MONITORING JUMLAH PENGGUNA *ACCESS POINT* BERBASIS *SIMPLE OBJECT ACCESS PROTOCOL* (SOAP) *WEB SERVICE*

(Studi Kasus: Universitas Brawijaya)

SKRIPSI

Konsentrasi : Jaringan Komputer

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun oleh :

DEDI TIARNO

NIM. 105090607111037

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2015

LEMBAR PERSETUJUAN

MONITORING JUMLAH PENGGUNA *ACCESS POINT* BERBASIS *SIMPLE OBJECT ACCESS PROTOCOL* (SOAP) *WEB SERVICE*

Studi Kasus: Universitas Brawijaya

SKRIPSI

Untuk memenuhi sebagian persyaratan untuk mencapai gelar Sarjana Komputer

Disusun Oleh:

DEDI TIARNO

NIM.105090607111037

Skripsi ini telah diperiksa dan disetujui oleh dosen pembimbing
Pada tanggal 19 Juni 2015

Dosen Pembimbing I,	**Dosen Pembimbing II,**
Eko Sakti P., S.Kom., M.Kom	**Kasyful Amron, ST., M.Sc**
NIK. 86080506110252	**NIP. 19750803 200312 1 003**

LEMBAR PENGESAHAN

MONITORING JUMLAH PENGGUNA *ACCESS POINT* BERBASIS *SIMPLE OBJECT ACCESS PROTOCOL* (SOAP) *WEB SERVICE*

(Studi Kasus: Universitas Brawijaya)

SKRIPSI

Untuk memenuhi sebagian persyaratan untuk mencapai gelar Sarjana Komputer

Disusun oleh :

DEDI TIARNO
NIM. 105090607111037

Skripsi ini telah diuji dan dinyatakan lulus pada tanggal 19 Juni 2015

Dosen Penguji I,	Dosen Penguji II,
Achmad Basuki, ST., MMG., Ph.D	**Aswin Suharsono, ST.,M.T.**
NIP. 197411182003121002	**NIK. 84091906110251**

Dosen Penguji III,

Aryo Pinandito, ST., M.MT
NIP. 198305192014041001

Mengetahui,
Ketua Program Studi Informatika/Ilmu Komputer

Drs. Marji, MT
NIP. 196708011992031001

LEMBAR PERNYATAAN
ORISINILITAS SKRIPSI

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

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

Malang, 19 Juni 2015
Yang menyatakan,

Dedi Tiarno
NIM. 105090607111037

KATA PENGANTAR

Syukur dan alhamdulillah penulis panjatkan ke hadirat Allah SWT yang telah melimpahkan segala Rahmat, Karunia dan Hidayah-Nya sehingga Penulis dapat menyelesaikan skripsi dengan judul: "**Monitoring Jumlah Pengguna *Access Point* Berbasis *Simple Object Access Protocol* (SOAP) *Web Service* (Studi Kasus: Universitas Brawijaya)** ".

Skripsi ini diajukan sebagai syarat ujian seminar skripsi dalam rangka untuk memperoleh gelar Sarjana Komputer di Program Teknologi Informasi dan Ilmu Komputer (PTIIK), Program Studi Ilmu Komputer, Universitas Brawijaya Malang. Atas terselesaikannya skripsi ini, Penulis mengucapkan terima kasih kepada:

1. Ibu, Bapak, dan seluruh keluarga tercinta yang selalu memberikan doa, kasih sayang dan motivasi baik moral maupun materi sehingga penulis dapat menyelesaikan pendidikan dengan baik.
2. Bapak Eko Sakti P., S.Kom., M.Kom., selaku Dosen Pembimbing Skripsi pertama yang telah meluangkan waktu dan juga memberikan pengarahan bagi penulis.
3. Bapak Kasyful Amron, ST., M.Sc selaku Dosen Pembimbing Skripsi kedua yang telah meluangkan waktu dan juga memberikan pengarahan bagi penulis.
4. Bapak Achmad Basuki, ST.,MMG.,Ph.D selaku Dosen yang telah meluangkan waktu dan juga memberikan pengarahan bagi penulis.
5. Drs. Marji, MT., selaku Ketua Program Studi Teknik Informatika Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
6. Reza Andria Siregar, ST selaku Dosen Penasehat Akademik.
7. Bapak Ir. Sutrisno, MT., selaku Ketua Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
8. Segenap Bapak dan Ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada Penulis selama menempuh pendidikan di Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.
9. Bapak Ivan Yulfrian,S.Kom, Alfan Mariyanto, A.Md dan segenap staff dan karyawan di Program Studi Teknologi Informasi & Ilmu Komputer

Universitas Brawijaya yang telah banyak membantu Penulis dalam pelaksanaan penyusunan skripsi ini.

10. Bapak R. Arief Setyawan, S.T., M.T yang memberikan izin penelitian di PPTI, Divisi Infrastruktur Teknologi Informasi PPTI Bapak Alan Stevrie Balantimuhe, S.T, Bapak Akhmad Mukhtarom, S.T, Bapak Rahmad Tsalaatsa, S.Kom, Bapak Kharisma Kumbarakarta, S.T, Bapak Abdurohman M.,S.T, Bapak Alfa Anggawasita, S.Kom, Bapak Iwanto yang telah membantu penulis dalam pelaksanaan penyusunan skripsi ini serta segenap staf dan karyawan PPTI Universitas Brawijaya.
11. Teman-teman kontrakan Kembang Turi (Awang, Ryan, Wildan, Brian) yang selalu memberi canda tawa di sela-sela kejenuhan mengerjakan skripsi ini.
12. Teman-teman kontrakan sigura hill E-10 Dwi Saputro Nugroho yang selalu membantu dalam pengerjaan skripsi, Ryan Dimas, Yhoga Asmara, Selly Johansyah, Afrizal Adytia, Restu Wandiro dan teman-teman Program Studi Informatika/Ilmu Komputer 2010 yang selalu memberikan semangat, dukungan, dan kebersamaan selama Penulis menempuh pendidikan di Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
13. Semua pihak yang tidak dapat disebutkan satu per satu, yang telah membantu dalam penulisan skripsi ini.

Penulis menyadari bahwa skripsi ini tentunya tidak terlepas dari berbagai kekurangan dan kesalahan. Oleh karena itu, segala kritik dan saran yang bersifat membangun sangat Penulis harapkan dari berbagai pihak demi penyempurnaan penulisan skripsi ini. Akhirnya penulis berharap agar skripsi ini dapat memberikan sumbangan dan manfaat bagi semua pihak yang berkepentingan.

Malang, 19 Juni 2015

Dedi Tiarno

Penulis

ABSTRAK

Dedi Tiarno. 2015 : Monitoring Jumlah Pengguna Access Point Berbasis Simple Object Access Protocol (SOAP) Web Service (Studi Kasus: Universitas Brawijaya)

Dosen Pembimbing : Eko Sakti Pramukantoro, S.Kom, M.Kom dan Kasyful Amron, ST., M.sc

Universitas Brawijaya merupakan salah satu universitas terkemuka di Indonesia yang menyediakan fasilitas dan layanan berbasis teknologi informasi untuk pengguna yaitu dosen, karyawan dan mahasiswa. *Access point* yang digunakan adalah produk Unifi yang mempunyai kinerja tinggi dan mampu menampung seratus pengguna secara bersamaan. *Access point* Unifi tersebar di beberapa titik lokasi yang mencakup seluruh area Universitas. Terdapat masalah yang sering di keluhkan pengguna yaitu pengguna tidak mengetahui jumlah pengguna *access point* sehingga kecepatan akses internet lambat. Permasalahan lain pengguna tidak mengetahui seluruh titik lokasi access point. Berdasarkan permasalahan di atas, akan dibuat sistem untuk menonitoring jumlah pengguna *access point*, lokasi *access point*, dan lokasi pengguna. Dengan adanya sistem ini pengguna bisa mengetahui titik lokasi *acess point* yang tidak mempunyai jumlah pengguna sehingga pengguna mendapatkan koneksi internet yang cepat dan optimal. Implementasi sistem menggunakan SOAP web service. Berdasarkan pengujian sistem yang telah dilakukan di dapatkan rata-rata *request time* sistem adalah 27973 ms, *throughput* sebesar 56,45 KBps, dan *latency* sebesar 2267ms.

Kata Kunci: Monitoring Access Point, Web Service, SOAP, Unifi

Dedi Tiarno. 2015 : Monitoring the Number of Users Access Point Based Simple Object Access Protocol Web Service (Case Study: Universitas Brawijaya)

Advisor: Eko Sakti Pramukantoro, S.Kom, M.Kom dan Kasyful Amron, ST., M.sc

ABSTRACT

Brawijaya University is one of the leading university in Indonesia that provides Internet service to support teaching and learning activities for lectures, student, and employee. Access point is used Unifi product that has high performance and can accommodate one hundred users simultaneously. Unifi access points spread over several locations that cover the entire area of the University. There are many problems that are often happened such as complain about the user does not know the number of users that use the access point. It makes speed of internet access slow. Another problem users is do not know the whole point of the location of the access point. Based on the above problems, the system will be made to know number of users, the location of the access point, and users location. It will make user know the point acess point locations that do not have the number of users. So users can get fast and optimal internet connection. Implementation of the system use SOAP web service. Based on system testing that has been done in getting the average request time system is 27 973 ms, throughput is 56.45 KBps, and latency of 2267 ms.

Keywords *:* Monitoring Access Point, Web Service, SOAP, Unifi

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi membuat manusia tidak akan lepas akan kebutuhan untuk saling berbagi informasi. Munculnya standar *Wireless Local Area Network* (WLAN) IEEE 802.11 menjawab tantangan bahwa komunikasi tidak hanya menggunakan kabel melainkan berbagai alat tanpa kabel secara *real time* tanpa batas akan jarak, ruang dan waktu. *Access point* merupakan salah satu alat yang diwujudkan sebagai jalur akses tanpa kabel, dimana menyambungkan alat berperangkat *wireless* untuk terhubung ke internet. *Access point* memancarkan koneksi melalui gelombang radio dengan mengunakan kekuatan sinyal, semakin besar kekuatan sinyal semakin luas jangkauannya. Jangkauan sinyal dari satu atau beberapa *access point* yang mencakup suatu area disebut dengan *Hotspot*. Dengan adanya *Hotspot* pengguna yang masih berada jangkauan sinyal *Access point* masih bisa terhubung ke jaringan.

Universitas Brawijaya merupakan salah satu universitas terkemuka di Indonesia yang menyediakan fasilitas dan layanan berbasis teknologi informasi untuk dosen, karyawan dan mahasiswa. Layanan ini ditujukan untuk menunjang aktivitas kegiatan belajar mengajar. Universitas Brawijaya membuat area *Hotspot* untuk menyediakan layanan internet yang mencakup seluruh area kampus melalui alat berupa *access point* yang tersebar di Universitas Brawijaya [BIT-14]. *Access point* yang digunakan adalah produk Unifi yang mempunyai kinerja tinggi dan mampu menampung seratus pengguna secara bersamaan [UBN-14]. *Access point* Unifi tersebar di beberapa titik lokasi yang mencakup seluruh area Universitas. Terdapat permasalahan yang sering dikeluhkan oleh pengguna yaitu kecepatan akses internet yang didapatkan lambat. Hal ini disebabkan access point kelebihan kapasitas jumlah pengguna yang menyebabkan pemakaian *bandwidth* bersama yang menurunkan kecepatan transfer data. Permasalahan lain pengguna tidak mengetahui semua titik lokasi *aceess point* dan hanya beberapa titik lokasi yang diketahui pengguna untuk mengakses internet. Berdasarkan permasalahan di atas, akan dibuat sistem untuk menonitoring jumlah pengguna yang terhubung ke

sebuah *access point*. Pengguna akan mendapatkan informasi tentang lokasi *access point*, jumlah pengguna yang terhubung ke *access point*, dan lokasi pengguna. Dengan adanya sistem ini pengguna bisa mengetahui titik lokasi acess point yang tidak mempunyai jumlah pengguna sehingga pengguna bisa menuju ke titik lokasi tersebut dan mendapatkan koneksi internet yang cepat dan optimal.

Metode yang digunakan dalam penelitian ini menggunakan SOAP *web service* merupakan sistem yang dirancang untuk mendukung interopabilitas interaksi antar aplikasi tanpa terpengaruh oleh *platform*, bahasa pemrogaman, dan teknologi. Sedangkan SOAP merupakan protocol yang digunakan untuk pertukaran data melalui protocol http. Dengan adanya SOAP *web service* akan memberikan layanan data berupa informasi *Access Point* yang tersebar di Universitas Brawijaya. Keluaran dari penelitian ini berupa sistem monitoring jumlah pengguna *access point* dengan menggunakan bahasa pemrograman PHP dengan database Mongodb sebagai DBMS. Pengguna mendapatkan informasi berupa pemetaan lokasi *access point* di Universitas Brawijaya, *access point* yang dijadikan koneksi oleh pengguna, dan jumlah pengguna yang terhubung ke *access point* berdasarkan waktu yaitu *real time*, hari dan bulan. Berdasarkan uraian diatas penelitian tugas akhir ini akan membahas mengenai "MONITORING JUMLAH PENGGUNA ACCESS POINT BERBASIS SIMPLE OBJECT ACCESS PROTOKOL (SOAP) WEB SERVICE" (Studi Kasus: Universitas Brawijaya Malang).

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang di atas, maka permasalahan yang akan dibahas dalam penulisan tugas akhir ini adalah sebagai berikut:

1. Bagaimana implementasi sistem *monitoring* jumlah pengguna *access point* dengan *simple object access protocol* (SOAP) *web service*?
2. Bagaimana performansi *web service* dan sistem berdasarkan *request time*, *throughput*, dan *latency* berdasarkan implementasi sistem *monitoring* jumlah pengguna *access point* berbasis *simple object access protocol* (SOAP) *web service*

1.3 Batasan Masalah

Batasan masalah dalam skripsi ini adalah sebagai berikut:

1. Studi kasus penelitian dilakukan di lingkungan Universitas Brawijaya.
2. Tidak membahas keamanan *web service*.
3. Perancangan dan implementasi *web monitoring access point* menggunakan protokol *Simple Object access Protocol* (SOAP) dan *Web Service Description Language* (WSDL).
4. *Access point* yang digunakan pada penelitian ini merupakan *product* dari Unifi sebanyak 4 buah *access point* yang ada di gedung PPTI.

1.4 Tujuan

Tujuan dari penelitian ini adalah membuat aplikasi web berbasis *Simple Object access Protocol* (SOAP) *web service* untuk memonitoring jumlah pengguna *access point*, lokasi *access point*, dan lokasi pengguna dengan memanfaatkan infrastruktur *access point* Unifi.

1.5 Manfaat

Manfaat dari penelitian ini adalah pengguna (dosen, mahasiswa, karyawan) bisa menentukan titik lokasi *access point* yang tidak memiliki jumlah pengguna sehingga performa pengguna dalam mengakses internet lebih cepat dan optimal.

1.6 Sistematika Penulisan

Sistematika penulisan pada laporan ini adalah sebagai berikut:

BAB I : PENDAHULUAN

Bab ini berisi latar belakang penulisan, perumusan masalah, tujuan, batasan masalah, manfaat penelitian dan sistematika penulisan skripsi.

BAB II : TINJAUAN PUSTAKA

Bab ini berisi jurnal, uraian dasar teori dan referensi yang mendukung dalam perancangan sistem *monitoring* jumlah

pengguna *access point* berbasis SOAP *web service* yang akan dijadikan acuan untuk pelaksanaan penelitian.

BAB III : METODOLOGI PENELITIAN DAN PERANCANGAN

Bab ini berisi metode dan analisa kebutuhan yang digunakan dalam perancangan sistem *monitoring* jumlah pengguna *access point* berbasis SOAP *web service* pada infrastruktur *Unifi*

BAB V : IMPLEMENTASI

Bab ini berisi penerapan sistem *monitoring* jumlah pengguna *access point* berbasis SOAP *web service* yang dibangun menggunakan bahasa pemrograman php dan *database* MongoDB.

BAB VI : PENGUJIAN DAN ANALISIS

Bab ini berisi pembahasan tentang pengujian dan analisa pengujian dari sistem *monitoring* jumlah pengguna *access point* berbasis SOAP *web service* pada arsitektur Unifi.

