TUGAS AKHIR – KI141502

RANCANG BANGUN SISTEM VIRTUALISASI SENSOR UNTUK MANAJEMEN SENSOR TERSEBAR BERBASIS KOMPUTASI AWAN

HADRIAN BAYANULHAQ SIREGAR
NRP 5112100145

Dosen Pembimbing I
Waskitho Wibisono, S.Kom., M.Eng,. Ph.D

Dosen Pembimbing II
Royyana Muslim Ijtihadie, S.Kom., M.Kom,. Ph.D

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

TUGAS AKHIR – KI141502

RANCANG BANGUN SISTEM VIRTUALISASI SENSOR UNTUK MANAJEMEN SENSOR TERSEBAR BERBASIS KOMPUTASI AWAN

HADRIAN BAYANULHAQ SIREGAR
NRP 5112100145

Dosen Pembimbing I
Waskitho Wibisono, S.Kom., M.Eng,. Ph.D

Dosen Pembimbing II
Royyana Muslim Ijtihadie, S.Kom., M.Kom,. Ph.D

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI1502

SENSOR VIRTUALIZATION FOR CLOUD BASED SENSOR MANAGEMENT SYSTEM

HADRIAN BAYANULHAQ SIREGAR
NRP 5112100145

Supervisor I
Waskitho Wibisono, S.Kom., M.Eng,. Ph.D

Supervisor II
Royyana Muslim Ijtihadie, S.Kom., M.Kom,. Ph.D

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2015

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

RANCANG BANGUN SISTEM VIRTUALISASI SENSOR UNTUK MANAJEMEN SENSOR TERSEBAR BERBASIS KOMPUTASI AWAN

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
HADRIAN BAYANULHAQ SIREGAR
NRP : 5112 100 145

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Waskitho Wibisono, S.Kom., M.Kom., Ph.D.
   NIP: 19741022 200003 1 002 ..... (Pembimbing 1)

2. Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.
   NIP: 19770824 200604 1 001 ..... (Pembimbing 2)

SURABAYA
JUNI, 2016

[Halaman ini sengaja dikosongkan]

RANCANG BANGUN SISTEM VIRTUALISASI SENSOR UNTUK MANAJEMEN SENSOR TERSEBAR BERBASIS KOMPUTASI AWAN

Nama Mahasiswa	**: HADRIAN BAYANULHAQ SIREGAR**
NRP	**: 5112100145**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**
Dosen Pembimbing 2	**: Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.**

Abstrak

Sensor merupakan sebuah objek yang berfungsi untuk mendeteksi kejadian ataupun perubahan pada lingkungan sensor tersebut. Saat ini, keberadaan sensor telah memungkinkan komputasi yang bersifat ubiquitous, yang artinya proses komputasi dapat terjadi dimana saja dengan memanfaatkan sensor tersebar. Untuk mengelola dan memantau sensor sensor yang tersebar tersebut, di butuhkan suatu sistem untuk mengelola, mengawasi, mengolah dan menganalisa data dari sensor tersebar tersebut. Sistem tersebut harus bisa di gunakan dari berbagai perangkat komputer dan darimana saja. Salah satu cara untuk menyediakan layanan sistem manajemen sensor tersebar adalah dengan melakukan virtualisasi data sensor tersebar. Dengan virtualisasi data sensor tersebar, pengguna dapat melihat dan mengawasi data data sensor mereka dari sebuah layanan aplikasi berbasis web. Untuk menyediakan layanan aplikasi web tersebut, salah satu cara yang dapat menghemat dan memudahkan penyedia tanpa harus mengelola perangkat keras dan jaringan sendiri adalah dengan memanfaatkan teknologi komputasi awan.

Komputasi awan memungkinkan penyediaan layanan yang bisa di akses darimana saja dan kapan saja selama memiliki akses internet. Dengan memanfaatkan infrastruktur komputasi awan yang ada, layanan dapat disediakan dengan availability

yang tinggi, dan dapat mengakomodasi bertumbuhnya jumlah pengguna dengan cara menambahkan sumber daya komputasi sesuai dengan kebutuhan.

Dengan layanan manajemen sensor tersebar ini, pengguna dapat mendaftarkan, mengelola, mengawasi, dan menganalisa data sensor yang mereka miliki dari perangkat komputer dimana saja, dan kapan saja selama memiliki koneksi internet.

Data dari sensor yang terdaftar yang berupa hasil bacaan dari sensor tersebut yang di bungkus dalam format JSON dikirimkan ke server melalui HTTP POST. Kemudian server akan mengolah dan menyimpan data ke sebuah database. Data sensor yang terdapat pada database ini ditampilkan pada antarmuka aplikasi web. Pada aplikasi web ini, pengguna dapat melakukan pengawasan dan analisa data sensor yang mereka miliki.

Berdasarkan hasil uji coba rancang bangun sistem tersebut, dapat disimpulkan bahwa sistem manajemen sensor tersebar ini mampu menyediakan layanan pengawasan, pengolahan, dan analisa data sensor. Serta sistem mampu mengakomodasi traffic sebanyak 250 sensor sekaligus dalam satu detik tanpa ada transaksi yang gagal dengan rata rata waktu respon 6.37 detik per transaksi.

Kata kunci: Virtualisasi Sensor, Pengawasan dan Pengelolaan Data Sensor, Komputasi Awan

SENSOR VIRTUALIZATION FOR CLOUD BASED SENSOR MANAGEMENT SYSTEM

Student's Name	**: HADRIAN BAYANULHAQ SIREGAR**
Student's ID	**: 5112100145**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**
Second Advisor	**: Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D.**

Abstract

Sensor is an object whose function is to detect an event or change in it's environment. Nowadays, sensors enabled ubiquitous computing, which means that computing can occure everywhere by leveraging distributed sensors. To manage and monitor those distributed sensors, a system whose function is to manage, monitor, process and analyze from those sensor is needed. The system has to be accessable from any location and at any time. On of the ways to provide that kind of service is by virtualizing the sensor data. By virtualizing the sensor, data from the sensor can be showed to user from a web application so that the user can manage and monitor the sensor's data from that web application. One of the ways to provide such service without the hassle of managing hardware and network is to leverage the capability of cloud computing.

Cloud computing allows the service to be provided in such way, so that it can be accessed from anywhere and anytime as long as the users have an internet connection. By leveraging cloud computing ability, the service can be provided with high availability and can accomodate the growing number of users and traffic by adding more computing resource on the fly as needed.

With this distributed sensor management service, users can register, manage, monitor, and analyze their sensor's data

from anywhere and anytime, as long as the user have an internet connection.

Sensor reading from the registered sensor will be packed as JSON and sent into the server by HTTP POST method. Then the server will store the data in a database. Sensor's data in the database will be displayed to users via a web application. With this web application, the user can monitor and analyze their sensor data.

Based on the test that have been conducted, it is concluded that the distributed sensor management system can provide sensor data monitoring, managing, and analyzing service. Also, the system can accomodate 250 conccurent sensors in a second without any failed transaction with average response time of 6.37 second per transaction.

Keyword: Sensor Virtualization, Sensor Data Management and Monitoring, Cloud Computing

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **"RANCANG BANGUN SISTEM VIRTUALISASI SENSOR UNTUK MANAJEMEN SENSOR TERSEBAR BERBASIS KOMPUTASI AWAN".**

Pengerjaan Tugas Akhir ini merupakan salah satu dari sekian banyak kesempatan yang saya dapatkan, untuk mendapatkan ilmu dan pengalaman berharga selama saya berada di kampus Teknik Informatika ITS ini.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Ibu, Ibu, Ibu, Ayah dan Adik yang selalu memberikan do'a, dukungan, serta motivasi, sehingga penulis selalu termotivasi untuk menyelesaikan Tugas Akhir.
3. Bapak Waskitho Wibisono, S.Kom., M.Eng., Ph.D selaku pembimbing I yang selalu menyemangati dan memotivasi dengan ilmu-ilmu yang diluar dugaan saya.
4. Bapak Royyana Muslim Ijtihadie, S.Kom., M.Kom., Ph.D selaku pembimbing II yang telah mengoreksi buku ini dengan cermat dan memberikan *insight* dan saran seputar komputasi awan.
5. Bapak, Ibu dosen Jurusan Teknik Informatika ITS yang telah banyak memberikan ilmu dan bimbingan yang tak ternilai harganya bagi penulis.
6. Teman – Teman Datangaja.com yang telah mengisi waktu-waktu penulis dengan tantangan, pengalaman, dan

kesempatan bergabung dan merasakan atmosfir menantang dalam sebuah *startup*.

7. Teman – teman angkatan 2012, tanpa mereka, saya tidak akan merasakan apa itu yang dinamakan "Angkatan".
8. Teman – teman Laboratorium Dasar Terapan Komputasi, yang telah mengajari saya banyak sekali hal secara tidak langsung.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Mei 2016

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1 Latar Belakang

Aplikasi dari sensor tersebar telah digunakan dibanyak bidang penting seperti pengawasan lingkungan, pengawasan infrastruktur penting, pertanian, bahkan di bidang militer. Namun karena keterbatasan *memory*, komunikasi, komputasi, dan skalabilitas pada sensor tersebar, dibutuhkan metode yang efisien untuk mengelola data data dari sensor. Manajemen sensor ini membutuhkan infrastruktur dengan kemampuan komputasi yang besar dan *scalable* untuk mengelola, menyimpan, dan melakukan analisa pada data data tersebut. Teknologi komputasi awan dapat menjadi solusi yang menjanjikan untuk masalah ini. Komputasi awan dapat menyediakan sumber daya komputasi, penyimpanan, dan layanan perangkat lunak secara fleksibel, dan dengan biaya yang relatif murah.

Komputasi awan (*cloud computing*) adalah sebuah sistem yang memungkinkan akses kepada sekumpulan sumber daya komputasi yang dapat diatur sesuai dengan kebutuhan pengguna dan bisa dengan cepat disediakan tanpa banyak pengaturan dan campur tangan penyedia. Sumber daya tersebut bisa berupa jaringan, *server*, penyimpanan (*storage*), aplikasi, dan layanan (*services*). [1]

Dengan teknologi komputasi awan ini, dapat di bangun sebuah sistem manajemen sensor dengan metode virtualisasi sensor. Virtualisasi sensor yang dimaksud adalah representasi data data hasil bacaan dari sensor fisik yang akan di tampilkan ke pengguna lewat HTTP. Representasi data tersebut berupa sebuah sensor virtual, dengan data-data yang didapatkan dari sensor sensor fisik yang tergabung di dalam suatu kumpulan sensor yang sebelumnya sudah didaftarkan oleh pemiliknya. Hasil akhir dari aplikasi ini berupa sebuah sistem yang dapat melakukan

virtualisasi sensor, dan menyediakan beberapa layanan seperti *monitoring* dan visualisasi data hasil sensor. Infrastruktur atau sistem ini disebut sebagai *sensor cloud.*

Istilah Sensor-Cloud sendiri sebenarnya masih baru. Menurut IntelliSys, Sensor-Cloud sendiri dapat didefinisikan sebagai sebuah infrastruktur yang memungkinkan komputasi yang *pervasive* dengan menggunakan sensor sensor sebagai *interface* antara dunia nyata dan dunia maya, dengan menggunakan data-data sebagai *backbone* dan internet sebagai media komunikasinya. [2, 3]. MicroStrain juga mendefenisikan Sensor-Cloud sebagai sebuah tempat penyimpanan data sensor yang unik, visualisasi, dan pengelolaan secara *remote* yang memanfaatkan kekuatan dari teknologi komputasi awan untuk menyediakan skalabilitas, visualisasi, dan analisis yang bisa di atur oleh penggunanya. [1]

Sebuah infrastruktur Sensor-Cloud mengumpulkan dan memproses informasi dari beberapa sensor tersebar. Ini memungkinkan pengguna untuk berbagi informasi, dan berkolaborasi menggunakan aplikasi dengan basis komputasi awan. Dengan kata lain, Sensor-Cloud memungkinkan pengguna untuk mengumpulkan, mengakses, memproses, memvisualisasi, menganalisa, menyimpan, membagi, dan mencari data dalam jumlah yang besar dengan memanfaatkan teknologi komputasi awan.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana cara mengumpulkan data dari sensor fisik untuk ditampilkan sebagai sensor virtual.
2. Bagaimana cara mendesain arsitektur sistem yang baik.
3. Bagaimana cara membuat abstraksi data dari kolaborasi antar sensor.

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan antara lain:

1. Interaksi dengan pengguna berbasis *web*.
2. Pemilik sensor harus mengikuti protokol yang disediakan oleh sistem ini agar data dari sensor nya bisa di simpan dan di tampilkan pada aplikasi ini
3. Sistem dibangun dengan Bahasa PHP, dengan *web framework* CodeIgniter dan menggunakan MySQL sebagai basis data
4. Sistem dibangun dengan model *Software as a Service* dengan memanfaatkan layanan infrastruktur komputasi awan yang sudah ada.
4. Uji coba akan dilakukan dengan mikrokontroler Arduino yang akan di*install* sebuah sensor.
5. Uji coba performa aplikasi akan dilakukan dengan program Python yang akan mensimulasikan banyak sensor.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini antara lain:

1. Membuat sistem yang dapat melakukan manajemen sensor berbasis komputasi awan.
2. Melakukan virtualisasi sensor.
3. Menyediakan layanan berupa visualisasi data, kolaborasi, dan data feed kepada pengguna yang dapat diaplikasikan sebagai pengawasan / *monitoring.*
4. Menyediakan layanan yang bersifat *multitenant.*

1.5 Tujuan

Manfaat dari pembuatan tugas akhir ini, antara lain:

1. Mempermudah dalam manajemen sensor tersebar.

2. Memanfaatkan teknologi komputasi awan untuk membangun sistem yang fleksibel dan *scalable*.

1.6 Metodologi

1. Penyusunan proposal tugas akhir

   Proposal tugas akhir ini berisi tentang deskripsi pendahuluan dari tugas akhir yang akan dibuat. Pendahuluan pada proposal tugas akhir ini terdiri dari latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir, tujuan dari pembuatan tugas akhir, dan manfaat hasil dari pembuatan tugas akhir. Selain itu dijelaskan pula tinjauan pustaka yang digunakan sebagai referensi pendukung implementasi tugas akhir. Pada proposal ini juga terdapat perencanaan jadwal pengerjaan tugas akhir.

2. Studi literatur

   Pada studi literatur ini, dipelajari sejumlah referensi yang diperlukan dalam implementasi sistem, yaitu mengenai mikrokontroler arduino, bahasa pemrograman PHP, kerangka kerja *web* CodeIgniter, JSON, dan MySQL

3. Analisis dan desain perangkat lunak

   Tahap ini meliputi perancangan sitem berdasarkan studi literature dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahapan ini dibuat prototype sistem, yang merupakan rancangan dasar dari sistem yang dibuat. Serta dilakukan desain suatu sistem dan desain proses-proses yang ada.

4. Implementasi perangkat lunak

Implementasi merupakan tahap membangun rancangan program yang telah dibuat. Pada tahapan ini merealisasikan apa yang terdapat pada tahapan sebelumnya, sehingga menjadi sebuah program yang sesuai dengan apa yang telah direncanakan.

5. Pengujian dan evaluasi

   Pada tahapan ini dilakukan uji coba pada alat yang telah dirancang. Tahapan ini dimaksudkan untuk mengevaluasi tingkat akurasi dari alat tersebut serta mencari masalah yang mungkin timbul dan mengadakan perbaikan jika terdapat kesalahan.

6. Penyusunan Buku Tugas Akhir

   Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat

1.7 Sistematika Penulisan Laporan Tugas Akhir

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini:

Bab I Pendahuluan

Bab yang berisi latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Dasar Teori

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

Bab III Perancangan Perangkat Lunak

Bab ini berisi implementasi dari perancangan perangkat lunak yang telah dibuat pada bab sebelumnya. Implementasi berupa *pseudocode* dari fungsi utama dan *screenshot* perangkat lunak.

Bab IV Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode yang digunakan untuk proses implementasi.

Bab V Uji Coba Dan Evaluasi

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

Bab VI Kesimpulan Dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan rancangan alat yang diajukan pada pengimplementasian program. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap alat yang dirancang dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1. Mikrokontroler Arduino Uno

Mikrokontroler Arduino Uno [4] merupakan sebuah kit elektronik yang bersifat *open source*. Mikrokontroller Arduino ini menggunakan chip mikrokontorler AVR dari perusahaan Atmel. Mikrokontroler itu sendiri adalah sebuah *chip* yang bisa diprogram menggunakan perangkat komputer. Mikrokontroler ada pada perangkat elektronik yang kita pakai sehari-hari, misalnya pada AC, TV, pemutar DVD, dll. Mikrokontroler juga dipakai untuk mengendalikan robot.

Arduino Uno merupakan jenis dari mikrokontroler Arduino yang paling banyak digunakan, terutama untuk pemula yang masih ingin melakukan eksplorasi pada mirkokontroler. Versi terakhir Arduino Uno adalah R3, dengan ATMEGA328 sebagai mikrokontroler nya. Arduino Uno memiliki 14 pin I/O digital dan 6 pin I/O analog.

Arduino juga menggunakan *Integrated Development Environment (IDE)* berbasis *processing* dimana *processing* adalah bahasa *open-source* untuk menuliskan program ke komputer lainnya. Jika ada sebuah *project* yang memerlukan beberapa komputer untuk berkomunikasi dengan Arduino, maka *processing* tersebut dapat digunakan, sehingga komputer-komputer tersebut dapat saling berkomunikasi dengan Arduino. Supaya mikrokontroler Arduino dapat berfungsi, Arduino juga dapat dipasangkan dengan berbagai macam sensor dan *actuator*. Pada tugas akhir ini, mikrokontroler Arduino akan digunakan pada uji

coba fungsionalitas dari sistem yang dibangun. Gambar 2.1 merupakan contoh mikrokontroler Arduino Uno R3.

Gambar 2.1 Mikrokontroler Arduino Uno [4]

2.2. Komputasi Awan

Komputasi awan [5] adalah sebuah gabungan dari dua teknologi, yaitu teknologi komputasi, dan pengembangan berbasis internet (awan). "Awan" merupakan metafora dari internet, seperti yang sering digambarkan pada diagram jaringan komputer. Awan dalam diagram jaringan komputer merepresentasikan infrastruktur kompleks yang disembunyikannya.

Teknologi komputasi awan merupakan sebuah teknologi yang menggunakan internet sebagai pusat *server* untuk mengelola data dan juga aplikasi pengguna. Teknologi ini mengizinkan pada pengguna untuk menjalankan program tanpa instalasi dan mengizinkan pengguna untuk mengakses data-data mereka melalui komputer dimana saja, selama mempunyai akses internet.

Gambar 2.2 menunjukkan beberapa pemanfaatan dari teknologi komputasi awan yang dapat berupa aplikasi, platform, dan infrastruktur.

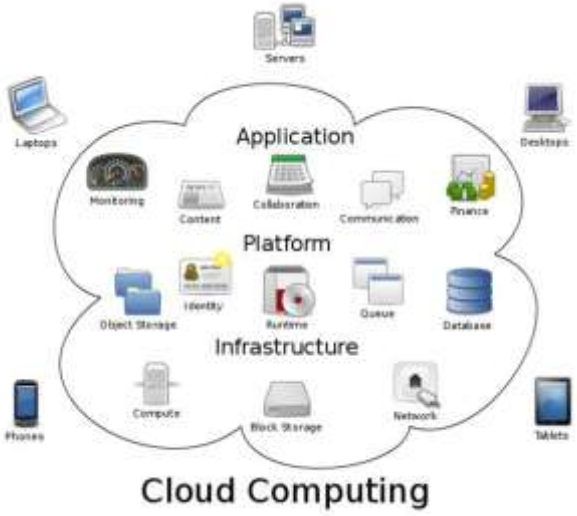
Beberapa karakteristik utama dari komputasi awan adalah:

- *On-demand self-service*. Seorang pengguna dapat mengatur kemampuan komputasi dari sebuah sistem, misalnya penyimpanan, secara otomatis sesuai dengan kebutuhan tanpa interaksi manusia.
- *Broad network access*. Bisa diakses dimana saja, dan dari perangkat apa saja selama mempunyai akses internet
- *Resource pooling*. Sumber daya komputasi yang disediakan oleh penyedia layanan di *pool* untuk melayani pengguna pengguna yang ada. Contoh dari sumber daya tersebut adalah sumber daya penyimpanan, pemrosesan, memori, dan *bandwidth* jaringan.
- *Rapid Elasticity*. Kemampuan dari sistem yang disediakan bisa dengan cepat diatur sesuai dengan kebutuhan.
- *Measured Service*. Sistem komputasi awan mengendalikan dan mengoptimasi sumber daya dengan menggunakan pengukuran secara otomatis. Penggunaan sumber daya bisa diawasi, dikendalikan, dan dilaporkan untuk menyediakan transparansi kepada kedua pihak, baik pengguna maupun penyedia. [14]

Beberapa model layanan pada komputasi awan adalah:

- *Software as a Service* (*SaaS*). Layanan yang disediakan kepada pengguna berupa aplikasi yang berjalan di atas sebuah infrastruktur komputasi awan. Layanan aplikasi yang disediakan bisa di akses dari berbagai perangkat, seperti *web browser*, maupun antar muka *desktop*. Pengguna layanan tidak mengelola dan mengatur infrastruktur dan sumber daya seperti jaringan, *server*, sistem operasi, penyimanan dimana aplikasi berjalan.

