3.

Keluaran 6.11 Pengujian fungsional menerima data tidak terstruktur dalam bentuk JPG (*chunks database*)

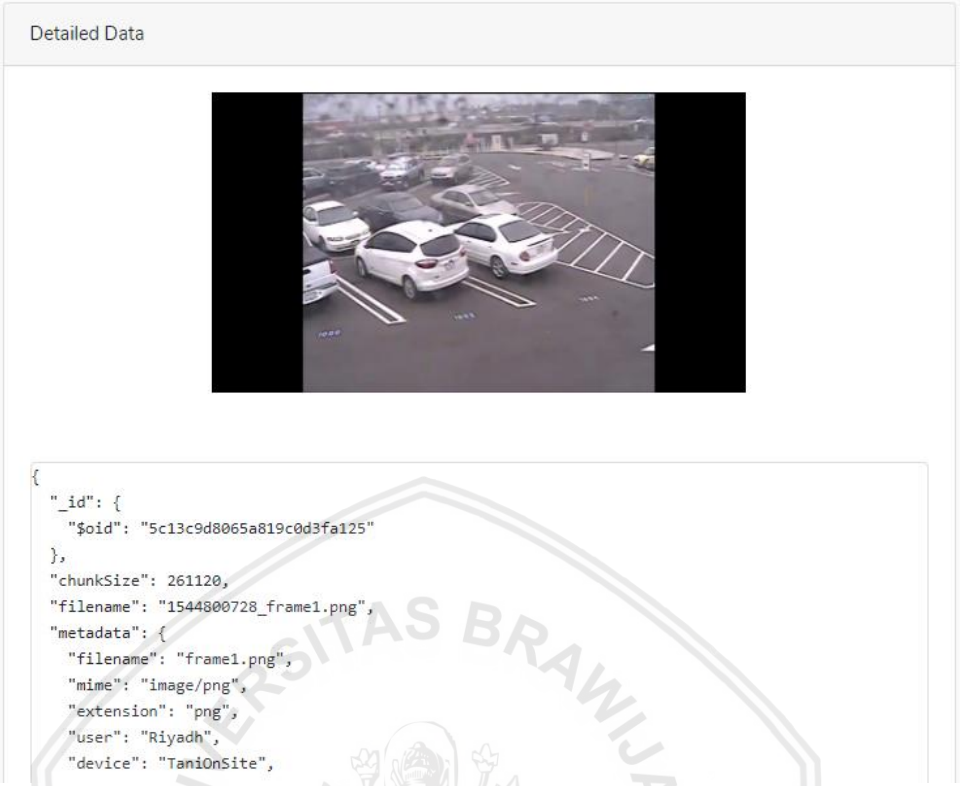
Keluaran *chunk database*

```
667cPj5uOa7POchBbo7cOyyjLjdlKZDSy9QmO1YiPQo4llHnU3Ie5pwl62Ut9GJBtyu
lhhVnitSw4kiRlI5b+zrcyD2I9FtD1qRvQagvFI7RabfmBVMeRYTv0RNxKcTyyyHH5V
G88Wl/FBkmhnMc5wycLo5z9ViDsLxzNZsl0liQKYNV+QdnnSWwJoA5v1vEKksJhmcc6
Zb+1w71vBV9a/SWg203Va4Z6KcXAlrmwrLLdGS6NZHMqUcrjpUPfqupJGtj6STFce4R
nFl/T7uOMx38xsmnn366+fzzz9NAhr9gMGG0o+FU9p/HW5FhrXjPuKqIPGbJMDnAFfi
Z8oDfg+r4i3ejQlJHyTEcRxxJ7VGgbko6Baa05ToMFTNL2BEk8JcMbkbRx+PdREzg4G
B6J8Gf5X8SIvcNYn8b9gV4ZBF9uF1CMsqvBbazDn/Iv6TSSK3cwzZDq/4WiJuzL0y31
r+t/K10JLU0JbXyWSnFcsVrpri7OLxDFS8gwY3/6WIS/Mupz2FiiihAQWJVFUTXT65u
tf8xnohu17WdHslkdxYNp68EznLKCeoU20roJ8uTKzNAnJLodI7jUQ4cE1Lkuwmryif
Qnf+RVB+//f775o8//kjzCseZyXVdrnecAjB8Eu2ezlJG2bld+kwgBvblBvfrhROxe4
ZllOWUfpZYTDlek1Z0HGtfurntlV6sNKeIRh/nhQd5AUL6yThz6Ik7hVweNduDjzuU5
FKw3l6q/hznJfGFlbMY2ljSYFruuVhp9UgvVppTZCq4omi9c7UavDv30YDJ1kterURb
ntpz7HKcMjU7csDzwsbjzoyDO974iEUuHZz5t3j2y/JwHwspucCKx3CuF+1p2R7kLqz
```

Keluaran 6.12 Pengujian fungsional menerima data tidak terstruktur dalam bentuk JPG (*metadata database*)

Keluaran metadata database

```
{ "_id" : ObjectId("5c13ca74065a81f6057c4724"), "chunkSize" :
261120, "filename" : "1544800884_frame0.jpg", "metadata" : {
"filename" : "frame0.jpg", "mime" : "image/PNG", "extension" :
"PNG", "user" : "Riyadh", "device" : "TaniOnSite", "topic" : "Gambar
CCTV" }, "length" : 325750, "md5" :
"60efffd9f291f1b0698eb32073318e3e", "uploadDate" : ISODate("2018-12-
14T15:21:24.543Z") }
```



Gambar 6.1 Pengujian fungsional menerima data tidak terstruktur dalam bentuk JPG (*webservice*)

6.1.6 Menampilkan Daftar Data yang Disimpan Dalam Suatu Topik Melalui Sebuah *API*

Kasus uji proses menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API* ditunjukkan pada Tabel 6.9 dan 6.10.

Tabel 6.9 Hasil pengujian fungsional menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API* (bagian 1)

Kode	FR_006
Nama Kasus Uji	Menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API*
Tujuan Pengujian	Mengetahui apakah sistem dapat menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API*

Tabel 6.10 Hasil pengujian fungsional menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API* (bagian 2)

Prosedur Pengujian	1. Sistem dijalankan 2. Pengguna menjalankan aplikasi Postman 3. Pengguna memasukkan JWT dari perangkat beserta topik yang ingin ditampikan 4. Pengguna melihat hasil respon dari aplikasi Postman 5. Pengguna mencocokan hasil respon dengan data dari *database*
Hasil yang Diharapkan	Sistem dapat menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API*
Hasil Pengujian	Sistem dapat menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API*

Setelah dilakukan pengujian berdasarkan prosedur pengujian yang telah ditentukan maka diketahui bahwa sistem dapat menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API*, seperti ditunjukkan pada Keluaran 6.13, 6.14, 6.15 dan dapat dilihat buktinya pada Lampiran B.6.1.2. Terlihat jika proses menampilkan daftar data berhasil apabila *framework* mengembalikan informasi daftar data dari sebuah topik

Keluaran 6.13 Pengujian menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API* (Postman bagian 1)

Keluaran Postman

```
{
    "sys": "Ok",
    "format": "Json",
    "data": [
        {
            "_id": {
                "$oid": "5c03455e065a81b7f6602fc4"
            },
            "chunkSize": 261120,
            "filename": "1543718238_frame0.jpg",
            "metadata": {

```

Keluaran 6.14 Pengujian menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API* (Postman bagian 2)


```
                "mime": "image/jpeg",
                "extension": "jpeg",
                "user": "Ahmad Riyadh Al Faathin",
                "device": "TaniOnSite",
                "topic": "Gambar CCTV"
            },
            "length": 83291,
            "md5": "ea3793ed50474e211b4930a86c3b60cf",
            "uploadDate": {
                "$date": {
                    "$numberLong": "1543718239576"
                }
            }
        },
        {
            "_id": {
                "$oid": "5c13c9d8065a819c0d3fa125"
            },
            "chunkSize": 261120,
            "filename": "1544800728_frame1.PNG",
            "metadata": {
                "filename": "frame1.PNG",
                "mime": "image/PNG",
                "extension": "PNG",
                "user": "Riyadh",
                "device": "TaniOnSite",
                "topic": "Gambar CCTV"
            },
            "length": 325750,
            "md5": "60efffd9f291f1b0698eb32073318e3e",
            "uploadDate": {
                "$date": {
                    "$numberLong": "1544800728393"
                }
            }
        },
        {
            "_id": {
                "$oid": "5c13ca74065a81f6057c4724"
            },
            "chunkSize": 261120,
            "filename": "1544800884_frame0.jpg",
            "metadata": {
                "filename": "frame0.jpg",
                "mime": "image/PNG",
                "extension": "PNG",
                "user": "Riyadh",
                "device": "TaniOnSite",
                "topic": "Gambar CCTV"
            },
            "length": 325750,
            "md5": "60efffd9f291f1b0698eb32073318e3e",
            "uploadDate": {
                "$date": {
                    "$numberLong": "1544800884543"
                }
            }
        }

```

Keluaran 6.15 Pengujian menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API* (Postman bagian 3)

```
72      ],
73      "timestamp": 1544802039
74  }
```

Langkah berikutnya adalah mencocokan dengan data di dalam *database*. Untuk melakukannya maka diperlukan login ke sistem *vps* untuk mendapatkan data dari dalam *database* MongoDB. setelah dilakukan *query* terhadap *database* maka dipastikan bahwa data dapat disimpan dan cocok dengan hasil keluaran dari proses sebelumnya seperti yang ditunjukkan pada Keluaran 6.16.

Keluaran 6.16 Pengujian menampilkan daftar data yang disimpan dalam suatu topik melalui sebuah *API* (*database*)

```
Keluaran database
1   { "_id" : ObjectId("5c03455e065a81b7f6602fc4"), "chunkSize" :
2   261120, "filename" : "1543718238_frame0.jpg", "metadata" : {
3   "filename" : "frame0.jpg", "mime" : "image/jpeg", "extension" :
4   "jpeg", "user" : "Ahmad Riyadh Al Faathin", "device" : "TaniOnSite",
5   "topic" : "Gambar CCTV" }, "length" : 83291, "md5" :
6   "ea3793ed50474e211b4930a86c3b60cf", "uploadDate" : ISODate("2018-12-
7   02T02:37:19.576Z") }
8   { "_id" : ObjectId("5c13c9d8065a819c0d3fa125"), "chunkSize" :
9   261120, "filename" : "1544800728_frame1.PNG", "metadata" : {
10  "filename" : "frame1.PNG", "mime" : "image/PNG", "extension" :
11  "PNG", "user" : "Riyadh", "device" : "TaniOnSite", "topic" : "Gambar
12  CCTV" }, "length" : 325750, "md5" :
13  "60efffd9f291f1b0698eb32073318e3e", "uploadDate" : ISODate("2018-12-
14  14T15:18:48.393Z") }
15  { "_id" : ObjectId("5c13ca74065a81f6057c4724"), "chunkSize" :
16  261120, "filename" : "1544800884_frame0.jpg", "metadata" : {
17  "filename" : "frame0.jpg", "mime" : "image/PNG", "extension" :
18  "PNG", "user" : "Riyadh", "device" : "TaniOnSite", "topic" : "Gambar
19  CCTV" }, "length" : 325750, "md5" :
20  "60efffd9f291f1b0698eb32073318e3e", "uploadDate" : ISODate("2018-12-
21  14T15:21:24.543Z") }
```

6.1.7 Menampilkan Data yang Disimpan Dalam Suatu Topik Melalui Sebuah *API*

Kasus uji proses menampilkan data yang disimpan dalam suatu topik melalui sebuah *API* ditunjukkan pada Tabel 6.11 dan Tabel 6.12.

Tabel 6.11 Hasil pengujian fungsional menampilkan data yang disimpan dalam suatu topik melalui sebuah *API* (bagian 1)

Kode	FR_007
Nama Kasus Uji	Menampilkan data yang disimpan dalam suatu topik melalui sebuah *API*

Tabel 6.12 Hasil pengujian fungsional menampilkan data yang disimpan dalam suatu topik melalui sebuah *API* (bagian 2)

Tujuan Pengujian	Mengetahui apakah sistem dapat menampilkan data yang disimpan dalam suatu topik melalui sebuah *API*
Prosedur Pengujian	1. Sistem dijalankan 2. Pengguna menjalankan aplikasi Postman 3. Pengguna memasukkan *Token* dan *Secret key* dari perangkat beserta topik dan id data yang ingin ditampikan 4. Pengguna melihat hasil respon dari aplikasi Postman 5. Pengguna mencocokan hasil respon dengan data dari *database*
Hasil yang Diharapkan	Sistem dapat menampilkan data yang disimpan dalam suatu topik melalui sebuah *API*
Hasil Pengujian	Sistem dapat menampilkan data yang disimpan dalam suatu topik melalui sebuah *API*

Setelah dilakukan pengujian berdasarkan prosedur pengujian yang telah ditentukan maka diketahui bahwa sistem dapat menampilkan data yang disimpan dalam suatu topik melalui sebuah *API*, seperti ditunjukkan pada Gambar 6.17 dan 6.18. Terlihat jika proses menampilkan data berhasil apabila *framework* mengembalikan informasi data dari sebuah topik. Untuk bukti Gambar dari proses ini dapat dilihat pada Lampiran B.1.7.1.

Keluaran 6.17 Pengujian menampilkan data yang disimpan dalam suatu topik melalui sebuah *API* (Postman bagian 1)

Keluaran Postman (bagian 1)

```
1  {
2      "sys": "Ok",
3      "format": "Json",
4      "data": [
5          {
6              "_id": {
7                  "$oid": "5bd1e62c065a81078a0bc595"
8
```

Keluaran 6.18 Pengujian menampilkan data yang disimpan dalam suatu topik melalui sebuah *API* (Postman bagian 2)

```
            },
            "device": "TaniOnSite",
            "data": {
                "ph": 7,
                "nitrogen": 38,
                "humidity": 78
            },
            "created_at": {
                "$date": {
                    "$numberLong": "1540482604000"
                }
            }
        }
    ],
    "timestamp": 1544802783
}
```

Langkah berikutnya adalah mencocokan dengan data di dalam *database*. Untuk melakukannya maka diperlukan login ke sistem vps untuk mendapatkan data dari dalam *database* MongoDB. setelah dilakukan *query* terhadap *database* maka dipastikan bahwa data dapat disimpan dan cocok dengan hasil keluaran dari proses sebelumnya seperti pada Keluaran 6.19. Untuk bukti Gambar dari proses ini dapat dilihat pada Lampiran B.1.7.1.

Keluaran 6.19 Pengujian menampilkan data yang disimpan dalam suatu topik melalui sebuah *API* (*database*)

Keluaran *database*

```
{ "_id" : ObjectId("5bd1e62c065a81078a0bc595"), "device" :
"TaniOnSite", "data" : { "ph" : 7, "nitrogen" : 38, "humidity" : 78
}, "created_at" : ISODate("2018-10-25T15:50:04Z") }
```

6.1.8 Registrasi Pengguna Baru

Kasus uji proses registrasi pengguna baru ditunjukkan pada Tabel 6.13 dan 6.14.

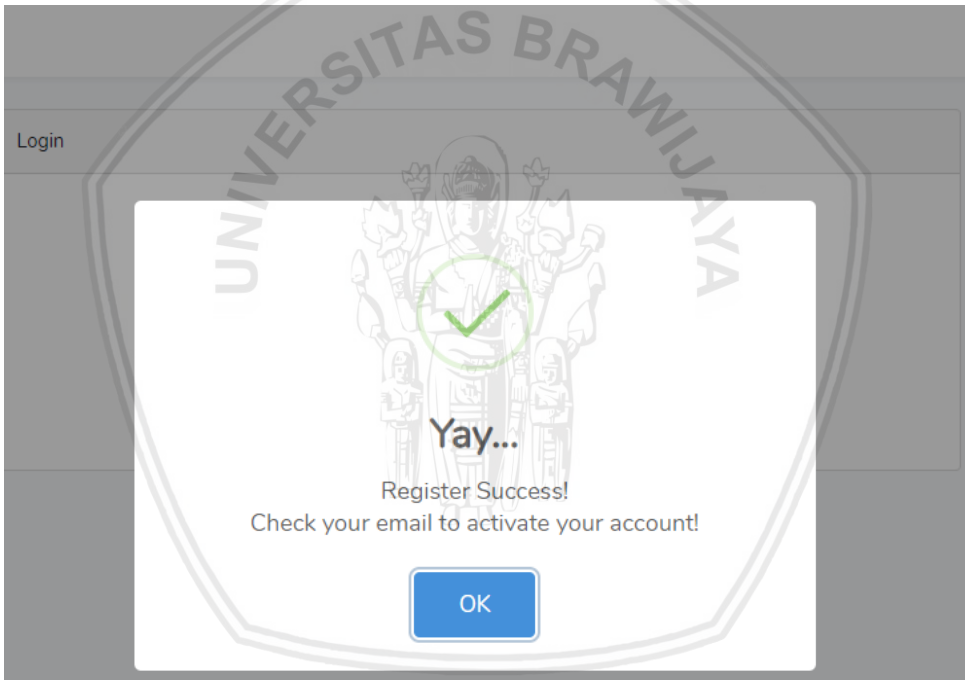
Tabel 6.13 Hasil pengujian fungsional registrasi pengguna baru (bagian 1)

Kode	FR_008
Nama Kasus Uji	Registrasi pengguna baru
Tujuan Pengujian	Mengetahui apakah sistem dapat registrasi pengguna baru
Prosedur Pengujian	1. Sistem dijalankan 2. Pengguna melakukan input data pengguna baru

Tabel 6.14 Hasil pengujian fungsional registrasi pengguna baru (bagian 2)

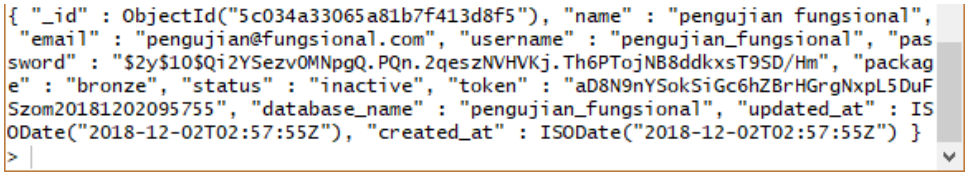
	3. Pengguna mencocokan data dengan database
Hasil yang Diharapkan	Sistem dapat registrasi pengguna baru
Hasil Pengujian	Sistem dapat registrasi pengguna baru

Setelah dilakukan pengujian berdasarkan prosedur pengujian yang telah ditentukan maka diketahui bahwa sistem dapat melakukan registrasi pengguna baru, informasi yang dimasukkan untuk pengguna baru adalah nama "pengujian fungsional", alamat email "pengujian@fungsional.com", dan usename "pengujian_fungsional" setelah dilakukan *submit* data maka sistem akan menampilkan respon seperti Gambar 6.2.