BAB VII : KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari sistem yang telah dirancang dan saran dari pembuatan sistem *monitoring* jumlah pengguna *access point* berbasis SOAP *web service* pada arsitektur Unifi.

BAB II
TINJAUAN PUSTAKA

2.1 Konsep Wi-Fi

Wireless Local Area Network (W-LAN) adalah suatu sistem komunikasi data melalui jaringan nirkabel dengan menggunakan *frekuensi radio* untuk mengirim dan menerima data melalui *free space* sehingga meminimalkan kebutuhan kabel. Dengan adanya W-LAN sebagai pengganti jaringan LAN maka memberikan beberapa manfaat, diantaranya adalah sebagai pengganti jaringan *Ethernet* atau kabel, kecepatan dan kemudahan dalam instalasi, mobilitas yang tinggi, mengurangi biaya, dan memperluas titik akses [RAB-08].

WiFi (*Wifeless fidelity*) merupakan standart diperkenalkan oleh WiFi Aliance yaitu organisasi nonprofit yang digunakan oleh jaringan W-LAN untuk melakukan pertukaran data tanpa kabel dengan interoperabilitas antar perangkat standart IEEE 802.11[WIF-14]. Standart terbaru muncul dengen spesifikasi 802.11 a,b, dan n dengan menawarkan banyak peningkatan mulai dari luas cakupan hingga kecepatan transfer. Tabel 2.1 menjelaskan standarisasi *wireless*, berikut tabelnya.

Tabel 2. 1 Perbandingan Standart Jaringan 802.11

Spesifikasi	Kecepatan (Mb/s)	Jarak Indor (m)	Jarak Outdor (m)	Frekuensi Band (GHz)	Type Penerima
802.11 a	54	35	120	5	a
802.11 b	11	38	140	2,4	b
802.11 g	54	38	140	2,4	b, g
802.11 n	150	70	250	2,4 dan 5	b,g,n

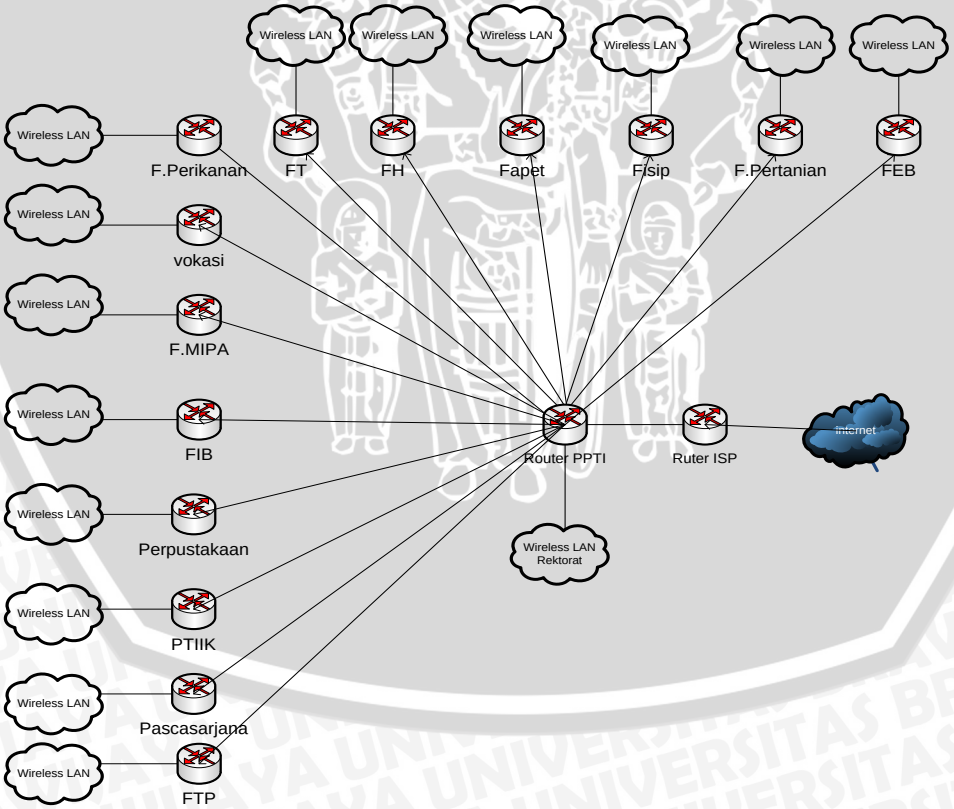
Sumber : [WID-13]

2.2 Infrastruktur W-LAN Universitas Brawijaya

PPTI merupakan pengkajian dan pengembangan teknologi informasi di Universitas Brawijaya. Salah satu tugasnya adalah merancang dan memelihara layanan yang terkait fasilitas jaringan internet baik kabel maupun *nirkabel* di

Universitas Brawijaya. Infrastruktur jaringan Universitas Brawijaya saat ini berpusat pada gedung PPTI, dimana gedung-gedung maupun fakultas terhubung dengan router PPTI menggunakan media transmisi berupa kabel *fiber optic*. Khusus fakultas Kedokteran dan Ilmu administrasi, jaringannya tidak terhubung ke router PPTI melainkan berdiri sendiri.

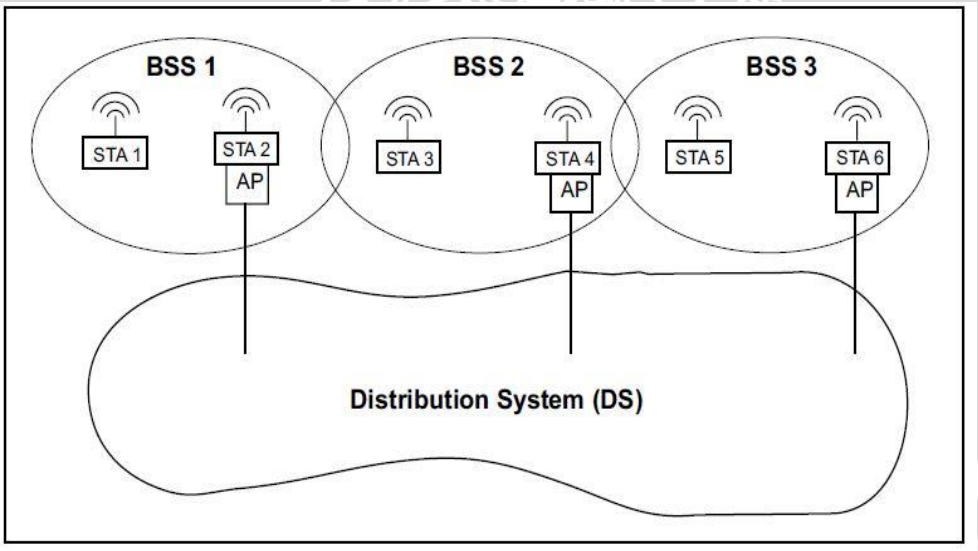
Koneksi internet di Universitas Brawijaya dilakukan perluasan dengan membuat area Hotspot di lingkungan kampus menggunakan *Wireless access point*. Hampir semua Fakultas kecuali Fakultas Pertanian diberikan hak untuk mengelola jaringannya sendiri. Hal ini dimaksudkan untuk mempermudah kontrol jaringan *wireless* yang berada di fakultas. PPTI mengelola *wireless acces point* sejumlah 106 yang tersebar di area public Universitas Brawijaya seperti gazebo perpustakaan, widyaloka, gedung rektorat, gedung PPTI, Fakultas Pertanian, dan beberapa gazebo di fakultas. Gambar infrastruktur jaringan W-LAN Universitas Brawijaya saat ini, secara umum ditunjukkan gambar 2.1 berikut ini.



Gambar 2. 1 Infrastruktur WLAN UB

Sumber : [PPTI-14]

Topologi jaringan *wireless* merupakan hubungan antara satu komponen dengan komponen lainnya dengan menggunakan frekuensi radio. Topologi di Universitas Brawijaya menggunakan *cover range* area untuk membentuk *extended service set* (ESS). *Extended Service Set* (ESS) terdiri dari beberapa *basic service set* (BSS) yang akan melakukan *forwarding* radio antar *access point* yang terkoneksi menggunakan jaringan kabel yang akan membentuk *distribubution system* (DS). Setiap *access point* diatur dengan menggunakan *Service Set Identifier* (SSID) yang sama yaitu *wifi-ub* dan *channel* yang berbeda yang bertujuan untuk menghindari interferensi sinyal. Pengguna dapat melakukan roaming sehingga ketika pengguna berpindah tempat selagi masih dalam range area dari *access point* pengguna tidak akan kehilangan koneksi. Topologi ini mirip dengan sistem komunikasi seluler, yang mana masing-masing BSS berlaku seperti sel dan masing - masing *access point* berlaku seperti *Base Tranceiver Station* (BTS) [RAB-08].



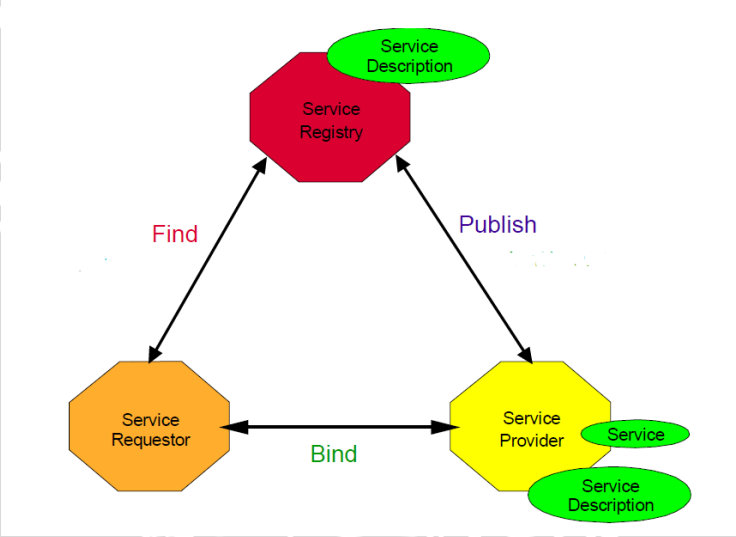
Gambar 2. 2 Topologi Extended Service Set (ESS)

Sumber : [RAB-08]

2.3 Web Service

Web service merupakan sistem yang dirancang untuk mendukung *interopabilitas* interaksi antar aplikasi tanpa terpengaruh oleh platform dan bahasa pemrogaman. *Web service* menyediakan layanan berupa fungsi-fungsi dengan

parameter tertentu. Layanan ini akan dipanggil oleh aplikasi lain secara langsung dengan menggunakan URI (*Uniform Resurce Identifier*) sesuai dengan otoritas yang ditentukan. *Web service* bisa memiliki antarmuka yang dijelaskan dengan WSDL (*Web service description language*) dimana aplikasi yang berinteraksi dengan *web service* akan di deskripsikan datanya dengan menggunakan protokol SOAP, yang disampaikan menggunakan dengan format data XML melalui protocol HTTP [BOO-04].



Gambar 2. 3 Architecture Web Service

Sumber :[KRE-01]

Gambar 2.3 merupakan *architecture web service*, dimana terdapat tiga *service* utama yaitu adalah *service requestor* (peminta layanan), *service provider* (penyedia layanan), dan *service registry* (mendeskripsikan layanan). *Service provider* menyediakan layanan dan mempublish ke *service registry* agar layanan tersebut dapat tersedia. *Service registry* mendeskripsikan semua layanan yang di berikan oleh *service provider*. *Service requestor* sebagai peminta layanan ke *service provider* serta menggunakan layanan tersebut. Interaksi antar service melibatkan tiga operasi yaitu *publish*, *find*, dan *bind*. Operasi *publish* untuk menerbitkan layanan-layanan yang di berikan *service provider* ke *service registry*. *Operasi find* digunakan service requestor untuk menemukan layanan yang disediakan oleh *service registry*. Operasi *bind* digunakan *service requestor* untuk

melakukan interaksi ke *service provider* segingga *service requestor* bisa mengakses layanan yang disediakan oleh *service provider* [KRE-01].

2.3.1 Extensible Markup Language (XML)

XML (Extensible Markup Language) adalah format pertukaran data berbasis text sederhana untuk mewakili informasi yang terstruktur antar aplikasi, antar komputer melalui jaringan [BOO-04]. XML menyediakan suatu cara terstandarisasi untuk menggambarkan isi dari dokumen. XML dapat digunakan untuk menggambarkan data dengan standart yang telah di tentukan. XML memiliki tiga tipe file yaitu [PUT-10] :

1. XML merupakan standar format dari struktur berkas/file.
2. XSL merupakan standar untuk memodifikasi data yang diimport atau dieksport.
3. XSD merupakan struktur dapat digunakan untuk mendefinisikan, menggambarkan dan kosakata katalog XML untuk kelas dokumen XML.

Kemampuan XML untuk mewakili atribut, *childrens* dan karakter data memberikan cara yang baik dalam mendeskripsikan struktur data dibandingkan JSON. XML memberikan informasi yang sangat rinci, termasuk informasi jenis data, tipe data, dan setiap elemen memiliki atribut yang mencakup lebih banyak informasi. XML merupakan pilihan yang cocok jika informasi dari tipe data dibutuhkan oleh aplikasi. Semakin besar format data XML membutuhkan koneksi yang cepat. XML adalah pilihan yang tepat digunakan untuk pertukaran data antara komputer ketika koneksi jaringan yang digunakan cepat. Berikut adalah kelebihan format XML [QUI-10] :

1. Redudancy

   XML markup sangat terinci. Misalnya, setiap tag akhir harus diberikan seperti </ description>, hal ini memungkinkan komputer menangkap kesalahan.

2. Self-describing

   Pembacaan XML adalah format berbasis teks, nama elemen dan atribut dalam XML membuat sebuah dokumen XML dapat dilihat mulai dari *head* sehingga membantu pengguna untuk menemukan kesalahan.

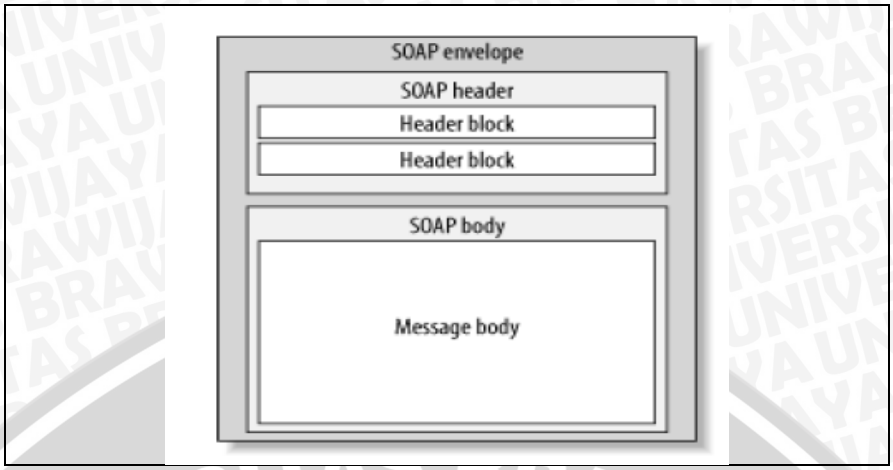
3. *Network effect* dan *XML Promise*

   Setiap dokumen XML dapat dibaca dan diproses dengan menggunakan tool XML sehingga setiap dokumen XML dapat ditambahkan nilainya.

2.3.2 *Simple Object Access Protocol* (SOAP)

SOAP merupakan protocol pertukaran data secara tersentralisasi maupun terdistribusi. SOAP juga merupakan standar proses komunikasi antar sistem yang berbeda bahasa pemrogaman maupun platform melalui jaringan dengan protocol HTTP, SMTP, FTP menggunakan format pengiriman pesan berbasis XML. Tujuan utama dari SOAP adalah berfokus penyediaan paket data dan menentukan aturan *encoding* yang digunakan untuk transmisi data. Tujuan lain adalah menyediakan model yang dapat digunakan untuk melaksanakan operasi *Remote Procedure Call* (RPC) menggunakan SOAP. SOAP menggunakan RPC-style untuk mempresentasikan parameter dan nilai-nilai yang dikembalikannya. RPC adalah sebuah metode yang memungkinkan aplikasi untuk mengakses sebuah prosedur yang berada di komputer lain. Sebuah server menyediakan layanan *remote procedure,* server membuka *socket* kemudian menunggu pengguna meminta prosedur yang disediakan server. Pengguna akan meminta ke RPC port ketika pengguna tidak tahu harus menghubungi port yang digunakan. RPC port akan memberikan port yang diminta pengguna, kemudian nilai-nilai yang diminta pengguna akan dikirimkan oleh server sesuai dengan prosedur yang diminta [JAN-13].