- *Platform as a Service (PaaS)*. Layanan yang disediakan kepada pengguna berupa *deploy* aplikasi yang dibuat atau di dapatkan oleh pengguna. Pengguna tidak mengelola dan mengatur sumber daya infrastruktur komputasi awan seperti jaringan, *server*, sistem operasi, maupun penyimpanan, tetapi memiliki kontrol penuh atas aplikasi yang dijalankannya di atas infrastruktur komputasi awan.
- *Infrastructure as a Service (IaaS)*. Layanan yang disediakan pada pengguna adalah kemampuan pengguna untuk mengatur dan menetapkan sumber daya pemrosesan, penyimpanan, jaringan, and sumber daya komputasi lainnya dimana pengguna dapat menjalankan aplikasi nya diatas infrastruktur tersebut. [14]



Gambar 2.2 Komputasi Awan [5]

2.3. Visualisasi Data

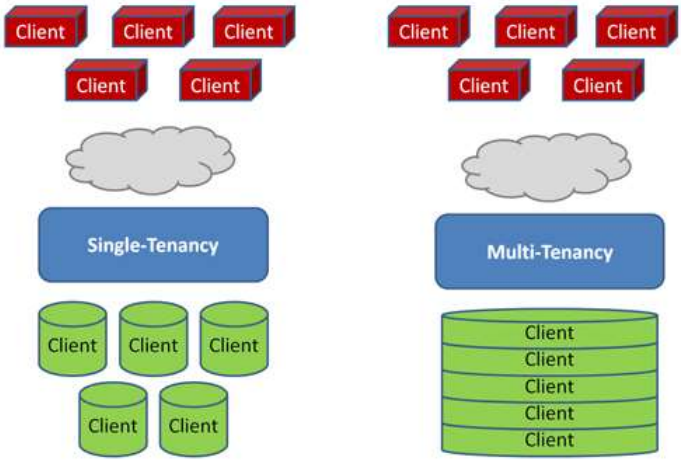
Visualisasi data [5] merupakan sebuah komunikasi visual modern yang digunakan oleh banyak bidang ilmu. Visualisasi data tidak berada di bawah bidang manapun, melainkan interpretasi diantara banyak bidang. Visualisasi data mengikutkan pembuatan dan kajian dari representasi visual data, yang artinya merupakan suatu "informasi yang telah di abstraksikan dalam bentuk skematis, termasuk atribut atau variabel dari unit informasi". Pada tugas akhir ini, visualisasi data menjadi salah satu fitur yang ditawarkan oleh sistem yang dibangun. Gambar 2.3 menunjukkan salah satu bentuk dari visualisasi data pada aplikasi yang dibangun, dimana data direpresentasikan kedalam bentuk grafik.

Gambar 2.3 Contoh Visualisasi Data

2.4. Multitenancy

Pada bidang rekayasa perangkat lunak (*software engineering*), *multitenancy* merujuk pada sebuah arsitektur perangkat lunak dimana sebuah *instance* dari sebuah perangkat lunak yang berjalan pada suatu *server* bisa melayani lebih dari satu *tenant* dalam suatu waktu. *Tenant* itu sendiri merupakan

sekumpulan atau suatu pengguna yang mempunyai akses yang sama pada sebuah perangkat lunak atau layanan, dengan hak akses yang spesifik pada masing masing pengguna dalam *group* tersebut. Dengan arsitektur *multitenant* ini, sebuah layanan di desain agar bisa menyediakan semua fungsionalitas dari sebuah layanan kepada semua *tenant* secara penuh. Gambar 2.4 merupakan ilustrasi dari arsitektur *multitenat.*



Gambar 2.4 Diagram Arsitektur *Multi-tenant* [6]

2.5. Javascript Object Notation (JSON)

JSON [7] merupakan singkatan dari *Javascript Object Notation*. JSON merupakan sebuah format data yang mudah diolah oleh berbagai bahasa pemrograman berbeda. JSON memudahkan manusia untuk membaca dan menulis dan memudahkan mesin untuk mengurai dan menghasilkan data. JSON merupakan sebuah format teks yang independen, namun memiliki pustaka yang dapat dikenali oleh berbagai macam bahasa

pemrograman, seperti C++, PHP, Python, Ruby, Perl, dan lain-lain. JSON dibangun dalam dua bentuk, antara lain:

1. Sebuah objek, merupakan sekumpulan pasangan nama dan nilai. Sebuah objek dimulai dengan karakter "{" dan diakhiri dengan karakter "}". Setiap nama diikuti oleh karakter ":" dan setiap elemen JSON yang berisi nama dan nilai dipisahkan dengan karakter ",".
2. Sebuah nilai yang berurutan, dalam bahasa pemrograman dikenal dengan istilah *array vector*, atau *list*. Sebuah *array* dimulai dengan karakter "[" dan diakhiri dengan karakter "]". Setiap nilai yang berbeda dipisahkan dengan tanda koma. Pada tugas akhir ini JSON digunakan dalam komunikasi antara sensor dengan server untuk pengiriman data.

Contoh kode JSON dapat dilihat pada Gambar 2.5.

```
{
  "arguments" : { "number" : 10 },
  "url" : "http://localhost:8080/restty-tester/collection",
  "method" : "POST",
  "header" : {
    "Content-Type" : "application/json"
  },
  "body" : [
    {
      "id" : 0,
      "name" : "name 0",
      "description" : "description 0"
    },
    {
      "id" : 1,
      "name" : "name 1",
      "description" : "description 1"
    }
  ],
  "output" : "json"
}
```

Gambar 2.5 Contoh kode JSON

2.6. PHP

PHP [8] merupakan sebuah bahasa pemrograman yang berjalan pada HTTP *server* dan digunakan pada pengembangan *web*. Selain itu, PHP juga dapat digunakan untuk membuat berkas XML dan melakukan pengolahan basis data seperti menyeleksi, menambah, membaca, menghapus, maupun memperbaharui data yang ada pada basis data tersebut. PHP biasanya dikombinasikan dengan HTML (*Hyptertext Markup Language)* untuk mengasilkan interaksi pengguna dari sebuah aplikasi web.

Pada tugas akhir ini, PHP digunakan untuk membangun sistem virtualisasi sensor untuk manajemen sensor tersebar. Sistem dibuat seutuhnya dengan PHP dan kerangka kerja *web* CodeIgniter. Gambar 2.6 merupakan contoh dari kode PHP.

```php
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

class Test extends CI_Controller {

    public function chart_test()
    {
        $sensorId = 1;
        $this->load->view('chart_test');
    }

    public function ajax_getChartData($sensorId)
    {
        $this->load->model('Data_model');
        $data = $this->Data_model->getSensorReading($sensorId);
        foreach($data as $row)
        {

        }
        $this->output
            ->set_content_type('application/json')
            ->set_output(json_encode($data));
    }
```

Gambar 2.6 Contoh kode PHP

2.7. CodeIgniter

Codeigniter (CI) [9] adalah *framework* pengembangan aplikasi (*Application Development Framework*) berbasis bahasa pemrograman PHP. CI merupakan suatu kerangka kerja *web* untuk membuat aplikasi *web* dengan bahasa pemrograman PHP yang lebih sistematis dan cepat dalam hal pengembangannya. CI menggunakan pola pengembangan MVC (*Model – View – Controller*) [9].

- *View*, merupakan bagian yang menangani *presentation logic*. Pada suatu aplikasi *web*, bagian ini biasanya berupa file *template* HTML, yang diatur oleh *Controller*. *View* berfungsi untuk menerima dan merepresentasikan data yang disusun oleh *Controller* kepada pengguna. Bagian ini tidak mempunyai akses langsung pada bagian *Model*.
- *Model*, merupakan bagian yang berhubungan langsung dengan basis data dari suatu aplikasi. Karena bagian ini berhubungan langsung dengan basis data, *Model* biasanya digunakan untuk melakukan manipulasi data seperti CRUD (*Create, Read, Update, Delete*)
- *Controller*, merupakan bagian yang mengatur hubungan antar *model* dan *view*. *Controller* berfungsi menerima dan memproses *request* yang diterima dari *web browser* pengguna. [9]

2.8. MySQL

MySQL [10] merupakan sebuah RDBMS (*Relational Database Management System*) yang bersifat *open source*. MySQL merupakan pilihan yang cukup populer untuk sistem basis data, dan merupakan sebuah komponen utama pada LAMP (Linux, Apache, MySQL, PHP/Perl/Python) *stack*. Beberapa aplikasi web yang menggunakan MySQL adalah Joomla, Wordpress, phpBB, Drupal, Facebook, Twitter, Flickr, dan Youtube [11][12]. Pada tugas akhir ini, MySQL digunakan

sebagai basis data untuk menyimpan data-data dari sensor yang terdaftar.

2.9. Web Service

Web Service [13] merupakan antarmuka yang menggambarkan operasi-operasi yang bisa di akses dalam jaringan, dan mengembalikan respon dalam bentuk XML atau JSON. Antarmuka *web service* menyembunyikan detail implementasi dari layanan, yang memungkinkan pengunaannya secara independen dari *platform hardware* atau *software* dimana *web service* itu di implementasikan. Pada tugas akhir ini, *web service* digunakan untuk menerima data dari sensor yang terdaftar.

BAB III
PERANCANGAN PERANGKAT LUNAK

Pada bab ini akan dijelaskan mengenai analisis dan perancangan perangkat lunak yang akan dikembangkan. Perancangan merupakan bagian penting dari pengembangan perangkat lunak karena merupakan perencanaan perangkat lunak secara teknis. Adapun hal-hal yang dibahas dalam bab ini adalah deskripsi umum perangkat lunak, arsitektur perangkat lunak, diagram kasus penggunaan, perancangan basis data, diagram alur, dan desain antar muka perangkat lunak.

3.1. Deskripsi Umum Sistem

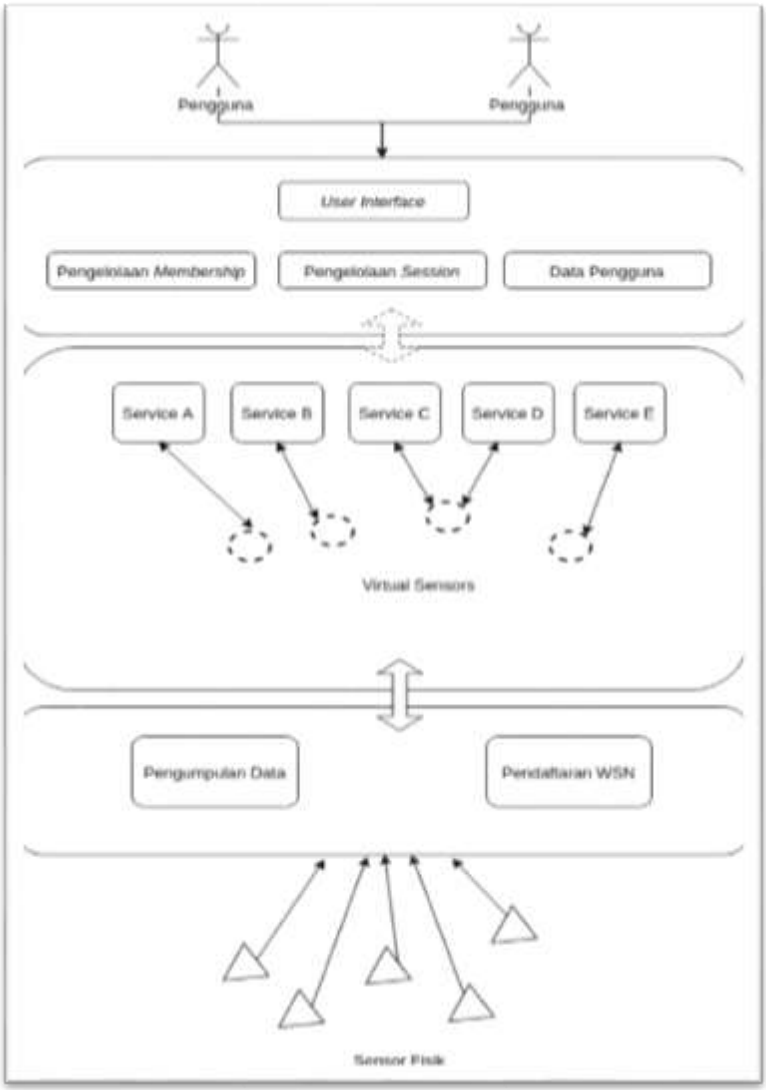
Pada Tugas Akhir ini dibangun sebuah sistem yang mampu melakukan virtualisasi sensor fisik untuk keperluan manajemen sensor tersebar. Pengguna bisa berupa aplikasi yang memerlukan akses data ke sebuah sensor tanpa harus membuat *server* data nya sendiri, atau orang yang ingin mengawasi data sensor yang dimiliknya (misalnya sensor di rumah, kebun, dll).

Pada aplikasi ini, pengguna dapat menggunakan sensor yang terpasang pada mikrokontroler Arduino, maupun Raspberry Pi selama mengikuti protokol yang disediakan oleh sistem. Data dari sensor yang tergabung dalam sistem akan di *record* ke dalam basis data, dan pengguna bisa melakukan pengawasan dan analisa terhadap data data tersebut.

Terdapat fitur kolaborasi sensor, yang berarti data dari dua atau tiga sensor bisa di kolaborasikan (bisa melakukan operasi matematika terhadap data-data antar sensor tersebut).

3.2. Arsitektur Umum Sistem

Perangkat lunak yang akan dibangun terdiri dari tiga *layer* seperti pada Gambar 3.1



Gambar 3.1 Arsitektur Umum Sistem

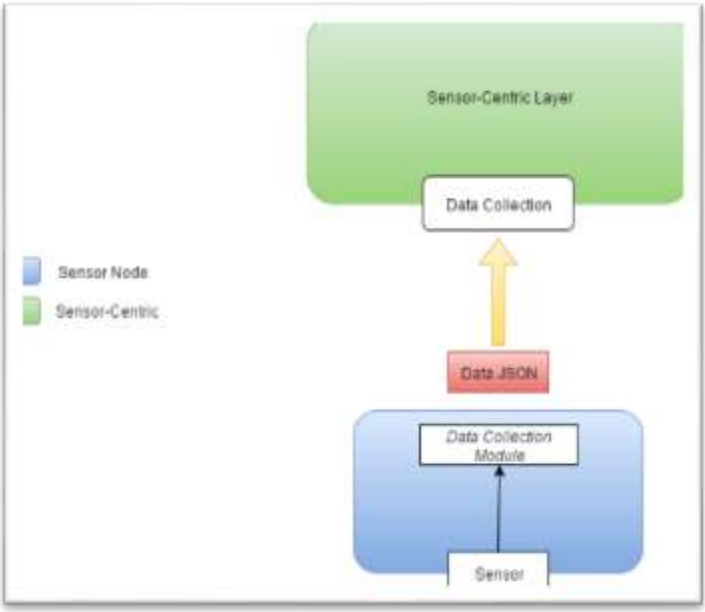
Layer pertama bersifat *client-centric*. *Layer* ini merupakan "jembatan" antara sistem dan pengguna (antarmuka pengguna). Komponen komponen pada *layer* ini memfasilitasi dan mengelola interaksi pengguna dengan sistem, mengelola *membership* dan *session* dari tiap pengguna yang terdaftar pada sistem. Hak akses pada layer ini juga diatur pada *layer* ini.

Layer kedua adalah *middleware layer*. *Layer* ini berfungsi sebagai penghubung antara *client-centric layer* dan *sensor-centric layer*. Proses virtualisasi sensor fisik yang tergabung pada sistem akan dilakukan pada *layer* ini.

Layer ketiga adalah layer yang bersifat *sensor-centric*, artinya *layer* ini berhubungan langsung dengan sensor sensor fisik yang terdaftar pada sistem. Beberapa fungsi yang dijalankan oleh *layer* ini salah satunya adalah pengumpulan data dari sensor fisik. Agar bisa melakukan pengumpulan data, *node* sensor fisik harus mengikuti protokol yang ditentukan oleh sistem. Pengiriman data dari sensor fisik ke sistem dilakukan dalam format JSON.

Ketiga *layer* di atas akan berjalan di atas sebuah infrastruktur *cloud* dengan model layanan *Software as a Service* (SaaS) yang artinya pengguna hanya bisa menggunakan layanan aplikasi yang disediakan, yang berjalan di atas infrastruktur komputasi awan. Namun penyedia dapat menambah sumberdaya komputasi jika diperlukan, tanpa sepengetahuan pengguna layanan.

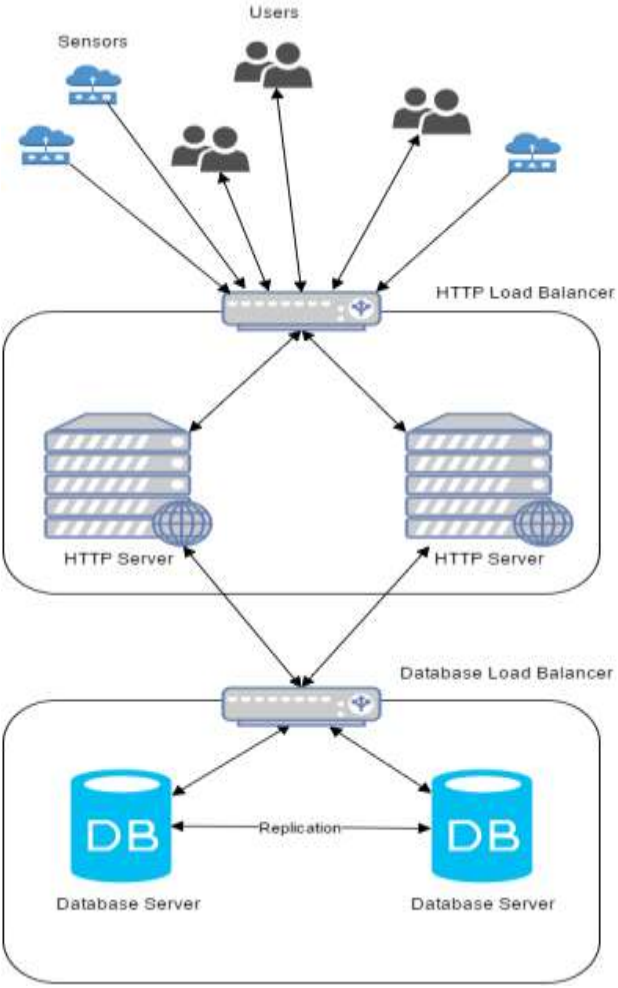
Modul pengumpulan data (*Data Collection Module*), merupakan modul yang harus di*install* pada *node* sensor yang akan mengirimkan data ke sistem. *Module* ini berfungsi untuk melakukan enkapsulasi data yang telah dibaca oleh sensor ke format JSON, dan mengirimkan data tersebut ke modul pengumpulan data yang ada pada sistem. Gambar 3.2 merupakan ilustrasi dari modul pengumpulan data.



Gambar 3.2 Modul Pengumpulan Data

3.3. Arsitektur Jaringan Sistem

Arsitektur jaringan sistem menggambarkan arsitektur dari jaringan tempat aplikasi web akan dijalankan. Karena model *deployment* bersifat *Software as a Service* (*SaaS*), sesuai dengan defenisi *SaaS* sendiri, aplikasi dijalankan di atas infrastruktur komputasi awan, tanpa kemampuan pengguna aplikasi mengatur sumber daya komputasi yang digunakan. Lingkungan tempat aplikasi akan dijalankan berupa *Virtual Machine* yang berjalan di atas infrastruktur komputasi awan *Amazon Web Services* dan *Microsoft Azure*. Diagram arsitektur jaringan bisa dilihat pada Gambar 3.3



Gambar 3.3 Arsitektur Jaringan Sistem

Sistem virtualisasi sensor untuk manajemen sensor tersebar yang akan dibangun bersifat *web based*, artinya aplikasi harus berjalan di atas sebuah HTTP *Server*. Pada arsitektur

jaringan ini, terdapat dua HTTP *Server* yang akan menjadi tempat berjalannya aplikasi ini, serta melayani *request* pengguna. Pemanfaatan kedua HTTP *Server* ini menggunakan sebuah *load balancer* dimana pengguna akan melakukan *request* ke *load balancer*, kemudian *load balancer* tadi meneruskan *request* pengguna ke HTTP *Server* yang tersedia. Hal ini dapat mempercepat dan mengurangi beban masing masing HTTP *Server* yang ada, karena *request* tidak hanya ditangani oleh satu *server* saja.