Gambar 6.2 Pengujian fungsional registrasi pengguna baru (*webservice*)

Langkah berikutnya adalah mencocokan dengan data di dalam *database*. Untuk melakukannya maka diperlukan login ke sistem vps untuk mendapatkan data dari dalam *database* MongoDB. setelah dilakukan *query* terhadap *database* maka dipastikan data dapat disimpan dan cocok dengan hasil *input* dari pengguna seperti pada Gambar 6.3.



Gambar 6.3 Pengujian fungsional registrasi pengguna baru (*database*)

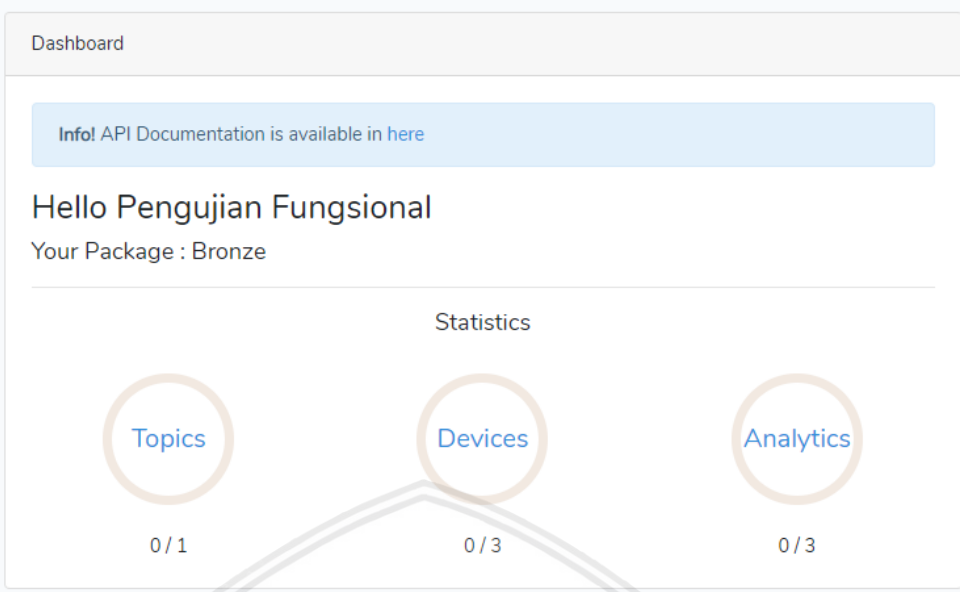
6.1.9 Login Pengguna

Kasus uji proses login pengguna ditunjukkan pada Tabel 6.15.

Tabel 6.15 Hasil pengujian fungsional login pengguna

Kode	FR_009
Nama Kasus Uji	Login pengguna
Tujuan Pengujian	Mengetahui apakah sistem dapat login pengguna
Prosedur Pengujian	1. Sistem dijalankan 2. Pengguna melakukan input data pengguna 3. Pengguna mencocokan respon
Hasil yang Diharapkan	Sistem dapat login pengguna
Hasil Pengujian	Sistem dapat login pengguna

Setelah dilakukan pengujian berdasarkan prosedur pengujian yang telah ditentukan maka diketahui bahwa sistem dapat melakukan login pengguna, informasi yang dimasukkan untuk login pengguna adalah alamat email pengujian@fungsional.com dan password “pengujian@fungsional”. setelah dilakukan *submit* data maka sistem akan menampilkan respon seperti Gambar 6.4.



Gambar 6.4 Pengujian fungsional login pengguna

6.1.10 CRUD (Manajemen) Perangkat IoT

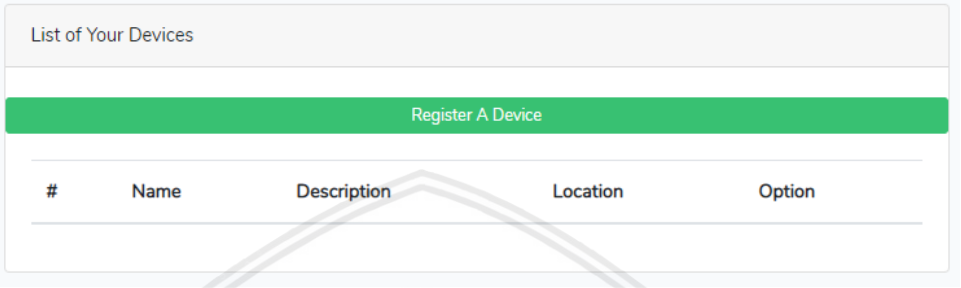
Kasus uji proses CRUD (manajemen) perangkat IoT ditunjukkan pada Tabel 6.16.

Tabel 6.16 Hasil pengujian fungsional CRUD (manajemen) perangkat IoT

Kode	FR_010
Nama Kasus Uji	CRUD (manajemen) perangkat IoT
Tujuan Pengujian	Mengetahui apakah sistem dapat CRUD (manajemen) perangkat IoT
Prosedur Pengujian	1. Sistem dijalankan 2. Pengguna melakukan CRUD dalam perangkat IoT 3. Pengguna mencocokan data dengan *database*
Hasil yang Diharapkan	Sistem dapat CRUD (manajemen) perangkat IoT
Hasil Pengujian	Sistem dapat CRUD (manajemen) perangkat IoT

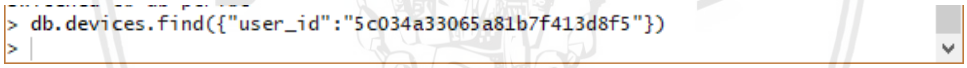
Dalam bagian CRUD atau manajemen terdapat beberapa komponen diantaranya *create* (pembuatan data), *read* (pembacaan data), *update* (pengubahan data), *delete* (penghapusan data) dan juga selain itu ada sebuah dashboard yang menyatukan itu semua yang bisa disebut sebagai *index.*

Setelah melakukan pengujian pada bagian *index*, maka dapat disimpulkan bahwa sistem dapat melakukan pengujian pada komponen itu, terlihat dari Gambar 6.5 yang menunjukan halaman perangkat dari pengguna.



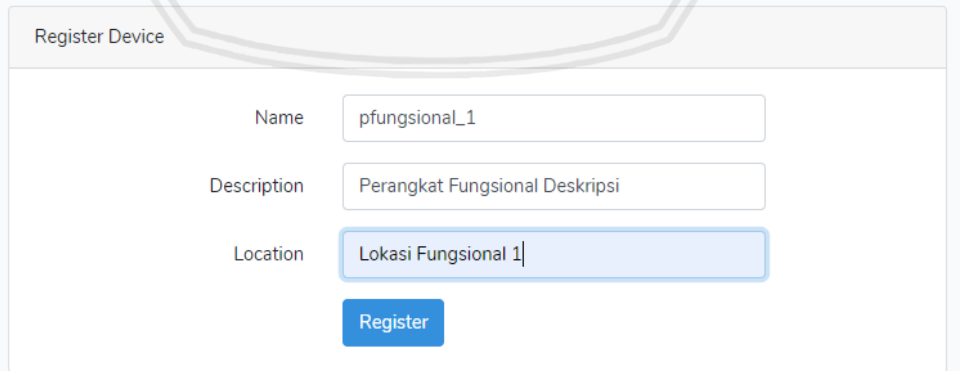
Gambar 6.5 Pengujian fungsional CRUD perangkat IoT (*index*)

Pada Gambar 6.5 ditunjukan tidak ada perangkat apapun yang dimiliki oleh pengguna yang terdaftar dalam *framework*, hal ini sesuai dengan kenyataan memang pengguna tersebut belum memiliki perangkat dalam sistem *framework*. Hal ini dapat divalidasi jika kita mencocokan dengan data di *database* seperti yang ada pada Gambar 6.6 dibawah ini.



Gambar 6.6 Pengujian fungsional CRUD perangkat IoT (*index database*)

Kemudian pada komponen *create* akan dilakukan pengujian fungsional, setelah diuji diketahui bahwa komponen tersebut dapat berfungsi dan dapat digunakan dengan baik. Seperti saat melakukan pendaftaran perangkat baru dengan nama “pfungsional_1”, seperti pada Gambar 6.7.



Gambar 6.7 Pengujian fungsional CRUD perangkat IoT (*create*)

Setelah dilakukan *submit* pada data Gambar 6.7, kemudian dilakukan pengecekan pada *database* maka dapat divalidasi bahwa perangkat tersebut berhasi didaftarkan dalam *framework* seperti yang tampak pada Gambar 6.8.

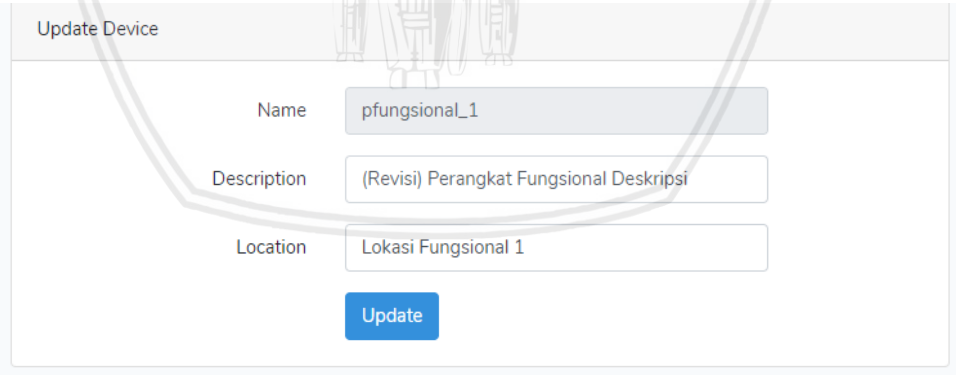
```
> db.devices.find({"user_id":"5c034a33065a81b7f413d8f5"})
{ "_id" : ObjectId("5c035266065a81b7f6602fc6"), "name" : "pfungsional_1", "nicen
ame" : "pfungsional_1", "description" : "Perangkat Fungsional Deskripsi", "locat
ion" : "Lokasi Fungsional 1", "token" : "2065cf55b9ec2a0d74418538a5ece027caca5c5
c", "secret" : "327d702463b25859c3d3c6a3a5f53966", "user_id" : "5c034a33065a81b7
f413d8f5", "updated_at" : ISODate("2018-12-02T03:32:54Z"), "created_at" : ISODat
e("2018-12-02T03:32:54Z") }
```


Gambar 6.8 Pengujian fungsional CRUD perangkat IoT (*create database*)

Kemudian untuk memvalidasi bahwa data juga dapat ditampilkan dalam *framework*, maka langkah selanjutnya adalah melakukan pengujian pada komponen *read* atau pembacaan data. setelah dilakukan pengujian maka dapat disimpulkan bahwa kebutuhan fungsional CRUD komponen *read* dapat dipenuhi seperti yang ditunjukkan pada Gambar 6.9.

Gambar 6.9 Pengujian fungsional CRUD perangkat IoT (*read*)

Hal yang perlu dilakukan pengujian berikutnya adalah pada komponen *update* atau pengubahan data dari sebuah perangkat IoT. setelah dilakukan pengujian dengan mencoba mengubah deskripsi pada perangkat "pfungsional_1" seperti pada ditunjukkan pada Gambar 6.10.



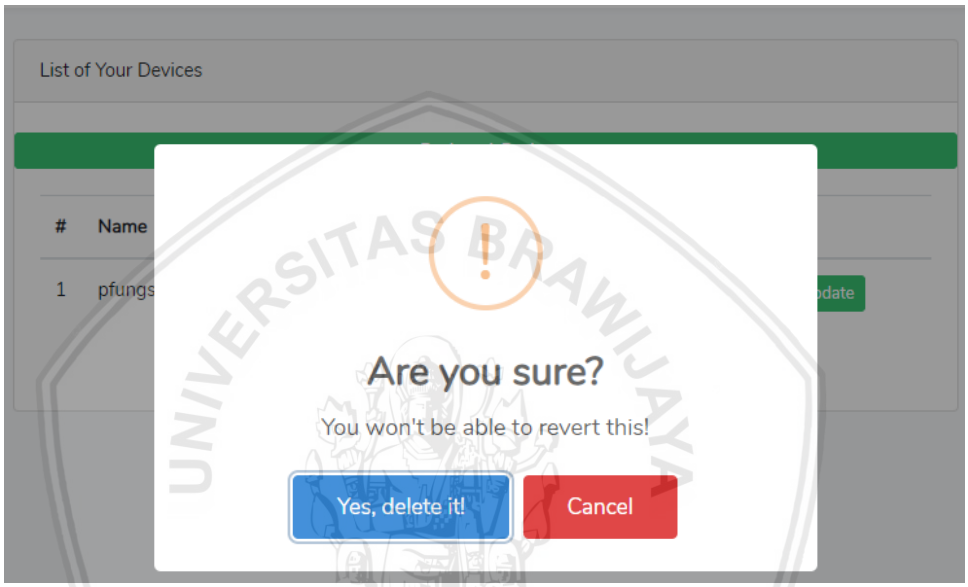
Gambar 6.10 Pengujian CRUD perangkat IoT (*update*)

Setelah dilakukan *submit* pada data Gambar 6.10, kemudian dilakukan pengecekan pada *database* maka dapat divalidasi bahwa perangkat tersebut berhasi diubah dalam *framework* seperti yang tampak pada Gambar 6.11.

```
> db.devices.find({"user_id":"5c034a33065a81b7f413d8f5"})
{ "_id" : ObjectId("5c035266065a81b7f6602fc6"), "name" : "pfungsional_1", "nicen
ame" : "pfungsional_1", "description" : "(Revisi) Perangkat Fungsional Deskripsi
", "location" : "Lokasi Fungsional 1", "token" : "2065cf55b9ec2a0d74418538a5ece0
27caca5c5c", "secret" : "327d702463b25859c3d3c6a3a5f53966", "user_id" : "5c034a3
3065a81b7f413d8f5", "updated_at" : ISODate("2018-12-02T03:51:47Z"), "created_at"
 : ISODate("2018-12-02T03:32:54Z") }
>
```


Gambar 6.11 Pengujian CRUD perangkat IoT (*update database*)

Komponen terakhir yang perlu diuji adalah komponen *delete* atau penghapusan data, dalam pengujian ini data yang akan diuji adalah perangkat dengan nama “pfungsional_1”seperti pada Gambar 6.12.



Gambar 6.12 Pengujian CRUD perangkat IoT (*delete*)

Setelah dilakukan konfirmasi penghapusan pada Gambar 6.12 maka perangkat akan terhapus pada sistem. Untuk memvalidasinya akan dilakukan pengecekan pada *database*. Berdasarkan pengecekan pada *database* seperti pada Gambar 6.13 diketahui bahwa data perangkat sudah terhapus sehingga dapat disimpulkan bahwa komponen *delete* pada CRUD sudah terpenuhi.


```
> db.devices.find({"user_id":"5c034a33065a81b7f413d8f5"})
>
```


Gambar 6.13 Pengujian CRUD perangkat IoT (*delete database*)

Berdasarkan pengujian pada setiap komponen CRUD maka dipastikan bahwa bahwa kebutuhan fungsional CRUD perangkat IoT sudah terpenuhi semuanya.

6.1.11 CRUD (Manajemen) Topik IoT

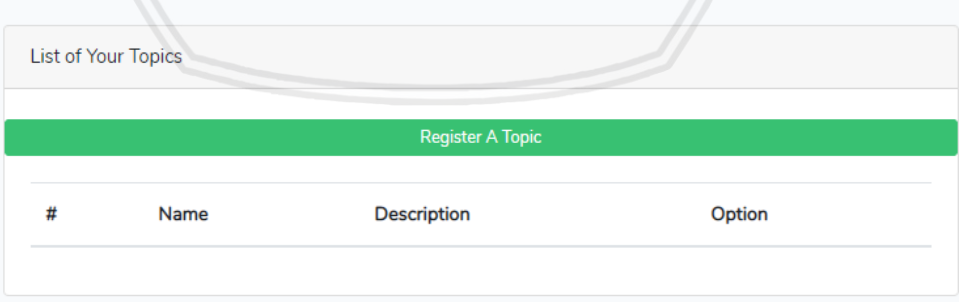
Kasus uji proses CRUD (manajemen) topik IoT ditunjukkan pada Tabel 6.17.