SOAP memiliki struktur pesan yang terdiri dari *envelope* sebagai amplop yang membungkus pesan dari SOAP. Sebuah *envelope* berisi *header* dan *body* yang diperlukan, seperti yang ditunjukkan pada Gambar 2.4. Header berisi blok informasi yang terkait bagaimana pesan diproses termasuk routing, otentikasi, dan otorisasi. Body berisi pesan yang dikirim dan diproses, semua informasi yang berada pada body menggunakan format XML [WIL-10].



Gambar 2. 4 Struktur Pesan SOAP
Sumber : [JAM-01]

Sintaks XML untuk menampilkan pesan SOAP didasarkan pada *http://www.w3.org/2001/XMLSchema*. Identifier namespace XML ini menunjuk ke XML Schema yang mendefinisikan struktur pesan SOAP terlihat seperti gambar 2.5.

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <device soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <ap_mac xsi:type="xsd:string">14:5a:05:dd:17:1c</ap_mac>
    </device>
  </soapenv:Body>
</soapenv:Envelope>
```

Gambar 2. 5 Document Style SOAP

Sumber : [W3C-14]

2.2.3 Web Service Description Language (WSDL)

WSDL merupakan document xml dengan tampilan antar muka yang digunakan untuk mendeskripsikan type data, lokasi dan operasi yang diberikan oleh web service ke pengguna [W3C-14]. WSDL menggunakan kombinasi antara SOAP dan *XML Schema* untuk mendeskripsikan layanan web service. WSDL bisa dikatakan sebagai pintu (gateway) komunikasi antara web service dengan pengguna. Penggguna web service membaca file WSDL untuk menentukan operasi yang tersedia di server. Tipe-tipe data yang digunakan di-embed pada file WSDL dalam bentuk *XML Schema*. Pengguna menggunakan SOAP *request* untuk

memanggil operasi-operasi yang terdaftar pada file WSDL dengan format pesan XML melalui protokol HTTP [JAN-13].

WSDL bermanfaat ketika server mengalami penambahan service maupun perubahan operasi sehingga pengguna akan menyesuaikan service yang ada pada WSDL. Gambar 2.6 menunjukkan struktur dari WSDL yang didefinisikan dengan beberapa element yaitu type, message, porttype, dan binding [WIL-10].

Gambar 2. 6 Struktur WSDL

Sumber : [WIL-10]

Element	Keterangan
<types>	Mendefinisikan tipe data yang digunakan oleh layanan web. WSDL menggunakan sintaks XML Schema untuk mendefinisikan tipe data
<message>	Mendefinisikan data yang akan dikomunikasikan
<portType>	Menjelaskan *web service*, operasi yang dapat dilakukan, dan pesan yang terlibat.
<binding>	Protokol komunikasi yang digunakan *web service* untuk mengikat operasi-operasi yang ada di SOAP.

Tabel 2. 2 Element WSDL

Sumber : [WIL-10]

2.4 Infrastruktur Access Point Unifi

Unifi (Unified Enterprise WiFi) merupakan perusahaan WiFi yang menawarkan perangkat *access point* yang mudah dikelola dan digunakan. *Access point* Unifi menggunakan software *Unifi Controller* untuk mengontrol seluruh aktifitas *access point* yang berada dalam jaringan sehingga menghemat waktu dan tenaga pengelola untuk memonitoring aktifitas user dan *access point* secara *real time* dan konfigurasi perangkat *access point*. Software Unifi Controller menggunakan database MongoDB untuk mengelola data dari keseluruhan aktifitas *access point*. Dengan adanya database ini controller Unifi mempunyai performa yang tinggi dalam menampilkan aktifitas *access point* secara *real time*.

Controller Unifi dapat diinstall di jaringan komputer dengan berbagai platform (Windows, Mac, dan Linux). Media akses *controller* melalui web browser dan mempunyai *user interface* yang interaktif dan dinamis. Dengan adanya controller pengelola *access point* dapat dengan mudah untuk melakukan konfigurasi *access point*. *Access point* Unifi menggunakan *hardware* yang handal yaitu WiFi 802.11ac MIMO technology dengan kecepatan sampai Gigabit dan range area mencapai 144 meter. Sedangkan untuk WiFi 802.11 n MIMO technology bekerja dengan menggunakan frekuensi 2,4 dan 5 GHz [UBN-14].

2.4.1 Model Unifi

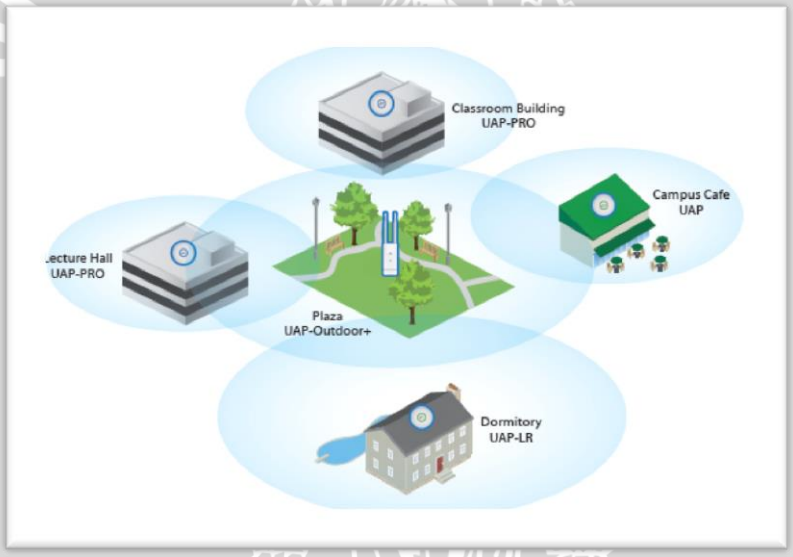
Unifi access point yang digunakan di Universitas Brawijaya menggunakan dua model yaitu model *indoor* dan model *outdoor*. Model *indoor* digunakan di dinding atau langit-langit sebuah bangunan yang di desain ramping, menyediakan LED berbentuk cincin dan persegi di setiap *access point*. LED digunakan sebagai penanda setiap perangkat dan penentu lokasi *access point* di map. Model *indoor* unifi di dukung oleh *Power over Ethernet* (PoE) yang fungsi memungkinkan kedua daya dan data akan dilewatkan melalui kabel Ethernet tunggal. Model Unifi indoor mempunyai jangkauan sinyal 122 m dengan kecepatan 450 Mbps dalam frekuensi 2,4 GHz dan kecepatan 1300 Mbps dalam frekuensi 5 GHz [UBN-14].

Model *outdor* setidaknya mempunyai feature yang sama dengan model indoor akan tetapi ada beberapa fitur tambahan. Model outdor menggunakan antena omni directional pada sehingga bias coverage area mencapai 360 derajat. Model Unifi *outdor* mempunyai jangkauan sinyal 183 m dengan kecepatan 450

Mbps dalam frekuensi 2,4 GHz dan kecepatan 1300 Mbps dalam frekuensi 5 GHz [UBN-14].

2.4.2 Topologi Unifi

Unifi menggunakan topologi *Zero Handoff Roaming* atau *Extended Service Set* (ESS) yaitu metode yang menggunakan *cover range* area dari *access point* untuk mengcover koneksi penguna. Ketika pengguna berpindah tempat yang masih berada dalam cakupan coverage area *access point*, *access point* akan melakukan *roaming* (proses perpindahan konektifitas) terhadap *access point* terdekat dari perangkat pengguna sehingga pengguna tidak akan kehilangan koneksi dan mempertahankan koneksi [UBN-14]. Di bawah ini Gambar 2.7 Topologi Zero Handoff Roaming.



Gambar 2. 7 Topologi Zero Handoff Roaming

Sumber : [UBN-14]

2.4 MongoDB

MongoDB merupakan database yang berbentuk *document oriented* yang *open source* dimana database ini mempunyai kemampuan yang tinggi *(hight performance)*, fleksibel, ketersediaan data yang tinggi, dan *scalable* dalam penyimpanan data. MongoDB memberikan kinerja tinggi untuk baca dan tulis karena memanfaatkan memori komputer. MongoDB dikatakan database NoSQL

(Not only SQL) artinya database ini tidak menggunakan relasional sebagai penyambung data-data antar table di database. NoSQL tidak perlu mendefinisikan terlebih dahulu *table* yang akan digunakan dan strukturnya *fleksibel*. MongoDB tidak menggunakan table, kolom dan baris, namun menggunakan *collection, document, dan field* [CHO-10].

MongoDB mempunyai data unit yang dikatakan sebagai document, ini sama dengan row dalam *relational database,* kumpulan dokumen disimpan dalam collection, ini sama dengan table dalam relational database. MongoDB mempunyai database yang dinamakan *collection* dimana *collection* ini dapat menerima lebih dari satu database secara independen. MongoDB memiliki javascript shell yang sangat berguna untuk urusan administrasi dan manipulasi data. Setiap document mempunyai key khusus yang di sebut _id untuk membedakan antar document, dengan adanya key ini memudahkan untuk melakukan pencarian [KCM-10]. Pada Table 2.4.1 menunjukkan perbedaan konsep antara SQL dan MongoDB.

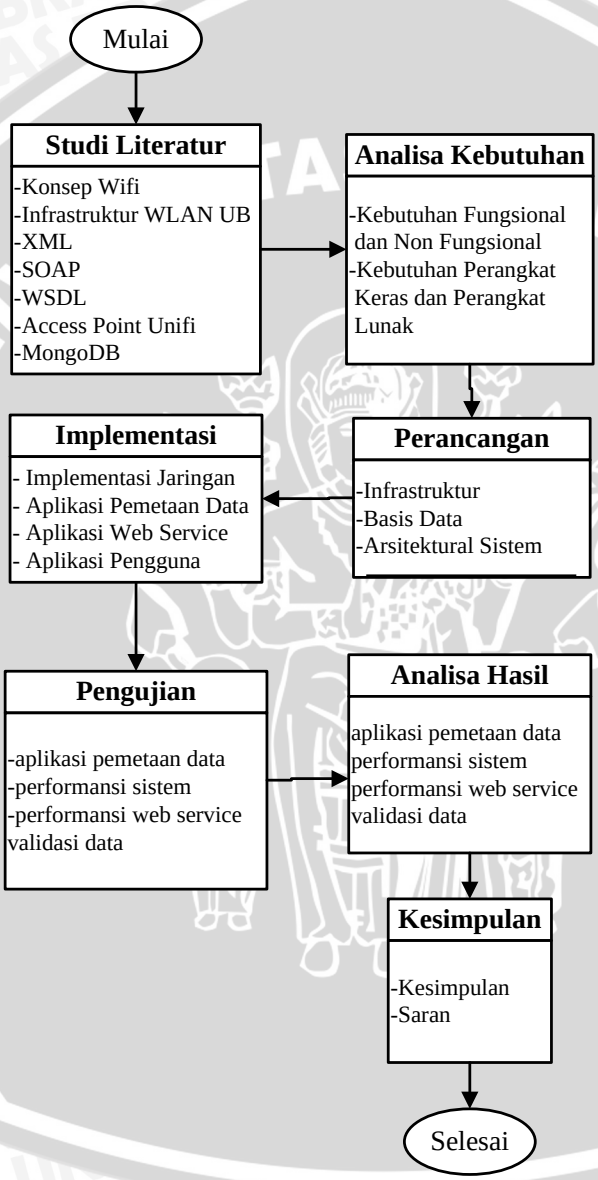
Tabel 2. 3 Perbedaan SQL dan NoSQL

SQL atau Relational Database	**NoSQL atau MongoDB**
Tabel	Collection
Row/Record	Document/ BSON Document
Column	Field
Index	Index
Primary Key (ditentukan oleh sebuah field yang unix)	Primary key (field di buat otomatis dengan nama_id)

Sumber : [CHO-10]

BAB III
METODOLOGI PENELITIAN DAN PERANCANGAN

Metode penelitian yang akan dilakukan pada penelitian ini terdiri dari beberapa tahap seperti berikut :



Gambar 3. 1 Diagram Alir Metodologi Penelitian

3.1 Studi Literatur

Studi literatur digunakan untuk mempelajari teori-teori yang menunjang dan berkaitan dengan penelitian yang sedang dilakukan. Teori-teori pendukung pendukung tersebut meliputi konsep *WiFi*, Infrastruktur W-LAN Universitas Brawijaya, Konsep SOAP, Arsitektur *UniFi*, dan MongoDB.

3.2 Analisa Kebutuhan

Analisa kebutuhan dilakukan untuk mengidentifikasi dan menentukan kebutuhan yang diperlukan untuk merancang sistem *monitoring* jumlah pengguna *access point* berbasis SOAP *web service.* Maka dilakukan analisis kebutuhan mengenai sistem yang akan di bangun.

3.2.1 Analisa Kebutuhan Sistem

Analisa kebutuhan sistem dibagi menjadi dua yaitu kebutuhan fungsional dan non fungsioanal, sebagai berikut :

1. Kebutuhan fungsional

   Kebutuhan fungsional dibutuhkan oleh sistem sebagai fitur yang diberikan sistem ke pengguna. Kebutuhan fungsional dalam pembuatan sistem dibagi menjadi dua, yaitu kebutuhan fungsional disisi pengguna dan kebutuhan fungsional disisi admin.

   a. Pengguna

      Kebutuhan fungsional yang harus disediakan sistem disisi pengguna yaitu:

      - Sistem memberikan informasi ke pengguna berupa lokasi *access point* pada map Universitas Brawijaya.
      - Sistem mampu memberikan informasi jumlah pengguna pada *access point.*
      - Sistem mampu memberikan lokasi *access point* yang di jadikan koneksi pengguna.
      - Sistem mampu menunjukkan jumlah pengguna yang terhubung ke *access point* berdasarkan waktu yaitu *real time*, hari dan bulan.

b. Admin

Admin merupakan pengguna yang menggunakan sistem dan memiliki tambahan akses terhadap sistem. Tambahan akses sistem disisi admin adalah menentukan atau *update* lokasi *access point* dengan menggunakan *user interface* di sistem. Admin menentukan lokasi *access point* di map yang telah di sediakan aplikasi dengan melakukan drag *access point*.