Data pada aplikasi ini akan disimpan pada basis data, yang berjalan di atas *database server*. Pada arsitektur ini, terdapat dua buah *database server* yang akan melayani transaksi basis data dari kedua HTTP *Server* yang ada. Hal ini bisa dilakukan dangan memanfaatkan *load balancer* yang mampu meneruskan request dari HTTP *Server* yang ada ke *database server*. Selain itu, dilakukan juga replikasi basis data yang bersifat *master – master*.

3.4. Perancangan Diagram Kasus Penggunaan

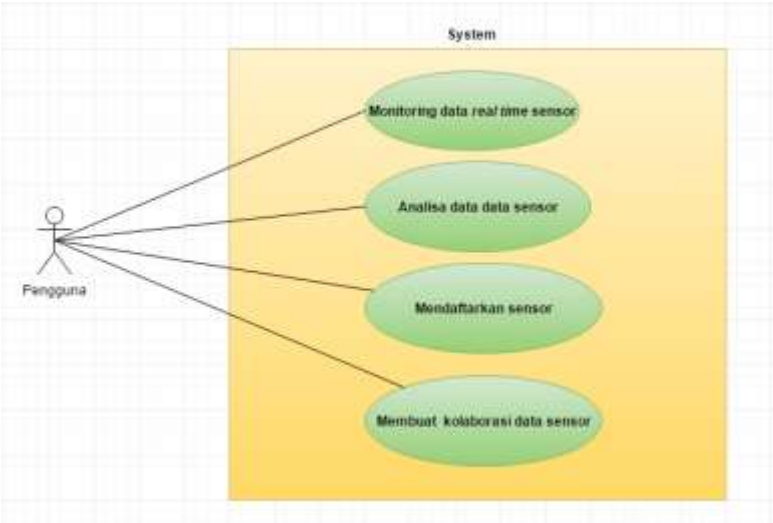
Diagram kasus penggunaan menggambarkan peran aktor yang terlibat dalam fungsionalitas perangkat lunak yang dibangun. Adapun perancangan diagram kasus perangkat lunak dalam Tugas Akhir ini dapat dilihat pada Gambar 3.4.

Pada Tugas Akhir ini, aktor yang merupakan pengguna sistem manajemen sensor ini berperan sebagai pengguna, dan pemilik sensor tersebar yang ada pada sistem. Pengguna dapat menjalankan fitur yang ada pada perangkat lunak, diantaranya:

- *Monitoring* data *real time* sensor
  Pengguna dapat menampilkan dan mengawasi data sensor yang dimiliknya secara *real time* melalui antar muka sistem yang berbasis *web*. Data yang ditampilkan berupa grafik / *chart* data sensor dengan *series* berupa waktu, dan *value* dari hasil bacaan sensor.
- Analisa data-data sensor

Pengguna dapat melakukan analisa data-data sensor yang telah dikumpulkan sebelumnya. Analisa berupa rerata, maksimum, dan minimum dari data-data sebuah sensor per harinya.

- Mendaftarkan sensor
  Pengguna juga bertindak sebagai pemilik sensor, artinya pengguna dapat mendaftarkan sensor fisik yang dimilikinya agar bisa memanajemen sensor fisiknya dari internet.
- Membuat kolaborasi data sensor
  Pengguna bisa membuat kolaborasi antar dua atau tiga sensor. Data-data sensor yang dikolaborasikan bisa di jumlah, kurang, bagi, dan kali kan. Fitur analisa dan *monitoring real time* data juga berlaku pada sensor yang dikolaborasi kan, karena sensor yang dikolaborasikan di anggap sebagai satu sensor.



Gambar 3.4 Diagram Kasus Penggunaan.

3.5. Perancangan Basis Data

Pada perancangan basis data merupakan tahapan untuk merancang basis data yang akan digunakan pada Tugas Akhir ini. Basis data ini berfungsi untuk menyimpan data-data sensor, data pengguna, dan data-data *rules* sensor.

3.5.1. Tabel Sensor

Tabel 3.1 Perancangan Tabel *sensor*

No	Nama Atribut	Tipe Data	Keterangan
1	*id*	*integer*	Sebagai *primary key* untuk sensor
2	*user_id*	*integer*	Sebagai *id* dari akun pemilik sensor
3	*sensor_key*	*varchar*	Sebagai *token* dari sebuah sensor agar bisa mengirim data ke sistem
4	*sensor_name*	*varchar*	Nama sensor yang di daftarkan
5	*sensor_description*	*varchar*	Deskripsi dari sensor
6	*created_on*	*timestamp*	Tanggal dan waktu di daftarkan nya sensor
7	*last_updated*	*timestamp*	Tanggal dan waktu sensor terakhir kali di update

Tabel 3.1 merupakan tabel sensor yang berfungsi untuk menyimpan data dari sebuah sensor. Yang dimaksud dari data sebuah sensor adalah informasi tentang sensor tersebut. Data yang disimpan berupa *id*, *user_id*, *sensor_key*, *sensor_name*, *sensor_description*, *created_on*, dan *last_updated*.

3.5.2. Tabel *sensor_data*

Tabel 3.2 Perancangan Tabel *sensor_data*

No.	Nama Atribut	Tipe Data	Keterangan
1	*id*	*integer*	*Primary key* dari sebuah data bacaan sensor
2	*sensor_id*	*integer*	*id* dari sensor yang mengirim data
3	*sensor_reading*	*float*	data hasil bacaan sensor yang dikirim
4	*timestamp*	*timestamp*	Waktu data masuk ke sistem

Tabel 3.2 merupakan tabel *sensor_data* yang berfungsi untuk menyimpan data bacaan dari sebuah sensor yang telah terdaftar. Server akan menyimpan data bacaan sensor ke tabel ini selama sensor aktif mengirimkan data.

3.5.3. Tabel *users*

Tabel 3.3 Perancangan Tabel *users*

No.	Nama Atribut	Tipe Data	Keterangan
1	*id*	*int*	*Primary key* dari tabel *users*.
2	*ip_address*	*varchar*	Alamat *IP* dari pengguna ketika login terakhir
3	*username*	*varchar*	*username* dari pengguna yang digunakan untuk login ke sistem
4	*password*	*varchar*	*password* pengguna yang digunakan utnuk login ke sistem
5	*email*	*varchar*	*email* yang digunakan saat pendaftaran

6	*created_on*	*integer*	Waktu pembuatan / pendaftaran user dalam *unix time*
7	*last_login*	*integer*	Waktu login terakhir pengguna dalam *unix time*.

Tabel 3.3 adalah tabel *users* yang berfungsi untuk menyimpan data pengguna dari aplikasi ini. Aplikasi web ini bisa melayani banyak pengguna.

3.5.4. Tabel *sensor_collab*

Tabel 3.4 Perancangan Tabel *sensor_collab*

No.	Nama Atribut	Tipe Data	Keterangan
1	*sensor_collab_id*	*integer*	*Primary key* dari tabel *sensor_collab_id*
2	*user_id*	*integer*	Pemilik dari sensor kolaborasi
3	*sensor_collab_name*	*varchar*	Nama dari sensor kolaborasi
4	*sensor_collab_desc*	*text*	Deskripsi dari sensor kolaborasi
5	*sensor_x_id*	*integer*	*id* dari sensor yang akan di kolaborasikan
6	*sensor_y_id*	*integer*	*id* dari sensor yang akan di kolaborasikan
7	*sensor_z_id*	*integer*	*id* dari sensor yang akan di kolaborasikan
8	*sensor_x_rule_id*	*integer*	*id* dari *rule* sensor yang akan di kolaborasikan
9	*sensor_y_rule_id*	*integer*	*id* dari *rule* sensor yang akan di kolaborasikan
10	*sensor_z_rule_id*	*integer*	*id* dari *rule* sensor yang akan di kolaborasikan
11	*comp_operator*	*varchar*	*operator* komparasi antar sensor (AND , OR)

12	*operator*	*varchar*	*operator* matematika untuk operasi data antar sensor (penjumlahan, pengurangan, pembagian, perkalian)

Tabel 3.4 adalah tabel *sensor_collab* yang berfungsi untuk menyimpan informasi mengenai sebuah kolaborasi sensor yang telah dibuat. Tabel ini menyimpan data-data seperti *id* dari kolaborasi sensor, *id* dari sensor-sensor yang dikolaborasikan, dan ekspresi matematik yang digunakan untuk memproses data hasil bacaan sensor yang masuk.

3.5.5. Tabel *sensor_collab_data*

Tabel 3.5 Perancangan Tabel *sensor_collab_data*

No.	Nama Atribut	Tipe Data	Keterangan
1	*sensor_collab_data_id*	*integer*	*Primary key* dari dari data hasil kolaborasi
2	*sensor_x_value*	*float*	Data bacaan salah satu sensor kolaborasi
3	*sensor_y_value*	*float*	Data bacaan salah satu sensor kolaborasi
4	*sensor_z_value*	*float*	Data bacaan salah satu sensor kolaborasi
5	*sensor_collab_id*	*integer*	*id* dari sensor kolaborasi
6	*sensor_reading*	*float*	Data hasil operasi kolaborasi dari dua atau tiga sensor yang dikolaborasikan
7	*timestamp*	*timestamp*	Waktu data masuk.

Tabel 3.5 merupakan tabel *sensor_collab_data* yang berfungsi untuk menyimpan data hasil dari kolaborasi sensor. Data yang disimpan pada tabel ini merupakan hasil olahan ekspresi matematik yang ada pada tabel *sensor_collab*.

3.5.6. Tabel *sensor_rules*

Tabel 3.6 Perancanga Tabel *sensor_rules*

No.	Nama Atribut	Tipe Data	Keterangan
1	*rule_id*	*integer*	*Primary key* dari *rules* sensor
2	*sensor_id*	*integer*	*id* dari sensor yang memiliki rule
3	*rule_type*	*varchar*	Tipe dari *rule* sensor (*high, low, equal*)
4	*rule_value*	*float*	Batas nilai / *threshold* dari *rule* sensor.

Tabel 3.6 merupakan tabel *sensor_rules* yang berfungsi menyimpan data aturan dari sebuah sensor. Aturan ini berfungsi pada kolaborasi sensor, dimana jika sebuah aturan ter-*trigger*, maka kolaborasi tersebut akan melakukan operasi yang telah didaftarkan sebelumnya.

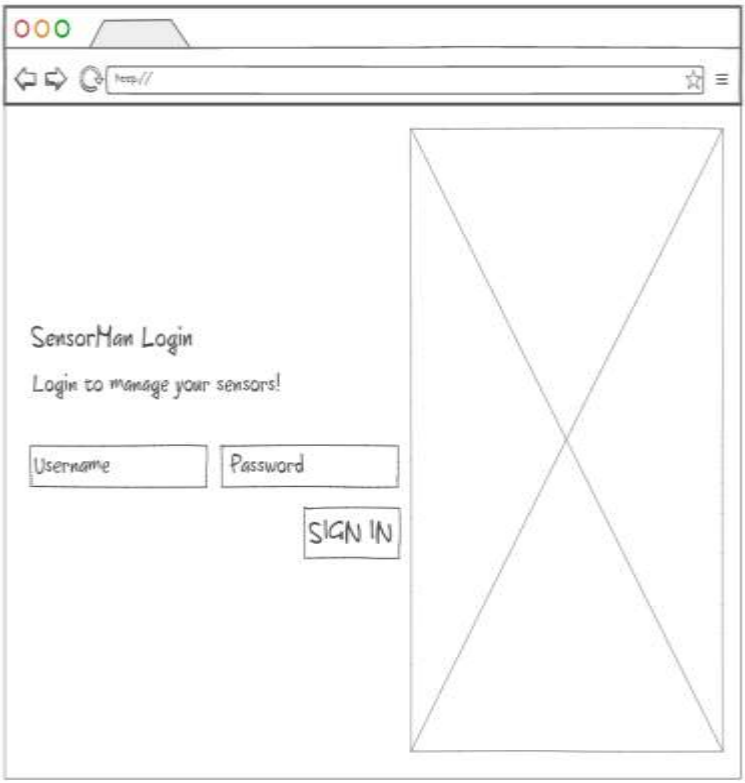
3.6. Perancangan *Web Service*

Web service pada Tugas Akhir ini digunakan untuk menerima data dari *node* sensor fisik yang terdaftar. *Web service* yang dirancang merupakan *web service* sederhanda dengan memanfaatkan metode HTTP POST untuk mengirimkan data ke sistem dalam format JSON. Parameter yang ditangkap oleh *web service* ini adalah *sensor_key* dan *sensor_reading*. Kemudian data *sensor_reading* akan di simpan pada tabel *sensor_data* di basis data.

3.7. Perancangan Antarmuka Sistem

Pada Tugas Akhir ini, antar muka perangkat lunak berupa halaman *web* yang bertujuan untuk segala aktivitas pengguna, baik

monitoring, analisa, dan pendaftaran. Berikut adalah gambar rancangan antar muka sistem berupa *wireframe*.



Gambar 3.5 Rancangan Antarmuka Halaman Login

Gambar 3.5 merupakan rancangan antarmuka halaman login ketika pengguna membuka *web* dari aplikasi ini. Untuk menggunakan sistem, pengguna harus terdaftar terlebih dahulu, kemudian melakukan login untuk bisa masuk ke halaman *dashboard* yang bisa dilihat rancangannya pada gambar 3.6

Gambar 3.6 Rancangan Antarmuka Halaman *Dashboard*

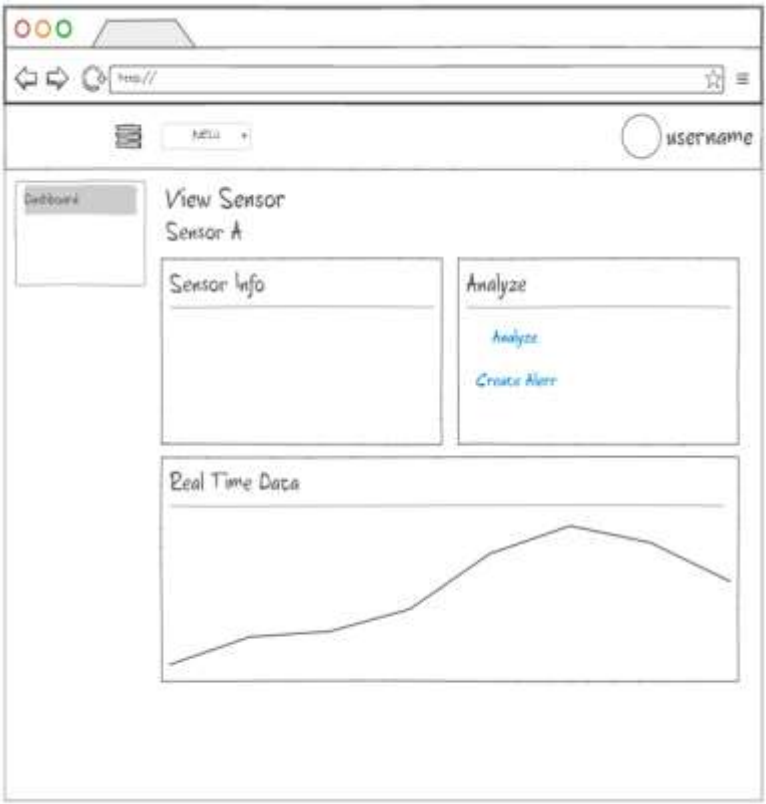
Gambar 3.6 merupakan rancangan antarmuka halaman *dashboard*. Halaman ini merupakan halaman utama dari sistem ketika pengguan sudah melakukan *login*. Pada halaman ini terdapat menu *NEW*, yang jika dilakukan *mouse hover* akan menampilkan *dropdown* yang berisi opsi apakah ingin membuat Sensor baru atau Sensor Kolaborasi baru. Kedua halaman tersebut bisa dilihat pada gambar 3.7 dan 3.8. Di halaman ini pengguna juga dapat melihat semua sensor dan sensor kolaborasi yang dimiliki oleh pengguna tersebut.

Gambar 3.7 Rancangan Antarmuka Halaman *Create New Sensor*

Gambar 3.7 merupakan rancangan antarmuka halaman *create new sensor*. Halaman ini berfungsi untuk memasukkan data sensor yang akan di daftarkan. Data yang diperlukan adalah Nama, Deskripsi, dan sebuah *checkbox* apakah pengguna ingin membuat sensor ini sebagai sensor publik.

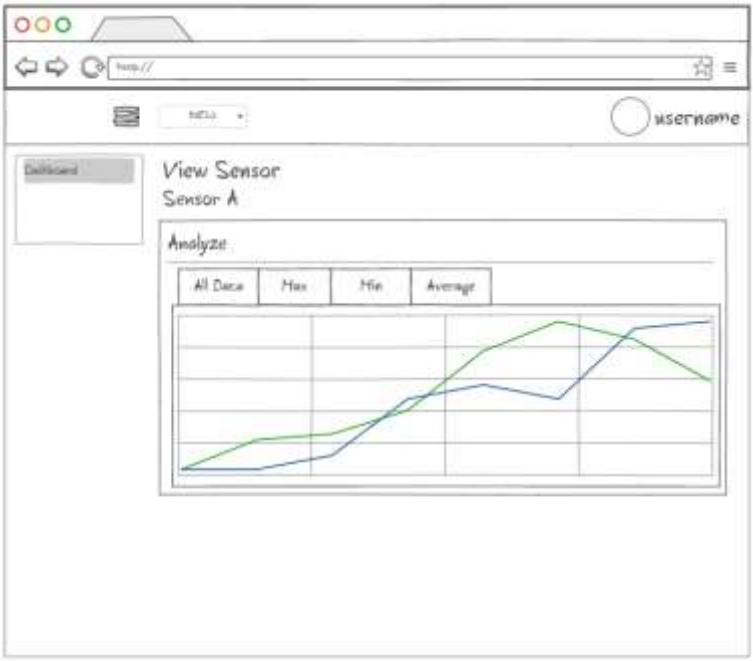
Gambar 3.8 Rancangan Antarmuka Halaman *Create New Collaborative Sensor*

Gambar 3.8 merupakan rancangan antarmuka halaman *craete new collaborative sensor*. Halaman ini berfungsi untuk memasukkan data data yang diperlukan untuk membuat sebuah kolaborasi sensor. Data yang diperlukan adalah kombinasi dari sensor yang ingin dikolaborasikan, *rules* dari masing masing sensor tersebut, serta ekspresi matematik yang akan digunakan untuk melakukan kalkulasi data gabungan dari sensor yang ingin dikolaborasikan.



Gambar 3.9 Rancangan Antarmuka Halaman *View Sensor*

Gambar 3.9 merupakan rancangan antar muka halaman *view sensor*, yang bisa dikases ketika pengguna melakukan klik pada salah satu sensor yang ditampilkan pada halaman *dashboard*. Halaman ini berisi Informasi sensor, tautan menuju halaman *analyze* sensor, tautan untuk membuat *rules* sensor, serta grafik yang menunjukkan data sensor secara *real time*.



Gambar 3.10 Rancangan Antarmuka Halaman *Analyze Sensor*

Gambar 3.10 merupakan rancangan antarmuka halaman *analyze sensor*. Halaman ini bisa di akses melalui tautan yang ada pada halaman *View Sensor*. Pada halaman ini terdapat 4 *tab* yang masing masing berisi grafik dan tabel data. Masing-masing tab tersebut adalah:

- *All Data*, untuk menampilkan semua data sensor yang tersimpan.
- *Max*, untuk menampilkan data nilai maksimum dari sebuah sensor per hari.
- *Min*, untuk menampilkan data nilai minimum dari sebuah sensor per hari.
- *Average*, untuk menampilkan data nilai rata-rata dari sebuah sensor per hari.

BAB IV
IMPLEMENTASI

Pada bab ini akan dijelaskan tentang implementasi perangkat lunak yang dikembangkan untuk Tugas Akhir ini. Implementasi perangkat lunak merupakan bentuk realisasi dari perancangan perangkat lunak yang telah di jelaskan pada bab sebelumnya. Adapun hal-hal yang akan dibahas pada bab ini adalah lingkungan implementasi perangkat lunak, *pseudocode* dari perangkat lunak, dan *screenshot* hasil implementasi perangkat lunak.