Tabel 6.17 Hasil pengujian fungsional CRUD (manajemen) topik IoT

Kode	FR_011
Nama Kasus Uji	CRUD (manajemen) topik IoT
Tujuan Pengujian	Mengetahui apakah sistem dapat CRUD (manajemen) topik IoT
Prosedur Pengujian	1. Sistem dijalankan 2. Pengguna melakukan CRUD dalam topik IoT 3. Pengguna mencocokan data dengan *database*
Hasil yang Diharapkan	Sistem dapat CRUD (manajemen) topik IoT
Hasil Pengujian	Sistem dapat CRUD (manajemen) topik IoT

Dalam bagian CRUD atau manajemen terdapat beberapa komponen diantaranya *create* (pembuatan data), *read* (pembacaan data), *update* (pengubahan data), *delete* (penghapusan data) dan juga selain itu ada sebuah dashboard yang menyatukan itu semua yang bisa disebut sebagai *index.*

Setelah melakukan pengujian pada bagian *index*, maka dapat disimpulkan bahwa sistem dapat melakukan pengujian pada komponen itu, terlihat dari Gambar 6.14 yang menunjukan halaman perangkat dari pengguna.



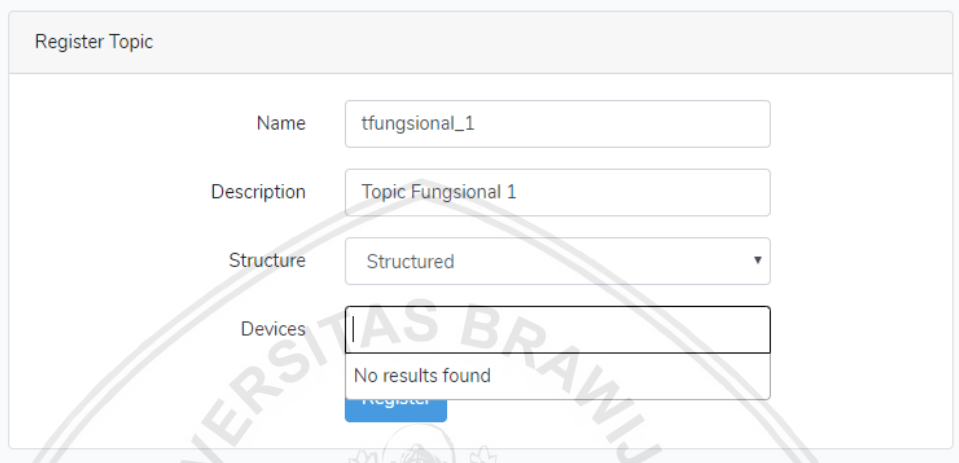
Gambar 6.14 Pengujian fungsional CRUD topik IoT (*index*)

Pada Gambar 6.14 ditunjukan tidak ada topik apapun yang dimiliki oleh pengguna yang terdaftar dalam *framework*, hal ini sesuai dengan kenyataan memang pengguna tersebut belum memiliki topik dalam sistem *framework*. Hal ini dapat divalidasi jika kita mencocokan dengan data di *database* seperti yang ada pada Gambar 6.15 dibawah ini.

```
> db.topics.find({"user_id":"5c034a33065a81b7f413d8f5"})
>
```

Gambar 6.15 Pengujian fungsional CRUD topik IoT (*index database*)

Kemudian pada komponen *create* akan dilakukan pengujian fungsional, setelah diuji diketahui bahwa komponen tersebut dapat berfungsi dan dapat digunakan dengan baik. Seperti saat melakukan pendaftaran topik baru dengan nama "tfungsional_1", seperti pada Gambar 6.16.



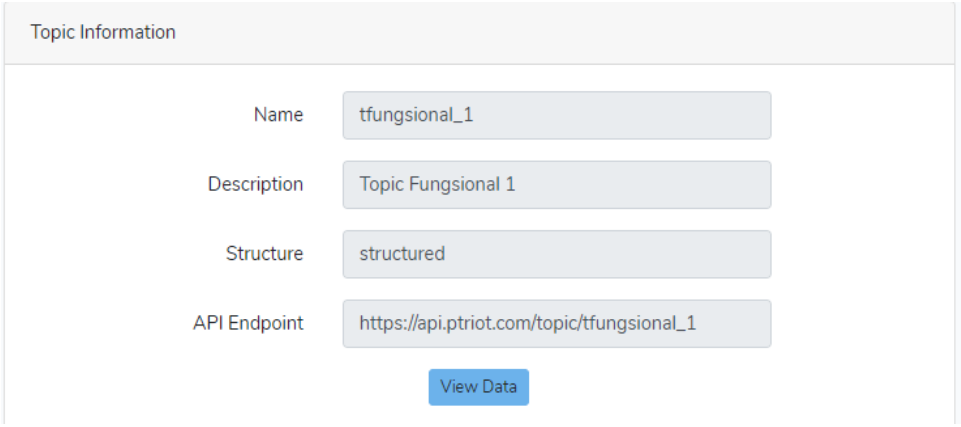
Gambar 6.16 Pengujian fungsional CRUD topik IoT (*create*)

Setelah dilakukan *submit* pada data Gambar 6.16, kemudian dilakukan pengecekan pada *database* maka dapat divalidasi bahwa topik tersebut berhasi didaftarkan dalam *framework* seperti yang tampak pada Gambar 6.17.

```
> db.topics.find({"user_id":"5c034a33065a81b7f413d8f5"})
{ "_id" : ObjectId("5c035e7a065a81b7f413d8f6"), "name" : "tfungsional_1", "nicen
ame" : "tfungsional_1", "description" : "Topic Fungsional 1", "structure" : "str
uctured", "user_id" : "5c034a33065a81b7f413d8f5", "updated_at" : ISODate("2018-1
2-02T04:24:26Z"), "created_at" : ISODate("2018-12-02T04:24:26Z") }
>
```

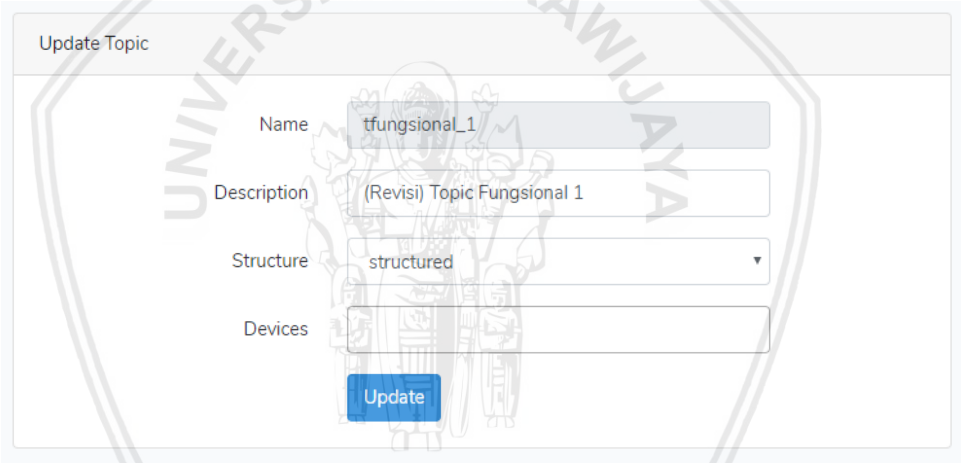
Gambar 6.17 Pengujian fungsional CRUD topik IoT (*create database*)

Kemudian untuk memvalidasi bahwa data juga dapat ditampilkan dalam *framework*, maka langkah selanjutnya adalah melakukan pengujian pada komponen *read* atau pembacaan data. setelah dilakukan pengujian maka dapat disimpulkan bahwa kebutuhan fungsional CRUD komponen *read* dapat dipenuhi seperti yang ditunjukkan pada Gambar 6.18.



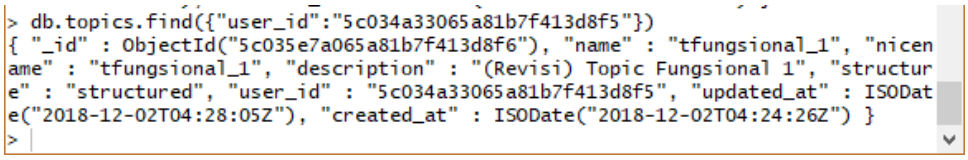
Gambar 6.18 Pengujian fungsional CRUD topik IoT (*read*)

Hal yang perlu dilakukan pengujian berikutnya adalah pada komponen *update* atau pengubahan data dari sebuah perangkat IoT. setelah dilakukan pengujian dengan mencoba mengubah deskripsi pada topik "tfungsional_1" seperti pada ditunjukkan pada Gambar 6.19.



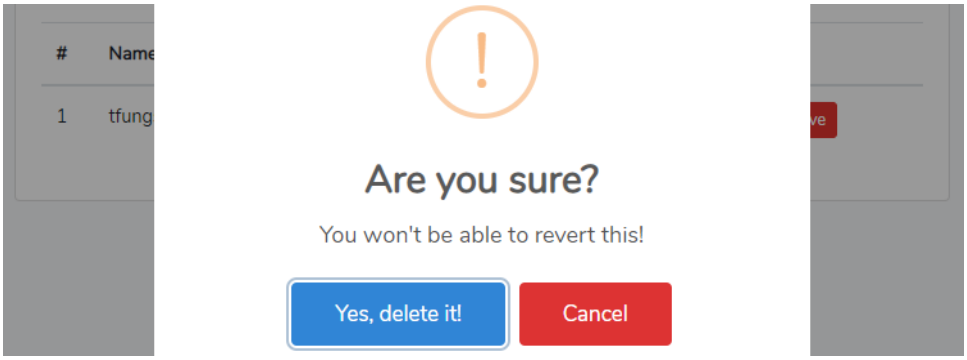
Gambar 6.19 Pengujian fungsional CRUD topik IoT (*update*)

Setelah dilakukan *submit* pada data Gambar 6.19, kemudian dilakukan pengecekan pada *database* maka dapat divalidasi bahwa topik tersebut berhasi diubah dalam *framework* seperti yang tampak pada Gambar 6.20.



Gambar 6.20 Pengujian fungsional CRUD topik IoT (*update database*)

Komponen terakhir yang perlu diuji adalah komponen *delete* atau penghapusan data, dalam pengujian ini data yang akan diuji adalah topik dengan nama "tfungsional_1"seperti pada Gambar 6.21.



Gambar 6.21 Pengujian fungsional CRUD topik IoT (*delete*)

Setelah dilakukan konfirmasi penghapusan pada Gambar 6.21 maka topik akan terhapus pada sistem. Untuk memvalidasinya akan dilakukan pengecekan pada *database*. Berdasarkan pengecekan pada *database* seperti pada Gambar 6.22 diketahui bahwa data topik sudah terhapus sehingga dapat disimpulkan bahwa komponen *delete* pada CRUD sudah terpenuhi.

```
> db.topics.find({"user_id":"5c034a33065a81b7f413d8f5"})
>
```

Gambar 6.22 Pengujian fungsional CRUD topik IoT (*delete database*)

Berdasarkan pengujian pada setiap komponen CRUD maka dipastikan bahwa bahwa kebutuhan fungsional CRUD topik IoT sudah terpenuhi semuanya.

6.1.12 CRUD (Manajemen) Pengguna

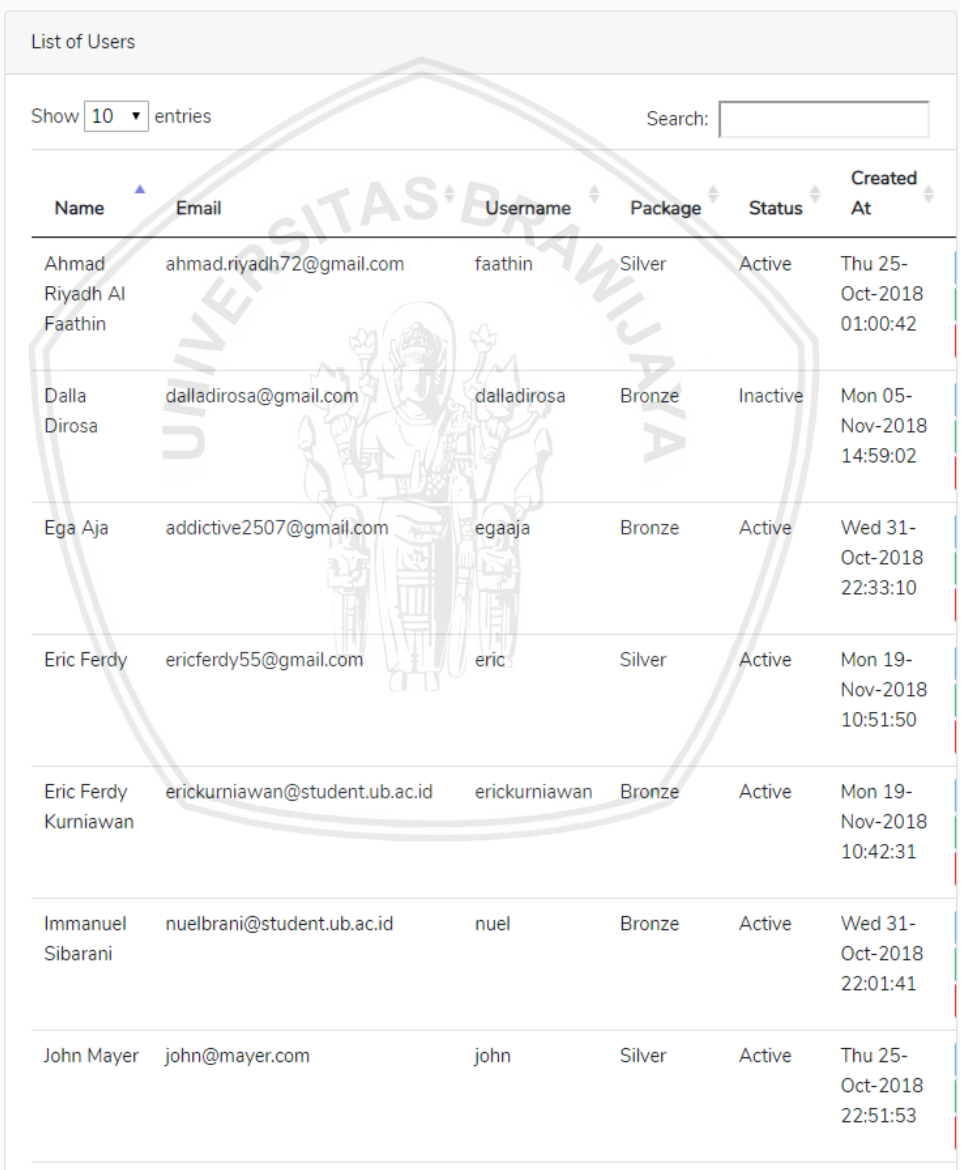
Kasus uji proses CRUD (manajemen) pengguna ditunjukkan pada Tabel 6.18.

Tabel 6.18 Hasil pengujian fungsional CRUD (manajemen) pengguna

Kode	FR_012
Nama Kasus Uji	CRUD (manajemen) pengguna
Tujuan Pengujian	Mengetahui apakah sistem dapat CRUD (manajemen) pengguna
Prosedur Pengujian	1. Sistem dijalankan 2. Pengguna melakukan CRUD dalam pengguna 3. Pengguna mencocokan data dengan *database*
Hasil yang Diharapkan	Sistem dapat CRUD (manajemen) pengguna
Hasil Pengujian	Sistem dapat CRUD (manajemen) pengguna

Dalam bagian CRUD atau manajemen terdapat beberapa komponen diantaranya *create* (pembuatan data), *read* (pembacaan data), *update* (pengubahan data), *delete* (penghapusan data) dan juga selain itu ada sebuah dashboard yang menyatukan itu semua yang bisa disebut sebagai *index.* Akan tetapi terdapat sedikit pengecualian pada kebutuhan fungsional ini, pada CRUD atau manajemen pengguna tidak akan ada komponen *create*, hal ini dikarenakan *create* akan dilakukan oleh pengguna sendiri melalui fitur registrasi pengguna.

Setelah melakukan pengujian pada bagian *index*, maka dapat disimpulkan bahwa sistem dapat melakukan pengujian pada komponen itu, terlihat dari Gambar 6.23 yang menunjukan halaman pengguna pada sistem.