2. Kebutuhan Non-fungsional

Kebutuhan non-fungsional adalah sebagai berikut:

a. *Availability* data yang tersimpan dalam database server terjamin ketersediaannya.
b. Database memiliki performa yang baik dalam melakukan proses query.
c. Sistem mampu memberikan informasi ke pengguna berdasarkan data yang sebenarnya.

Berdasarkan analisa kebutuhan yang telah didefinisikan dibutuhkan lingkungan penelitian dimana ditentukan dengan melakukan identifikasi kebutuhan infrastruktur yaitu perangkat keras (hardware) maupun perangkat lunak (software) yang diperlukan untuk membangun sistem.

3.2.2 Analisa Kebutuhan Infrastruktur

Analisa kebutuhan infrastruktur dalam penelitian ini dibagi menjadi 2 yaitu kebutuhan perangkat keras digunakan untuk proses perancangan infrastruktur dan kebutuhan perangkat lunak digunakan untuk proses perancangan sistem.

1. Kebutuhan Perangkat Keras

Kebutuhan akan perangkat keras dalam pembuatan sistem ini yaitu *access point* dan 2 buah server. *Access point* yang digunakan oleh adalah produk *Unifi Ubiquiti* dan 4 buah *access point* yang di kelola PPTI di gedung PPTI. Selain itu dibutuhkan 2 server dengan rincian satu server digunakan sebagai server *controller access point* yang berada di jaringan

PPTI dan satu komputer digunakan sebagai server aplikasi pengguna yang berada di laboratorium jaringan PTIIK.

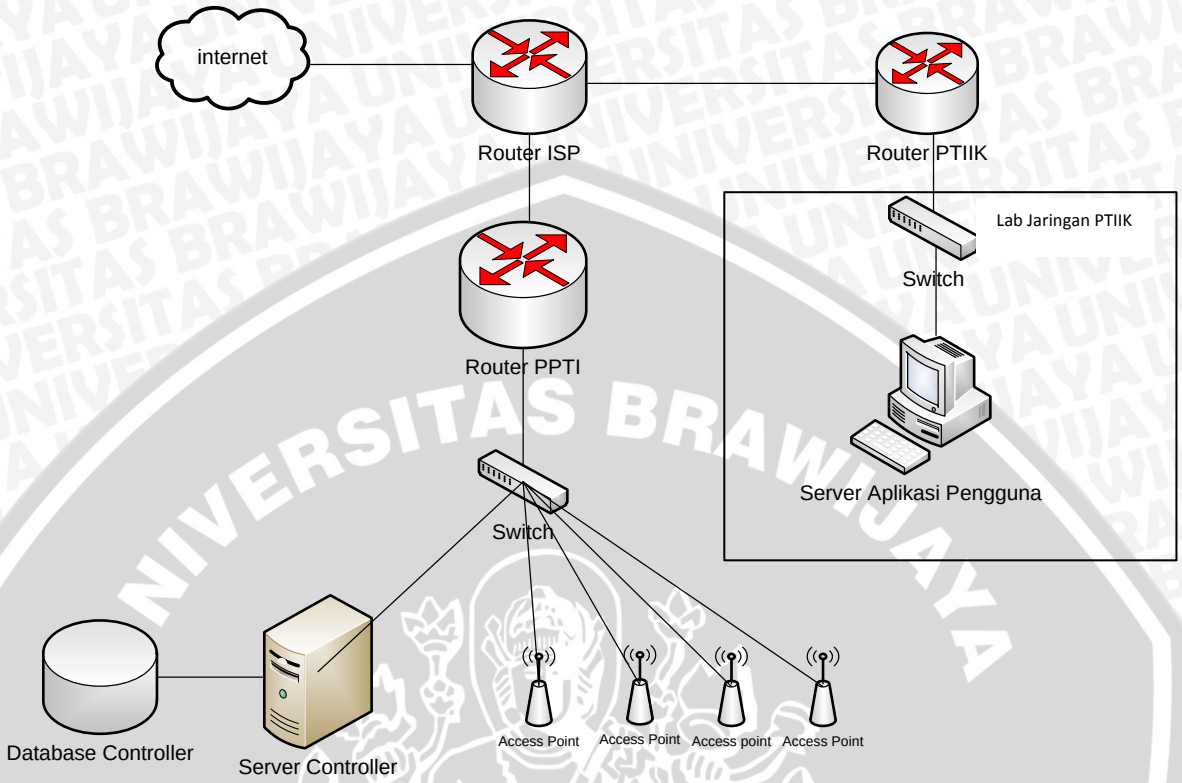
2. Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak yang di butuhkan dalam pembuatan sistem pada *server controller* yaitu *operating system* linux centos 5.0 dan ubuntu 14.04 digunakan sebagai *server controller* dan server aplikasi pengguna. *Apache web server* di gunakan sebagai web server dari sistem yang akan dibuat. Dibutuhkan controller unifi versi 3.2.1 sebagai *controller* dan konfigurasi *access point Unifi* secara terpusat dan mongodb sebagai databasenya. Sistem yang akan di buat juga menggunakan database MongoDB karena bersifat *open source* dan mempunyai *high performance* jika di bandingkan database SQL. Toolkit Nusoap yang untuk membuat SOAP *web service* dengan menggunakan bahasa pemrogaman PHP. CodeIgniter 2.2.0 sebagai framework MVC (model, view, controller) untuk membuat website secara dinamis dengan bahasa pemrogaman PHP. Terakhir dibutuhkan Google Map API versi-3 merupakan library yang digunakan memetakan lokasi *access point* di map pada framework codeigniter.

3.3 Perancangan

Perancangan merupakan tahapan yang menjelaskan tentang proses kerja dari keseluruhan sistem. Sekaligus proses interaksi yang terjadi antara server dengan aplikasi. Perancangan pada penelitian ini meliputi 3 tahapan yaitu : perancangan infrastruktur jaringan, perancangan basis data, dan perancangan Arsitektural sistem.

3.3.1 Perancangan Infrastruktur Jaringan



Gambar 3. 2 Perancangan Jaringan

Perancangan jaringan sistem monitoring jumlah pengguna *access point* seperti gambar 3.5 menggunakan sampel infrastruktur *access point* yang berada di ruang PPTI. Proses monitoring aktifitas seluruh *access point* bersifat terpusat, dimana aktifitas dari semua *access point* dimonitoring dengan *aplikasi controller* kemudian data disimpan ke database controller. Aplikasi controller merupakan aplikasi dari access point unifi yang digunakan untuk mengatur aktifitas *access point*. Perancangan jaringan penelitian ini dilakukan dengan membuat 2 server yaitu server *controller* dan server aplikasi pengguna. Server *controller* digunakan untuk mengontrol seluruh aktifitas *access point* dengan menggunakan aplikasi controller dari *access point*. Server controller akan disisipkan aplikasi yang akan di buat yaitu aplikasi aplikasi *web service* dan aplikasi pemetakan data. Aplikasi *web service* merupakan aplikasi yang digunakan untuk memberikan layanan data ke pengguna. Aplikasi pemetaan data merupakan aplikasi yang digunakan untuk memetakan data dari database controller ke database web service atau database

sistem yang akan di buat. Pemetaan data di lakukan karena tidak semua data dari database *controller* di butuhkan sehingga hanya data yang di butuhkan saja yang akan diambil. Aplikasi *web service* tidak menggunakan database langsung dari controller. Hal ini karena faktor keamanan dan peforma yang bisa mengakibatkan server database *controller* menjadi *down* akibat permintaan secara terus-menerus sehingga bisa mengakibatkan infrastruktur dari *access point* tidak berfungsi.

Server *controller* berada di jaringan PPTI dengan infrastruktur seperti gambar 3.2. Pada penelitian nanti menggunakan 4 access point yang berada di gedung PPTI. Sedangakan server aplikasi pengguna berada di laboratorium jaringan gedung PTIIK. Server aplikasi pengguna digunakan untuk menangani permintaan informasi dari pengguna dalam dalam bentuk web. Server dari aplikasi pengguna di buat terpisah dengan server aplikasi pemetaan data dan *web service* untuk mengurangi beban server yang di timbulkan dari proses aplikasi pengguna akibat permintaan dari pengguna.

3.3.2 Perancangan Basis Data

Perancangan basis data di dapatkan dari *database controller* yang merupakan database dari *access point unifi*. *Database controller* merupakan database dari *access point* unifi digunakan menyimpan semua informasi tentang aktifitas dari *access point* dan *mongodb* sebagai databasenya. Pada database *controller* mempunyai 34 *collection*, dimana setiap *collection* menyimpan berbagai macam informasi mengenai aktifitas dari *access point*.

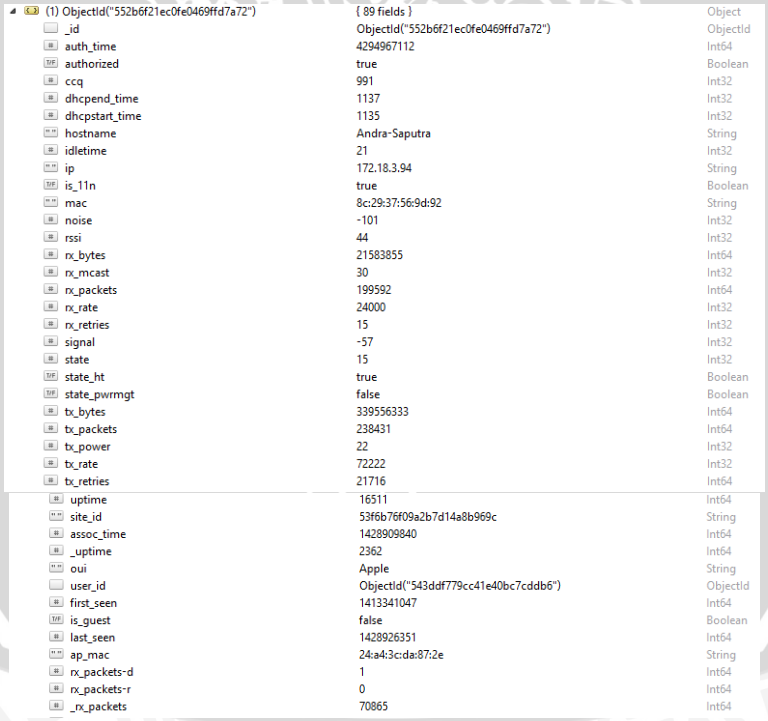
Berdasarkan analisa yang telah di lakukan, penelitian ini membutuhkan data dari 3 *collection* dari database *controller* yaitu *collection cache_stat, device,* dan *stat_daily*. Ketiga *collection* tersebut menyimpan berbagai macam informasi sehingga mempunyai struktur data yang kompleks, sehingga tidak semua data pada 3 *collection* tersebut di butuhkan dalam pembuatan sistem. Berdasarkan hal tersebut perlu dilakukan pemetaan data pada database baru sebagai database sistem. Jika sistem menggunakan database dari *controller* dan terjadi permintaan yang berlebihan dari pengguna yang mengakibatkan database *controller* mengalami *down* sehingga berdampak semua *access point* tidak berfungsi. Pemetaan data dilakukan karena faktor keamanan data, karena sistem akan diakses

secara publik dan berhubungan langsung dengan berbagai macam karakter dari pengguna. Sehingga jika semua informasi dari *access point* yang ada di *controller* sampai diketahui oleh pihak yang tidak bertanggung jawab maka akan berbahaya pada jaringan. Berdasarkan alasan tersebut maka dibuat pemetaan data dari database *controller* ke database sistem.

Hasil dari pemetaan data akan disimpan pada database sistem yaitu database *web service*. Berikut ini perancangan struktur data database web service yang merupakan hasil dari pemetaan struktur data dari 3 collection database controller.

1. Perancangan Struktur Data Users

   Gambar 3.3 merupakan struktur data dari *collection cache_sta*. Collection ini menyimpan semua informasi tentang pengguna yang terhubung ke access point secara *real time*. Berdasarkan analisa data yang telah dilakukan maka di buat struktur data yang akan di gunakan dalam pembuatan sistem seperti pada tabel 3.1.



Gambar 3. 3 Struktur Data *Collection cache_sta*

Tabel 3.1 merupakan hasil struktur data yang di petakan dari *collection cache_sta* database *controller* ke *collection users* database *web service*. *Collection users* digunakan sistem untuk menympan informasi pengguna

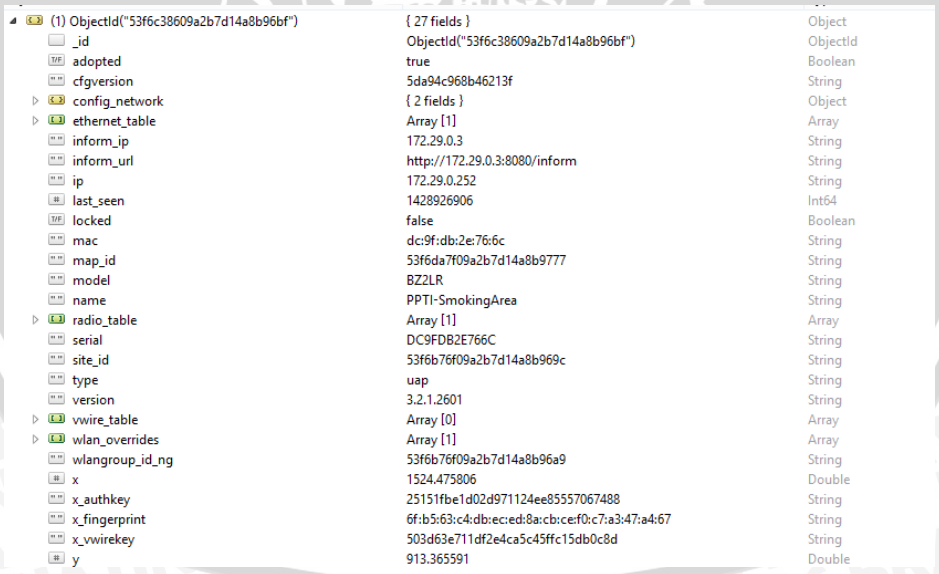
yang terhubung ke *access point* secara *real time*. Struktur data yang di simpan pada *collection users* antara lain MAC merupakan MAC *address* dari *users*, *ap_mac* merupakan MAC *address* dari *access point*, dan IP merupakan IP *address* yang digunakan pengguna.

Tabel 3. 1 Hasil Perancangan Struktur Data Users

Field	Type
_id	ObjectId
Mac	String
ap_mac	String
Ip	String

2. Perancangan Struktur Data *Access Point*

   Gambar 3.4 merupakan struktur data dari *collection device*. Collection ini menyimpan semua informasi tentang *access point*. Berdasarkan analisa data yang telah dilakukan maka di buat struktur data yang akan di gunakan dalam pembuatan sistem seperti pada table 3.2.



Gambar 3. 4 Struktur Data *Collection Device*

Tabel 3.2 merupakan hasil struktur data yang di petakan dari *collection device* database *controller* ke *collection access_point* database *web service*. *collection access_point* digunakan sistem untuk menyimpan informasi

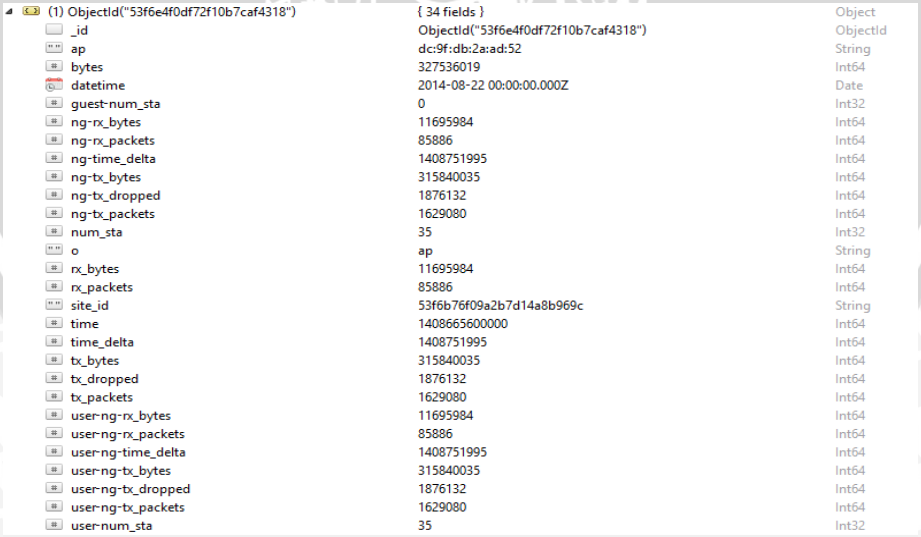
tentang *access point*. Struktur data yang di simpan pada *collection access_point* antara lain MAC merupakan MAC *address* dari *access point*, *name* merupakan nama dari *access point*, dan lokasi map dari *acces point* (x dan y).

Tabel 3. 2 Hasil Perancangan Struktur Data *Users*

Field	Type
_id	ObjectId
Mac	String
Name	String
X	Double
Y	Double

3. Perancangan Struktur Data *Users Daily*

   Gambar 3.5 merupakan struktur data dari *collection stat_daily*. *Collection* ini menyimpan semua informasi tentang jumlah pengguna yang terhubung ke *access point* berdasarkan waktu yaitu hari. Berdasarkan analisa data yang telah dilakukan maka di buat struktur data yang akan di gunakan dalam pembuatan sistem seperti pada table 3.3.