4.1. Lingkungan Implementasi

Pada Tugas Akhir ini, lingkungan untuk mengembangkan perangkat lunak ini terdiri atas perangkat keras dan perangkat lunak. Adapun lingkungan implementasi adalah sebagai berikut:

4.1.1. Lingkungan Implementasi Perangkat Keras

Lingkungan implementasi perangkat keras yang digunakan pada Tugas Akhir ini adalah sebuah *laptop* dan mikrokontroler Arduino Uno R3 untuk keperluan pengujian. Adapun spesifikasi dari perangkat keras tersebut adalah sebagai berikut:

- Laptop Apple MacBook Pro
  - OSX 10 El Capitan 10.11.12
  - Processor 2.6 GHz Intel Core i5
  - Memory 8 GB 1600 MHz DDR3

4.1.2. Lingkungan Implementasi Perangkat Lunak

Lingkungan implementasi perangkat lunak yang digunakan untuk menunjang pengembangan perangkat lunak ini terdiri dari

sebuah sistem operasi, IDE, dan beberapa pustaka. Adapun lingkungan implementasi perangkat lunak tersebut adalah sebagai berikut:

- OSX 10 El Capitan 10.11.12 sebagai sistem operasi
- Ninjamock sebagai alat bantu perancangan antarmuka perangkat lunak yang berupa *web*.
- Draw.io sebagai alat bantu perancangan diagram kasus penggunaan perangkat lunak.
- SublimeText sebagai IDE utama dalam pengembangan perangkat lunak.
- MySQL versi 5.6 sebagai RDBMS (*Relational Database Management System*) untuk menyimpan data dan informasi sensor.
- Apache 2.4.17 sebagai *platform webserver* untuk *deployment* situs *web* secara lokal.
- HighCharts sebagai pustaka Javascript utama untuk menampilkan hasil olahan data sebagai grafik.
- Ion_auth sebagai pustaka untuk *authentication* dan *authorization* pada perangkat lunak.

4.2. Implementasi Perangkat Lunak

Pada subbab ini dijelaskan implementasi yang telah dilakukan pada sistem. Berikut dijelaskan mengenai implementasi perangkat lunak berupa *web*.

4.2.1. Implementasi *Sensor-Centric Layer*

Bagian ini menjelaskan bagaimana *server* menerima parameter data yang dikirim oleh sensor fisik. *Web Service* yang di implementasikan pada *layer* ini berupa sebuah *method* yang meggunakan HTTP POST untuk menerima data dalam format JSON, yang kemudian akan di *parse* dan di masukkan kedalam basis data. Prosedurnya adalah sebagai berikut:

1. Membuat *method* yang menunggu data POST masuk.
2. Jika ada data masuk, data JSON di *parse* kemudian dilakukan *checking* terhadap parameter *sensor_key* untuk mengetahui apakah ada sensor yang memiliki *sensor_key* yang sama di basis data. Jika tidak ditemukan, *server* akan mengembalikan pesan *error* ke *client*, dalam hal ini sensor fisik.
3. Jika terdapat *sensor_key* yang cocok pada *database*, maka *server* akan menerima parameter data yang dikirim, dan melakukan *checking* untuk *entry* kolaborasi pada database.
4. Jika terdapat data kolaborasi pada sensor tersebut, maka data kolaborasi akan dimasukkan juga kedalam *database.*

```
1  void function RetrieveData
2    load model
3    raw_data ← raw_input_stream
4    data ← json_decode(raw_data)
5    if checkSensorKey(data['sensor_key') == FALSE
6      then
7        return error response
8    else
9      then
10       if checkCollab(data['sensor_key') == TRUE
11         then
12           model->insert_collab_data()
13     model->insertData(data)
14     return success response
```

Gambar 4.1 Implementasi Fungsi *RetrieveData*

```
1  function checkSensorKey(sensor_key)
2    load model
3    set data to sensor_key
4    set query to db->query("SELECT sensor.id as
5      SENSORID FROM 'sensor' WHERE
6      sensor.sensor_key='sensor_key')
```

```
7    return query->result()
```

Gambar 4.2 Implementasi Fungsi *checkSensorKey*

```
1  function checkCollab(sensor_id)
2    load model
3    set data to sensor_id
4    set db->from to 'sensor_collab'
5    set db->where to "'sensor_x_id', data"
6    set db->or_where to "'sensor_y_id', data"
7    set query to db->get()
8    return query->result()
```

Gambar 4.3 Implementasi Fungsi Model *checkCollab*

```
1  function insertData(data)
2    load model
3    select database
4    insert data to database
```

Gambar 4.4 Implementasi Fungsi *insertData*

Fungsi *layer sensor-centric* lainnya adalah pendaftaran sensor. Prosedur pendaftaran sensor adalah sebagai berikut:

1. Sebelumnya, pengguna harus sudah mempunyai akun pada aplikasi web ini.
2. Jika telah mempunyai akun, pengguna dapat mendaftarkan sensor pada menu "New" dan memilih "Data Point / Sensor" dan memasukkan data sensor yang di minta.
3. Kemudian aplikasi *web* akan menampilkan *sensor_key* yang telah di*generate* untuk sensor anda.

```
1  function registerSensor(data)
2    load model
```

```
3     initialize newSensorData
4     set user_id to getUserData()->id;
5     set sensor_name to input->post('name')
6     set sensor_description to
7       input->post('description')
8     set sensor_key to sha1(sensor_name + time()
9       + user_id)
10    set newSensorData to array_push(user_id,
11      sensor_name, sensor_description,
12      sensor_key)
13    model->createNewSensor(newSensorData)
14    return newSensorData;
```

Gambar 4.5 Implementasi Fungsi *registerSensor*

```
1  function createNewSensor(newSensorData)
2    load model
3    select database
4    insert newSensorData to database
```

Gambar 4.6 Implementasi Fungsi *createNewSensor*

4.2.2. Implementasi *Middleware Layer*

Layer middleware merupakan bagian yang berfungsi melakukan virtualisasi sensor fisik. Fungsi yang dijalankan oleh *layer* ini adalah melakukan proses *fetching* dan kalkulasi data dari *database* untuk kemudian ditampilkan pada *client-centric layer*.

Untuk *fetching* data dari *database*, diimplementasikan *controller getAllSensorData*, *getRealTimeSensorData*, dan beberapa *model* pendukungnya.

Fungsi *getAllSensorData* berfungsi untuk mengambil semua data sensor yang tersimpan dengan *model getSensorReading* berdasarkan *sensor_id*, dan menyajikannya dalam format JSON agar bisa diproses *HighCharts* pada *client-centric layer*.

```
1   function getAllSensorData(sensor_id)
2     load model
3     initialize data
4     initialize querydata
5     initialize datareading
6     set querydata to
7       model->getSensorReading(sensor_id)
8     foreach querydata as row
9     do
10      set datetime to strtotime(row->timestamp)
11       * 1000 //konversi ke javascript time
12      set datareading to row->datareading
13      set data to array_push(datetime,
14       datareading)
15    return json_encode(data)
```

Gambar 4.7 Implementasi Fungsi *getAllSensorData*

Fungsi *getRealTimeSensorData* berfungsi untuk mengambil data sensor secara *real time*. Hal ini dilakukan dengan memanfaatkan pustaka *HighCharts* yang memanggil fungsi *getRealTimeSensorData* secara berkala dan menggunakan data yang dikembalikan oleh fungsi ini untuk me *render* grafik secara berkala.

```
1   function getRealTimeSensorData(sensor_id)
2     load model
3     initialize data
4     initialize result
5     set data to
6       model->getSensorReading(sensor_id)
7     foreach data as row
8     do
9       set result to
10       array(strtotime(row->timestamp) * 1000 ,
11       row->datareading)
12    return json_encode(result)
```

Gambar 4.8 Implementasi Fungsi *getRealTimeSensorData*

```
function getSensorReading(sensor_id)
  set db->select to
     'sensor_data.sensor_reading as
     datareading, sensor_data.timestamp as
     timestamp'
  set db->from to 'sensor_data'
  set db->where to
    " 'sensor_data.sensor_id', sensor_id "
  set db->order_by to
    " 'sensor_data.timestamp', 'ASC' "
  set query to db->get()
  return query->result()
```

Gambar 4.9 Implementasi Fungsi Model *getSensorReading*

Fungsi kalkulasi data juga dilakukan pada *layer* ini. Kalkulasi yang dilakukan berupa nilai rerata, maksimum, dan minimum dari data sensor yang tersimpan di *database*.

Untuk melakukan kalkulasi tersebut, diimplementasikan fungsi *getMaxSensorReading*, *getMinSensorReading*, dan *getAverageSensorReading*, dan *model* untuk transaksi *database* dengan nama fungsi yang sama.

```
function getMaxSensorReading(sensor_id)
  load model
  initialize data
  initialize datetime
  initialize querydata
  initialize datareading
  set querydata to
    model->getMaxSensorReading(sensor_id)
  foreach querydata as row
  do
    set datetime to strtotime(row->sensordate)
     * 1000 //konversi ke javascript time
    set datareading to row->sensordata
    set data to array_push(datetime,
     datareading)
```

```
16    return json_encode(data)
```

Gambar 4.10 Implementasi Fungsi *getMaxSensorReading*

```
1   function getMinSensorReading(sensor_id)
2     load model
3     initialize data
4     initialize datetime
5     initialize querydata
6     initialize datareading
7     set querydata to
8       model->getMinSensorReading(sensor_id)
9     foreach querydata as row
10    do
11      set datetime to strtotime(row->sensordate)
12        * 1000 //konversi ke javascript time
13      set datareading to row->sensordata
14      set data to array_push(datetime,
15        datareading)
16    return json_encode(data)
```

Gambar 4.11 Implementasi Fungsi *getMinSensorReading*

```
1   function getAverageSensorReading(sensor_id)
2     load model
3     initialize data
4     initialize datetime
5     initialize querydata
6     initialize datareading
7     set querydata to
8       model->getAverageSensorReading(sensor_id)
9     foreach querydata as row
10    do
11      set datetime to strtotime(row->sensordate)
12        * 1000 //konversi ke javascript time
13      set datareading to row->sensordata
14      set data to array_push(datetime,
15        datareading)
16    return json_encode(data)
```

Gambar 4.12 Implementasi Fungsi *getAverageSensorReading*

```
1   function getMaxSensorReading(sensor_id)
2     set db->select to
3       'MAX(sensor_data.datareading) as
4       sensordata, DATE(sensor_data.timestamp) as
5       sensordate'
6     set db->from to 'sensor_data'
7     set db->where to
8       " 'sensor_data.sensor_id', sensor_id "
9     set db->group_by to
10      'DATE(sensor_data.timestamp)'
11    set db->order_by to
12      " 'DATE(sensor_data.timestamp)', 'DESC' "
13    set query to db->get()
14    return query->result()
```

Gambar 4.13 Implmenetasi Fungsi Model *getMaxSensorReading*

```
1   function getMinSensorReading(sensor_id)
2     set db->select to
3       'MIN(sensor_data.datareading) as
4       sensordata, DATE(sensor_data.timestamp) as
5       sensordate'
6     set db->from to 'sensor_data'
7     set db->where to
8       " 'sensor_data.sensor_id', sensor_id "
9     set db->group_by to
10      'DATE(sensor_data.timestamp)'
11    set db->order_by to
12      " 'DATE(sensor_data.timestamp)', 'DESC' "
13    set query to db->get()
14    return query->result()
```

Gambar 4.14 Implementasi Fungsi Model *getMinSensorReading*

```
1   function getAverageSensorReading(sensor_id)
2     set db->select to
3       'AVG(sensor_data.datareading) as
4       sensordata, DATE(sensor_data.timestamp) as
5       sensordate'
```

```
6     set db->from to 'sensor_data'
7     set db->where to
8       " 'sensor_data.sensor_id', sensor_id "
9     set db->group_by to
10      'DATE(sensor_data.timestamp)'
11    set db->order_by to
12      " 'DATE(sensor_data.timestamp)', 'DESC' "
      set query to db->get()
      return query->result()
```

Gambar 4.15 Implementasi Fungsi Model *getAverageSensorReading*

Salah satu fitur pada perangkat lunak ini adalah kolaborasi data antar sensor yang artinya data dari beberapa sensor (dalam hal ini dibatasi dua atau tiga sensor) dapat dikolaborasikan dalam bentuk operasi matematik (penjumlahan, pengurangan, perkalian, dan pembagian). Operasi ini dilakukan pada *sensor-centric layer* ketika data masuk, kemudian dikalkulasikan dan di *record* ke dalam *database*. Pada *layer* ini, fungsi yang terkait fitur kolaborasi adalah *fetching* data hasil kolaborasi. Fungsi yang di implementasikan untuk *fetching* data kolaborasi adalah *model getCollabData*. *Model getCollabData* mengambil data kolaborasi dari *database* berdasarkan parameter *collab_id* dan mengembalikan isi tabel *sensor_collab* untuk disajikan ke pengguna pada *client-centric* baik dalam bentuk tabel maupun grafik.

```
1   function getCollabData(collab_id)
2     set db->select to
3       'sensor_collab.sensor_collab_id as
4       COLLABID, sensor_collab.sensor_collab_name
5       as COLLABNAME,
6       sensor_collab.sensor_collab_desc as
7       COLLABDESC, sensor_collab_data.timestamp
8       as TIMESTAMP,
9       sensor_collab_data.sensor_reading as
10      DATAREADING'
11    set db->from to 'sensor_collab
12
```

```
13    set db->join to " 'sensor_collab_data',
14      'sensor_collab_data.sensor_collab_id =
15      sensor_collab.sensor_collab_id', 'left' "
16    set db->where to
17      " 'sensor_collab.sensor_collab_id',
18      collab_id "
19    set query to db->get()
20    return query->result()
```

Gambar 4.16 Implementasi Fungsi Model *getCollabData*

4.2.3. Implementasi *Client-Centric Layer*

Layer client-centric merupakan bagian yang berhubungan langsung dengan pengguna. *Layer* ini memfasilitasi dan mengelola interaksi pengguna dengan sistem, mengelola *membership* dan *session*, dan menyajikan data yang telah di *fetch* dan di olah pada *middleware layer*. Layanan yang dijalankan oleh *layer* ini adalah pendaftaran, *authentication*, dan *authorization* akun pengguna, serta penyajian data yang telah diolah dalam bentuk tabel dan grafik. Layanan tersebut direalisasikan dengan implementasi fungsi *viewSensor*, dan *analyzeSensor*. Kedua fungsi tersebut mengambil data hasil olahan berdasarkan parameter *sensor_id*.

```
1  function viewSensor(sensor_id)
2    load model
3    initialize data
4    set data->userdata to getUserData();
5    set data->sensordata to
6      model->getSensorData(sensor_id)
7    set load->view(data)
```

Gambar 4.17 Implementasi Fungsi View *viewSensor*

```
1  function analyzeSensor(sensor_id)
2    load model
```

```
3    initialize data
4    set data->userdata to getUserData();
5    set data->sensordata to
6      model->getSensorData(sensor_id)
7    set data->maxsensordata to
8      model->getMaxSensorReading(sensor_id)
9    set data->minsensordata to
10     model->getMinSensorReading(sensor_id)
11   set data->averagesensordata to
12     model->getAverageSensorReading(sensor_id)
13   set load->view(data)
```

Gambar 4.18 Implementasi Fungsi View *analyzeSensor*

4.3. Implementasi Arsitektur Jaringan Perangkat Lunak

Sesuai dengan rancangan arsitektur jaringan sisitem pada bab sebelumnya, arsitektur jaringan perangkat lunak akan di implementasikan pada subbab berikut. Hal yang akan dibahas pada subbab ini adalah implementasi arsitektur dan lingkungan *http server* dan *http load balancer*, serta implementasi arsitektur dan lingkungan *database server* dan *load balancer*.

4.3.1. Implementasi Arsitektur Jaringan dan Lingkungan *HTTP Server* dan *HTTP Load Balancer*

Pada subbab ini akan dibahas implementasi arsitektur jaringan dan lingkungan untuk *http server* dan *http load balancer*. Pada bab sebelumnya, dapat dilihat bahwa arsitektur jaringan terdiri dari dua *HTTP Server*, dua *database server*, dan dua *load balancer*. *HTTP Server* dan *HTTP load balancer* di implementasikan dengan memanfaatkan *virtual machine (VM) Elastic Compute Cloud (EC2)* yang dijalankan di atas infrastruktur komputasi awan yang disediakan oleh *Amazon Web Services (AWS)*. Spesifikasi *instance VM* yang dibuat pada infrastuktur komputasi awan *AWS* adalah sebagai berikut:

- Model : t2.micro
- vCPU : 1
- Memori : 1 GB
- Storage : 10 GB
- Sistem Operasi : Ubuntu Server 14.04 64 bit

Pada gambar 4.19 dapat dilihat bahwa kedua buah *VM instance* pada *AWS* telah berhasil di buat.

Gambar 4.19 Daftar *EC2 VM Instance* yang berhasil dibuat

Kemudian pada masing-masing *instance VM* tadi di lakukan instalasi perangkat lunak sebagai berikut:

- *HTTP Server* : Nginx versi 1.4.6
- PHP
- PHP5-FPM
- PHP5-MySQL

Setelah proses instalasi selesai, dilakukan konfigurasi *nginx http server*. Gambar 4.20 adalah kode konfigurasi *Nginx* untuk perangkat lunak ini.

```
server {
    listen 80 default_server;
    listen [::]:80 default_server ipv6only=on;

    root /usr/share/nginx/html;
    index index.php index.html index.htm;

    server_name sensorman-lb-1143910599.ap-
    southeast-1.elb.amazonaws.com;

    location / {
        try_files $uri $uri/ /index.php?args;
    }

    error_page 404 /404.html;
    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
        root /usr/share/nginx/html;
    }

    location ~ \.php$ {
        try_files $uri =404;
        fastcgi_split_path_info
    ^(.+\.php)(/.+)$;
        fastcgi_pass unix:/var/run/php5-
    fpm.sock;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME
    $document_root$fastcgi_script_name;
        include fastcgi_params;
    }
}
```

Gambar 4.20 Kode Konfigurasi *Nginx HTTP Server*

Setelah konfigurasi *Nginx*, untuk memanfaatkan dua *HTTP server* yang ada, di butuhkan sebuah *HTTP load balancer*. Fungsi dari *load balancer* adalah membagi *incoming traffic* ke *HTTP Server* yang tersedia. Pada *AWS*, sudah tersedia layanan *load balancer* dengan nama *Elastic Load Balancer*. Gambar 4.21 merupakan sebuah *load balancer instance* yang berhasil dibuat dan dikonfigurasi untuk mengarahkan *traffic* yang masuk ke dua

buah *EC2 VM instance* yang telah dibuat dan dikonfigurasi sebelumnya.

Gambar 4.21 *Load Balancer instance* dan konfigurasinya

4.3.2. Implementasi Arsitektur dan Lingkungan *Database Server* dan *Database Load Balancer*

Pada subbab ini akan dibahas mengenai implementasi arsitektur jaringan dan lingkungan untuk *database server* dan *load balancer*. Sesuai dengan rancangan aristektur pada bab sebelumnya, pada lingkungan ini akan diimplementasi dua *database server* dan sebuah *load balancer*. Implementasi arsitektur jaringan dan lingkunan *database server* ini dilakukan dengan memanfaatkan tiga buah *virtual machine* yang berjalan di atas infrasturktur komputasi awan *Microsoft Azure*. Spesifikasi *virtual machine* yang dibuat pada infrastruktur komputasi awan *Microsoft Azure* adalah sebagai berikut:

- *Database Server*:
  - Model : Standard A1
  - CPU *Core* : 1
  - Memori : 1.75 GB
  - *Storage* : 70 GB
  - Sistem Operasi : Ubuntu Server 14.04 64 bit
  - MySQL 5.6 *Relational Database Management System*.

- *Load Balancer*:
    - Model : Basic A0
    - CPU *Core* : 1
    - Memori : 0.75 GB
    - *Storage* : 20 GB
    - Sistem Operasi : Ubuntu Server 14.04 64 bit
    - *Load Balancer* : HAProxy 1.6

Dari spesifikasi di atas dapat dilihat bahwa spesifikasi *virtul machine* untuk *database server* relatif tinggi. Hal ini disebabkan oleh sumber daya yang dibutuhkan oleh MySQL cukup tinggi.