List of Users

Show 10 entries Search:

Name	Email	Username	Package	Status	Created At
Ahmad Riyadh Al Faathin	ahmad.riyadh72@gmail.com	faathin	Silver	Active	Thu 25-Oct-2018 01:00:42
Dalla Dirosa	dalladirosa@gmail.com	dalladirosa	Bronze	Inactive	Mon 05-Nov-2018 14:59:02
Ega Aja	addictive2507@gmail.com	egaaja	Bronze	Active	Wed 31-Oct-2018 22:33:10
Eric Ferdy	ericferdy55@gmail.com	eric	Silver	Active	Mon 19-Nov-2018 10:51:50
Eric Ferdy Kurniawan	erickurniawan@student.ub.ac.id	erickurniawan	Bronze	Active	Mon 19-Nov-2018 10:42:31
Immanuel Sibarani	nuelbrani@student.ub.ac.id	nuel	Bronze	Active	Wed 31-Oct-2018 22:01:41
John Mayer	john@mayer.com	john	Silver	Active	Thu 25-Oct-2018 22:51:53

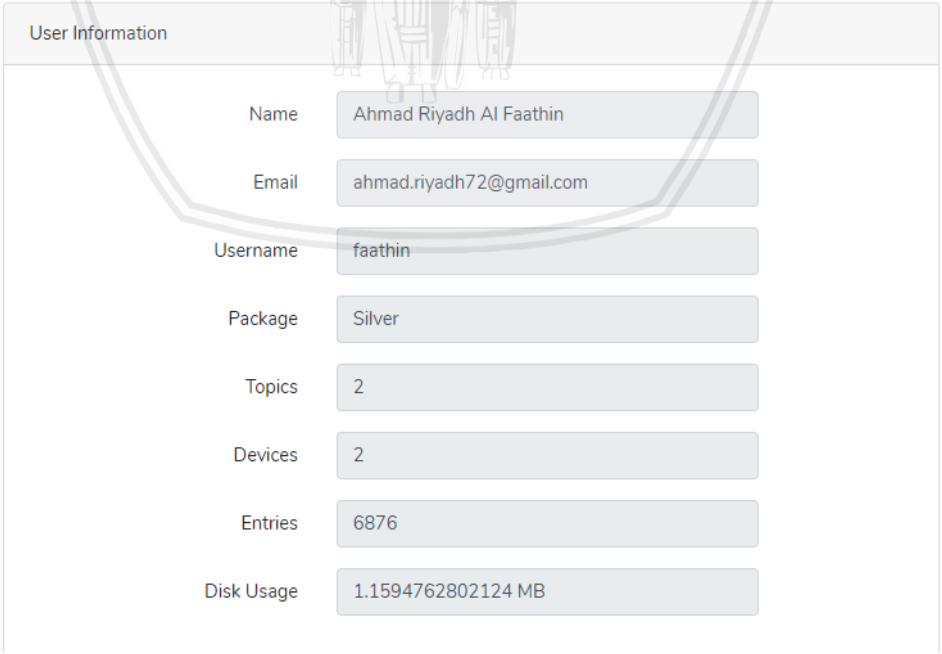
Gambar 6.23 Pengujian fungsional CRUD pengguna (*index*)

Pada Gambar 6.23 ditunjukan ada 11 pengguna yang terdaftar dalam *framework*, hal ini sesuai dengan kenyataan memang terdapat 11 pengguna dalam sistem *framework*. Hal ini dapat divalidasi jika kita mencocokan dengan data di *database* seperti yang ada pada Gambar 6.24 dibawah ini.

```
> db.users.find()
{ "_id" : ObjectId("5bd0b34a065a813063133a33"), "name" : "Ahmad Riyadh Al Faathi
n", "email" : "ahmad.riyadh72@gmail.com", "username" : "faathin", "password" : "
$2y$10$F1q/fpqBPBbfgLRUjndH/OGA7Uz4iGGXc7fZpNhBMom4HqVh2TeZi", "package" : "silv
er", "database_name" : "faathin", "updated_at" : ISODate("2018-11-06T07:04:26Z")
, "created_at" : ISODate("2018-10-24T18:00:42Z"), "remember_token" : "hXxR6VOjs7
WXFRq8h9Qu4ZswII4Y9ARU9mvWSJgmgx2afY1GTIbPCmGm7kTm", "status" : "active" }
{ "_id" : ObjectId("5bd1e699065a81078c4614c2"), "name" : "John Mayer", "email" :
 "john@mayer.com", "username" : "john", "password" : "$2y$10$ClAFOCWa.hOLBojhc18
6TOHyNyV3SNrlXeQDO/EmOymaq1WiUNPIm", "package" : "silver", "status" : "active",
"database_name" : "john", "updated_at" : ISODate("2018-11-08T09:40:11Z"), "creat
ed_at" : ISODate("2018-10-25T15:51:53Z"), "remember_token" : "JQaBkg6O69s9i89sin
ZjyOTrQPxF8IZPKlGXsmqzI84ghawgUmAFkNI6j559" }
{ "_id" : ObjectId("5bd9c3d5065a81a83512e05b"), "name" : "Immanuel Sibarani", "e
mail" : "nuelbrani@student.ub.ac.id", "username" : "nuel", "password" : "$2y$10$
7KF/RropiZLfHbRwg2kPC.Cgxi6SuOGEN72tbEoUu4mWObqyLwvbO", "package" : "bronze", "s
tatus" : "active", "database_name" : "nuel", "updated_at" : ISODate("2018-11-08T
05:54:32Z"), "created_at" : ISODate("2018-10-31T15:01:41Z"), "remember_token" :
"ihYYM1aOQQ4W6K6L6UM2mSOLa3fbiICp1PMxLsLGKs9vkci2IpOPIlc2pnuC" }
{ "_id" : ObjectId("5bd9cb36065a81bbcc0d6ed5"), "name" : "Ega Aja", "email" : "a
ddictive2507@gmail.com", "username" : "egaaja", "password" : "$2y$10$IjcHRr2MBMD
miEHYyGJOGOHzV3SuQgur.e2ld61ksEqBh9H871kka", "package" : "bronze", "status" : "a
ctive", "database_name" : "egaaja", "updated_at" : ISODate("2018-11-01T03:29:17Z
"), "created_at" : ISODate("2018-10-31T15:33:10Z"), "remember_token" : "2C6FZwBq
```

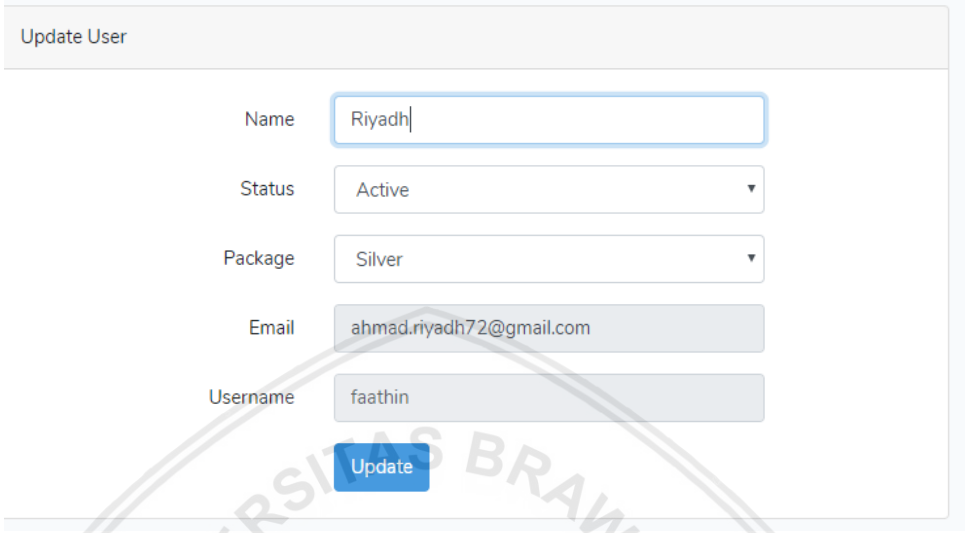

Gambar 6.24 Pengujian fungsional CRUD pengguna (*index database*)

Kemudian untuk memvalidasi bahwa data juga dapat ditampilkan dalam *framework*, maka langkah selanjutnya adalah melakukan pengujian pada komponen *read* atau pembacaan data. setelah dilakukan pengujian maka dapat disimpulkan bahwa kebutuhan fungsional CRUD komponen *read* dapat dipenuhi seperti yang ditunjukkan pada Gambar 6.25.



Gambar 6.25 Pengujian fungsional CRUD (*read*)

Hal yang perlu dilakukan pengujian berikutnya adalah pada komponen *update* atau pengubahan data dari sebuah pengguna. setelah dilakukan pengujian dengan mencoba mengubah nama pada pengguna “Ahmad Riyadh Al Faathin” seperti pada ditunjukkan pada Gambar 6.26.

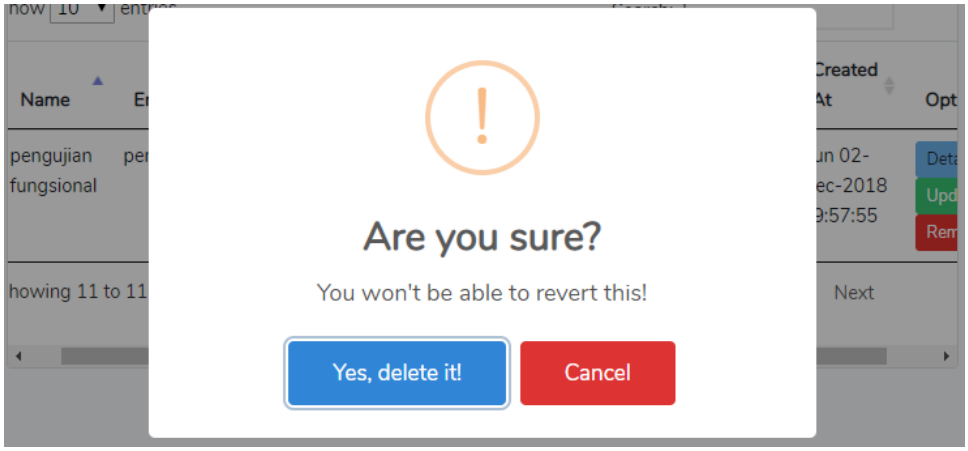


Gambar 6.26 Pengujian fungsional CRUD pengguna (*update*)

Setelah dilakukan *submit* pada data Gambar 6.26, kemudian dilakukan pengecekan pada *database* maka dapat divalidasi bahwa pengguna tersebut berhasi diubah dalam *framework* seperti yang tampak pada Gambar 6.27.

Gambar 6.27 Pengujian fungsional CRUD pengguna (*update database*)

Komponen terakhir yang perlu diuji adalah komponen *delete* atau penghapusan data, dalam pengujian ini data yang akan diuji adalah pengguna dengan nama “pengujian fungsional”seperti pada Gambar 6.28.



Gambar 6.28 Pengujian fungsional CRUD pengguna (*delete*)

Setelah dilakukan konfirmasi penghapusan pada Gambar 6.28 maka pengguna akan terhapus pada sistem. Untuk memvalidasinya akan dilakukan pengecekan pada *database*. Berdasarkan pengecekan pada *database* seperti pada Gambar 6.29 diketahui bahwa data pengguna sudah terhapus sehingga dapat disimpulkan bahwa komponen *delete* pada CRUD sudah terpenuhi.

```
> db.users.find({"name":"pengujian fungsional"})
>
```

Gambar 6.29 Pengujian fungsional CRUD pengguna (*delete database*)

Berdasarkan pengujian pada setiap komponen CRUD maka dipastikan bahwa bahwa kebutuhan fungsional CRUD topik IoT sudah terpenuhi semuanya.

6.1.13 Menampilkan Daftar Data yang Disimpan Dalam Suatu Topik

Kasus uji proses menampilkan daftar data yang disimpan dalam suatu topik ditunjukkan pada Tabel 6.19 dan 6.20.

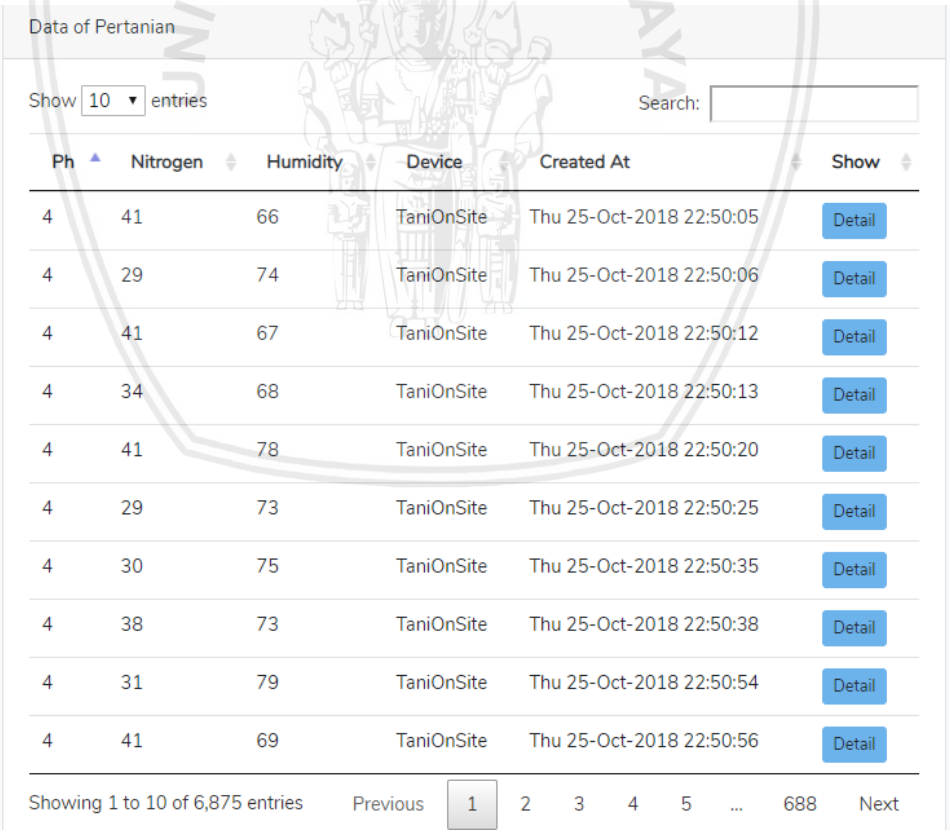
Tabel 6.19 Hasil pengujian fungsional menampilkan daftar data yang disimpan dalam suatu topik (bagian 1)

Kode	FR_013
Nama Kasus Uji	Menampilkan daftar data yang disimpan dalam suatu topik
Tujuan Pengujian	Mengetahui apakah sistem dapat menampilkan daftar data yang disimpan dalam suatu topik

Tabel 6.20 Hasil pengujian fungsional menampilkan daftar data yang disimpan dalam suatu topik (bagian 2)

Prosedur Pengujian	1. Sistem dijalankan 2. Pengguna memilih topik 3. Pengguna mencocokan hasil keluaran dari sistem dengan *database*
Hasil yang Diharapkan	Sistem dapat menampilkan daftar data yang disimpan dalam suatu topik
Hasil Pengujian	Sistem dapat menampilkan daftar data yang disimpan dalam suatu topik

Setelah dilakukan pengujian berdasarkan prosedur pengujian yang telah ditentukan maka diketahui bahwa sistem dapat menampilkan daftar data yang disimpan dalam suatu topik, seperti ditunjukkan pada Gambar 6.30. Terlihat jika proses menampilkan data berhasil apabila *framework* mengembalikan informasi daftar data dari sebuah topik.



Gambar 6.30 Pengujian fungsional menampilkan daftar data yang disimpan dalam suatu topik

Langkah berikutnya adalah mencocokan dengan data di dalam *database*. Untuk melakukannya maka diperlukan login ke sistem vps untuk mendapatkan data dari dalam *database* MongoDB. setelah dilakukan *query* terhadap *database* maka dipastikan bahwa data dapat disimpan dan cocok dengan hasil keluaran dari proses sebelumnya seperti pada Gambar 6.31.

Gambar 6.31 Pengujian menampilkan daftar data yang disimpan dalam suatu topik (*database*)

6.1.14 Menampilkan Data yang Disimpan Dalam Suatu Topik

Kasus uji proses menampilkan data yang disimpan dalam suatu topik ditunjukkan pada Tabel 6.21 dan 6.22.

Tabel 6.21 Hasil pengujian fungsional menampilkan data yang disimpan dalam suatu topik (bagian 1)

Kode	FR_014
Nama Kasus Uji	Menampilkan daftar data yang disimpan dalam suatu topik
Tujuan Pengujian	Mengetahui apakah sistem dapat menampilkan data yang disimpan dalam suatu topik
Prosedur Pengujian	1. Sistem dijalankan 2. Pengguna memilih topik dan id data yang dipilh 3. Pengguna mencocokan hasil keluaran dari sistem dengan *database*

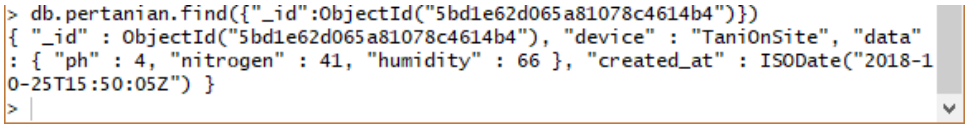
Tabel 6.22 Hasil pengujian fungsional menampilkan data yang disimpan dalam suatu topik (bagian 2)

Hasil yang Diharapkan	Sistem dapat menampilkan data yang disimpan dalam suatu topik
Hasil Pengujian	Sistem dapat menampilkan data yang disimpan dalam suatu topik

Setelah dilakukan pengujian berdasarkan prosedur pengujian yang telah ditentukan maka diketahui bahwa sistem dapat menampilkan data yang disimpan dalam suatu topik, seperti ditunjukkan pada Gambar 6.32. Terlihat jika proses menampilkan data berhasil apabila *framework* mengembalikan informasi data dari sebuah topik.

Gambar 6.32 Pengujian fungsional menampilkan daftar data yang disimpan dalam suatu topik

Langkah berikutnya adalah mencocokan dengan data di dalam *database*. Untuk melakukannya maka diperlukan login ke sistem vps untuk mendapatkan data dari dalam *database* MongoDB. setelah dilakukan *query* terhadap *database* maka dipastikan bahwa data dapat disimpan dan cocok dengan hasil keluaran dari proses sebelumnya seperti pada Gambar 6.33.