Gambar 3. 5 Struktur Data *Collection stat_daily*

Tabel 3.3 merupakan hasil struktur data yang di petakan dari *collection stat_daily* database *controller* ke *collection users_daily* database *web*

service. *Collection users_daily* digunakan sistem untuk menyimpan informasi tentang jumlah pengguna yang terhubung ke *access point* berdasarkan hari. Struktur data yang di simpan pada *collection users_daily* antara lain *ap* merupakan MAC *address* dari *access point*, *datetime* merupakan waktu berdasarkan hari, dan num_sta merupakan jumlah pengguna.

Tabel 3. 3 Hasil Perancangan Struktur Data *users_daily*

Field	Type
_id	ObjectId
Ap	String
Datetime	Date
num_sta	Int

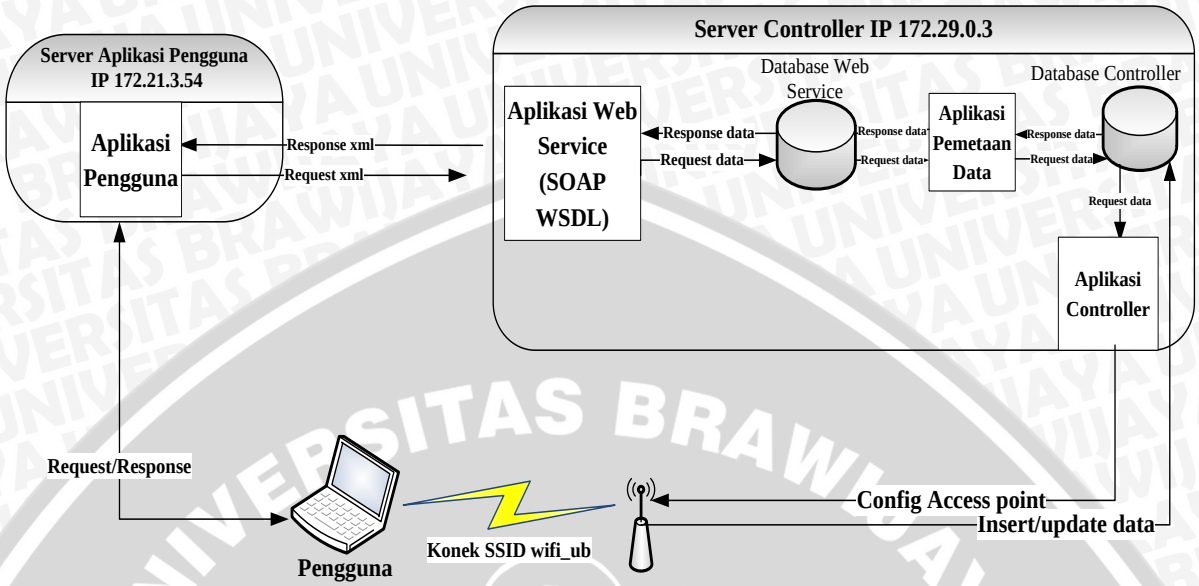
4. Perancangan Struktur Data Admin

   Data admin akan dibuat ke database *web service*, dimana collection admin untuk menyimpan data login admin ke sistem yaitu username dan password. Data admin bukan hasil dari pemetaan data dari database controller melainkan struktur data dibuat sendiri oleh admin sebagai data login ke sistem.

Tabel 3. 4 Struktur Data *Collection admin*

Field	Type
username	String
x_password	String

3.3.2 Perancangan Arsitektural Sistem



Gambar 3. 6 Perancangan Arsitektur Sistem

Gambar 3.6 merupakan gambaran dari arsitektur dari keseluruhan sistem yang telah di buat. Terdapat dua server yang berbeda jaringan yang di buat yaitu server controller IP 172.29.0.3 dan server aplikasi pengguna IP 172.21.3.54. Kedua server saling terhubung satu dengan yang lainnya dengan menggunakan DNS Universitas Brawijaya. Server controller IP 172.29.0.3 mempunyai tiga aplikasi yang di jalankan yaitu aplikasi controller, aplikasi pemetaan data, dan aplikasi *web service*. Aplikasi controller merupakan aplikasi bawaan dari access point unifi untuk memanagement aktifitas *access point* melalui web, aplikasi pemetaan data digunakan untuk memetakan data dari database controller ke database *web service,* dan aplikasi *web service* digunakan untuk menyediakan layanan data ke aplikasi pengguna. Server aplikasi pengguna IP 172.21.3.54 mempunyai satu aplikasi yang dijalankan yaitu aplikasi pengguna. Aplikasi pengguna digunakan untuk memberikan layanan ke pengguna berupa tampilan web untuk mendapatkan informasi tentang lokasi access point, jumlah pengguna yang terhubung ke *access point,* dan lokasi *access point* yang dijadikan koneksi oleh pengguna.

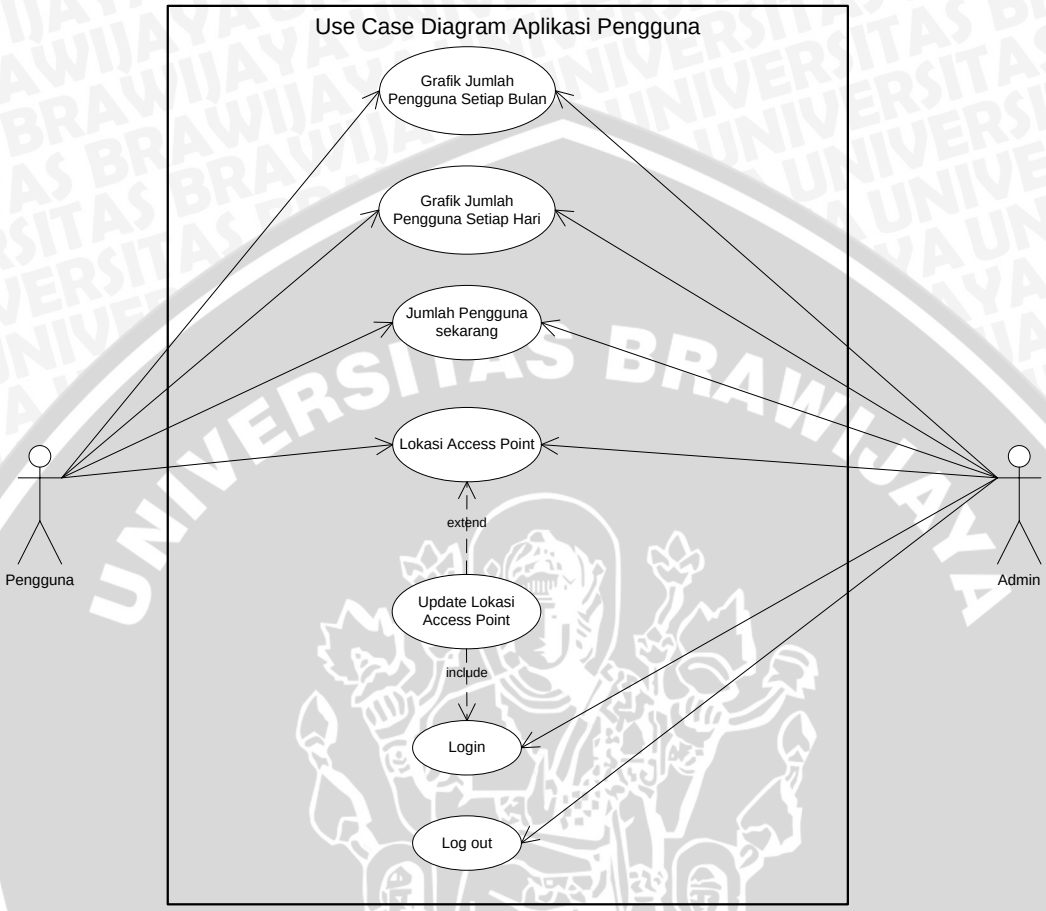
Cara kerja keseluruhan system diawali dari pengguna melakukan koneksi ke SSID wifi_ub untuk terhubung ke jaringan wifi di Universitas Brawijaya.

Kemudian aktifitas dari pengguna dan *access point* akan di simpan oleh sistem ke database controller. Access point unifi mempunyai aplikasi controller untuk melakukan management aktifitas *access point* menggunakan web dan data hasil management *access point* akan di simpan ke database controller. Selanjutnya data dari database controller akan di petekan ke database sistem/web service menggunakan aplikasi pemetaan data. Pemetaan data di lakukan karena tidak semua data dari database controller di butuhkan sehingga hanya data yang di butuhkan saja yang akan di ambil. Hal ini juga karena faktor keamanan dan peforma yang bisa mengakibatkan server database controller menjadi *down* akibat permintaan secara terus-menerus sehingga bisa mengakibatkan infrastruktur dari *access point* tidak berfungsi. Alur proses dan algoritma aplikasi pemetaan data akan dijelaskan secara detail pada perancangan basis data. Kemudian pengguna menjalankan aplikasi monitoring *access point* dengan melakuakan akses web ke alamat URL yang mengacu pada halaman web dari aplikasi pengguna yaitu http://172.21.3.54/apmonitor. Aplikasi pengguna akan melakukan *envelope* data request dari pengguna menggunakan format xml. Sebuah *envelope* berisi *header* dan *body*, header merupakan blok informasi yang terkait bagaimana pesan diproses termasuk routing pengiriman, otentikasi, otorisasi dan body berisi pesan yang dikirim dan diproses. Kemudian *request* yang sudah di envelope dari aplikasi pengguna akan di teruskan oleh aplikasi pengguna melalui protocol http ke aplikasi *web service*. Aplikasi *web service* melakukan ektraksi paket yang telah dikirimkan oleh aplikasi pengguna. Jika data yang di minta di sediakan oleh *web service* maka *web service* akan mengambil data dari database. Kemudian data di *envole* lagi oleh SOAP dan di kirimkan ke aplikasi pengguna oleh SOAP *response*. Ketika sampai ke aplikasi pengguna data akan di ekstrak oleh aplikasi pengguna untuk tampilkan dalam bentuk antarmuka dalam bentuk web ke pengguna.

3.3.2.1 *Use Case Diagram*

Use Case diagram merupakan salah satu model UML yang mendeskripsikan kebutuhan fungsional sistem dari perspektif *end-user* dan

menunjukkan aksi–aksi yang dapat dilakukan oleh pengguna. Gambar 3.7 merupakan *use case* diagram sistem web *monitoring access point*.



Gambar 3. 7 Use Case Diagram

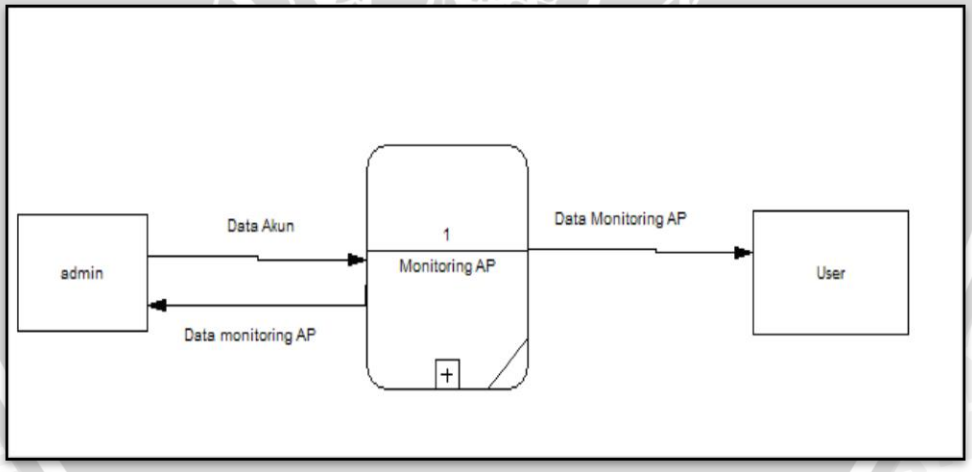
Aktor yang terlibat dalam sistem ada dua yaitu pengguna dan admin. Pengguna merupakan aktor yang memiliki aksi untuk mendapatkan informasi lokasi *access point*, jumlah pengguna yang terhubung ke *access point secara real time*, dan grafik jumlah pengguna berdasarkan hari dan bulan. Admin merupakan aktor yang memiliki aksi yang sama dengan pengguna dan mempunyai hak akses tambahan terhadap aplikasi yaitu dapat mengubah lokasi *access point* yang di map dengan melakukan login terlebih dahulu. Admin keluar dari halaman admin dengan menggunakan log out.

3.3.2.2 Data Flow Diagram (DFD)

DFD (*Data Flow Diagram)* menggambarkan proses keseluruhan sistem yaitu terkait proses aliran data dan interaksi yang terjadi antara pengguna dan sistem. Terdapat tiga level pemodelan *Data Flow Diagram* (DFD) yang akan digunakan dalam proses aliran data sistem sistem monitoring access point, yaitu: DFD level 0 (Konteks Diagram), DFD level 1, dan DFD level 2. Ditunjukkan seperti gambar berikut :

1. DFD Level 0

   Diagram level 0 atau diagram konteks akan didekomposisi menjadi beberapa sub proses yang terdapat pada diagram level 1 seperti gambar 3.8. Sub proses yang terdapat dalam diagram level 1 yaitu login dan informasi *access point*. Diagram level 1 juga memiliki beberapa data *store* yang berfungsi untuk memodelkan data-data yang tersimpan dalam *database* sistem. Data *store* yang terdapat dalam diagram level 1 diantaranya: data *users*, data *access point*, data *users daily* dan data admin.

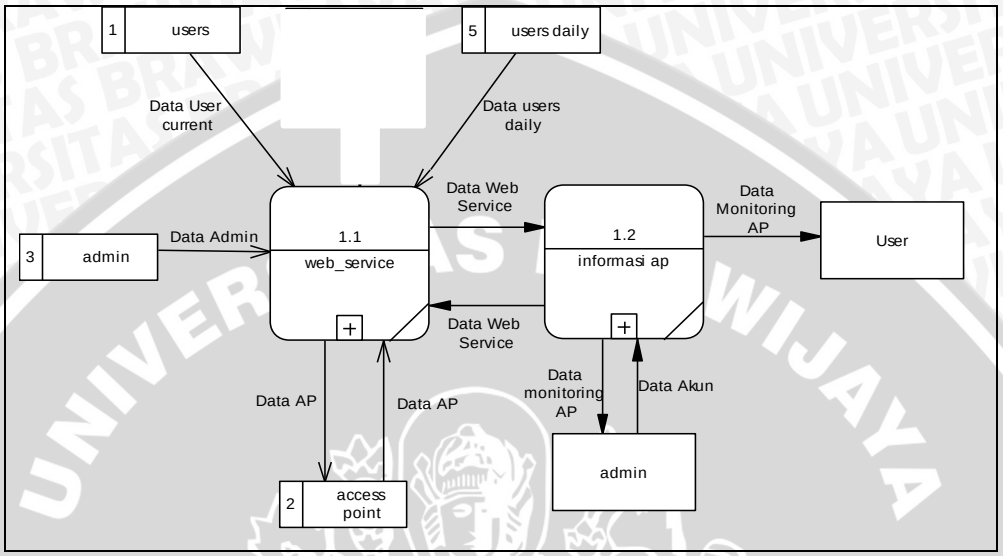


Gambar 3. 8 DFD Level 0

2. DFD Level 1

   DFD level 1 seperti gambar 3.9 terdapat dua proses yaitu *web service* dan informasi *access point*. Proses web service mempunyai informasi mengenai data admin, access point, users daily dan data user. Proses informasi *access point* berinteraksi langsung dengan dua actor yaitu

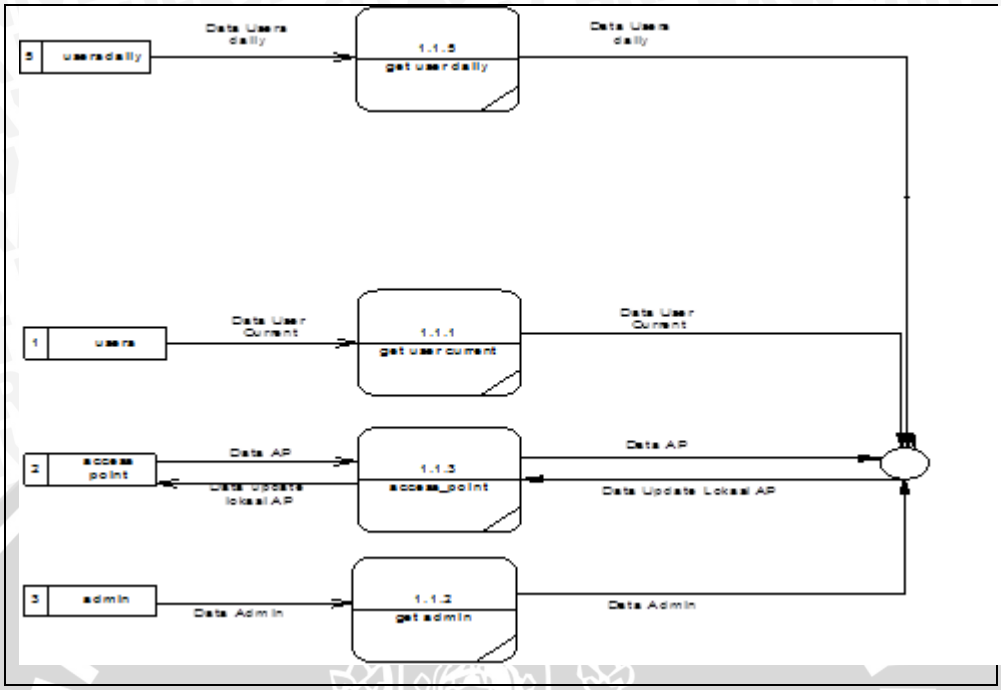
admin dan user. Proses informasi *access point* mempunyai informasi mengenai lokasi *access point*, jumlah pengguna yang terhubung di *access point* berdasarkan waktu, dan lokasi *access point* yang digunakan pengguna.



Gambar 3. 9 DFD Level 1

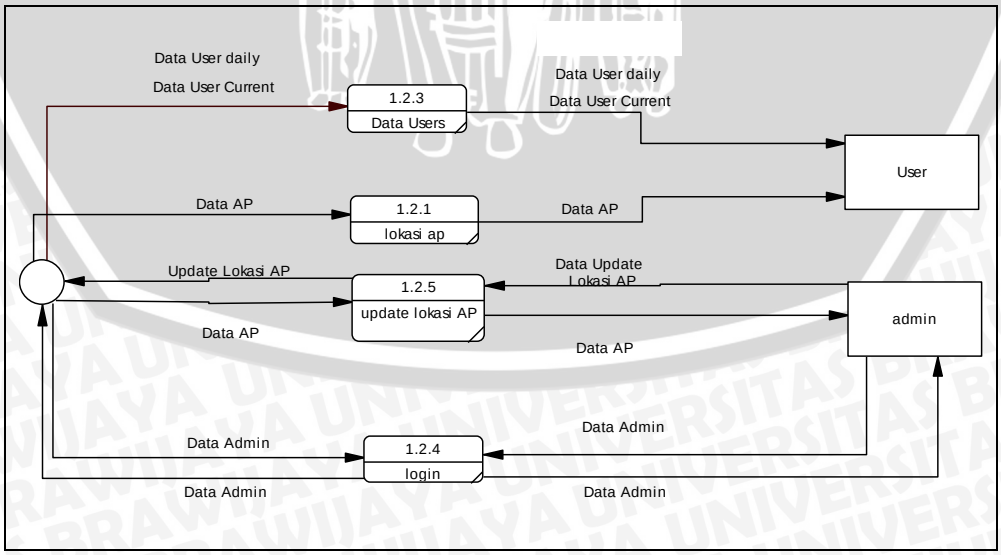
3. DFD level 2

   DFD level 2 yang merupakan aliran proses dari sistem *web service* seperti yang di tunjukkan gambar 3.10 diantaranya get user current , get user daily, get access point, dan get admin. Proses get user current digunakan untuk mengambil data tentang jumlah pengguna yang terhubung ke *access point* berdasarkan waktu sekarang. Proses get user daily digunakan untuk mengambil data jumlah pengguna yang terhubung ke *access point* berdasarkan hari dan bulan. Proses get access point digunakan untuk mengambil data tentang *access point* baik berupa nama, MAC *addres,* dan lokasi *access point* yang di petakan di map. Proses get admin digunakan untuk mengambil data admin yang nantinya digunakan login ke aplikasi pengguna untuk proses *update* lokasi dari *access point* di map.



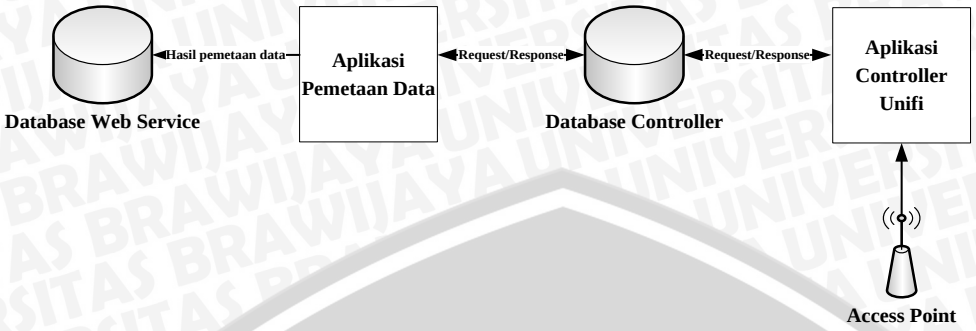
Gambar 3. 10 DFD Level 2 Web Service

DFD level 2 berupa proses informasi sistem di tunjukkan seperti gambar 3.11. Proses data user akan memberikan informasi kepada pengguna tentang jumlah pengguna yang terhubung ke access point. Proses lokasi AP memberikan informasi lokasi *access point* yang telah di petakan di map. Proses *update* lokasi di lakukan oleh aktor admin untuk update lokasi *access point* dengan melakukan drag ke map.



Gambar 3. 11 DFD Level 2 Aplikasi Pengguna

3.3.2.3 Perancangan Proses Aplikasi Pemetaan Data



Gambar 3. 12 Proses Pemetaan Data

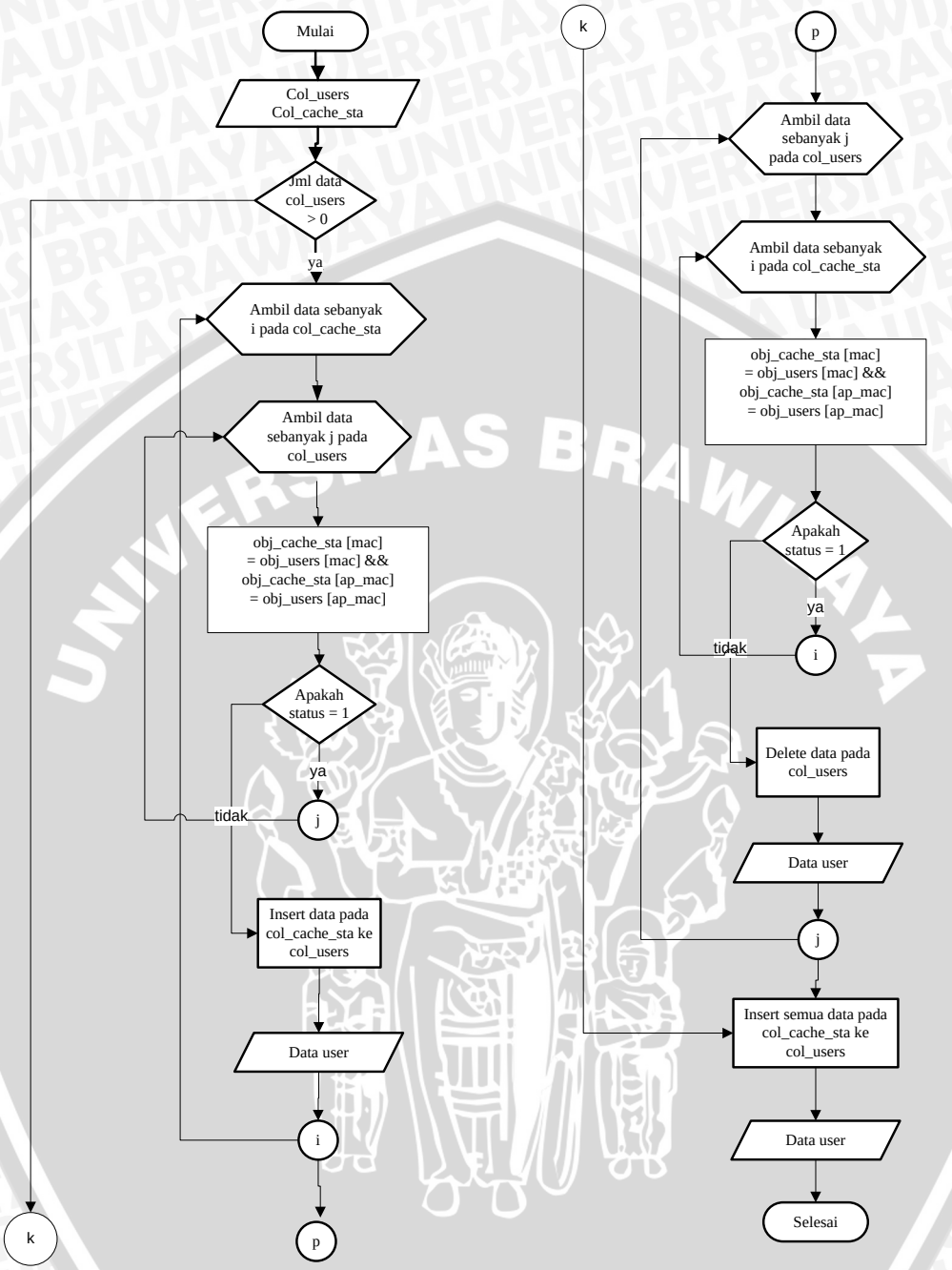
Gambar 3.12 merupakan proses pemetaan data dari database controller ke database *web service*. Diawali dari seluruh aktifitas *access point* di simpan di database *controller*. Kemudian aplikasi pemetaan data melakukan *request* data yang di inginkan ke database controller. Kemudian database controller akan memberikan *response* ke aplikasi pemetaan data sejumlah data yang di minta. Terakhir aplikasi pemetaan data akan menyimpan data dari database controller ke database *web service*. Berikut ini merupakan perancangan data yang dipetakan dari database controller ke database *web service* oleh aplikasi pemetaan data.

1. Pemetaan data Users/Pengguna