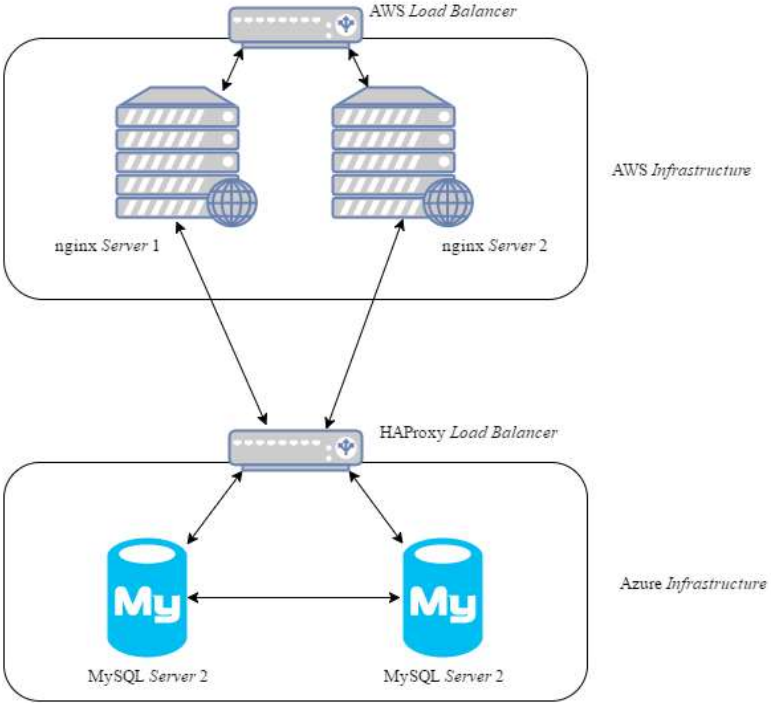
Setelah berhasil membuat *virtual machine*, pada *virtual machine database server* dilakukan instalasi dan konfigurasi MySQL. Konfigurasi yang dilakukan pada MySQL adalah konfigurasi replikasi dengan skema *master – master*, yang artinya kedua *server database* tersebut dapat dimanfaatkan sekaligus dengan memanfaatkan *load balancer* untuk mengatur transaksi basis data yang masuk.

Pada *virtual machine load balancer*, dilakukan instalasi dan konfigurasi perangkat lunak HAProxy untuk *load balancing*. Gambar 4.22 adalah kode konfigurasi HAProxy.

```
1  global
2      log 127.0.0.1 local0 notice
3      user haproxy
4      group haproxy
5
6  defaults
7      log global
8      retries 2
9      timeout connect 3000
10     timeout server 5000
11     timeout client 5000
12
13 listen mysql-cluster
14     bind 10.0.0.6:3306
15     mode tcp
16     option mysql-check user haproxycheck
17     balance roundrobin
18     server mysql-1 10.0.0.4:3306 check
19     server mysql-2 10.0.0.5:3306 check
20
21 listen stats-server
22     bind 10.0.0.6:8080
23     mode http
24     stats enable
25     stats uri /
26     stats realm Strictly\ Private
27     stats auth bayan:bayan12345
```

Gambar 4.22 Kode konfigurasi HAProxy

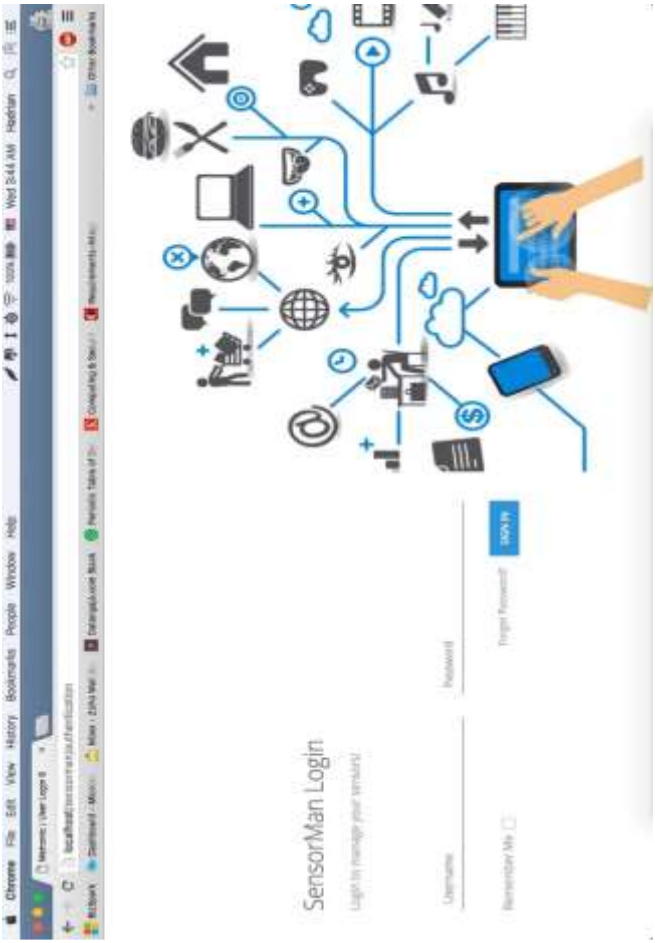
Dari konfigurasi HAProxy di atas dapat dilihat HAProxy mengarahkan *traffic* yang datang ke *server* mysql-1 pada IP 10.0.0.4 dan mysql-2 pada ip 10.0.0.4



Gambar 4.23 Implementasi arsitektur jaringan perangkat lunak

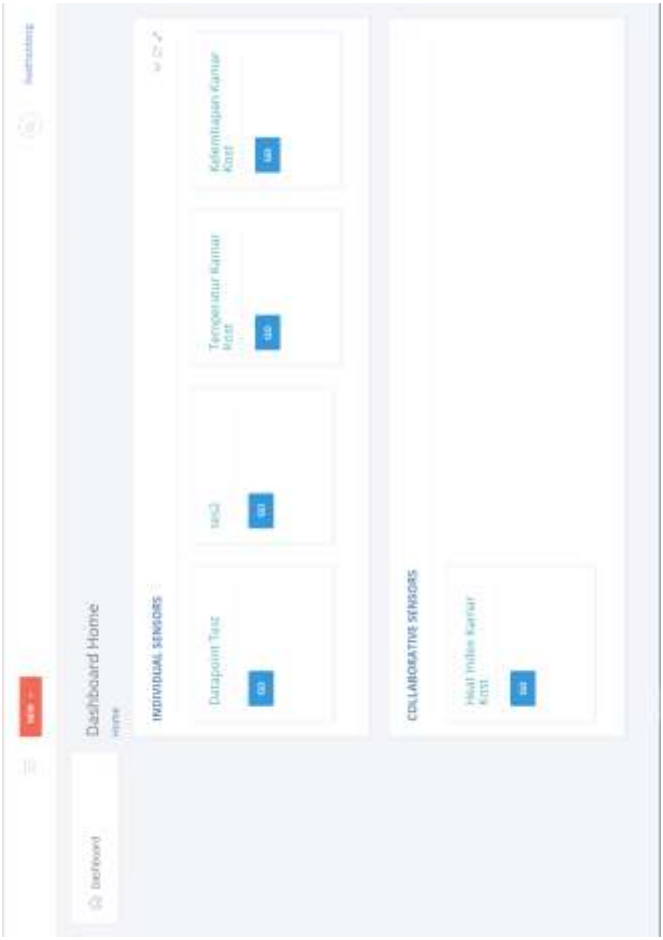
4.4. Implementasi Antarmuka Perangkat Lunak

Implementasi antarmuka perangkat lunak yang dilakukan hanya pada aplikasi *web* karena perangkat lunak memang berbasis *web*. *Web* tersebut memiliki satu *sidebar menu* yaitu Dashboard, dan sebuah menu *dropdown* New yang berisi menu New Data Point / Sensor, 2 Sensor Collaboration, dan 3 Sensor Collaboration.



Gambar 4.24 Implementasi Antarmuka Halaman *Login*

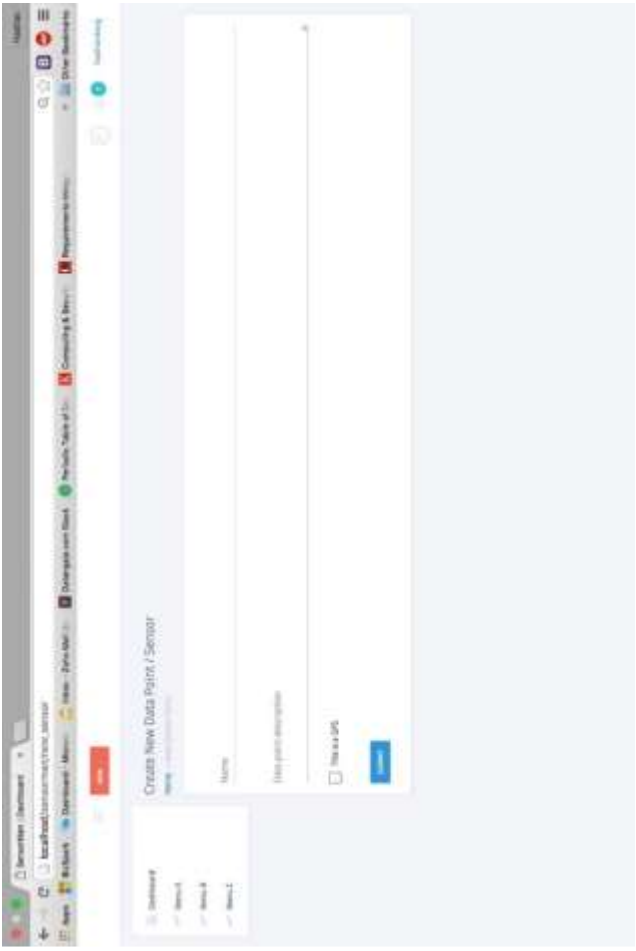
Gambar 4.24 merupakan implementasi dari antarmuka halaman login ketika pengguna membuka *web* dari aplikasi ini. Untuk menggunakan sistem, pengguna harus terdaftar terlebih dahulu, kemudian melakukan login untuk bisa masuk ke halaman *dashboard*.



Gambar 4.25 Implementasi Antarmuka Halaman *Dashboard*

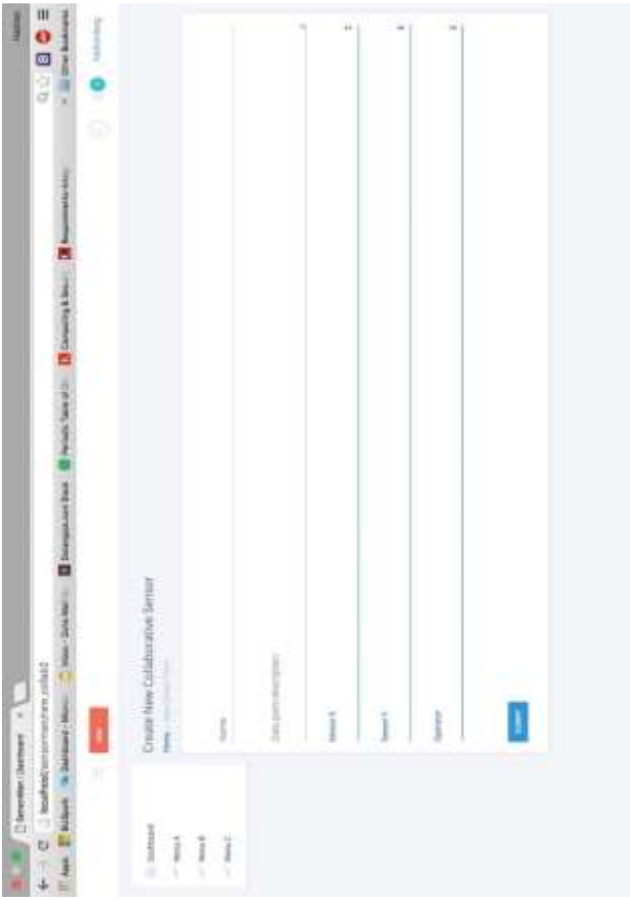
Gambar 4.25 merupakan implementasi dari antarmuka halaman *dashboard*. Halaman ini merupakan halaman utama dari sistem ketika pengguan sudah melakukan *login*. Pada halaman ini terdapat menu *NEW*, yang jika dilakukan *mouse hover* akan

menampilkan *dropdown* yang berisi opsi apakah ingin membuat Sensor baru atau Sensor Kolaborasi baru. Kedua halaman tersebut bisa dilihat pada Gambar 4.26 dan Gambar 4.27. Di halaman ini pengguna juga dapat melihat semua sensor dan sensor kolaborasi yang dimiliki oleh pengguna tersebut.



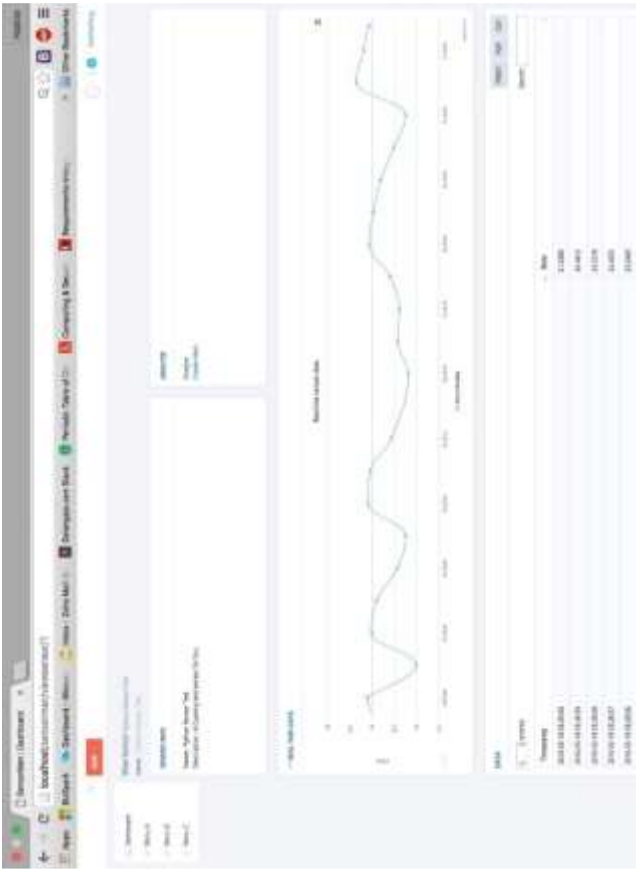
Gambar 4.26 Implementasi Antarmuka Halaman *Create New Sensor*

Gambar 4.26 merupakan implementasi dari antarmuka halaman *create new sensor*. Halaman ini berfungsi untuk memasukkan data sensor yang akan di daftarkan. Data yang diperlukan adalah Nama, Deskripsi, dan sebuah *checkbox* apakah pengguna ingin mengirimkan data GPS juga pada sensor ini.



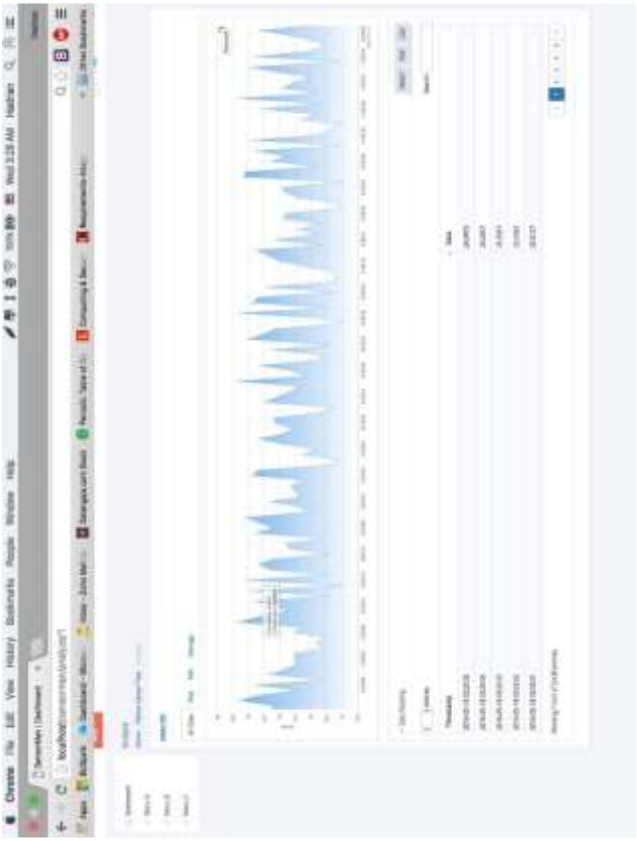
Gambar 4.27 Implementasi Antarmuka Halaman *Create New Collaborative Sensor*

Gambar 4.27 merupakan implementasi dari antarmuka halaman *craete new collaborative sensor*. Halaman ini berfungsi untuk memasukkan data data yang diperlukan untuk membuat sebuah kolaborasi sensor. Data yang diperlukan adalah kombinasi dari sensor yang ingin dikolaborasi kan, *rules* dari masing masing sensor tersebut, serta ekspresi matematik yang akan digunakan untuk melakukan kalkulasi data gabungan dari sensor yang ingin dikolaborasikan.



Gambar 4.28 Implementasi Antarmuka Halaman *View Sensor*

Gambar 4.28 merupakan implementasi dari antarmuka halaman *view sensor*, yang bisa dikases ketika pengguna melakukan klik pada salah satu sensor yang ditampilkan pada halaman *dashboard*. Halaman ini berisi Informasi sensor, tautan menuju halaman *analyze* sensor, tautan untuk membuat *rules* sensor, grafik yang menunjukkan data sensor secara *real time*, serta sebuah tabel yang berisi semua data dari sensor terkait yang bisa di unduh.



Gambar 4.29 Implementasi Antarmuka Halaman *Analyze Sensor*

Gambar 4.29 merupakan rancangan antarmuka halaman *analyze sensor*. Halaman ini bisa diakses melalui tautan yang ada pada halaman *View Sensor*. Pada halaman ini terdapat 4 *tab* yang masing masing berisi grafik dan tabel data. Masing masing tab tersebut adalah:

- *All Data*, untuk menampilkan semua data sensor yang tersimpan.
- *Max*, untuk menampilkan data nilai maksimum dari sebuah sensor per hari.
- *Min*, untuk menampilkan data nilai minimum dari sebuah sensor per hari.
- *Average*, untuk menampilkan data nilai rata-rata dari sebuah sensor per hari

[Halaman ini sengaja dikosongkan]

BAB V
UJI COBA DAN EVALUASI

Bab ini berisi penjelasan mengenai hasil uji coba dan evaluasi dari implementasi perangkat lunak yang telah dilakukan. Adapun hal-hal yang akan dibahas pada bab ini adalah lingkungan uji coba perangkat lunak, uji coba fungsionalitas perangkat lunak, dan uji coba performa perangkat lunak. Uji coba akan dilakukan dengan beberapa skenario.

5.1. Lingkungan Uji Coba

Subbab Lingkungan Uji Coba menjelaskan mengenai lingkungan tempat pengujian perangkat lunak. Lingkungan uji coba pada kasus ini terdiri dari perangkat keras dan perangkat keras dengan spesifikasi sebagai berikut:

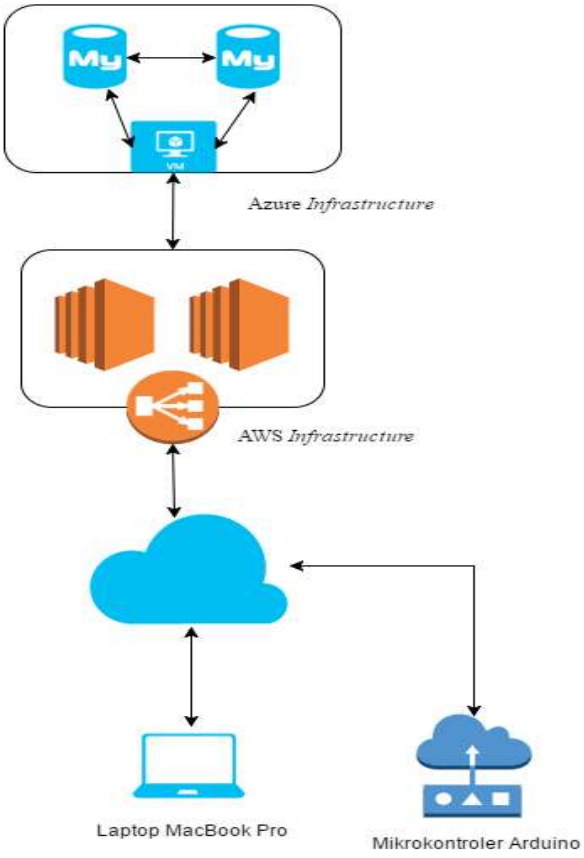
- Perangkat Keras
  - Laptop MacBook Pro, dengan spesifikasi:
    - Prosesor 2.6 GHz Intel Core i5
    - Memori 8 GB 1600 MHz DDR3

- Perangkat Lunak
  - Sistem Operasi Mac OSX El Capitan 10.11.12
  - *Web Broswer* Google Chrome
  - SublimeText *text editor*
  - Python untuk keperluan simulasi dan pengujian
  - *Webserver* Apache, dan *database* MySQL
  - Postman yang berfungsi untuk menguji *web service* dan fungsi yang menggunakan HTTP POST.
  - *Siege* yang berfungsi untuk menguji performa dan ketahanan.
  - *htop* untuk mengetahui tingkat utilisasi CPU server.