Gambar 6.33 Pengujian fungsional menampilkan daftar data yang disimpan dalam suatu topik (*database*)

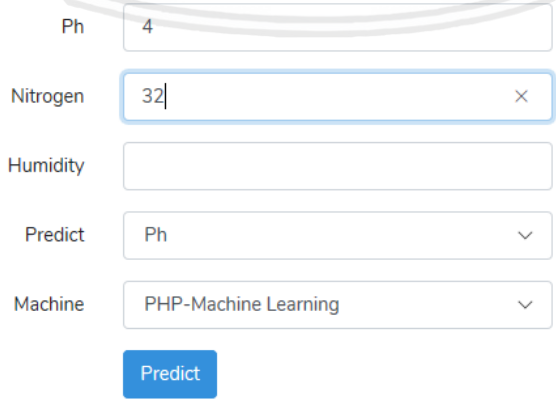
6.1.15 Prediksi Data Menggunakan Least Square (personalisasi)

Kasus uji proses prediksi data menggunakan least square ditunjukkan pada Tabel 6.23.

Tabel 6.23 Hasil pengujian fungsional prediksi data menggunakan least square

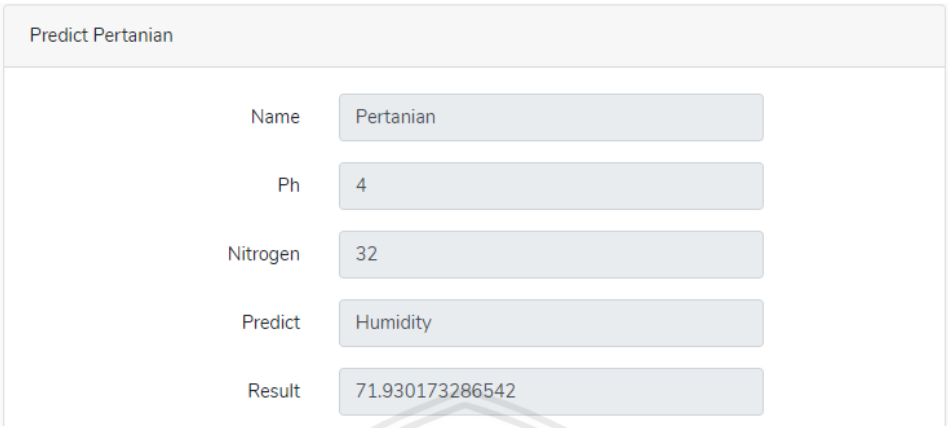
Kode	FR_015
Nama Kasus Uji	Prediksi data menggunakan least square
Tujuan Pengujian	Mengetahui apakah sistem dapat prediksi data menggunakan least square
Prosedur Pengujian	1. Sistem dijalankan 2. Pengguna memilih topik dan memilih modul analitik least square 3. Pengguna memasukkan data yang ingin diprediksi 4. Pengguna melihat hasil keluaran dari sistem
Hasil yang Diharapkan	Sistem dapat prediksi data menggunakan least square
Hasil Pengujian	Sistem dapat prediksi data menggunakan least square

Setelah dilakukan pengujian berdasarkan prosedur pengujian yang telah ditentukan maka diketahui bahwa sistem dapat prediksi data menggunakan least square. Seperti ditunjukkan pada Gambar 6.34 saat melakukan input data.



Gambar 6.34 Pengujian fungsional prediksi data menggunakan least square (masukkan pada PHP-ML)

Terlihat jika proses prediksi data menggunakan least square berhasil apabila hasil prediksi dapat dikeluarkan seperti pada Gambar 6.35.



Gambar 6.35 Pengujian fungsional prediksi data menggunakan least square (keluaran pada PHP-ML)

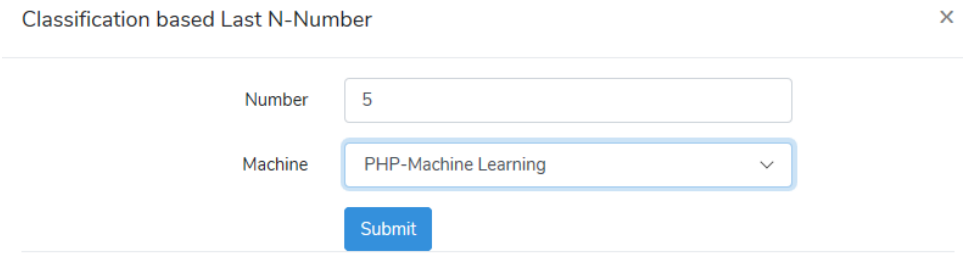
6.1.16 Klasifikasi Data Menggunakan Naïve bayes (personalisasi)

Kasus uji proses klasifikasi data menggunakan Naïve bayes ditunjukkan pada Tabel 6.24.

Tabel 6.24 Hasil pengujian fungsional klasifikasi data menggunakan Naïve bayes

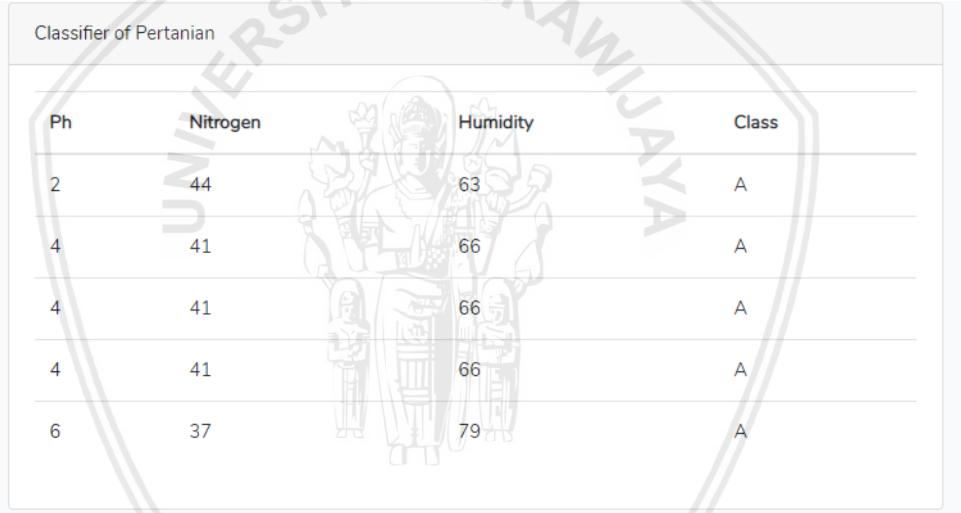
Kode	FR_016
Nama Kasus Uji	Klasifikasi data menggunakan Naïve bayes
Tujuan Pengujian	Mengetahui apakah sistem dapat klasifikasi data menggunakan Naïve bayes
Prosedur Pengujian	1. Sistem dijalankan 2. Pengguna memilih topik dan memilih modul analitik Naïve bayes 3. Pengguna memasukkan data yang ingin diklasifikasikan 4. Pengguna melihat hasil keluaran dari sistem
Hasil yang Diharapkan	Sistem dapat klasifikasi data menggunakan Naïve bayes
Hasil Pengujian	Sistem dapat klasifikasi data menggunakan Naïve bayes

Setelah dilakukan pengujian berdasarkan prosedur pengujian yang telah ditentukan maka diketahui bahwa sistem dapat klasifikasi data menggunakan naïve bayes, seperti ditunjukkan pada Gambar 6.36 saat melakukan input data.



Gambar 6.36 Pengujian fungsional klasifikasi data menggunakan naïve bayes (masukkan pada PHP-ML)

Terlihat jika proses klasifikasi data menggunakan naïve bayes berhasil apabila hasil klasifikasi dapat dikeluarkan seperti pada Gambar 6.37.



Gambar 6.37 Pengujian fungsional klasifikasi data menggunakan naïve bayes (keluaran pada PHP-ML)

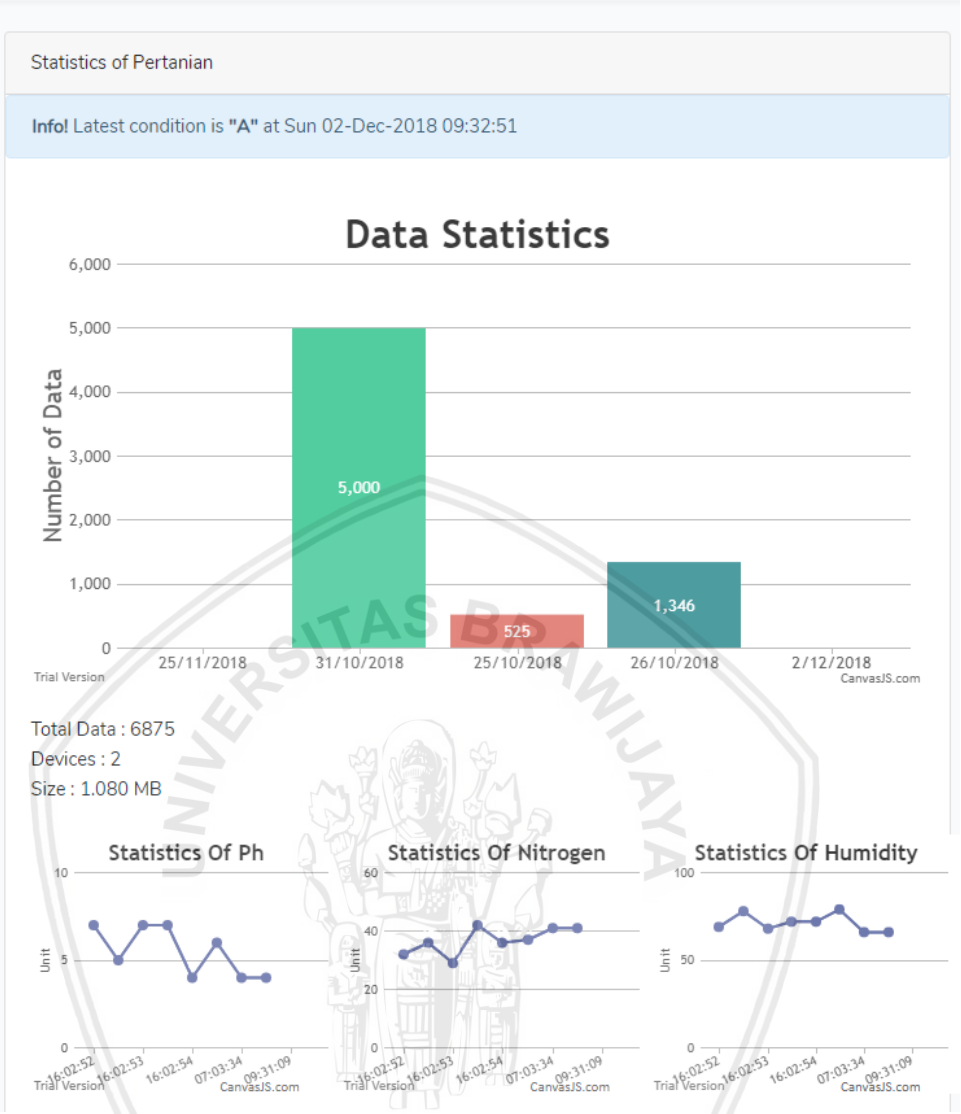
6.1.17 Menampilkan Informasi Statistik dari Sebuah Topik

Kasus uji proses menampilkan informasi statistik dari sebuah topik ditunjukkan pada Tabel 6.25.

Tabel 6.25 Hasil pengujian fungsional menampilkan informasi statistik dari sebuah topik

Kode	FR_016
Nama Kasus Uji	Menampilkan informasi statistik dari sebuah topik
Tujuan Pengujian	Mengetahui apakah sistem dapat menampilkan informasi statistik dari sebuah topik
Prosedur Pengujian	1. Sistem dijalankan 2. Pengguna memilih topik 3. Pengguna melihat hasil keluaran dari sistem
Hasil yang Diharapkan	Sistem dapat menampilkan informasi statistik dari sebuah topik
Hasil Pengujian	Sistem dapat menampilkan informasi statistik dari sebuah topik

Setelah dilakukan pengujian berdasarkan prosedur pengujian yang telah ditentukan maka diketahui bahwa Sistem dapat menampilkan informasi statistik dari sebuah topik, seperti ditunjukkan pada Gambar 6.38 saat menampilkan hasil statistik data.



Gambar 6.38 Pengujian fungsional menampilkan informasi statistik dari sebuah topik

6.2 Pengujian Skalabilitas

Pengujian skalabilitas dilakukan untuk mengetahui seberapa banyak data yang dapat dikirim dengan menggunakan *POST*, seberapa banyak data yang dapat diambil dengan menggunakan *GET* dan juga kemampuan sistem dalam melakukan analitik data. Pengujian ini dilakukan menggunakan tool yaitu JMeter untuk pengujian *POST dan GET*, dan juga Chrome untuk pengujian analitik data.

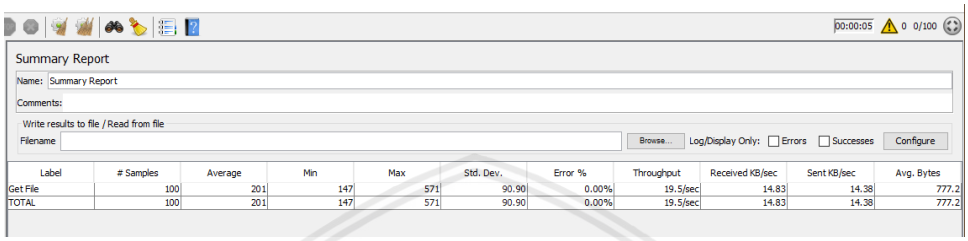
6.2.1 Pengujian Pengambilan Data

Dalam melakukan pengujian ini, penguji melakukan pengambilan data yang ada pada *data storage*. Skenario yang diujikan dengan membuat 100, 300, 500, 700 dan 1000 *request* data ke *framework*. Banyak data yang dapat berhasil diambil dan lamanya proses pengambilan data bergantung pada koneksi internet yang digunakan. Data yang akan diambil sendiri dapat dilihat pada Lampiran

B.2.1.1, sedangkan parameter pengambilan data dapat dilihat pada Lampiran B.2.1.2.

6.2.1.1 Skenario 1 : Melakukan pengambilan data sebanyak 100 request data

Pada skenario pengujian ini, penguji melakukan eksperimen pengambilan data sebanyak 100 *request* dan diperoleh hasil seperti pada Gambar 6.39 yang menunjukkan persentase keberhasilan pengambilan data beserta waktu yang digunakan untuk mengambil data.

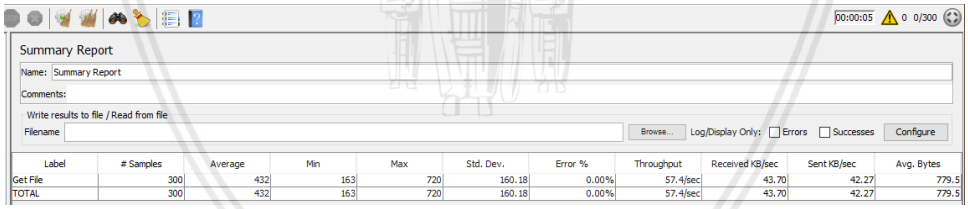


Gambar 6.39 Pengujian skalabilitas *request* 50 data

Dari eksperimen pengujian 100 *request* diperoleh hasil bahwa keseluruhan data berhasil diambil dengan persentase error 0%. Eksperimen menghabiskan waktu selama 5 detik.

6.2.1.2 Skenario 2 : Melakukan pengambilan data sebanyak 300 request data

Pada skenario pengujian ini, penguji melakukan eksperimen pengambilan data sebanyak 300 *request* dan diperoleh hasil seperti pada Gambar 6.40 yang menunjukkan persentase keberhasilan pengambilan data beserta waktu yang digunakan untuk mengambil data.

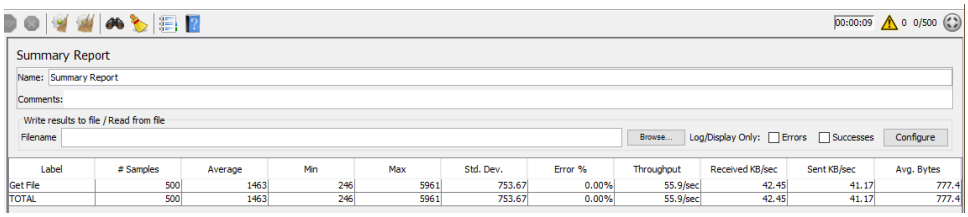


Gambar 6.40 Pengujian skalabilitas *request* 300 data

Dari eksperimen pengujian 300 *request* diperoleh hasil bahwa keseluruhan data berhasil diambil dengan persentase error 0%. Eksperimen menghabiskan waktu selama 5 detik.

6.2.1.3 Skenario 3 : Melakukan pengambilan data sebanyak 500 request data

Pada skenario pengujian ini, penguji melakukan eksperimen pengambilan data sebanyak 500 *request* dan diperoleh hasil seperti pada Gambar 6.41 yang menunjukkan persentase keberhasilan pengambilan data beserta waktu yang digunakan untuk mengambil data.

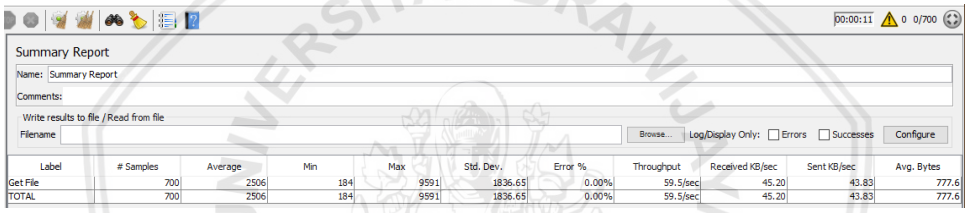


Gambar 6.41 Pengujian skalabilitas *request* 500 data

Dari eksperimen pengujian 500 *request* diperoleh hasil bahwa keseluruhan data berhasil diambil dengan persentase error 0%. Eksperimen menghabiskan waktu selama 9 detik.