   Pemetaan data dilakukan dari database controller pada *collection cache_sta* kemudian hasil pemetaan disimpan ke database *web service* pada *collection users*. Tabel 3.5 merupakan objek data hasil dari pemetaan yang di simpan di *collection users*.

ObjectId		
Id	Mac	ap_mac

Tabel 3. 5 Collection Users



Gambar 3. 13 Algoritma Pemetaan Data users

Algoritma pemetaan data pengguna dapat di lihat pada gambar 3.13. Pemetaan data dimulai dari aplikasi akan melakukuan pengecekan data pada *collection users*. Jika data *collection users* bernilai kosong maka semua data pada collection *cache sta* akan di *insertkan* semua ke *collection users*. Akan tetapi jika terdapat data pada *collection users* akan di lakukan pengecekan.

Pengecekan pertama dilakukan untuk mengecek ketersediaan data yang ada pada *collection users* akibat dari penambahan data pada *collection cache_sta*. Dengan cara membandingkan objek data *mac address* pengguna dan *mac address access point* secara berulang ulang pada *collection chace_sta* dan *collection user*. Jika data yang ada pada kedua *collection* sama maka proses akan berhenti, akan tetapi jika data pada *collection users* tidak sama dengan *collection cache_sta.* Maka akan di lakukan *insert* data dari *collection cache_sta* ke *collection users* sejumlah data yang tidak ada pada *collection users*.

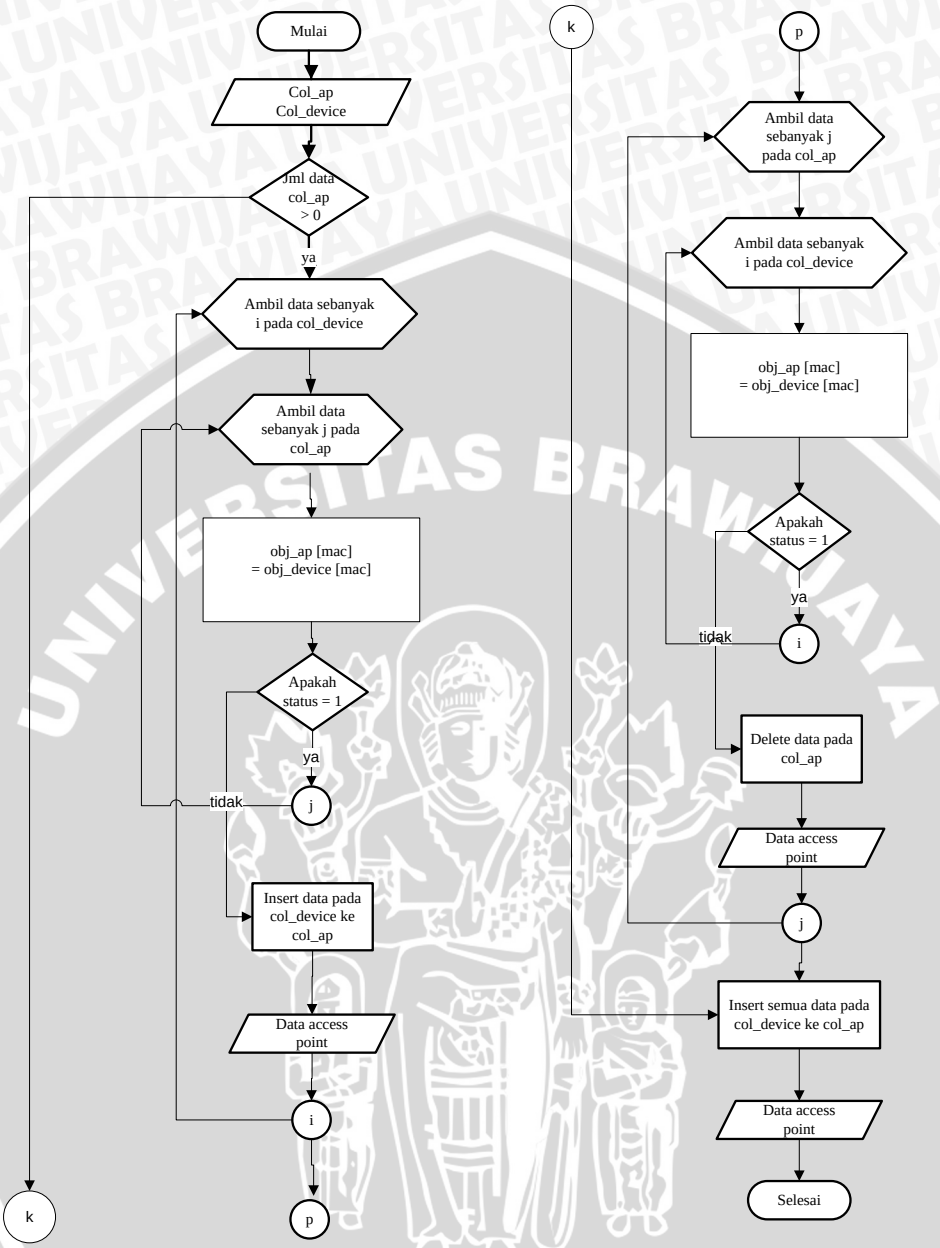
Pengecekan kedua dilakukan untuk mengecek ketersediaan data yang ada pada *collection users* akibat dari terhapusnya data pada *collection_cache_sta* dengan cara membandingkan objek data *mac address* pengguna dan *mac address access point* secara berulang ulang pada *collection chace_sta* dan *collection user*. Jika data yang ada pada kedua *collection* sama maka proses akan berhenti, akan tetapi jika data pada *collection users* tidak sama dengan *collection cache_sta* dimana ada pengurangan atau terhapusnya data pada *collection cache_sta* maka akan di lakukan *remove* data pada *collection users*. Sehingga algoritma pemetaan ini dilakukan agar data pada *collection cache_sta* sama dengan data pada *collection users.*

2. Pemetaan data *Access Point*

Pemetaan data dilakukan dari database *controller* pada *collection device* ke database *web service* pada *collection access point.* Table 3.6 merupakan objek data hasil dari pemetaan yang di simpan di collection access point.

ObjectId				
Id	ap_mac	Name	x	Y

Tabel 3. 6 *Collection access point*



Gambar 3. 14 Algoritma Pemetaan Data Access Point

Algoritma pemetaan data *access point* dapat di lihat pada gambar 3.14. Pemetaan data dimulai dari aplikasi akan melakukuan pengecekan data pada *collection access_point*, jika data pada *collection access_point* bernilai kosong maka semua data pada *collection device* akan di *insertkan* semua ke *collection access point*. Akan tetapi jika terdapat data pada *collection access point* akan di lakukan pengecekan.

Pengecekan pertama dilakukan untuk mengecek ketersediaan data yang ada pada *collection access point* akibat dari penambahan data pada *collection device*, dengan cara membandingkan *mac address access point* secara berulang ulang pada *collection device* dan *collection access point*. Jika data yang ada pada kedua *collection* sama maka proses akan berhenti, akan tetapi jika data pada *collection access point* tidak sama dengan *collection device* dimana ada penambahan data pada *collection device* maka akan di lakukan *insert* data dari *collection device* ke *collection access point* sejumlah data yang tidak ada pada *collection access point*.

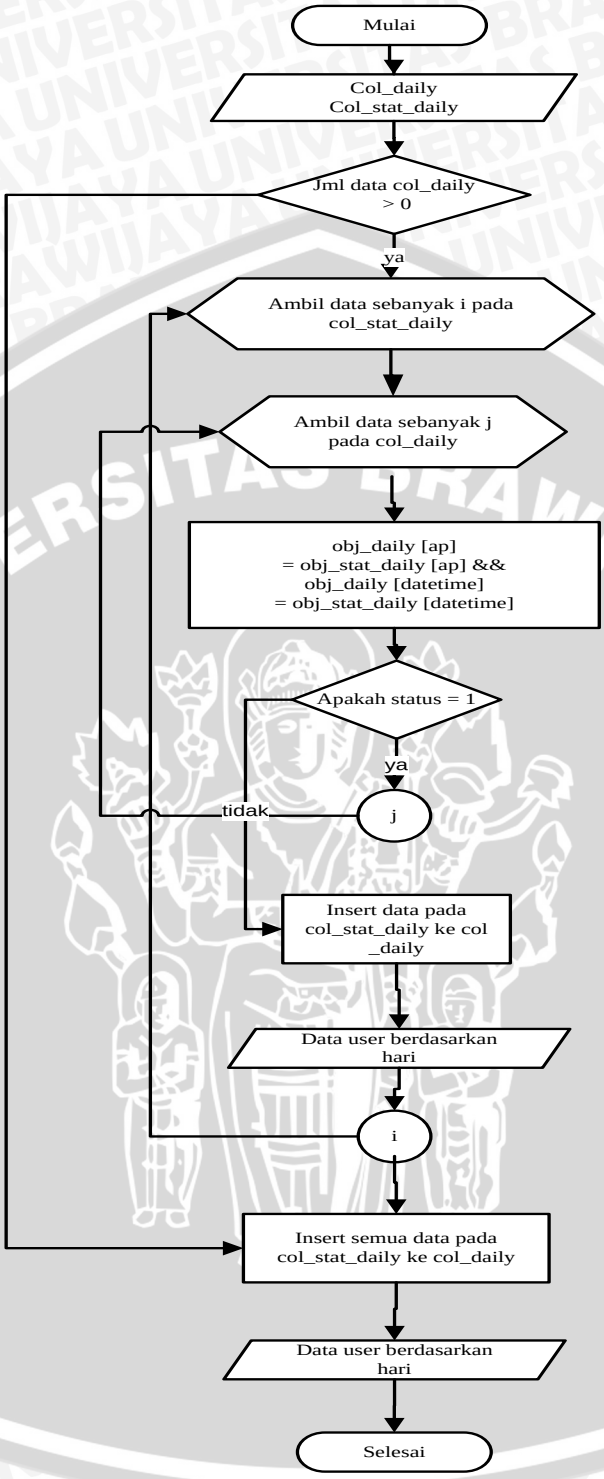
Pengecekan kedua dilakukan untuk mengecek ketersediaan data yang ada pada *collection access point* akibat dari berkurangnya data pada *collection device* dengan cara membandingkan *mac address access point* secara berulang ulang pada *collection device* dan *collection access point*. Jika data yang ada pada kedua *collection* sama maka proses akan berhenti, akan tetapi jika data pada *collection access point* tidak sama dengan *collection device* dimana ada pengurangan atau terhapusnya data pada *collection device* maka akan di lakukan *remove* data pada *collection access point*. Sehingga algoritma pemetaan ini dilakukan agar jumlah data dan nilai data pada *collection device* dan *collection access point* selalu sama.

3. Pemetaan data Jumlah Pengguna berdasarkan Hari

Hasil pemetaan data dari database controller pada *collection stat_daily* ke database web service pada *collection users_daily*. Pada data ini berisikan informasi jumlah pengguna yang terhubung ke sebuah access point berdasarkan waktu yaitu hari. Table 3.7 merupakan objek data hasil dari pemetaan yang di simpan di *collection users daily*.

ObjectId			
id	ap_mac	Datetime	Jml

Tabel 3. 7 Collection users daily



Gambar 3. 15 Algoritma Pemetaan Data Pengguna Setiap Hari

Hasil pemetaan data berisikan informasi jumlah pengguna berdasarkan hari yaitu MAC *address access point*, waktu, dan jumlah pengguna. Algoritma pemetaan data dapat di lihat pada gambar 3.15.

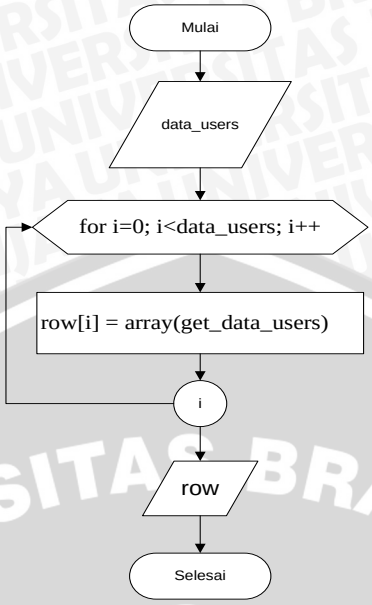
Pemetaan data dimulai dari aplikasi akan melakukuan pengecekan data pada *collection users_daily*, jika data pada *collection users_ daily* bernilai kosong maka semua data pada *collection stat_ daily* akan di tambahkan semua ke *collection users_ daily*. Akan tetapi jika terdapat data pada *collection users_ daily* akan di lakukan pengecekan.

Pengecekan dilakukan untuk mengecek ketersediaan data yang ada pada *collection users_ daily* akibat dari penambahan data pada *collection stat_ daily*, dengan cara membandingkan *mac address access point* dan *datetime* secara berulang ulang pada *collection users_ daily* dan *collection stat_ daily*. Jika data yang ada pada kedua *collection* sama maka proses akan berhenti, akan tetapi jika data pada *collection users_ daily* tidak sama dengan *collection stat_ daily* dimana ada penambahan data pada *collection stat_ daily* maka akan di lakukan penambahan data dari *collection stat_ daily* ke *collection users_ daily* sejumlah data baru yang ada pada *collection stat_ daily*

3.3.2.4 Perancangan Proses Aplikasi Web Service

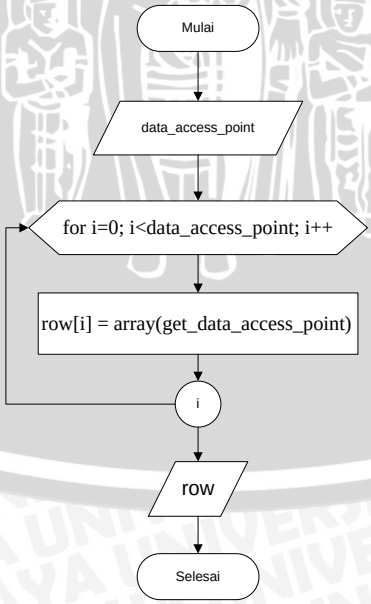
Aplikasi web service digunakan untuk memberikan layanan/fungsi ke aplikasi pengguna dengan menggunakan konsep SOAP. Fungsi akan memproses sejumlah data yang nantinya akan di berikan oleh pengguna. Fungsi yang di berikan oleh aplikasi *web service* ke pengguna antara lain fungsi users, fungsi access point, fungsi daily, dan fungsi admin. Algoritma proses pemanggilan fungsi dari aplikasi web service adalah sebagai berikut.

Gambar 3.16 merupakan diagram alur proses yang akan dilakukan oleh fungsi users, dimana ketika pengguna memanggil fungsi users akan dilakukan proses perulangan sejumlah data users. Kemudian data akan di kirimkan ke aplikasi pengguna sejumlah data users berupa mac address , IP users, dan MAC address access point.



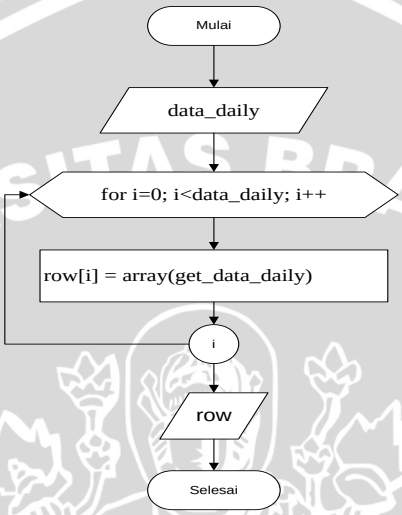
Gambar 3. 16 Alur Proses Fungsi Users

Gambar 3.17 merupakan diagram alur proses yang akan dilakukan oleh fungsi access point, dimana ketika pengguna memanggil fungsi access point akan dilakukan proses perulangan sejumlah data access point. Kemudian data akan di kirimkan ke aplikasi pengguna sejumlah data access point berupa MAC address access point, nama access point, latitude, dan longtitude.



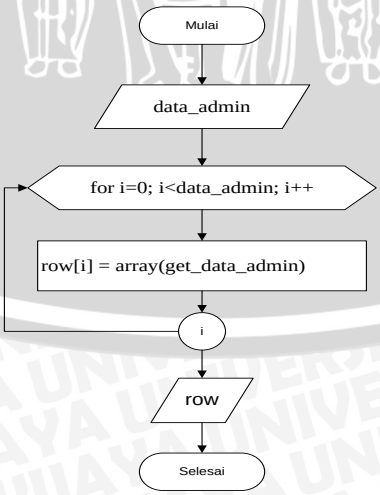
Gambar 3. 17 Alur Proses Fungsi Access Point

Gambar 3.18 merupakan diagram alur proses yang akan dilakukan oleh fungsi daily, dimana ketika pengguna memanggil fungsi daily akan dilakukan proses perulangan sejumlah data daily. Kemudian data akan di kirimkan ke aplikasi pengguna sejumlah data daily berupa mac address access point , waktu, dan jumlah pengguna yang terhubung ke access point berdasarkan jam.



Gambar 3. 18 Alur Proses Fungsi *Daily*

Gambar 3.19 merupakan diagram alur proses yang akan dilakukan oleh fungsi admin, dimana ketika pengguna memanggil fungsi admin akan dilakukan proses perulangan sejumlah data admin. Kemudian data akan di kirimkan ke aplikasi pengguna sejumlah data admin berupa username dan password.

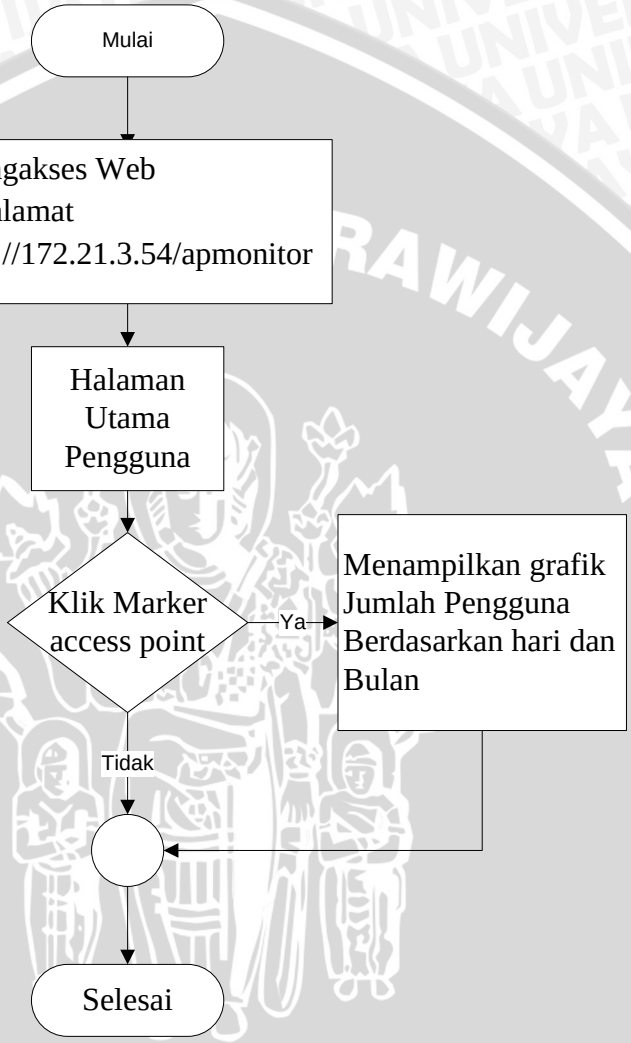


Gambar 3. 19 Alur Proses Fungsi Admin

3.3.2.5 Perancangan Proses Aplikasi Pengguna

Proses kerja aplikasi pengguna dibagi menadi 2 yaitu di sisi pengguna dan admin. Berikut rancangan dari aplikasi penguna yang akan dibuat :

1. Disisi Pengguna

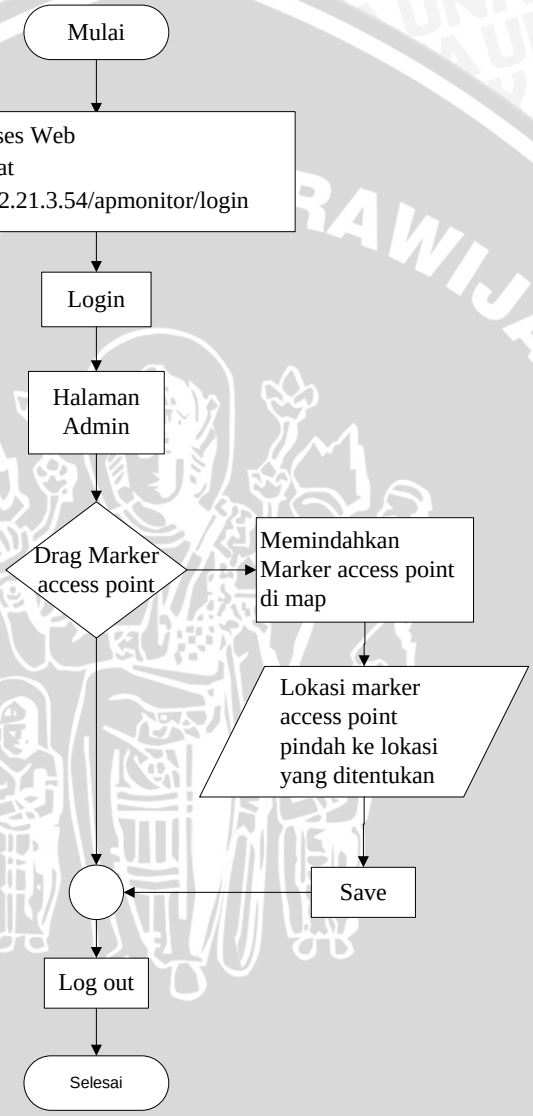


Gambar 3. 20 Flowchart Aplikasi Pengguna disisi pengguna

Gambar 3.20 merupakan proses kerja aplikasi pengguna disisi pengguna (mahasiswa, dosen, karyawan). Diawali dari pengguna membuka halaman web http://172.21.3.54/apmonitor menggunakan web browser, sistem akan menampilkan halaman utama aplikasi penguna. Halaman utama menampilkan informasi lokasi *access point* yang di petakan di map, jumlah pengguna yang terhubung pada sebuah

access point secara *real time*, dan *access point* yang di gunakan oleh pengguna. Pengguna juga bisa melihat informasi mengenai jumlah pengguna yang terhubung ke *access point* berdasarkan waktu yaitu hari dan bulan dengan cara melakukan klik pada marker *access point*.

2. Disisi Admin



Gambar 3. 21 Flowchart Aplikasi Pengguna Disisi Admin