Untuk keperluan uji coba performa, perangkat lunak di *deploy* pada lingkungan yang telah diimplementasi pada bab sebelumya. Lingkungan dijalankannya aplikasi, seperti yang bisa dilihat pada Gambar 5.1, memiliki spesifikasi sebagai berikut:

- Dua *instance virtual machine* yang berfungsi sebagai *web server* dengan spesifikasi:
  - CPU : 1 *Core*
  - Memori : 1 GB
  - *Storage* : 10 GB
  - Sistem Operasi : Ubuntu Server 14.04 64 bit
  - *HTTP server* : Nginx versi 1.4.6
  - PHP Versi 5.5.9-1ubuntu4.17
  - IP Privat VM-1 : 172.31.16.58
  - IP Privat VM-2 : 172.31.16.17
  - IP Publik VM-1 : 52.221.251.29
  - IP Publik VM-2 : 54.169.232.169
- Satu *instance AWS load balancer* dengan konfigurasi:
  - IP Publik : 52.76.91.136
- Dua *instance virtual machine* yang berfungsi sebagai *database server* dengan spesifikasi:
  - CPU : 1 *Core*
  - Memori : 1.75 GB
  - *Storage* : 70 GB
  - Sistem Operasi : Ubuntu Server 14.04 64 bit
  - MySQL 5.6 sebagai *Relational Database Management System*
  - IP Privat DB-1 : 10.0.0.4
  - IP Privat DB-2 : 10.0.0.5
  - IP Publik DB-1 : 52.163.89.96
  - IP Publik DB-2 : 52.163.92.99
- Satu *instance virtual machine* yang berfungsi sebagai *database load balancer* dengan spesifikasi:
  - CPU : 1 *Core*
  - Memori : 0.75 GB
  - *Storage* : 20 GB

- Sistem Operasi : Ubuntu Server 14.04 64 bit
- HAProxy sebagai perangkat lunak *load balancer*
- IP Publik *HAProxy* : 10.0.0.6
- IP Publik *HAProxy* : 52.163.90.54



Gambar 5.1 Lingkungan Uji Coba.

5.2. Skenario Uji Coba

Pada subbab ini akan dijelaskan tentang skenario uji coba prangkat lunak yang telah dibangun. Skenario uji coba dibagi menjadi dua bagian, yaitu uji coba fungsionalitas dan uji coba performa perangkat lunak.

5.2.1. Uji Coba Fungsionalitas

Uji coba fungsionalitas bertujuan untuk menguji apakah fungsi-fungsi utama pada perangkat lunak berhasil di implementasikan dan berjalan sesuai dengan yang diharapkan.

5.2.1.1. Pengujian *Sensor-centric Layer*

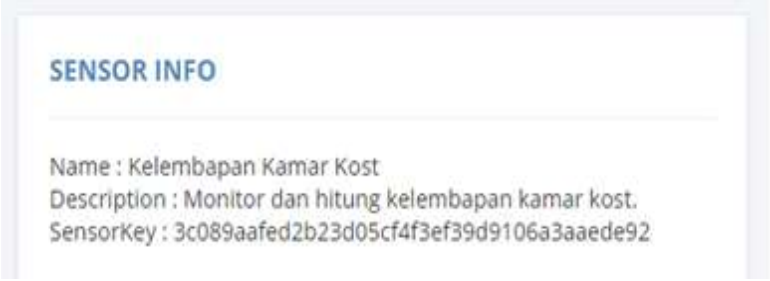
Pengujian fungsi fungsi pada *sensor-centric layer* dilakukan dengan mecoba mendaftarkan sensor, dan melakukan pengiriman data dari sensor yang telah didaftarkan. Pendaftaran sensor dilakukan melalui antarmuka *web*, dan pengiriman data sensor akan dilakukan dengan program Python untuk simulasi dan mikrokontroler Arduino untuk pengujian dengan data yang *real*. Berikut adalah rincian prosedur uji coba *sensor-centric layer*.

Tabel 5.1 Prosedur uji coba pendaftaran sensor baru

ID	**UJ-01**
Nama	Uji Coba Pendaftaran Sensor Baru
Tujuan Uji Coba	Menguji fungsionalitas perangkat lunak untuk mendaftarkan dan menggenrasi *sensor_key* untuk sensor baru.
Kondisi Awal	Perangkat lunak dijalankan
Skenario	Perangkat lunak dijalankan, kemudian di akses melalui *web browser* Google Chrome, dan melakukan pendaftaran sensor.
Masukan	Data sensor baru
Keluaran	*sensor_key* yang telah di *generate*.

Hasil Uji Coba	Sensor berhasil di daftarkan dan perangkat lunak berhasil melakukan *generate sensor_key* baru.

Tabel 5.1 diatas merupakan rincian prosedur pengujian Pendaftaran Sensor Baru pada perangkat lunak. Pada Gambar 5.2 dan Gambar 5.3 ditampilkan keluaran dari perangkat lunak, yaitu *sensor_key* dan data sensor baru yang telah ter-*record* pada *database*.



Gambar 5.2 Tampilan antarmuka *View Sensor* menunjukkan sensor yang baru di daftarkan

Gambar 5.3 Tampilan *record* pada *database* menunjukkan sensor yang baru di daftarkan

Fungsi berikutnya pada *sensor-centric layer* adalah fungsi penerimaan data. Fungsi ini dijalankan oleh sebuah *web service* yang memanfaatkan metode HTTP POST untuk menerima, *parsing*, mengecek *sensor_key* dan memasukkan data kedalam *database*.

Tabel 5.2 Prosedur uji coba menerima data daru sensor

ID	**UJ-02**
Nama	Uji Coba Penerimaan Data pada *Server*.
Tujuan Uji Coba	Menguji fungsionalitas perangkat lunak untuk menerima data dari sensor yang telah di daftarkan.

Kondisi Awal	Perangkat lunak dijalankan
Skenario	1. Perangkat lunak dijalankan, kemudian kemudian sebuah program Python akan mengakses *web service* perangkat lunak dan mengirim data dalam format JSON yang berisi *sensor_key* dan *sensor_reading*. 2. Perangkat lunak dijalankan, kemudian dengan menggunakan sebuah mikrokontroler Arduino dan sensor kelembapan, mikrokontroler Arduino akan mengirimkan data hasil bacaan sensor kelembapan ke server aplikasi.
Masukan	1. Data dalam format JSON dari program Python untuk simulasi. 2. Data dalam format JSON dari mikrokontroler Arduino.
Keluaran	Respon berhasil dari server, dan data yang berhasil di terima oleh server dalam format JSON.
Hasil Uji Coba	*Web service* berhasil menerima, *parsing*, mengecek, dan memasukkan data kedalam *database*.

Berdasarkan skenario pada tabel diatas, perangkat lunak melakukan penerimaan data dari sensor. Gambar 5.4 adalah sepuluh data percobaan yang dikirim dari sensor Python:

Gambar 5.4 Data yang dikirim dari sensor

Gambar 5.4 menunjukkan sepuluh data percobaan yang dikirim dari sensor. Sepuluh data tersebut dikirim dalam bentuk JSON, kemudian *server* melakukan *parsing*, pengecekan *sensor_key*, kemudian memasukkan data tersebut ke *database*. Sensor terkait mempunyai *sensor_key* yang bernilai "0d1b6542eeef87922f3661402611ea89ef411cdf". Pada Gambar 5.5 dapat dilihat bahwa *id* dari sensor ini adalah "19".

id	user_id	sensor_key	sensor_name	sensor_description
11	2	3b623224a84a1dd88de344037e17c20ddfb72725	Datapoint Test	Test datapoint.
16	2	cb3e705e318dafc74a95ca4c725c249b63bbadc8	tes2	
17	2	b82644431a453ae5a02503aef529f86a74effb0d	Temperatur Kamar Kost	Monitoring temperatur kamar kost.
18	2	3c089aafed2b23d05cf4f3ef39d9106a3aaede92	Kelembapan Kamar Kost	Monitor dan hitung kelembapan kamar kost.
19	3	0d1b6542eeef87922f3661402611ea89ef411cdf	Uji Coba Pengiriman	Test Pengiriman Data

Gambar 5.5 *Database record* yang menunjukkan *id* sensor terkait

```
[[1465165584000,28.5502],
[1465165585000,24.6016],
[1465165587000,31.108],
[1465165588000,33.9212],
[1465165589000,34.2686],
[1465165590000,29.6188],
[1465165591000,30.9384],
[1465165592000,31.6707],
[1465165593000,34.8946],
[1465165594000,34.6046]]
```

Gambar 5.6 Hasil keluaran fungsi *getAllSensorData*

Gambar 5.6 diatas menunjukkan hasil tampilan fungsi *getAllSensorData* yang mengembalikan semua data sensor berdasarkan *id* sensor dalam bentuk JSON. Data yang

dikembalikan berupa waktu saat data masuk ke server dalam *unix time*, dan nilai dari bacaan sensor.

Pada skenario kedua, dilakukan pengiriman data dengan sensor sungguhan, dengan menggunakan sebuah mikrokontroler Arduino dan sensor kelembapan. Gambar 5.7 adalah kode sumber untuk mengirim data ke server.

```
#include <EtherCard.h>
#include <dht.h>

dht DHT;
#define DHT11_PIN 7

static byte mymac[] = { 0x74,0x69,0x69,0x2D,0x30,0x31 };
#define BUFFERSIZE 350
byte Ethernet::buffer[BUFFERSIZE];
Stash stash;
const char website[] PROGMEM = "sensorman.bayanulhaq.me";
#define SENSORKEY   "df000bc04cb16c65265ba52ede78028d2c451abb"
//float DATA = 22.00;
static byte session;

uint32_t nextSeq;
static void my_callback (byte status, word off, word len) {

    if (strncmp_P((char*) Ethernet::buffer+off,PSTR("HTTP"),4) == 0) {
        Serial.println(">>>");
        // first reply packet
        nextSeq = ether.getSequenceNumber();
    }

    if (nextSeq != ether.getSequenceNumber())
    {
      Serial.print(F("<IGNORE DUPLICATE(?) PACKET>"));
      return;
    }

    uint16_t payloadlength = ether.getTcpPayloadLength();
    int16_t chunk_size   = BUFFERSIZE-off-1;
    int16_t char_written = 0;
    int16_t char_in_buf  = chunk_size < payloadlength ? chunk_size : payloadlength;

    while (char_written < payloadlength)
    {
        Serial.write((char*) Ethernet::buffer+off,char_in_buf);
        char_written += char_in_buf;

        char_in_buf = ether.readPacketSlice((char*)
        Ethernet::buffer+off,chunk_size,off+char_written);
    }

    nextSeq += ether.getTcpPayloadLength();
}
```

```
void setup () {
    Serial.begin(57600);
    Serial.println(F("\n[Persistence+readPacketSlice]"));


    if (ether.begin(sizeof Ethernet::buffer, mymac) == 0)
        Serial.println(F("Failed to access Ethernet controller"));
    if (!ether.dhcpSetup())
        Serial.println(F("DHCP failed"));

    ether.printIp("IP:  ", ether.myip);
    ether.printIp("GW:  ", ether.gwip);
    ether.printIp("DNS: ", ether.dnsip);

    if (!ether.dnsLookup(website))
        Serial.println("DNS failed");

    ether.printIp("SRV: ", ether.hisip);
    ether.persistTcpConnection(true);
}

void loop () {
  ether.packetLoop(ether.packetReceive());
  Serial.println("Data....");
  float DATA = DHT.Temperature;
  byte sd = stash.create();
  stash.print("sensor_key=");
  stash.print(SENSORKEY);
  stash.print("&sensor_reading=");
  stash.println(DATA);
  stash.save();
  int stash_size = stash.size();
  Stash::prepare(PSTR("POST http://$F/send-data HTTP/1.0" "\r\n"
    "Host: $F" "\r\n"
    "Content-Length: $D" "\r\n"
    "Content-Type: application/x-www-form-urlencoded" "\r\n"
    "\r\n"
    "$H"),
  website, website, stash_size, sd);
  session = ether.tcpSend();
  delay(1000);
}
```

Gambar 5.7 Kode sumber pada mikrokontroler Arduino

←T→	id ▾ 1	sensor_id	sensor_reading
Edit Copy Delete	5983	15	27
Edit Copy Delete	5982	15	27
Edit Copy Delete	5981	15	27
Edit Copy Delete	5980	15	27
Edit Copy Delete	5979	15	27
Edit Copy Delete	5978	15	27
Edit Copy Delete	5977	15	27
Edit Copy Delete	5976	15	27
Edit Copy Delete	5975	15	27
Edit Copy Delete	5974	15	27

Gambar 5.8 *Database record* yang menunjukkan data hasil uji coba pengiriman dengan mikrokontroler Arduino.

Gambar 5.8 merupakan data hasil uji coba pengiriman data dengan mikrokontroler Arduino Uno. Data yang dikirim oleh Arduino adalah data temperatur yang didapatkan dengan menggunakan sensor suhu DHT11.

5.2.1.2. Pengujian *Middleware Layer*

Pada *middleware layer*, layanan yang dijalankan berupa fungsi *data fetching* dan kalkulasi data. *Core* dari layanan perangkat lunak ini terdapat pada *layer* ini.

Fungsionalitas untuk *data fetching* yang akan diuji antara lain adalah fungsi *getAllSensorData* yang berfungsi untuk mengambil seluruh data sensor berdasarkan *sensor_id*, dan *getRealTimeSensorData* yang berfungsi untuk mengambil data sensor secara *real time* berdasarkan *sensor_id*.

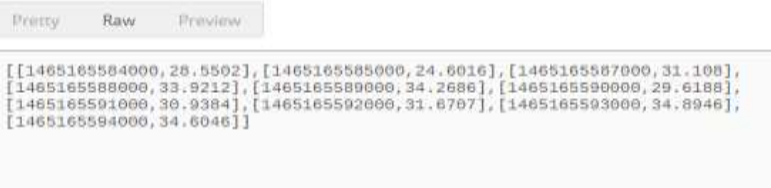
Tabel 5.3 Prosedur uji coba *data fetching* dengan fungsi *getAllSensorData*

ID	**UJ-03**
Nama	Uji Coba *data fetching* dengan fungsi *getAllSensorData*.
Tujuan Uji Coba	Menguji fungsionalitas perangkat lunak untuk mengambil data dari *database* berdasarkan *sensor_id*
Kondisi Awal	Perangkat lunak dijalankan
Skenario	Uji coba dengan *Postman* dengan mengakses secara langsung fungsi *getAllSensorData*.
Masukan	Parameter *sensor_id*
Keluaran	Keluaran pada uji coba dengan Postman berupa data sensor terkait dalam format JSON
Hasil Uji Coba	Fungsi *getAllSensorData* berhasil mengambil dan mengembalikan data sensor terkait.

Seperti yang terdapat pada Tabel 5.3, skenario uji coba dilakukan dengan kakas bantu Postman untuk melihat keluaran data mentah dalam format JSON.

Gambar 5.9 Perintah uji coba yang dikirim

Gambar 5.9 menunjukkan perintah yang dikirim ke *server* untuk menguji fungsi *getAllSensorData*. Angka "19" pada akhir perintah yang dikirim merupakan *id* dari sensor yang telah terdaftar pada sistem.



Gambar 5.10 Hasil keluaran berupa data JSON

Gambar 5.10 menunjukkan hasil keluaran dari perintah yang dikirim sesuai dengan Gambar 5.9. Keluaran dari fungsi *getAllSensorData* berupa data waktu *timestamp* data masuk dalam *unix time* dan *value* dari bacaan sensor yang dibungkus dalam format JSON.

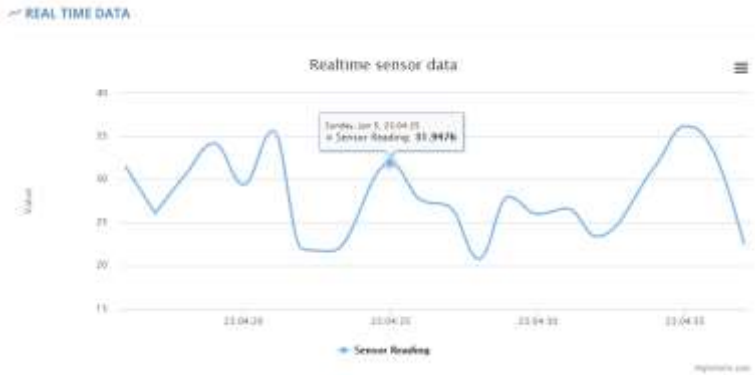
Fungsionalitas *middleware layer* berikutnya adalah *getRealTimeSensorData* yang berfungsi untuk mengambil data sensor dari *database* secara *real time*, artinya keluaran data mentah akan berupa satu data pada setiap *request*.

Tabel 5.4 Prosedur uji coba *data fethcing* secara *real time* dengan fungsi *getRealTimeSensorData*

ID	**UJ-04**
Nama	Uji Coba *data fetching* dengan fungsi *getRealTimeSensorData*
Tujuan Uji Coba	Menguji fungsionalitas perangkat lunak untuk mengambil data dari *database* berdasarkan *sensor_id* scara *real time*.
Kondisi Awal	Perangkat lunak dijalankan
Skenario	Perangkat lunak dijalankan, kemudian : Uji coba menggunakan *web browser* Google Chrome dengan cara mengakses halaman "View Sensor".
Masukan	Parameter *sensor_id*
Keluaran	Keluaran pada uji coba dengan *web browser* berupa grafik *real time* yang merepresentasikan tiap data baru yang masuk ke *database*.
Hasil Uji Coba	Fungsi *getRealTimeSensorData()* berhasil mengambil dan mengembalikan data sensor terkait.

Sesuai dengan Tabel 5.4, pengujian akan dilakukan dengan menggunakan peramban *web* Google Chrome, karena fungsi ini khusus dibuat agar *Highcharts* dapat menampilkan grafik real time pada halaman *View Sensor*. Agar bisa me-*render*

data *real time*, *Highcharts* memerulukan sepasang nilai tiap satuan waktu. Data tersebut didapat dengan melakukan *polling*, memanggil fungsi *getRealTimeSensorData* secara berkala.



Gambar 5.11 Grafik yang di *render Highcharts* pada halaman *View Sensor*

Pada Gambar 5.11 dapat dilihat *Highcharts* berhasil me-*render* grafik *real time* yang artinya fugnsi *getRealTimeSensorData* bekerja dengan semesti nya dan mengembalikan sepasang nilai yang berupa waktu dalam *unix time*, dan nilai bacaan sensor yang di bungkus dalam format JSON.

Fungsionalitas *middleware layer* berikutnya adalah fungsi kalkulasi. Fungsi kalkulasi merupakan sebuah fitur pengolahan data sensor yang tersimpan di dalam *database*. Fitur kalkulasi ini juga berupa fungsi *rules* sensor, yaitu nilai *threshold* dari sebuah sensor. Jika nilai tersebut melewati batas *threshold*, maka akan di tampilkan warning pada halaman "*View Sensor"*. Kolaborasi termasuk pada fungsi kalkulasi pada perangkat lunak ini. Kolaborasi memungkinkan pengguna untuk membuat abstraksi data dari beberapa sensor (dalam kasus ini dibatasi dua hingga tiga sensor), dengan operator pembanding (*AND* atau *OR*) dan operator

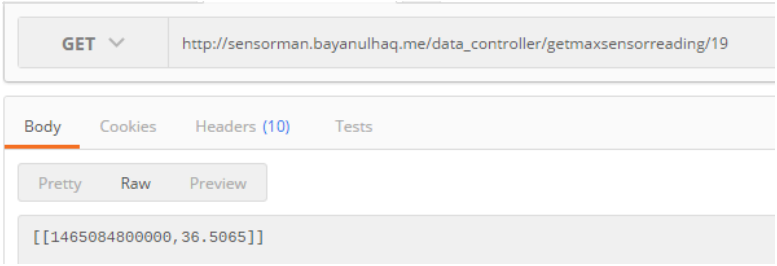
matematika (penjumlahan, pengurangan, perkalian, dan pembagian)

Pengolahan data pada perangkat lunak ini berupa fungsi maksimum, minimum, dan rerata dari data data sensor yang telah dikumpulkan di *database*. Data hasil olahan tersebut kemudian disajikan dalam bentuk grafik statis pada antarmuka *web*, dan data dalam bentuk JSON. Skenario uji coba UJ-05 di bawah akan membahas pengujian ketiga fungsi *getMaxSensorReading*, *getMinSensorReading*, dan *getAverageSensorReading*.