6.2.1.4 Skenario 4 : Melakukan pengambilan data sebanyak 700 request data

Pada skenario pengujian ini, penguji melakukan eksperimen pengambilan data sebanyak 700 *request* dan diperoleh hasil seperti pada Gambar 6.42 yang menunjukkan persentase keberhasilan pengambilan data beserta waktu yang digunakan untuk mengambil data.

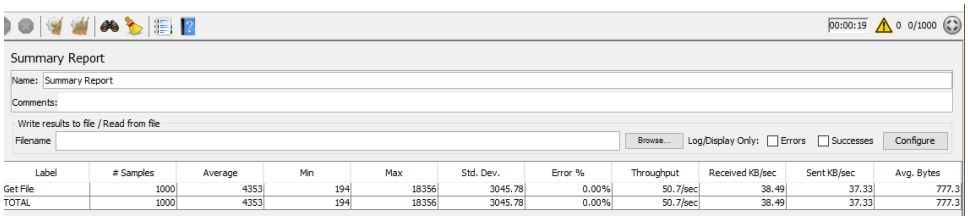


Gambar 6.42 Pengujian skalabilitas *request* 200 data

Dari eksperimen pengujian 700 *request* diperoleh hasil bahwa keseluruhan data berhasil diambil dengan persentase error 0%. Eksperimen menghabiskan waktu selama 11 detik.

6.2.1.5 Skenario 5 : Melakukan pengambilan data sebanyak 1000 request data

Pada skenario pengujian ini, penguji melakukan eksperimen pengambilan data sebanyak 1000 *request* dan diperoleh hasil seperti pada Gambar 6.43 yang menunjukkan persentase keberhasilan pengambilan data beserta waktu yang digunakan untuk mengambil data.

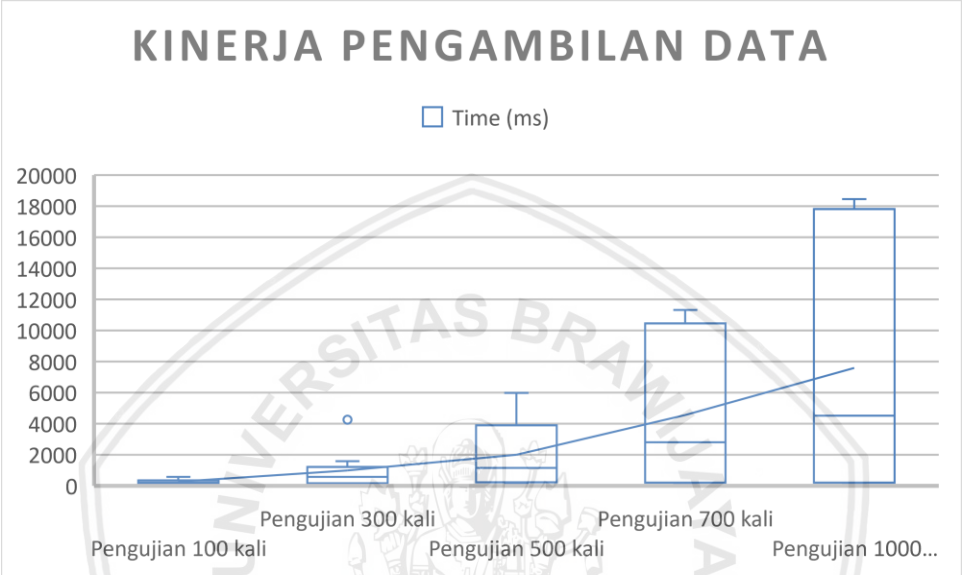


Gambar 6.43 Pengujian skalabilitas *request* 1000 data

Dari eksperimen pengujian 1000 *request* diperoleh hasil bahwa keseluruhan data berhasil diambil dengan persentase error 0%. Eksperimen menghabiskan waktu selama 19 detik.

6.2.1.6 Pembahasan

Setelah dilakukan percobaan eksperimen pengambilan data pada sistem menggunakan method *GET* menggunakan 5 (lima) skenario didapatkan hasil seperti berikut. Kinerja *framework* terpengaruh dalam 5 (lima) skenario yang dilakukan, terbukti dengan waktu *response* pada sistem yang cenderung naik seiring meningkatnya jumlah *request* yang dilakukan, seperti yang terlihat pada Gambar 6.45.



Gambar 6.44 Pembahasan kinerja pengambilan data

Terlihat saat dilakukan pengujian dengan 100 *request*, *average response time* adalah sekitar 201 ms, yang kemudian naik dalam pengujian dengan 300 *request* menjadi 983 ms, dan kemudian naik lagi menjadi 1991 ms saat dilakukan pengujian dengan 500 *request,* yang pada pengujian selanjutnya juga menunjukkan kenaikan *average time* menjadi 4543 ms saat dilakukan 700 *request*, dan berlanjut naik dalam titik tertinggi *average* menjadi 7569 ms pada pengujian dengan 1000 *request*. Dengan kinerja seperti ini maka dapat dikatakan bahwa *framework* bekerja cukup baik untuk standar yang ditetapkan oleh Jakob Nielsen dalam bukunya *Usability Engineering* yaitu 10.000 ms untuk pengguna merasakan bahwa sistem dapat digunakan secara nyaman (Jakob Nielsen, 1993). Akan tetapi dengan nilai seperti ini sistem juga dapat diperbaiki lagi khususnya untuk menangani *request* yang tinggi diatas 500 *request* secara serempak agar dapat mencapai nilai yang dianjurkan dalam buku Usability Engineering yaitu 1000 ms agar sistem dapat digunakan secara sangat interaktif (Jakob Nielsen, 1993).

Hasil lengkap dari pengujian dapat dilihat dalam Lampiran B.2.1.3.

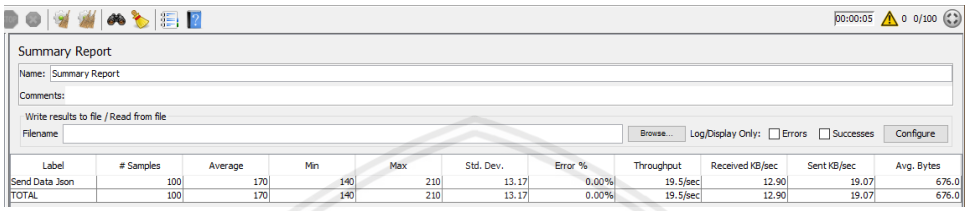
6.2.2 Pengujian Pengiriman Data

Dalam melakukan pengujian ini, penguji melakukan pengiriman data JSON ke dalam *data storage*. Skenario yang diujikan dengan membuat 100,300,500,700, dan 1.000 kali *POST* data ke *framework*. Banyak data yang dapat berhasil diambil

dan lamanya proses pengiriman data bergantung pada koneksi internet yang digunakan. Data yang akan diambil sendiri dapat dilihat pada Lampiran B.2.2.1, sedangkan parameter pengambilan data dapat dilihat pada Lampiran B.2.2.2.

6.2.2.1 Skenario 1 : Melakukan POST data sebanyak 100 kali

Pada skenario pengujian ini, penguji melakukan eksperimen *POST* data sebanyak 100 kali dan diperoleh hasil seperti pada Gambar 6.61 yang menunjukkan persentase keberhasilan pengiriman data beserta waktu yang digunakan untuk mengirim data.

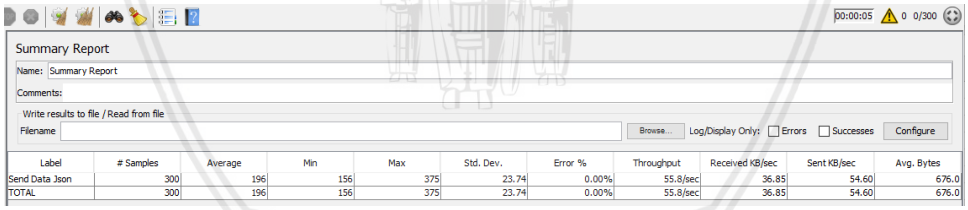


Gambar 6.45 Pengujian skalabilitas *POST* 100 data

Dari eksperimen pengujian 100 kali didapatkan hasil bahwa data yang berhasil dikirim yaitu 100 data dengan persentase error 0%. Eksperimen menghabiskan waktu selama 5 detik.

6.2.2.2 Skenario 2 : Melakukan POST data sebanyak 300 kali

Pada skenario pengujian ini, penguji melakukan eksperimen *POST* data sebanyak 300 kali dan diperoleh hasil seperti pada Gambar 6.62 yang menunjukkan persentase keberhasilan pengiriman data beserta waktu yang digunakan untuk mengirim data.

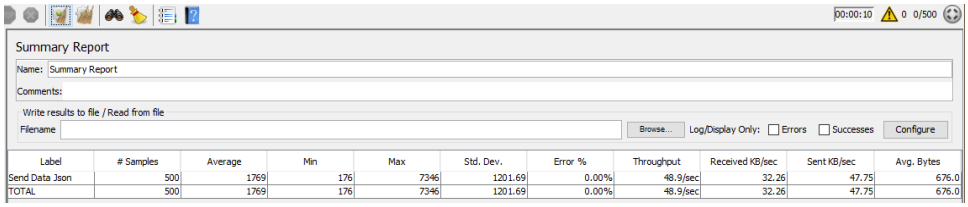


Gambar 6.46 Pengujian skalabilitas *POST* 300 data

Dari eksperimen pengujian 300 kali didapatkan hasil bahwa data yang berhasil dikirim yaitu 300 data dengan persentase error 0%. Eksperimen menghabiskan waktu selama 5 detik.

6.2.2.3 Skenario 3 : Melakukan POST data sebanyak 500 kali

Pada skenario pengujian ini, penguji melakukan eksperimen *POST* data sebanyak 500 kali dan diperoleh hasil seperti pada Gambar 6.63 yang menunjukkan persentase keberhasilan pengiriman data beserta waktu yang digunakan untuk mengirim data.

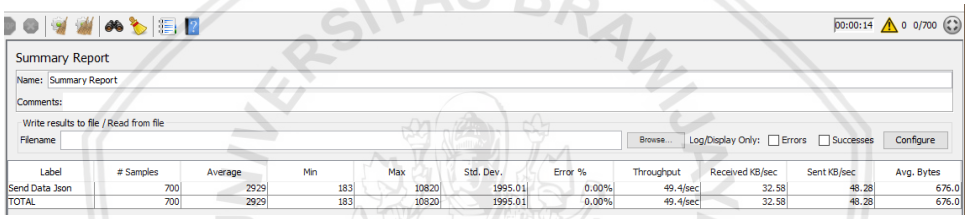


Gambar 6.47 Pengujian skalabilitas *POST* 500 data

Dari eksperimen pengujian 500 kali didapatkan hasil bahwa data yang berhasil dikirim yaitu 500 data dengan persentase error 0%. Eksperimen menghabiskan waktu selama 10 detik.

6.2.2.4 Skenario 4 : Melakukan POST data sebanyak 700 kali

Pada skenario pengujian ini, penguji melakukan eksperimen *POST* data sebanyak 700 kali dan diperoleh hasil seperti pada Gambar 6.64 yang menunjukkan persentase keberhasilan pengiriman data beserta waktu yang digunakan untuk mengirim data.

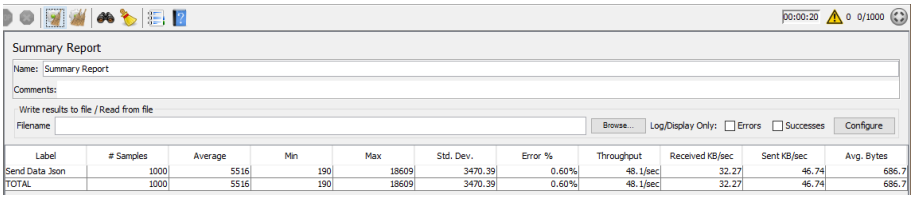


Gambar 6.48 Pengujian skalabilitas *POST* 700 data

Dari eksperimen pengujian 700 kali didapatkan hasil bahwa data yang berhasil dikirim yaitu 700 data dengan persentase error 0%. Eksperimen menghabiskan waktu selama 14 detik.

6.2.2.5 Skenario 5 : Melakukan POST data sebanyak 1.000 kali

Pada skenario pengujian ini, penguji melakukan eksperimen *POST* data sebanyak 1.000 kali dan diperoleh hasil seperti pada Gambar 6.65 yang menunjukkan persentase keberhasilan pengiriman data beserta waktu yang digunakan untuk mengirim data.

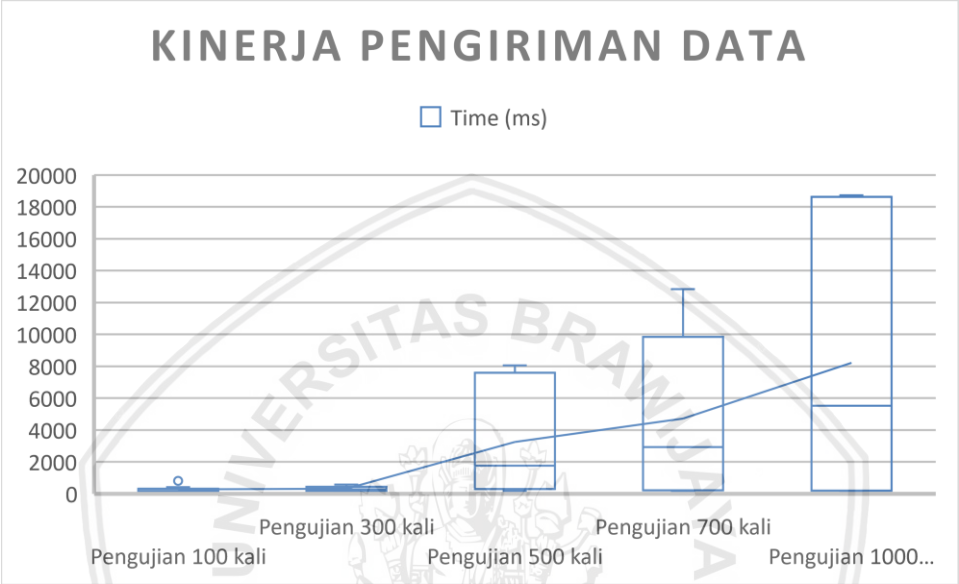


Gambar 6.49 Pengujian skalabilitas *POST* 1000 data

Dari eksperimen pengujian 1.000 kali didapatkan hasil bahwa data yang berhasil dikirim yaitu 993 data dengan persentase error 0.60%. Eksperimen menghabiskan waktu selama 20 detik.

6.2.2.6 Pembahasan

Setelah dilakukan percobaan eksperimen pengiriman data pada sistem menggunakan method *POST* menggunakan 5 (lima) skenario didapatkan hasil seperti berikut. Kinerja *framework* terpengaruh dalam 5 (lima) skenario yang dilakukan, terbukti dengan waktu *response* pada sistem yang cenderung naik secara eksponensial seiring jumlah data yang dikirim, seperti yang terlihat pada Gambar 6.66.



Gambar 6.50 Pembahasan kinerja pengiriman data

Terlihat saat dilakukan pengujian dengan 100 *POST*, *average response time* adalah sekitar 269 ms, yang kemudian naik sedikit dalam pengujian dengan 300 *POST* menjadi 294 ms, dan kemudian naik secara drastis menjadi 3242 ms saat dilakukan pengujian dengan 500 *POST*, yang pada pengujian selanjutnya terjadi peningkatan lagi *average time* menjadi 4206 ms saat dilakukan 700 *POST*, dan berlanjut naik dalam titik tertinggi *average* menjadi 8213 ms pada pengujian dengan 1000 *POST*. Dengan kinerja seperti ini maka dapat dikatakan bahwa *framework* bekerja dibawah standar yang ditetapkan oleh Jakob Nielsen dalam bukunya *Usability Engineering* yaitu 1000 ms untuk pengguna merasakan bahwa sistem tetap interaktif (Jakob Nielsen, 1993). Akan tetapi masih dapat dikatakan cukup dalam batas pengguna masih tetap bersedia menunggu sistem yaitu diangka 10.000 ms (Jakob Nielsen, 1993).

Hasil lengkap dari pengujian dapat dilihat dalam Lampiran B.2.2.3.

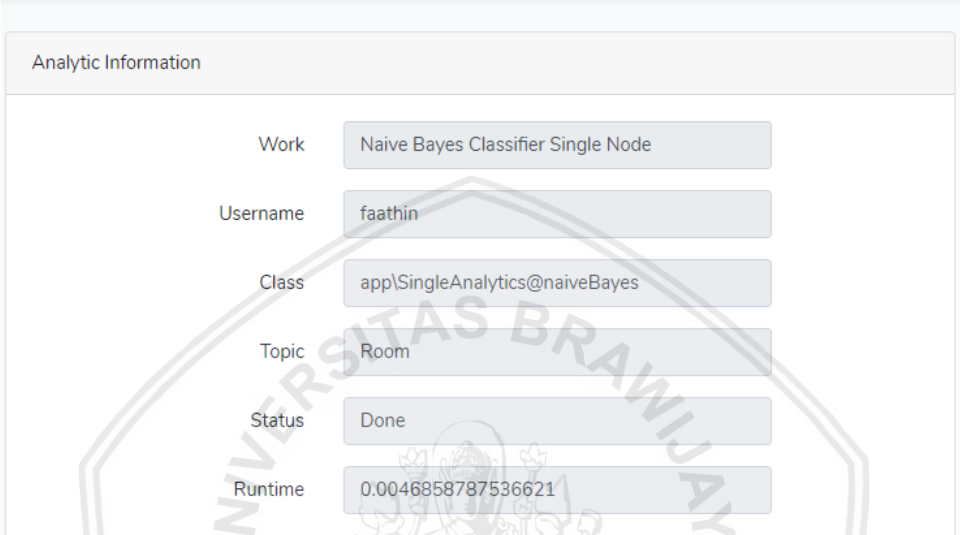
6.2.3 Pengujian Analitik Data

Dalam melakukan pengujian ini, penguji mencoba melakukan proses analitik data pada *data analytic*. Skenario yang diujikan dengan melakukan analitik terhadap 1.000,10.000,100.000,500.000, dan 1.000.000 data yang terdapat dalam *data storage*. Waktu pemrosesan dan juga kemampuan analitik akan sangat bergantung pada sumberdaya server. Pada pengujian ini akan dilakukan *klasifikasi*

menggunakan Naïve bayes, data yang akan dianalisis sendiri dapat dilihat pada Lampiran B.2.3.1, sedangkan data latih dapat dilihat pada Lampiran B.2.3.2.