Tujuan dibuat halaman admin karena data koordinat yang di dapatkan dari hasil pemetaan data merupakan koordinat static map bukan koordinat dari google map. Sehingga perlu menentukan koordinat google map pada masing-masing access point agar lokasi

access point bisa di tampilkan pada sistem. Agar admin tidak melakukan update koordinat di database web service secara manual pada setiap access point. Hal ini juga akan membutuhkan waktu yang lama maka diperlukan halaman admin. Halaman admin akan memudahkan admin menentukan lokasi *access point* di map dengan menggunakan *user interface* di aplikasi. Admin bisa menentukan lokasi *access point* di map dengan hanya melakukan drag *access point*. Dengan adanya fitur ini admin akan lebih dimudahkan dalam mengupdate lokasi *access point* di map.

Gambar 3.21 merupakan alur proses dari aplikasi pengguna di sisi admin. Diawali dari admin membuka halaman web http://172.21.3.54/apmonitor/login menggunakan web browser, sistem akan menampilkan form login. Admin diminta memasukkan username dan password yang telah disimpan di collection admin. Jika username dan password yang dimasukkan oleh admin benar maka akan tampil halaman admin, jika salah maka admin diminta untuk memasukkan username dan password kembali. Halaman admin menampilkan informasi lokasi access point yang telah di petakan di map. Admin bisa mengubah lokasi dari access point dengan melakukan drag pada marker access point. Saat di lakukan drag marker akan muncul form save untuk menyimpan lokasi access point. Jika admin menekan tombol save maka lokasi access point akan berubah di marker sesuai dengan lokasi yang di tentukan admin dan data akan simpan di collection access point pada database *web service*.

3.4 Implementasi

Tahapan Implementasi dimulai dengan konfigurasi dan instalasi perangkat lunak pada *server*. Tahap selanjutnya membuat aplikasi untuk memetakan data dari database *controller* ke database *web service* dimana database yang digunakan adalah MongoDB. Ketika database *web service* hasil dari pemetaan dari database controller sudah tersedia, tahap selanjutnya membuat layanan web service dengan *simple object access protocol* (SOAP) dan *web service description language*

(WSDL). Tahap terakhir membuat aplikasi pengguna dengan menerapkan MVC (model, view, controller) yang nantinya akan berinteraksi dengan *web service* dalam pemrosesan data.

3.5 Pengujian Sistem

Tahapan ini dilakukan untuk mengetahui bahwa sistem yang dibuat telah mampu bekerja sesuai dengan spesifikasi dari kebutuhan yang di inginkan. skenario pengujian yang dilakukan adalah sebagai berikut :

1. Pengujian Aplikasi Pemetaan Data

   Pengujian aplikasi pemetaan data dilakukan dengan melakukan pengujian *response time* yang di hasilkan oleh sistem dengan menggunakan Jmeter. Berdasarkan hasil *request time* untuk menentukan penjadwalan aplikasi dengan menggunakan bantuan daemon *cron*. *Cron* merupakan daemon yang di miliki oleh linux untuk menentukan penjadwalan secara berkala terhadap aplikasi. Dengan menggunakan cron aplikasi pemetaan data akan diproses secara otomatis berdasarkan jadwal yang telah di tentukan. Format dasar jadwal crontab terdiri dari 6 kolom, ditempatkan pada satu baris dan dipisahkan dengan spasi, formatnya adalah seperti `minute hour day month day-of-week command-line-to-execute`. Berikut ini merupakan keterangan dari masing-masing kolom dalam penulisan cron.

Field	Range Nilai
Minute	0-59
Hour	0-23
Day	1-31
Month	1-12
Day of week	0-7
command-line-to-execute	Perintah untuk menjalankan aplikasi atau letak file yang akan di eksekusi

Tabel 3. 8 Keterangan Perintah Cron

Sumber : [HEN-14]

2. Pengujian Web Service

   Pengujian web service dilakukan dengan dua hal yaitu fungsionalitas dan peformansi web service. Pengujian fungsionalitas web service menggunakan aplikasi SOAPUI yang merupakan aplikasi *open source* dan berjalan pada berbagai macam platform untuk mengintegrasikan semua fungsi secara otomatis. Tujuan pengujian ini untuk mengetahui response data yang di dapatkan dari request pada masing-masing fungsi. Pengujian peformansi web service dengan menggunakan Jmeter untuk dilakukan tes terhadap kekuatannya atau analisa performa web service.

3. Pengujian Peformansi Sistem

   Pengujian peformansi sistem dengan menggunakan aplikasi Jmeter. Jmeter merupakan sebuah aplikasi desktop yang bersifat open source digunakan untuk melakukan tes perilaku fungsional dan pengukuran performa pada aplikasi web. JMeter dapat digunakan untuk melakukan sebuah simulasi heavy load pada sebuah server, jaringan atau objek untuk dilakukan tes terhadap kekuatannya atau analisa performa dengan beberapa tipe load yang berbeda [JME-13].

4. Pengujian Validasi Pengguna

   Pengujian validasi Pengguna dilakukan dengan melakukan pengamatan langsung pada access point di gedung PPTI. Dimana akan di buat skenario uji dengan melakukan koneksi sejumlah device ke access point yang berada di gedung PPTI. Hal yang diamati antara lain jumlah pengguna yang konek ke access point yang berada di gedung PPTI dan lokasi access point yang di jadikan koneksi oleh pengguna untuk di bandingkan output yang di tampilkan oleh sistem yang di buat.

3.6 Analisa Hasil

Berdasarkan data yang diperoleh dari hasil pengujian, kemudian dilakukan analisa untuk mengetahui hasil yang akan digunakan untuk menarik kesimpulan dari penelitian yang dilakukan. Analisa yang dilakukan berupa data dari hasil pengujian pemetaan data, pengujian peforma sistem, pengujian fungsi web service, dan pengujian validasi data.

3.7 Kesimpulan dan saran

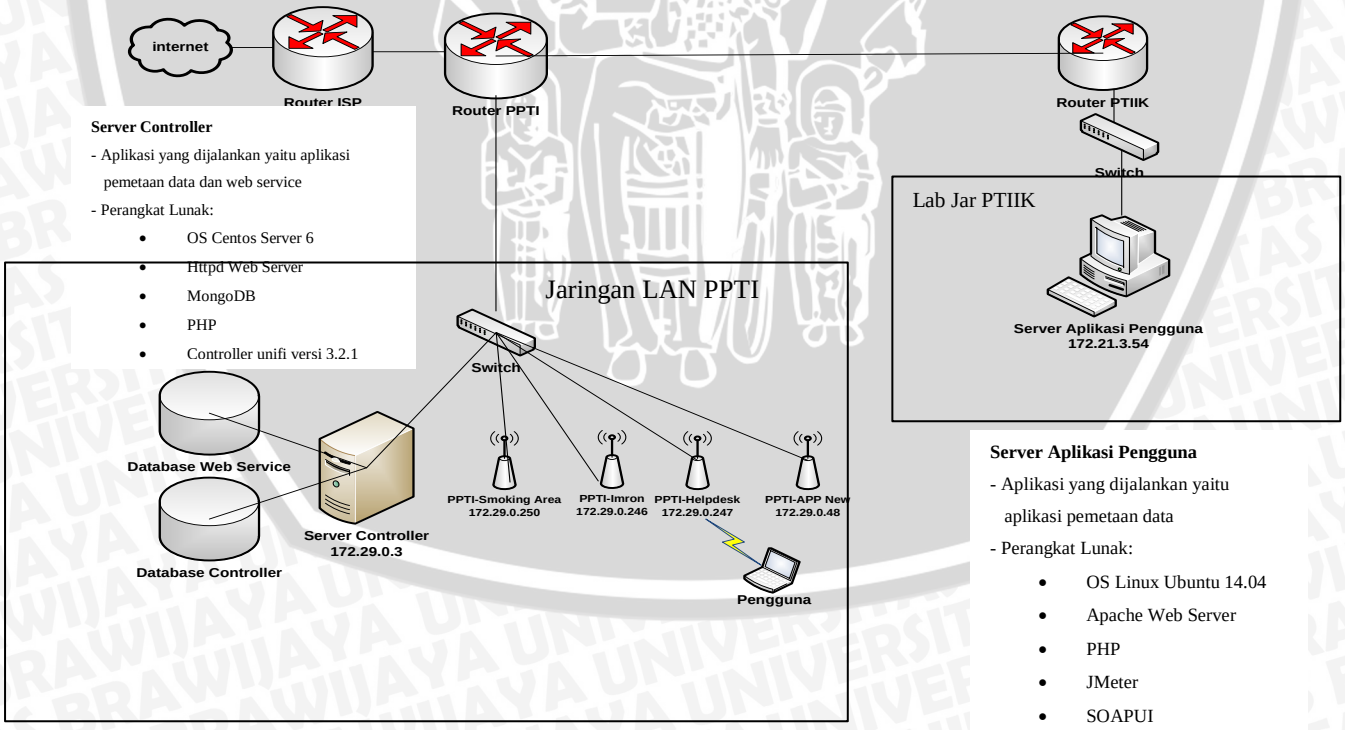
Kesimpulan dari penelitian ini diambil ketika semua tahapan penelitian telah selesai dilakukan. Penarikan kesimpulan didapatkan dari hasil pengujian beserta analisa hasil. Tahap terakhir dari penelitian adalah saran yang berisikan hal yang digunakan untuk mengembangkan penelitian maupun perbaikan.

BAB IV
IMPLEMENTASI

Bab ini menjelaskan implementasi dari perancangan yang telah di buat. Implementasi dimulai dari konfigurasi dan instalasi perangkat lunak pada *server*, kemudian pembuatan aplikasi. Aplikasi yang di buat antara lain aplikasi pemetaan data, aplikasi *web service* dan aplikasi pengguna.

4.1 Implementasi Topologi Jaringan

Dalam tahap implementasi jaringan, hanya menambahkan 2 buah server dari topologi yang sudah ada di Universitas Brawijaya. Server tersebuat antara lain server controller dengan IP 172.29.0.3 di install pada jaringan PPTI dan server aplikasi pengguna IP 172.21.3.54 di install pada laboratorium jaringan PTIIK. Topologi yang di buat seperti yang telah dijelaskan pada tahap perancangan jaringan. Gambar 4.1 merupakan implementasi topologi jaringan yang digunakan pada penelitian ini. Spesifikasi server yang digunakan pada penelitian ini adalah sebagai berikut:



Gambar 4. 1 Implementasi Topologi Jaringan

4.1.1 Server Controller

Server Controller berada pada jaringan LAN PPTI sebagaimana terlihat pada perancangan Gambar 3.2. *Server controller* digunakan sebagai *server* dari *controller* unifi, aplikasi pemetaan data, dan aplikasi *web service*. Spesifikasinya *sever controller* adalah sebagai berikut :

Sistem Operasi : Centos Server 6.0
Interface : eth0.2000
Memory : 2 GB
Alamat IP : 172.29.0.3
Subnet Mask : 255.255.255.0

Perangkat lunak yang digunakan oleh *server controller* antara lain *apache web server*, *controller* unifi, mongodb, dan modul PHP. *Apache web server* di gunakan sebagai server web untuk memberikan layanan web ke pengguna. Sedangakan *controller* unifi digunakan sebagai *controller* dari *access point* untuk memonitoring aktifitas access point melalui web. Mongodb digunakan sebagai database dari *controller* unifi dan database *web service*. Terakhir modul PHP digunakan sebagai bahasa pemrogaman yang digunakan untuk membuat aplikasi pemetaan data dan aplikasi *web service*.

4.1.2 Server Aplikasi Pengguna

Server aplikasi pengguna berada dalam satu unit PC yang berada pada jaringan PTIIK. Server aplikasi pengguna digunakan sebagai server dari aplikasi pengguna yaitu aplikasi yang berinteraksi langsung dengan pengguna.
Spesifikasinya adalah sebagai berikut :

Prosesor : Intel® Core™ i3
Harddisk : 500 GB
Memory : 4 GB
Sistem Operasi : Ubuntu 14.04
Interface : eth1
Alamat IP : 172.21.3.54
Subnet Mask : 255.255.255.192

Perangkat lunak yang di butuhkan oleh server pengguna antara lain *apache web server* dan modul PHP. Apache web server di gunakan sebagai server web untuk memberikan layanan web ke pengguna. Sedangkan modul PHP sebagai bahasa pemrogaman yang digunakan untuk membuat aplikasi pengguna.

4.2 Instalasi dan Konfigurasi *Server*

4.2.1 Konfigurasi IP dan Cek Koneksi

Konfigurasi IP pada masing-masing *server* dilakukan pada *interface ethernet card*. Pada *server controller* konfigurasi IP dilakukan pada interface eth0.2000 yang terhubung pada jaringan PPTI. Sedangkan pada *server aplikasi pengguna* konfigurasi IP dilakukan pada eth1 yang terhubung dengan jaringan PTIIK. Konfigurasi yang dilakukan meliputi alamat *ip, netmask, network, broadcast*, dan *dns server*. DNS yang digunakan adalah alamat DNS Universitas Brawijaya dengan tujuan agar masing-masing server bisa saling terhubung. Gambar 4.2 dan 4.3 merupakan hasil konfigurasi IP dari *server controller* dan *server aplikasi pengguna*:

1. *Server controller*