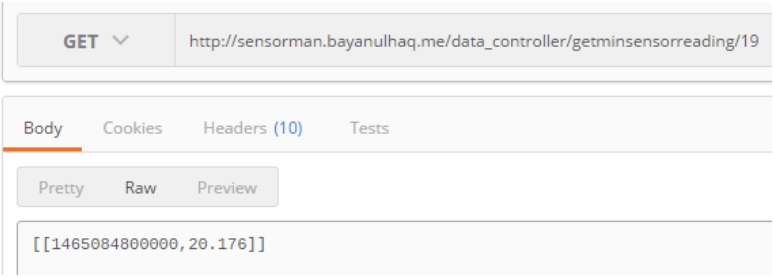
Tabel 5.5 Prosedur uji coba pengolahan data sensor menjadi *Max, Min* dan *Average*

ID	**UJ-05**
Nama	Uji Coba Pengolahan Data dengan Fungsi *getMaxSensorReading*, *getMinSensorReading*, dan *getAverageSensorReading*.
Tujuan Uji Coba	Menguji fungsionalitas pengolahan data sensor yang tersimpan di *database*.
Kondisi Awal	Perangkat lunak dijalankan
Skenario	Perangkat lunak dijalankan, kemudian masing masing dari tiga fungsi diatas akan diuji dengan: Uji coba dengan *Postman* dengan mengakses secara langsung fungsi *getRealTimeSensorData*.
Masukan	Parameter *sensor_id*
Keluaran	Keluaran pada uji coba dengan Postman berupa data sensor terkait dalam format JSON
Hasil Uji Coba	Ke tiga fungsi berhasil mengolah data sesuai dengan fungsinya.

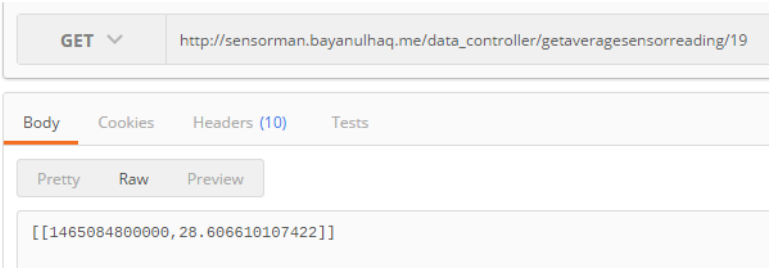
Dari Tabel 5.5, masing-masing dari tiga fungsi di atas diuji coba dengan kakas bantu Postman. Hasil uji coba ketiga fungsi diatas dapat dilihat pada Gambar 5.12, Gambar 5.13, dan Gambar 5.14.



Gambar 5.12 Hasil keluaran fungsi *getMaxSensorReading*



Gambar 5.13 Hasil keluaran fungsi *getMinSensorReading*



Gambar 5.14 Hasil keluaran fungsi *getAverageSensorReading*

Dari ketiga gambar di atas dapat dilihat bahwa fungsi unutk pengolahan data pada perangkat lunak ini bekerja sesuai dengan fungsinya. Masing-masing fungsi di atas mengembalikan

nilai maksimum, minimum, dan rata-rata dari sebuah data sensor dalam satuan waktu per hari.

Fungsi yang dijalankan *middleware layer* selanjutnya adalah fungsi *rules* pada sensor. Fungsi ini berupa *threshold* nilai dari sensor tersebut, baik nilai batas atas (*high*), batas bawah (*low*), maupun nilai spesifik yang ditentukan oleh pengguna (*equal*). Implementasi dari *rules* ini adalah fungsi *checkSensorRules*. Fungsi ini dijalankan secara terus menerus dengan *looping* karena berfungsi melakukan pengecekan secara *real time* pada data sensor yang ada. Tabel uji coba UJ-06 akan membahas secara rinci pengujian fungsi ini.

Tabel 5.6 Prosedur uji coba *rules* pada sensor

ID	**UJ-06**
Nama	Uji Coba *Rules* Sensor.
Tujuan Uji Coba	Menguji fungsionalitas *rules* dan *alert* pada perangkat lunak..
Kondisi Awal	Perangkat lunak dijalankan, *rule* high dengan *value* 50 telah ditambahkan untuk sensor terkait.
Skenario	Perangkat lunak dijalankan, kemudian sensor akan mengirimkan nilai yang berada di luar batas threshold dari *rule* sensor tersebut.
Masukan	Parameter *sensor_key* dan *sensor_reading.*
Keluaran	Berupa peringatan yang ditampilkan pada halaman *web* "*View Sensor*".
Hasil Uji Coba	Berhasil mendeteksi nilai yang berada diluar batas *rules*.

Berdasarkan Tabel 5.6 diatas, pengujian dilakukan dengan mengirim nilai bacaan sensor diatas 50, karena *rule* sensor yang berupa *high threshold value* di set pada nilai 50. Ketika sensor Python mengirim data dengan *value* diatas 50, maka *rule high* tadi akan ter-*trigger*, dan ditampilkan pada halaman "*View Sensor*".

Gambar 5.15 Tampilan halaman antarmuka *View Sensor* yang menampilkan *alert* "*High rule triggered!"*

Pada Gambar 5.15 dapat dilihat bahwa *rule* yang di daftarkan pada sensor ter – *trigger* karena ada nilai sensor yang melewati *threshold* yang telah di daftarkan ke tabel *rules* pada *database*.

Fungsi kolaborasi juga dijalankan pada *middleware layer*. Fungsi ini memungkinkan pengguna untuk melakukan abstraksi data dari dua hingga tiga sensor. Abstraksi data berupa operasi matematika dasar antar nilai sensor tersebut. Tabel UJ-07 akan membahas rincian prosedur uji coba fitur kolaborasi ini.

Tabel 5.7 Prosedur uji coba kolaborasi sensor

ID	**UJ-07**
Nama	Uji Coba Kolaborasi Sensor
Tujuan Uji Coba	Menguji fungsionalitas kolaborasi sensor pada pernagkat lunak.
Kondisi Awal	Perangkat lunak dijalankan, sudah terdapat sensor kolaborasi pada *database*.

Skenario	Perangkat lunak dijalankan, kemudian sensor akan mengirimkan data ke *server*.
Masukan	Parameter *sensor_key* dan *sensor_reading.*
Keluaran	Data hasil kalkulasi antar sensor dan respon JSON dari *server*.
Hasil Uji Coba	Berhasil melakukan abstraksi data berupa ekspresi matematika dasar pada data sensor.

Berdasarkan Tabel 5.7 diatas, pengujian dilakukan dengan mengirimkan data dari sensor yang telah terdaftar pada sebuah *instance* kolaborasi. Ketika data sensor tersebut masuk, *server* melakukan pengecekan terhadap *sensor_key* yang masuk, apakah terdaftar pada *server* dan terdaftar pada tabel *sensor_collab*. Jika ya, maka data akan di olah sesuai dengan aturan pada tabel *sensor_collab* terkait.

sensor_collab_id	sensor_collab_name	sensor_x_id	sensor_y_id	comp_operator	operator
1	Heat Index Kamar Kost	17	18	OR	x * (y/100)

Gambar 5.16 Tabel *sensor_collab*

Gambar 5.16 menunjukkan sebuah kolom data pada tabel *sensor_collab*. Data tersebut berupa sebuah kolaborasi sensor antara sensor yang memiliki *id* 17 dan 18, dengan komparator ‘OR’ dan ekspresi matematika untuk perhitungan “x * (y/100)”, artinya setiap data dari sensor dengan *id* 17 atau 18 akan di kalkulasi sesuai dengan ekspresi matematika yang di daftarkan.

sensor_collab_data_id	sensor_x_value	sensor_y_value	sensor_collab_id	sensor_reading	timestamp
23	42.1201	48.6729	1	20.5853	2016-06-05 23:42:56
22	42.1201	45.7529	1	19.2712	2016-06-05 23:42:55
21	42.1201	45.5728	1	19.1953	2016-06-05 23:42:54
20	42.1201	37.4649	1	15.7893	2016-06-05 23:42:52
19	42.1201	41.5709	1	17.5097	2016-06-05 23:42:51
18	42.1201	45.6705	1	19.2365	2016-06-05 23:42:50
17	42.1201	48.9781	1	20.6299	2016-06-05 23:42:49
16	42.1201	38.0000	1	16.0184	2016-06-05 23:42:48
15	42.1201	42.5741	1	17.9322	2016-06-05 23:42:47

Gambar 5.17 Tabel *sensor_collab_data*

Pada Gambar 5.17 dapat dilihat bahwa tabel *sensor_collab_data* berisi *record record* hasil kalkulasi kolaborasi sensor. Kolom *sensor_reading* merupakan nilai hasil dari kalkulasi kolaborasi antar sensor x dan sensor y terdaftar.

5.2.1.3. Pengujian *Client-centric Layer*

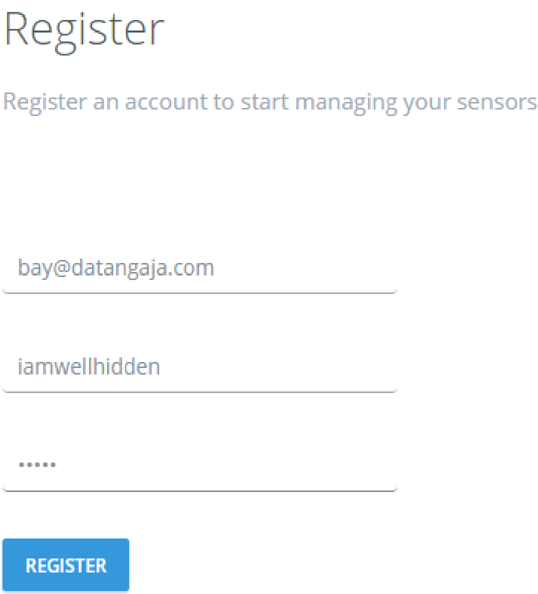
Client-centric layer, seperti namanya, merupakan *layer* yang berhubungan langsung dengan pengguna. *Layer* ini menjalankan fungsi manajemen *user*, pendaftaran *user*, dan menampilkan data-data yang telah di kumpulkan *Sensor-centric layer*, diolah oleh *Middleware layer*.

Implementasi utama dari fungsi *client-centric layer* ini adalah fungsi *viewSensor*, dan *analyzeSensor* yang keduanya berfungsi menampilkan data kepada pengguna melalui antarmuka *web* yang telah di implementasikan. Fungsi yang akan di uji adalah pendaftaran pengguna, karena perangkat lunak yang di bangun harus memenuhi kriteria, yaitu bersifat *multi-tenant*.

Tabel 5.8 Prosedur uji coba pendaftaran pengguna baru

ID	**UJ-08**
Nama	Uji Coba fungsionalitas pendaftaran
Tujuan Uji Coba	Menguji fungsionalitas *multi-tenancy* dari perangkat lunak.
Kondisi Awal	Perangkat lunak dijalankan
Skenario	Perangkat lunak dijalankan, kemudian di akses menggunakan *web browser* Google Chrome.
Masukan	Data pengguna baru
Keluaran	Pengguna baru terdaftar pada sistem
Hasil Uji Coba	Berhasil mendaftarkan pengguna baru.

Sesuai dengan Tabel 5.8, percobaan dilakukan dengan mendaftarkan pengguna baru.



Gambar 5.18 Form pendaftaran pengguna baru

Gambar 5.18 menunjukkan form pendaftaran pengguna baru yang terisi dengan *email* ‘bay@datangaja.com’ dan *username* ‘iamwellhidden’. Jika dilakukan klik pada tombol ‘REGISTER’, sistem akan mendaftarkan pengguna tersebut ke *database*.

id	ip_address	username	password	email	created_on	last_login
1	127.0.0.1	administrator	$2a$07$SeBkmtpZror9uyffVopmu61qg3ms8QxrYV6FG.kQOS...	admin@admin.com	1268889823	126888982
2	::1	hadriansrg	$2y$08$zexQjmkNuwkmn4lvRn7Deqtq3273/sbXzWbeazP5LH...	bay@datangaja.com	1460047354	146516996
3	::1	hadrianbs	$2y$08$kq7JAWgJHs5af3TsJFxO3f5hK/GXLmbLgVs8Jzhq...	hadrianbayanulhaq@gmail.com	1464633997	146516784
4	172.31.23.175	iamwellhidden	$2y$08$Gbz7zeAhXH7aWqpYumV5p.5S3xDe3N7relrhGJ/Su0...	bay@datangaja.com	1465171778	NUL

Gambar 5.19 Tabel *users*

Pada Gambar 5.19, terlihat bahwa *username* ‘iamwellhidden’ telah terdaftar pada sistem.

5.2.2. Uji Coba Performa

Uji coba performa merupakan pengujian yang bertujuan untuk menguji tingkat kehandalan suatu perangkat lunak, baik dalam segi kecepatan (*speed*) maupun ketahanan (*robustness*). Uji coba ini juga mengukur penggunaan *bandwidth* saat sensor melakukan pengiriman data ke *server*. Adapun penjelasan setiap bagian dari uji coba performa ini akan dijelaskan pada subbab berikut ini.

Uji coba performa ini akan dilakukan dengan kakas bantu *siege*. *Siege* merupakan kakas bantu untuk melakukan *load test* pada sebuah aplikasi web. Skenario yang dilakukan adalah mengirimkan *request* data, pengiriman data biasa (server melakukan *retrieve and store*) dan pengiriman data kolaborasi sensor (server menerima data, kemudian memproses sensor dan datanya, lalu disimpan pada basis data). Skenario tadi akan di lakukan masing masing dengan 100 hingga 1500 *concurrent users* selama 60 detik.

5.2.2.1. Pengujian *Request* Data

Pada uji coba ini, keseluruhan sistem baik perangkat lunak, perangkat keras, dan arsitektur jaringan diuji coba kecepatan dan ketahanannya terhadap *data request traffic* dari pengguna. Pengambilan data dari *database* akan dilakukan dengan fungsi *getAllSensorData* yang mengembalikan semua data sensor yang tersimpan berdasarkan *sensor_id* nya. Tabel 5.9 merupakan hasil uji coba *data request* ini.

Tabel 5.9 Hasil uji coba *data request*

Concurrent Users	*Average Response Time (sec)*	*Throughput (Mb/s)*	*Server 1 CPU Utilization (%)*	*Server 2 CPU Utilization (%)*
100	0,18	0.34	28	30
200	0,76	0.33	34	33
300	1,40	0.34	35	34
500	2,58	0.46	38	40
800	3,73	0.33	41	40
1300	4,78	0.33	42	47
1500	5,37	0.32	49	46

Pada Tabel 5.9 dapat dilihat hasil pengujian *data request* hingga 1500 *concurrent users*. Kolomn *Concurrent Users* merupakan jumlah pengguna yang mengakses *server* secara *concurrent*. *Average Response Time* adalah waktu respon rata-rata server dihitung dalam satuan detik. CPU *Utilization* merupakan presentasi penggunaan CPU pada masing masing *server* ketika uji coba sedang berlangsung.

5.2.2.2. Pengujian Pengiriman Data Sensor

Pada skenario uji coba ini, klien (dalam kasus ini *siege*) mengirimkan HTTP POST *request* yang berisi data dalam format JSON ke URL API aplikasi. Data tersebut kemudian akan diterima, diperiksa, dan disimpan kedalam basis data. Pada tabel ini terlihat pengujian kecepatan dan ketahanan *server* untuk aplikasi ini berdasarkan *concurrent sensors* yang melakukan transaksi pengiriman data. Kolom *concurrent sensors* merupakan jumlah sensor yang mengirimkan data per detik secara bersamaan, selama 60 detik. Rata-rata waktu respon adalah waktu respon

diterima oleh klien. Tabel 5.10 merupakan hasil uji coba pengiriman data sensor.

Tabel 5.10 Hasil uji coba pengiriman data sensor

Concurrent Sensors	*Average Response Time (sec)*	*Throughput (Mb/s)*	*Server 1 CPU Utilization (%)*	*Server 2 CPU Utilization (%)*
100	0,32	0.03	22	22
200	1,10	0.02	23	22
300	1,83	0.03	22	25
500	3,39	0.03	24	27
800	5,19	0.03	32	34
1300	6,40	0.03	41	37
1500	7,39	0.03	44	47

5.2.2.3. Pengujian Pengiriman Data Kolaborasi Sensor

Pada skenario uji coba ini, klien (dalam kasus ini *siege*) mengirimkan HTTP POST *request* yang berisi data dalam format JSON ke URL API aplikasi. Data tersebut kemudian akan diterima, diperiksa, kemudian diproses sesuai dengan ekspresi matematik yang tersimpan pada tabel kolaborasi sensor sebelum disimpan pada basis data. Pada tabel ini terlihat pengujian kecepatan dan ketahanan *server* untuk aplikasi ini berdasarkan *concurrent sensors* yang melakukan transaksi pengiriman data. Kolom *concurrent sensors* merupakan jumah sensor yang mengirimkan data per detik secara bersamaan, selama 60 detik. Rata-rata waktu respon adalah waktu respon diterima oleh klien. Tabel 5.11 merupakan hasil uji coba pengiriman data sensor.

Tabel 5.11 Hasil uji coba pengiriman data kolaborasi sensor

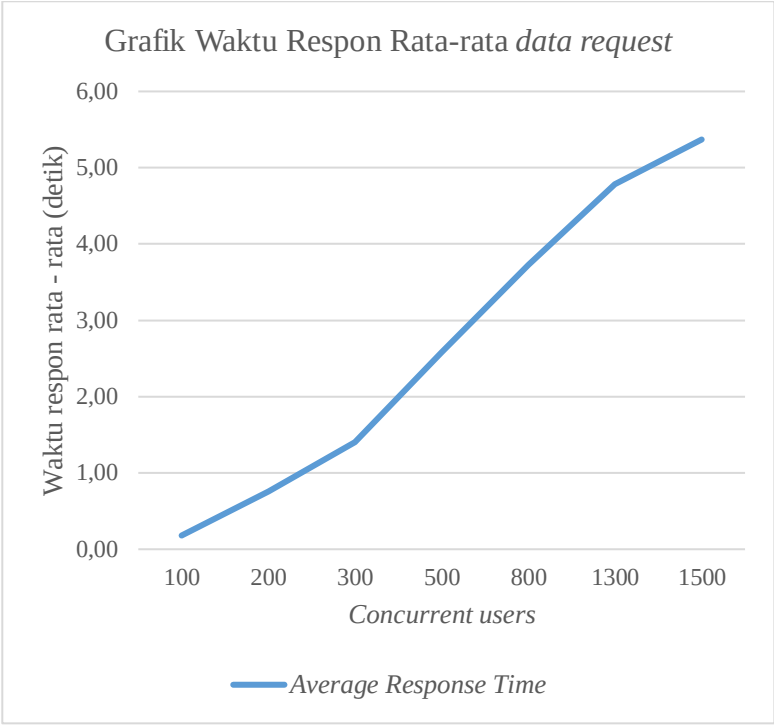
Concurrent Sensors	*Average Response Time (sec)*	*Throughput (Mb/s)*	*Server 1 CPU Utilization (%)*	*Server 2 CPU Utilization (%)*
100	0,42	0.03	27	22
200	1,92	0.03	28	26
300	2,32	0.02	28	27
500	3,39	0.02	31	29
800	5,97	0.03	33	36
1300	8,99	0.03	44	39
1500	12,73	0.03	49	44

5.3. Evaluasi Hasil Uji Coba

Evaluasi hasil uji coba performa dibagi menjadi dua bagian, yaitu evaluasi uji coba performa kecepatan, dan evaluasi uji coba performa ketahanan.