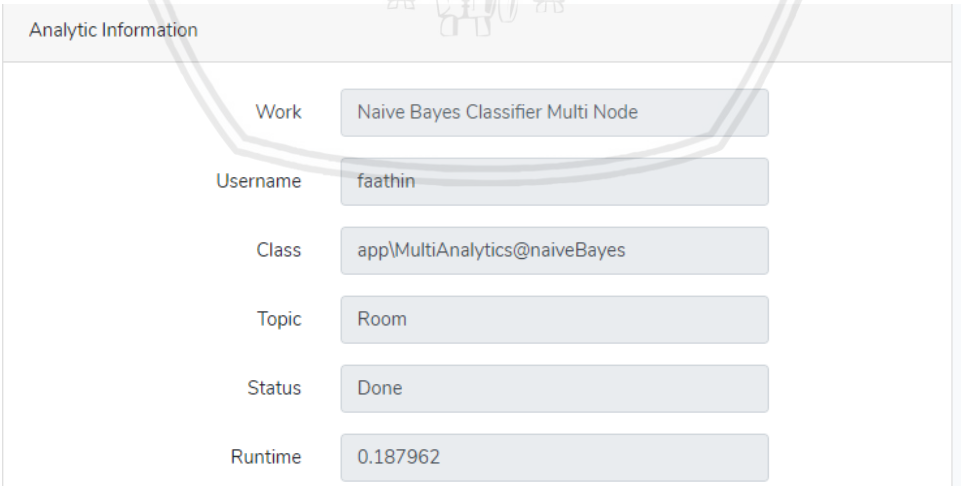
6.2.3.1 Skenario 1 : Melakukan analisis terhadap *1.000 entry data*

Pada skenario pengujian ini, penguji melakukan eksperimen analisis terhadap 1000 *entry* data dan diperoleh hasil seperti pada Gambar 6.67 dan 6.68 yang menunjukkan status keberhasilan, nama class modul beserta *runtime* dari proses.



Gambar 6.51 Pengujian skalabilitas analisis 1.000 Data (PHP-ML)

Gambar 6.67 menunjukkan hasil Analisis 1000 data menggunakan SingleAnalytic@naiveBayes yang menggunakan PHP-ML, berdasarkan data diatas diketahui bahwa proses memerlukan waktu 0.00468 detik untuk menyelesaikan keseluruhan proses.

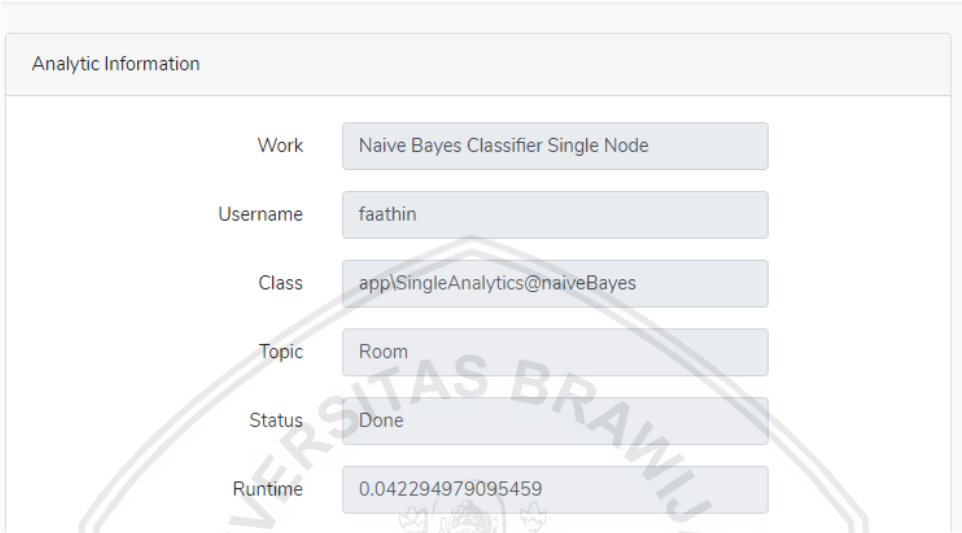


Gambar 6.52 Pengujian skalabilitas analisis 1.000 Data (Apache Spark)

Gambar 6.68 menunjukkan hasil Analisis 1000 data menggunakan MultiAnalytic@naiveBayes yang menggunakan Apache Spark, berdasarkan data diatas diketahui bahwa proses memerlukan waktu 0.18796 detik untuk menyelesaikan keseluruhan proses.

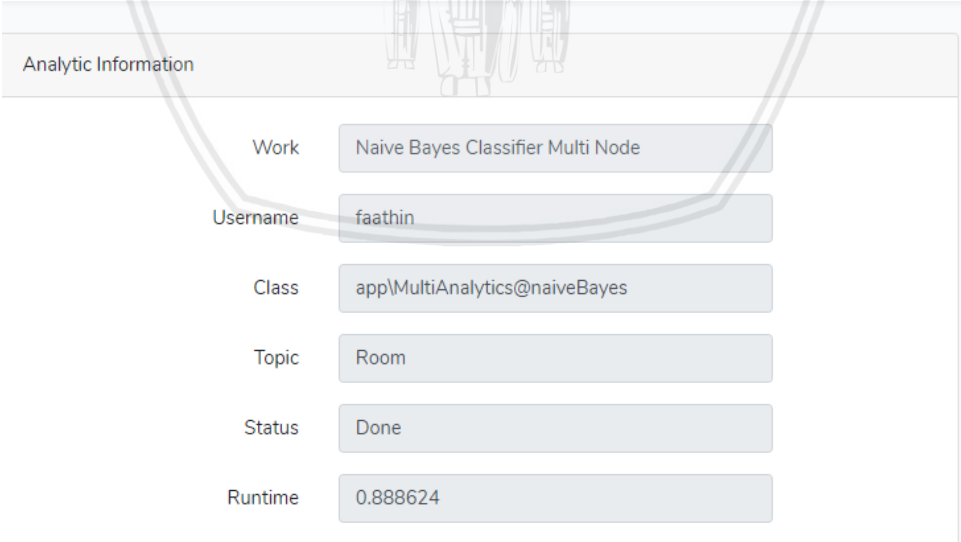
6.2.3.2 Skenario 2 : Melakukan analisis terhadap *10.000 entry data*

Pada skenario pengujian ini, penguji melakukan eksperimen analisis terhadap 10.000 *entry* data dan diperoleh hasil seperti pada Gambar 6.69 dan 6.70 yang menunjukkan status keberhasilan, nama class modul beserta *runtime* dari proses.



Gambar 6.53 Pengujian skalabilitas analisis 10.000 Data (PHP-ML)

Gambar 6.67 menunjukkan hasil Analisis 10.000 data menggunakan SingleAnalytic@naiveBayes yang menggunakan PHP-ML, berdasarkan data diatas diketahui bahwa proses memerlukan waktu 0.04229 detik untuk menyelesaikan keseluruhan proses.

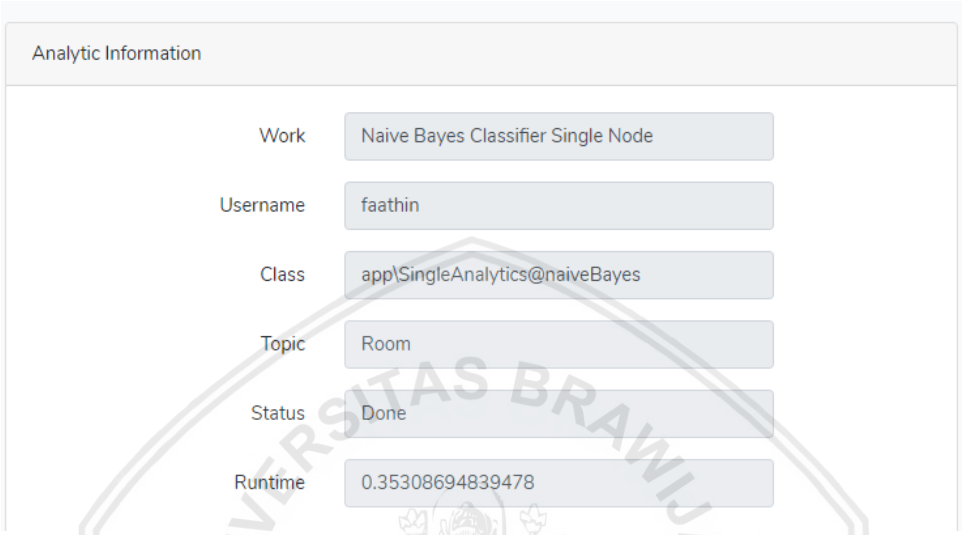


Gambar 6.54 Pengujian skalabilitas analisis 10.000 Data (Apache Spark)

Gambar 6.70 menunjukkan hasil analisis 10.000 data menggunakan MultiAnalytic@naiveBayes yang menggunakan Apache Spark, berdasarkan data diatas diketahui bahwa proses memerlukan waktu 0.88862 detik untuk menyelesaikan keseluruhan proses.

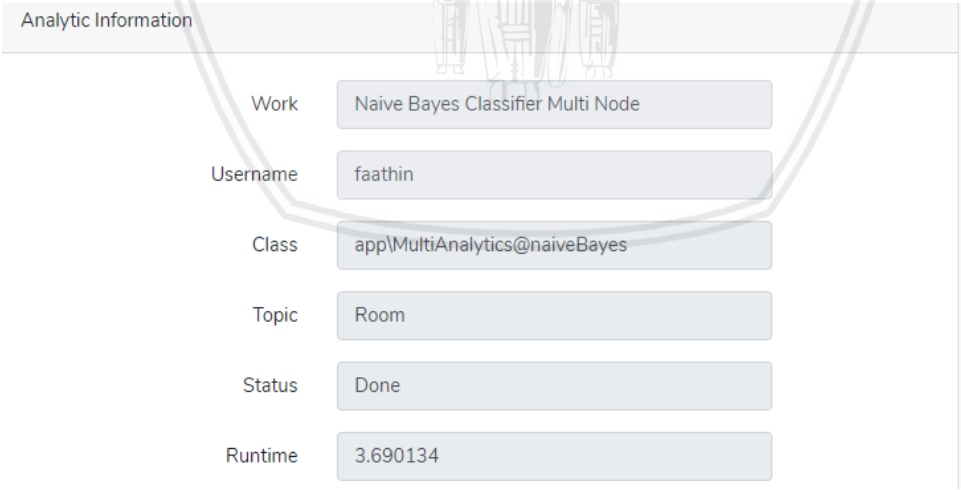
6.2.3.3 Skenario 3 : Melakukan analisis terhadap *100.000 entry data*

Pada skenario pengujian ini, penguji melakukan eksperimen analisis terhadap 100.000 *entry* data dan diperoleh hasil seperti pada Gambar 6.71 dan 6.72 yang menunjukkan status keberhasilan, nama class modul beserta *runtime* dari proses.



Gambar 6.55 Pengujian skalabilitas analisis 100.000 Data (PHP-ML)

Gambar 6.71 menunjukkan hasil Analisis 100.000 data menggunakan SingleAnalytic@naiveBayes yang menggunakan PHP-ML, berdasarkan data diatas diketahui bahwa proses memerlukan waktu 0.35308 detik untuk menyelesaikan keseluruhan proses.

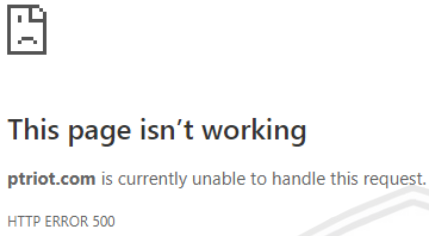


Gambar 6.56 Pengujian skalabilitas analisis 100.000 Data (Apache Spark)

Gambar 6.72 menunjukkan hasil analisis 100.000 data menggunakan MultiAnalytic@naiveBayes yang menggunakan Apache Spark, berdasarkan data diatas diketahui bahwa proses memerlukan waktu 3.6901 detik untuk menyelesaikan keseluruhan proses.

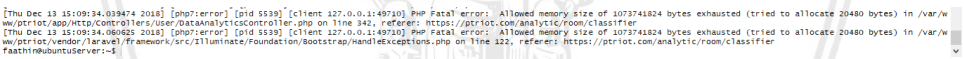
6.2.3.4 Skenario 4 : Melakukan analisis terhadap *500.000 entry data*

Pada skenario pengujian ini, penguji melakukan eksperimen analisis terhadap 500.000 *entry* data dan diperoleh hasil seperti pada Gambar 6.73 , 6.74, dan 6.75 yang menunjukkan status keberhasilan, nama class modul beserta *runtime* dari proses.



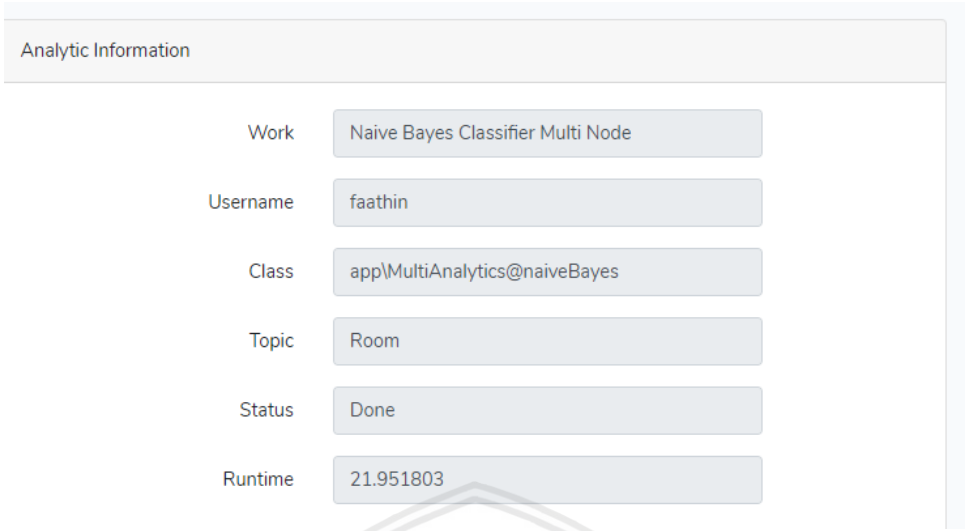
Gambar 6.57 Pengujian skalabilitas analisis 500.000 Data (PHP-ML)

Saat dilakukan proses Analisis data terhadap 500.000 data menggunakan PHP-ML terjadi sebuah kesalahan atau *error*, *browser* menunjukkan *error 500* yang menandakan bahwa terjadi kesalahan dari sisi server, kesalahan ini dapat dilihat pada Gambar 6.73. Untuk log kesalahan dari PHP dapat dilihat pada Lampiran B.2.3.3.



Gambar 6.58 Pengujian skalabilitas analisis 500.000 Data (Error PHP-ML)

Saat dilakukan pengecekan pada *log* webserver seperti pada Gambar 6.74 ternyata terdapat baris yang menandakan bahwa server error saat memproses data tersebut, hal ini dikarenakan proses tersebut berusaha menggunakan *memory* lebih dari yang diperbolehkan, sehingga webserver menutup proses tersebut.



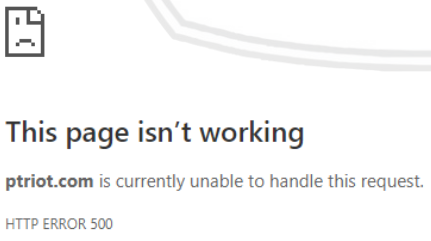
Gambar 6.59 Pengujian skalabilitas analisis 500.000 Data (Apache Spark)

Gambar 6.75 menunjukkan hasil analisis 500.000 data menggunakan MultiAnalytic@naiveBayes yang menggunakan Apache Spark, berdasarkan data diatas diketahui bahwa proses memerlukan waktu 21.95180 detik untuk menyelesaikan keseluruhan proses.

Akan tetapi setelah dilakukan 5 percobaan pada skenario ini hanyalah 1 saja yang berhasil selesai, 4 percobaan yang lain menggunakan Apache Spark menghasilkan kesalahan atau *hang*, penyebabnya adalah Apache Spark tidak dapat mengalokasikan *memory* seperti yang dibutuhkan.

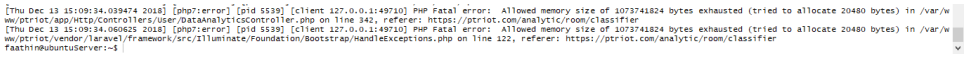
6.2.3.5 Skenario 5 : Melakukan analisis terhadap *1.000.000 entry data*

Pada skenario pengujian ini, penguji melakukan eksperimen analisis terhadap 1.000.000 *entry* data dan diperoleh hasil seperti pada Gambar 6.76 , 6.77, dan 6.78 dan 6.79 yang menunjukkan status keberhasilan, nama class modul beserta *runtime* dari proses.