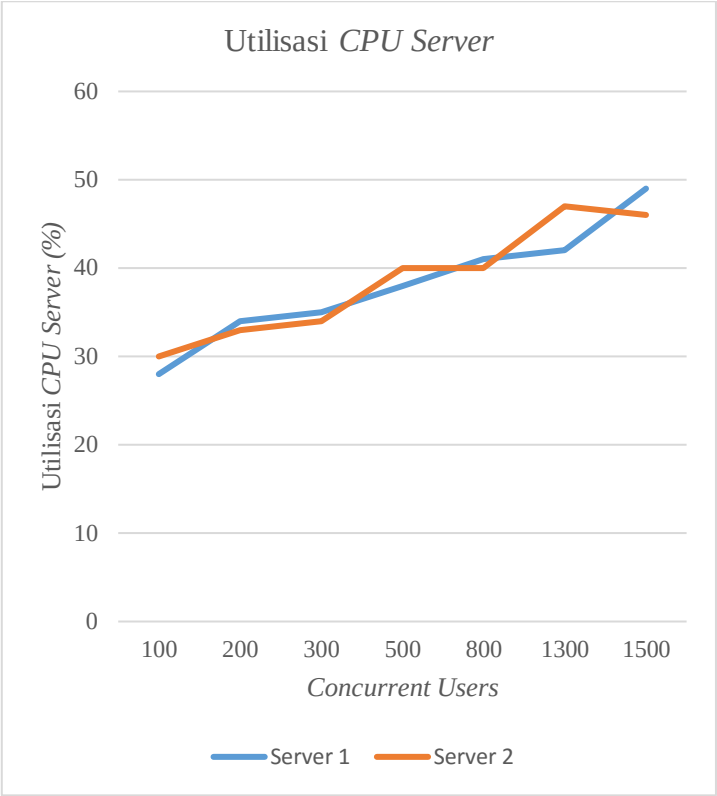
5.3.1. Evaluasi Hasil Uji Coba *Request Data*

Sesuai dengan hasil uji coba waktu *data request* pada Tabel 5.9, maka di dapat waktu respon rata-rata, transaksi terpanjang dengan satuan detik, dan tingkat utilisasi CPU. Grafik perbandingan waktu respon rata-rata dengan transaksi terpanjang serta grafik utilisasi CPU bisa dilihat pada Gambar 5.20 dan Gambar 5.21.



Gambar 5.20 Grafik Waktu Respon rata-rata *data request*.

Gambar 5.20 menunjukkan perbandingan waktu respon rata-rata untuk uji coba *data request*. Sumbu X pada grafik diatas merepresentasikan jumah *concurrent users*, sendangan sumbu Y pada grafik di atas merepresentasikan waktu dalam satuan detik. Dari grafik 5.17, dapat dilihat waktu respon bertambah lama seiring bertambahnya *concurrent user*. Hal ini dikarenakan *server* membutuhkan waktu lebih lama untuk menyelesaikan *request* yang masuk sebelumnya.

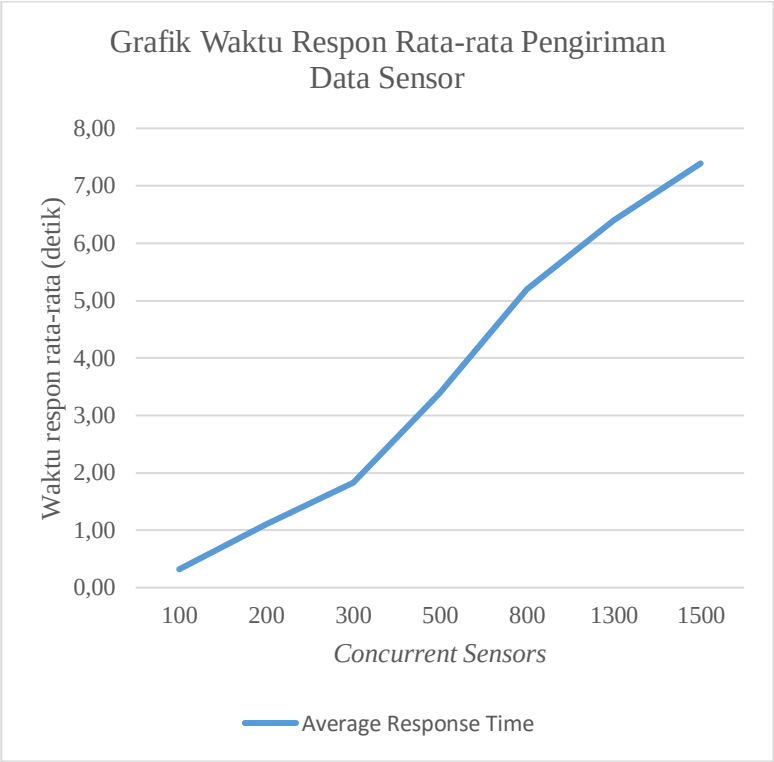


Gambar 5.21 Grafik Tingkat Utilisasi CPU pada Server pada uji coba *data request*.

Gambar 5.21 menunjukkan tingkat utilisasi CPU pada kedua server untuk uji coba *data request*. Sumbu X pada grafik diatas merepresentasikan jumlah *concurrent users*, sendangkan sumbu Y pada grafik diatas merepresentasikan persentase CPU utilization.

5.3.2. Evaluasi Hasil Uji Coba Pengiriman Data Sensor

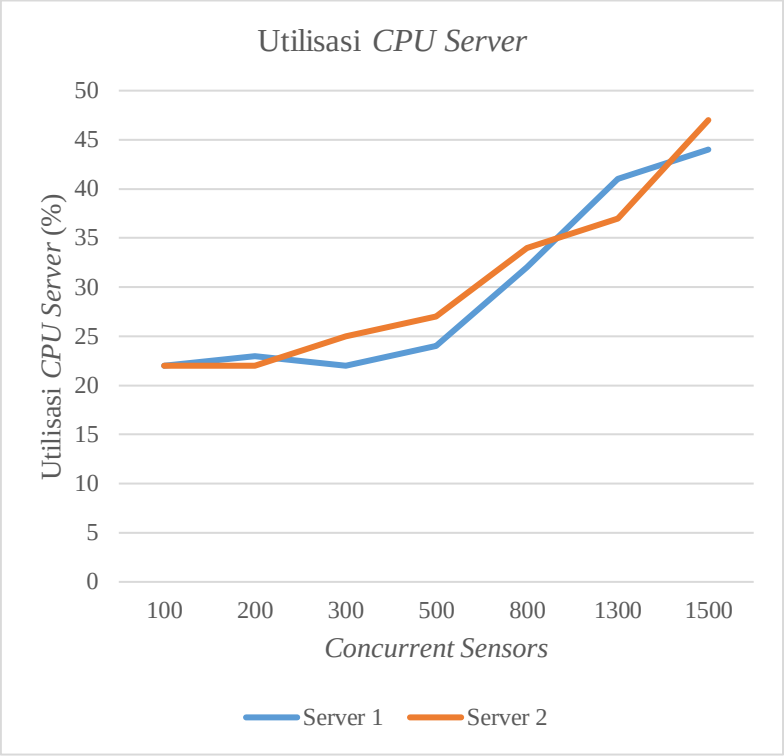
Sesuai dengan hasil uji coba waktu pengiriman data sensor pada Tabel 5.10, maka didapat waktu respon rata-rata, transaksi terpanjang dengan satuan detik, dan tingkat utilisasi CPU. Grafik waktu respon rata-rata serta grafik utilisasi CPU bisa dilihat pada Gambar 5.22 dan Gambar 5.23.



Gambar 5.22 Grafik Waktu Respon rata-rata pengiriman data sensor.

Gambar 5.22 menunjukkan perbandingan waktu respon rata-rata untuk uji coba pengiriman data sensor. Sumbu X pada grafik di atas merepresentasikan jumah *concurrent users*,

sendangan sumbu Y pada grafik diatas merepresentasikan waktu dalam satuan detik.

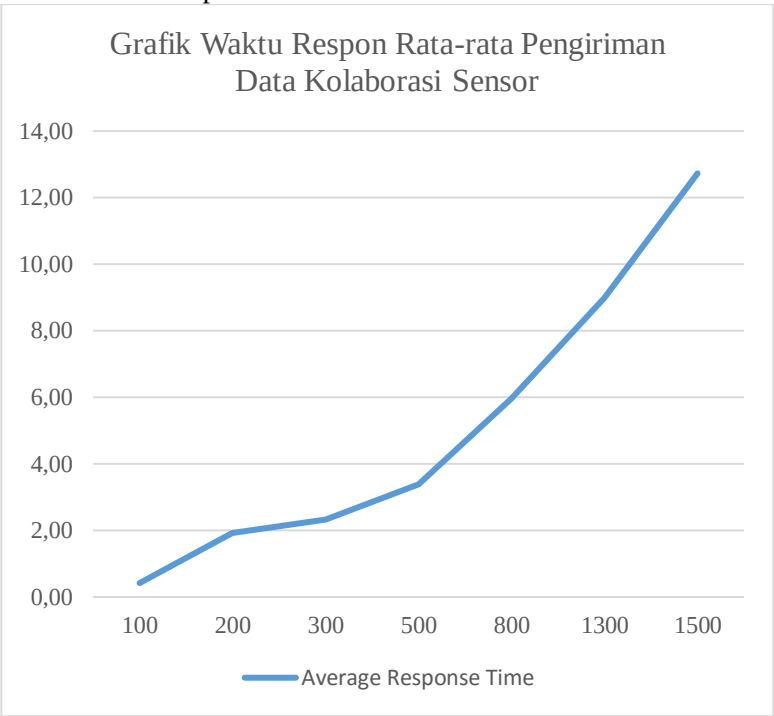


Gambar 5.23 Grafik Tingkat Utilisasi CPU pada Server

Gambar 5.23 menunjukkan tingkat utilisasi CPU pada kedua server untuk uji coba pengiriman data sensor. Sumbu X pada grafik di atas merepresentasikan jumah *concurrent users*, sendangan sumbu Y pada grafik diatas merepresentasikan persentase CPU utilization.

5.3.3. Evaluasi Hasil Uji Coba Pengiriman Data Kolaborasi Sensor

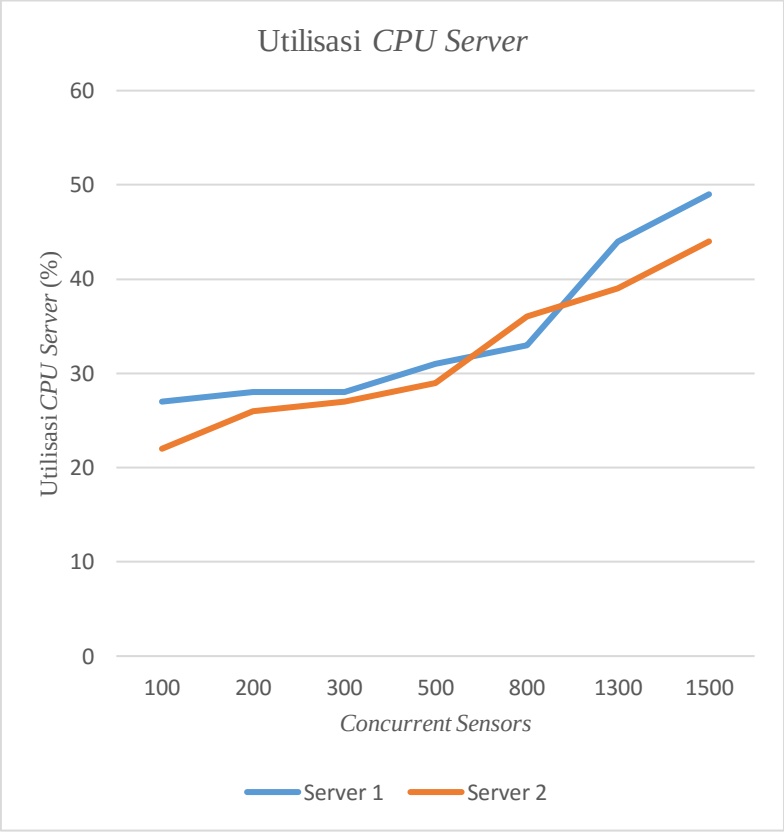
Sesuai dengan hasil uji coba waktu pengiriman data kolaborasi sensor pada Tabel 5.11, maka didapat waktu respon rata-rata, transaksi terpanjang dengan satuan detik, dan tingkat utilisasi CPU. Grafik waktu respon rata-rata serta grafik utilisasi CPU bisa dilihat pada Gambar 5.24 dan Gambar 5.25



Gambar 5.24 Grafik Waktu Respon rata-rata pengiriman data kolaborasi sensor.

Gambar 5.24 menunjukkan waktu respon rata-rata untuk uji coba pengiriman data kolaborasi sensor. Sumbu X pada grafik diatas merepresentasikan jumah *concurrent users*, sendangan

sumbu Y pada grafik diatas merepresentasikan waktu dalam satuan detik.



Gambar 5.25 Grafik Tingkat Utilisasi CPU pada Server

Gambar 5.25 menunjukkan tingkat utilisasi CPU pada kedua server untuk uji coba pengiriman data kolaborasi sensor. Sumbu X pada grafik diatas merepresentasikan jumah *concurrent users*, sendangan sumbu Y pada grafik diatas merepresentasikan persentase CPU utilization.

LAMPIRAN

1. Fungsi *getRealTimeSensorData* merupakan fungsi untuk mengambil data sensor terakhir dari database. Fungsi ini bertujuan agar data yang diambil bisa di*render* oleh pustaka *Highcharts* untuk menampilkan grafik real time.

```
public function getRealTimeSensorData($sensor_id){
        $this->load->model('Data_model');
        $data = $this->Data_model-
>getSensorReading($sensor_id);
        foreach($data as $row)
        {
                $ret = array(strtotime(
                $row->timestamp)*1000,
                (float)$row->datareading);
        }
        $this->output
                ->set_content_type('application/json')
                ->set_output(json_encode($ret));
}
```

2. Fungsi *getAllSensorData* berfungsi untuk mengambil semua sensor data dalam bentuk JSON

```
public function getAllSensorData($sensor_id){
        $this->load->model('Data_model');
        $querydata = $this->
                Data_model->getSensorReading($sensor_id);
        foreach($querydata as $row){
                $datetime = strtotime(
                        $row->timestamp)*1000;
                        $datareading = (float)$row-
                        >datareading;
                        $data[] = [$datetime,
                        $datareading];
                }

                $this->output
                ->set_content_type('application/json')
                ->set_output(json_encode($data,
JSON_NUMERIC_CHECK));
}
```

3. Fungsi *getMaxSensorReading* berfungsi untuk mengambil data rata-rata per hari dari sebuah sensor dalam bentuk JSON

```
public function getAverageSensorReading($sensor_id){
        $this->load->model('Data_model');
        $querydata = $this->Data_model-
>getAverageSensorReading($sensor_id);
        foreach($querydata as $row){
                $datetime = strtotime($row-
>sensordate)*1000;
                $datareading = (float)$row->sensordata;
                $data[] = [$datetime, $datareading];
        }
        echo json_encode($data, JSON_NUMERIC_CHECK);
}
```

4. Fungsi *getMinSensorReading* berfungsi untuk mengambil data minimum per hari dari sebuah sensor dalam bentuk JSON.

```
public function getMinSensorReading($sensor_id){
        $this->load->model('Data_model');
        $querydata = $this->Data_model-
>getMinSensorReading($sensor_id);
        foreach($querydata as $row){
                $datetime = strtotime($row-
>sensordate)*1000;
                $datareading = (float)$row->sensordata;
                        $data[] = [$datetime,
$datareading];
        }

        echo json_encode($data, JSON_NUMERIC_CHECK);
}
```

5. Fungsi *getMaxSensorReading* berfungsi untuk mengambil dara maksimum per hari dari sebuah sensor dan mengembalikannya dalam bentuk JSON

```
public function getMaxSensorReading($sensor_id){
        $this->load->model('Data_model');
        $querydata = $this->Data_model-
        >getMaxSensorReading($sensor_id);
        foreach($querydata as $row){
                $datetime = strtotime($row-
>sensordate)*1000;
                $datareading = (float)$row->sensordata;
                        $data[] = [$datetime,
$datareading];
        }

        echo json_encode($data, JSON_NUMERIC_CHECK);
}
```

[Halaman ini sengaja dikosongkan]

BAB VI
KESIMPULAN DAN SARAN

Bab ini berisi penjelasan mengenai kesimpulan yang dapat diambil pada pengerjaan Tugas Akhir ini, serta saran-saran tentang pengembangan yang dapat dilakukan pada Tugas Akhir ini di masa yang akan datang.

6.1 Kesimpulan

Dari hasil pengamatan dan percobaan selama perancangan, implementasi, dan uji coba aplikasi, maka dapat diambil kesimpulan sebagai berikut:

1. Perangkat lunak dapat menjalankan fungsi utama pada masing-masing *layer*.
2. Perangkat lunak dapat melakukan penerimaan dan pengolahan data dari sensor fisik. Pengolahan tersebut berupa fungsi fungsi statistik dasar seperti *Max*, *Min,* dan *Average*.
3. Perangkat lunak dapat membuat sebuah abstraksi data dari dua atau tiga sensor dengan operasi matematik dasar (penjumlahan, pengurangan, perkalian, dan pembagian).
4. Perangkat lunak dapat menyajikan data data hasil olahan ke dalam bentuk visual yang berupa grafik, dengan kata lain perangkat lunak bisa melakukan visualisasi data.
5. Perangkat lunak dapat menampilkan data sensor secara *real time* dalam bentuk grafik *real time*.
6. Kecepatan penerimaan dan pengolahan data sensor maupun data kolaborasi sensor dipengaruhi, walaupun tidak signifikan, oleh banyaknya *traffic* sensor yang aktif.

6.2 Saran

Saran yang diberikan untuk pengembangan aplikasi ini adalah:

1. Untuk melakukan optimasi pada fungsi penerimaan data agar bisa lebih optimal dalam rangkaian proses penerimaan data. Proses penerimaan data yang telah diimplementasi saat ini bersifat *synchronous*, dimana masing masing data yang masuk akan di cek apakah terdaftar pada *database record* kolaborasi sensor dan *rules* nya.
2. Untuk menambahkan fitur *alert* secara *real time*. Fungsionalitas *alert* yang telah diimplementasi tidak bersifat *real time* dan hanya bisa dilihat ketika pengguna membuka halaman *View Sensor*.

DAFTAR PUSTAKA

[1] Sensor-Cloud, [Online]. Avaliable: http://sensorcloud.com/system-overview [Diakses 2015]

[2] http://www.ntu.edu.sg/intellisys

[3] K. T. Lan, “What’s Next? Sensor+Cloud?” in *Proceeding of the 7th International Workshop on Data Management for Sensor Networks*, pp. 978-971, ACM Digital Library, 2010.

[4] Arduino Uno, [Online]. Avaliable: https://www.arduino.cc/en/main/arduinoBoardUno [Diakses 2015]

[5] F. Michael, "Milestones in the history of thematic cartography, statistical graphics, and data visualization", 2008

[6] “IT Asssets Management”, [Online]. Avaliable: http://www.itassetmanagement.net/ [Diakses 2015]

[7] "JSON," JSON. [Online]. Avaliable: http://www.json.org/ [Diakses 2016]

[8] Tatroe, K., MacIntyre, P. dan Lerdorf, R., 2013. Programming PHP, Third Edition. USA: O’Reilly Media, Inc.

[9] Vaswani, Vikram., 2007. Programming PHP Solutions, First Edition. USA: McGraw Hill companies, Inc.

[10] "What is MySQL?". *MySQL 5.1 Reference Manual*. Oracle.

[11] Urlocker, M. Zack (13 December 2005). "Google Runs MySQL". The Open Force. M. Zack Urlocker

[12] Sobel, Jason (21 December 2007). "Keeping Up". *The Facebook Blog*. Facebook

[13] H. Kreger, *Web Services* Conceptual Architecture, New York: International Business Machines Corporation, 2001

14 Mell, Peter, dan Grance, Timothy, "The NIST Definition of Cloud Computing", National Institute of Standards and Technology, 2011

BIODATA PENULIS

Hadrian Bayanulhaq Siregar, lahir di kota Tebing Tinggi pada tanggal 10 September 1994. Penulis adalah anak pertama dari tiga bersaudara dan dibersarkan di kota kelahiran Tebing Tinggi, Sumatera Utara. Penulis menempuh pendidikan formal di SD. Swasta F. Tandean Tebing Tinggi (2000-2006), SMP Negeri 1 Tebing Tinggi (2006-2009), SMA Negeri 1 Tebing Tinggi. Pada tahun 2012 penulis memulai pendidikan strata satu di jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya, Jawa Timur. Di jurusan Teknik Informatika, penulis mengambil bidang minat Komputasi Berbasis Jaringan dan memiliki ketertarikan di bidang keamanan jaringan, sistem operasi, dan jaringan secara umum. Selama menempuh masa perkuliahan, penulis juga aktif sebagai staf departemen Pengembangan Profesi di Himpunan Mahasiswa Teknik Computer (HMTC). Penulis dapat dihubungi melalui alamat email hadrianbayanulhaq@gmail.com