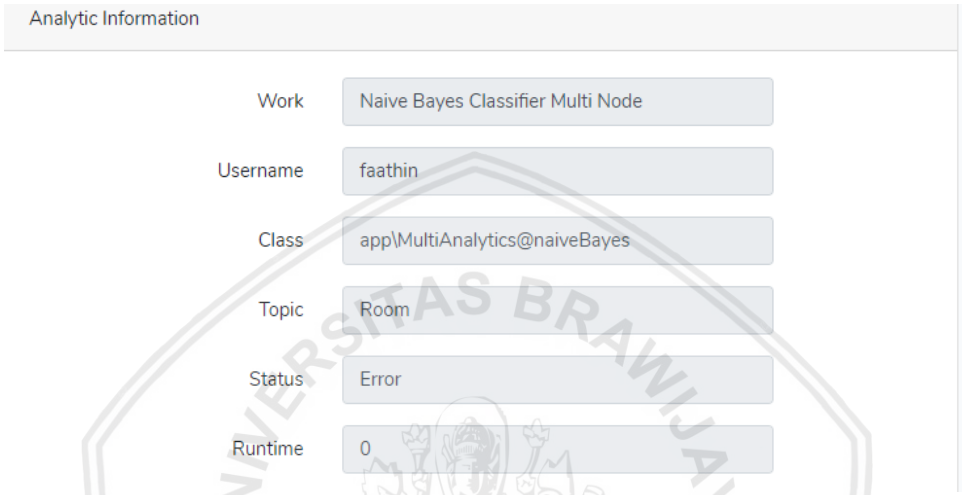
Gambar 6.60 Pengujian skalabilitas analisis 1.000.000 Data (PHP-ML)

Saat dilakukan proses Analisis data terhadap 1.000.000 data menggunakan PHP-ML terjadi sebuah kesalahan atau *error*, *browser* menunjukkan *error 500* yang menandakan bahwa terjadi kesalahan dari sisi server, kesalahan ini dapat dilihat pada Gambar 6.76. Untuk log kesalahan dari PHP dapat dilihat pada Lampiran B.2.3.3.



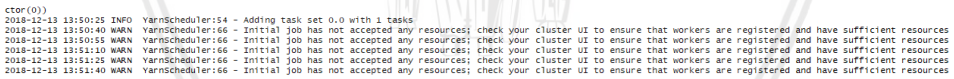
Gambar 6.61 Pengujian skalabilitas analisis 500.000 Data (Error PHP-ML)

Saat dilakukan pengecekan pada *log* webserver seperti pada Gambar 6.77 ternyata terdapat baris yang menandakan bahwa server error saat memproses data tersebut, hal ini dikarenakan proses tersebut berusaha menggunakan *memory* lebih dari yang diperbolehkan, sehingga webserver menutup proses tersebut.



Gambar 6.62 Pengujian skalabilitas analisis 1.000.000 Data (Apache Spark)

Gambar 6.78 menunjukkan hasil analisis 1.000.000 data menggunakan MultiAnalytic@naiveBayes yang menggunakan Apache Spark, berdasarkan data diatas diketahui bahwa proses tidak berhasil diproses ditandai dengan status *error*.



Gambar 6.63 Pengujian skalabilitas analisis 1.000.000 Data (Error Apache Spark)

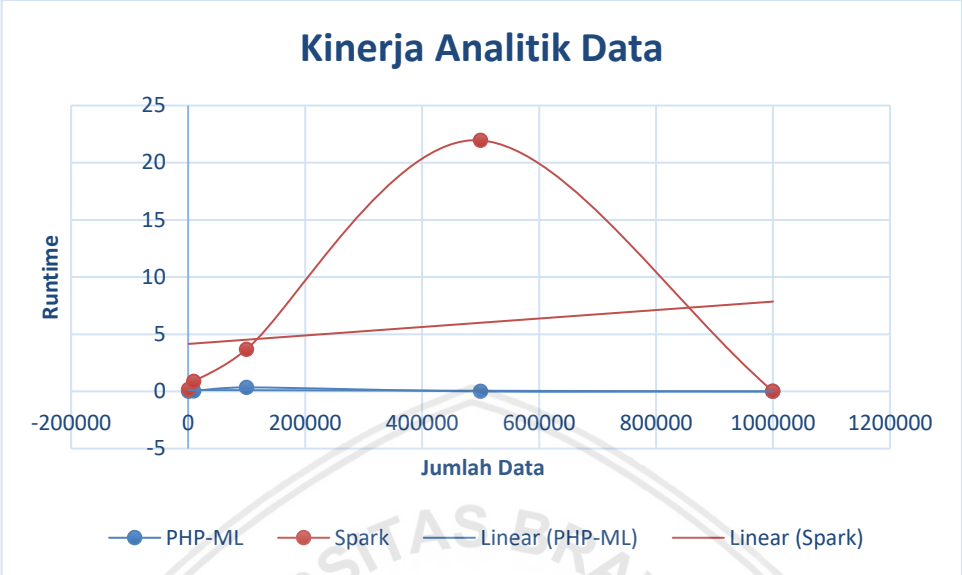
Saat dilakukan pengecekan dengan menjalankan program Apache Spark secara manual ternyata diketahui bahwa terjadi *hang* dalam Apache Spark seperti pada Gambar 6.64, hal ini dikarenakan *cluster* untuk menjalankan Apache Spark tidak mempunyai cukup sumberdaya untuk memproses, sumberdaya yang dimaksud adalah *memory.*

Untuk log kesalahan secara penuh dapat dilihat pada Lampiran B.2.3.4.

6.2.3.6 Pembahasan

Setelah dilakukan percobaan eksperimen analitik data pada sistem menggunakan metode Naïve bayes pada PHP-ML dan Apache Spark menggunakan 5 skenario didapatkan hasil seperti berikut. Kinerja *framework* sangat terpengaruh dalam 5 skenario yang dilakukan, terbukti dengan waktu *runtime* pada sistem yang cenderung naik secara eksponensial hingga akhirnya sistem tidak mampu

menangani dan *down* baik pada PHP-ML maupun Apache Spark seperti pada Gambar 6.80.



Gambar 6.64 Pembahasan kinerja analitik data

Terdapat temuan lain terkait hasil pengujian yaitu kinerja PHP-ML sangatlah mengungguli daripada Apache Spark pada data yang berukuran kecil, hingga akhirnya tidak mampu ditangani di ukuran 100.000 data. Akan tetapi Apache Spark dapat unggul untuk melakukan pemrosesan data berukuran besar, seperti ditunjukkan proses tetap berjalan pada pengujian dengan 500.000 data. Walaupun lambat Apache Spark dapat diandalkan untuk menangani data berukuran besar, untuk konfigurasi ram yang sama dalam sebuah sistem yang sama, sehingga disarankan jika ingin membuat sebuah sistem data analitik untuk data yang berukuran besar disarankan menggunakan Apache Spark. Sedangkan jika untuk membuat sistem data analitik yang *embed* didalam sistem lain dan data yang diproses tidaklah besar sangat disarankan menggunakan PHP-ML dikarenakan waktu *runtime* yang sangatlah baik, dan juga kemampuan menangani hingga 100.000 data.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan dari perancangan, implementasi, serta pengujian yang telah dilakukan oleh penulis maka penelitian dapat menghasilkan kesimpulan sebagai berikut :

1. Personal data analitik berbasis PHP-ML dan Apache Spark pada *IoT cloud apps* dapat diimplementasikan dengan cara menyediakan fitur personalisasi sistem. Personalisasi sistem ditandai dengan kemampuan manajeman akun, manajemen perangkat, manajemen topik dan juga pengolahan data melalui sebuah sistem data analytic. Dengan kemampuan personalisasi sistem maka sistem ini dapat menerima tidak hanya data yang *heterogen*, akan tetapi juga dapat menangani pengguna, topik, perangkat dan juga data yang *heterogen*.

   Personal data analitik berbasis PHP-ML dan Apache Spark pada *IoT cloud apps* dapat meningkatkan utitisasi sumberdaya pada lingkungan *cloud*, ditandai dengan penambahan kemampuan *data analytic* yang mengimplementasikan *machine learning*. Dengan penambahan ini maka pastinya sumberdaya pada lingkungan cloud computing akan dapat digunakan secara maksimal. Bahkan hingga terjadi *hang (error)* seperti yang terjadi saat melakukan analitik 1.000.000 data, yang dikarenakan penggunaan ram yang berlebihan.

2. Personal data analitik berbasis PHP-ML dan Apache Spark pada *IoT cloud apps* dapat memenuhi semua kebutuhan fungsional yang sudah ditentukan dan juga memiliki kemampuan skalabilitas yang cukup baik yang seperti :

   a. Kinerja *GET* data yang mampu menangani hingga 1000 *request* dalam 1 satuan waktu, dengan tingkat error mencapai 0%.

   b. Kinerja *GET* data terpengaruh oleh jumlah request yang dilakukan, dibuktikan dengan grafik pada Gambar 6.60, yang mana pada grafik tersebut hasil dari percobaan cenderung naik secara eksponensial.

   c. Kinerja *GET* data cukup baik yaitu diangka 7569 ms pada saat terdapat beban 1000 *request* yang mana dibawah standar yang ditetapkan oleh Jakob Nielsen 1.000 ms untuk dapat dikatakan sebagai sistem yang interaktif, akan tetapi masih diatas dari standar untuk dapat dikatakan sebagai sistem yang ditolerir oleh pengguna yaitu 10.000 ms.

   d. Kinerja *POST* data yang mampu menangani hingga 1000 pengiriman data dalam 1 satuan waktu, dengan tingkat error mencapai 0.7%.

e. Kinerja *POST* data terpengaruh oleh jumlah data yang dikirimkan, dibuktikan dengan grafik pada Gambar 6.66, yang mana pada grafik tersebut hasil dari percobaan cenderung naik secara eksponensial.

f. Kinerja *POST* data cukup baik yaitu diangka 8.213 ms yang mana dibawah standar yang ditetapkan oleh Jakob Nielsen 1.000 ms untuk dapat dikatakan sebagai sistem yang interaktif, akan tetapi masih diatas dari standar untuk dapat dikatakan sebagai sistem yang ditolerir oleh pengguna yaitu 10.000 ms.

g. Kinerja analitik data cukup baik dengan mampu menangani hingga 500.000 data untuk dianalitik menggunakan Apache Spark menggunakan metode Naïve bayes.

h. Kinerja analitik data sangat baik untuk mengolah data dengan jumlah dibawah 100.000 data untuk dianalitik menggunakan PHP-Machine Learning (PHP-ML) menggunakan metode Naïve bayes, ditandai dengan *runtime* dibawah 5 detik.

Selain itu didapatkan hasil lainnya mengenai kemampuan skalabilitas dari *Data analytic* pada sistem personal data analitik berbasis PHP-ml dan Apache Spark pada *IoT cloud apps* memiliki kinerja yang secara umum dapat dikatakan cukup baik. Didapatkan temuan sebagai berikut pada data analytic yang dikembangkan dalam *IoT cloud apps* yang antaranya adalah:

a. PHP-ML mampu menangani data hingga 100.000 data dalam waktu dibawah 5 detik.

b. PHP-ML mengalami *error* saat menangani 500.000 data, hal ini dikarenakan memory yang dialokasikan untuk PHP tidak mencukupi.

c. Apache Spark mampu menangani data hingga 500.000 data dalam waktu dibawah 1 menit.

d. Apache Spark mengalami *error* saat menangani 1.000.000 data, hal ini dikarenakan memory yang dialokasikan untuk proses Apache Spark tidak mencukupi.

e. Saat Apache Spark mengalami *error* sistem akan mengalami *hang*, yang mana sistem menjadi tidak responsive ditandai tidak dapat dilakukan ping, bahkan ssh.

f. Dengan hal seperti ini maka tidak disarankan untuk menggunakan Apache Spark dalam sebuah sistem yang terotomasi seperti pada penelitian ini.

g. Apache Spark lebih baik dikembangkan secara *standalone* dalam sebuah lingkungan yang terpisah, dan dijalankan

secara manual atau supervisi, sehingga jika terjadi *error*, maka tidak akan mempengaruhi sistem lain.

h. PHP-ML sangatlah cocok untuk diimplementasikan dalam sebuah sistem *webservice* atau *lightweight data analytical system*, hal ini dikarenakan kemampuannya yang cukup baik dan waktu pemrosesan sangatlah cepat.

7.2 Saran

Berdasarkan kesimpulan yang dihasilkan dari penelitian ini, maka direkomendasikan beberapa hal dengan harapan dapat dilanjutkan untuk penelitian selanjutnya yaitu :

1. Pada penelitian berikutnya dikembangkan sebuah kemampuan kolaborasi dalam sebuah sistem *IoT cloud apps*, dengan ini maka sistem akan mampu menangani banyak pengguna dalam sebuah lingkungan secara lebih baik.
2. Pada penelitian berikutnya dilakukan optimasi pada komponen *data analytic* sehingga tidak akan terjadi *hang* saat melakukan analisis data, khususnya data yang berukuran besar.
3. Pada penelitian berikutnya dilakukan pengujian sistem dari segi penggunaan *resources* secara lebih menyeluruh, sehingga kemampuan sistem dapat dilihat secara lebih detail lagi.
4. Pada penelitian berikutnya dibangun sebuah *data analytic* yang menggunakan Apache Spark secara *standalone,* akan tetapi tetap terkait dengan sistem yang sekarang sudah dikembangkan. Untuk mengetahui apakah *hang* yang terjadi dapat dihilangkan secara menyeluruh.
5. Pada penelitian berikutnya dilakukan penambahan modul-modul khususnya modul analitik untuk sistem ini, sehingga kemampuan personalisasi sistem menjadi lebih baik lagi.
6. Pada penelitian beriktunya dilakukan penambahan kemampuan untuk melakukan analisis data yang tidak terstruktur seperti data gambar, suara, video maupun data raw.

DAFTAR REFERENSI

Atzori, L., Iera, A. & Morabito, G., 2010. *The Internet of things: A survey*. Elsevier, pp. 2787-2805.

Biswas, A.R. & Giaffreda, R., 2014. *IoT and cloud convergence: Opportunities and challenges*. 2014 IEEE World Forum on *Internet of things* (WF-IoT), Seoul, pp. 375-376.

Pramukantoro, E.S., Yahya, W., Arganata, G., Bhawiyuga, G., & Basuki, A., 2017. *Topic based IoT data storage framework for heterogeneous sensor data*. 2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA), Lombok, pp. 1-4.

Hamid, T., Pramukantoro, E., & Siregar, A., 2018. *Pengembangan Web Service Pada IoT Data Storage Dengan Pendekatan Semantik dan Penambahan Mekanisme Autentikasi*. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, vol. 2, no. 12, p. 6824-6827.

Kim , J. & Lee, J., 2014. *OpenIoT: An open service framework for the Internet of things*. 2014 IEEE World Forum on *Internet of things* (WF-IoT), Seoul, pp. 89-93.

Onal, A.C, Sezer, B.O., Ozbayoglu, M. & Dogdu, E., 2017. *Weather data analysis and sensor fault detection using an extended IoT Framework with semantics, big data, and machine learning*.2017 IEEE International Conference on *Big data* (*Big data*), Boston, MA, 2017, pp. 2037-2046.

Khodadadi, F., Calheiros, R.N., & Buyya, R., 2015. *A data-centric framework for development and deployment of Internet of things applications in clouds*. 2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Singapore, pp. 1-6.

Dhoot, Vallabh, et al., 2016. *Efficient Dimensionality Reduction for Big data Using Clustering Technique*.Imperial Journal of Interdisciplinary Research 2.5.

Jiang, L., Xu, L.D., Cai, H., Jiang, Z., Bu, F. & Xu, B., 2014. *An IoT-Oriented Data Storage Framework in Cloud Computing Platform*. IEEE Transactions on Industrial In*form*atics, vol. 10, no. 2, pp. 1443-1451.

Grgić, K., Špeh, I. & Heđi, I., 2016. *A web-based IoT solution for monitoring data using MQTT protokol*. International Conference on Smart Systems and Technologies (SST), Osijek, pp. 249-253.

Li, QianQian, et al., 2017. *EnergIoT: A solution to improve network lifetime of IoT devices.* Pervasive and Mobile Computing 42, pp. 124-133.

Tanase, Diana Irina., 2003. *Documenting frameworks using pattern languages*. Diss. University of Northern Iowa.

Kim, J.T., 2014. *Analysis of RFID application for U-healthcare system in Internet of things.* International Journal of Smart Home, pp. 131–142.

Bashir, M.R., & Gill, A. Q., 2016. *Towards an IoT big data analytic framework: Smart buildings systems*. 2016 IEEE 18th InternationalConference on High Per*form*ance Computing and Communications, pp.1325–1332

Marjani, M. et al., 2017. *Big IoT Data Analytic: Architecture, Opportunities, and Open Research Challenges*. in IEEE Access, vol. 5, pp. 5247-5261.

Hewitt, C., 2008 *ORGs for Scalable, Robust, Privacy-Friendly Client Cloud Computing*. in IEEE Internet Computing, vol. 12, no. 5, pp. 96-99.

Reis, J. L., & Carvalho, J. A., 2014. *Personalization dimensions: A conceptual model for personalized information systems*. 2014 9th Iberian Conference on Information Systems and Technologies (CISTI).

Nielsen, J., 1993. *Usability Engineering.* Burlington: Morgan Kaufmann.