```
eth0.2000 Link encap:Ethernet  HWaddr 52:54:00:9A:35:02
          inet addr:172.29.0.3  Bcast:172.29.0.255  Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe9a:3502/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7541399 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2148294 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1545997536 (1.4 GiB)  TX bytes:332158177 (316.7 MiB)
```

Gambar 4. 2 Konfigurasi IP Server Controller

2. *Server Aplikasi Pengguna*

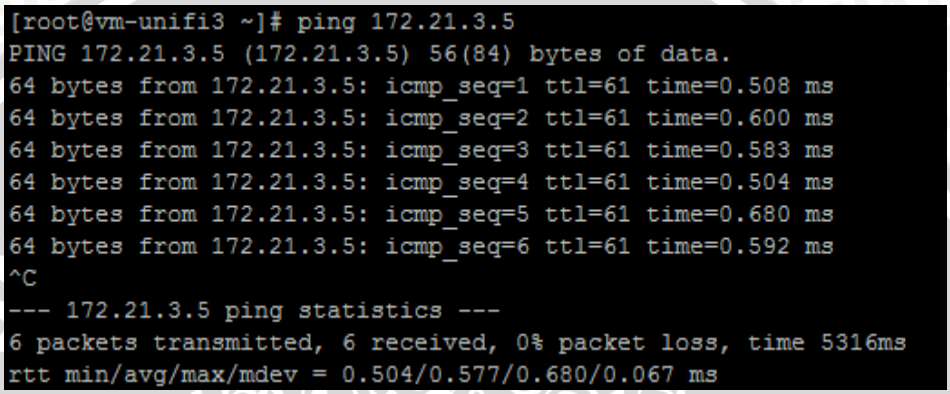
```
root@skripsweet:~# ifconfig
eth1      Link encap:Ethernet  HWaddr 10:78:d2:c6:fe:6e
          inet addr:172.21.3.5  Bcast:172.21.3.63  Mask:255.255.255.192
          inet6 addr: fe80::1278:d2ff:fec6:fe6e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:589 errors:0 dropped:0 overruns:0 frame:0
          TX packets:292 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:74426 (74.4 KB)  TX bytes:37675 (37.6 KB)
          Interrupt:20 Memory:fbec0000-fbee0000
```

Gambar 4. 3 Konfigurasi IP Server Aplikasi Pengguna

Setelah IP pada masing-masing server berhasil di konfigurasi, selanjutnya dilakukan tes koneksi untuk menunjukkan bahwa koneksi antara *server aplikasi pengguna* dan *server controller* sudah terhubung.

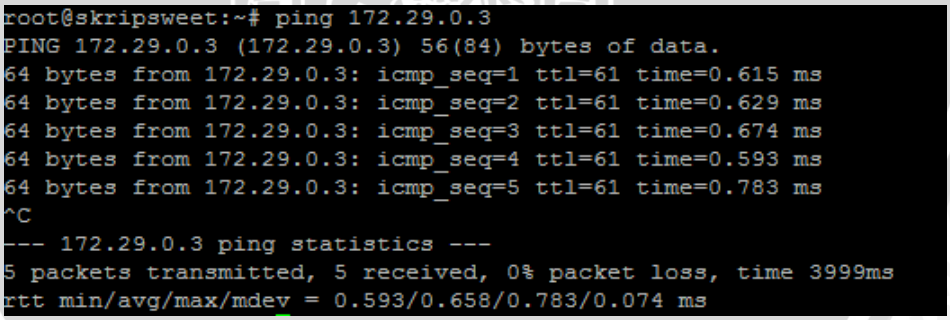
Tes Koneksi antara *server aplikasi pengguna* dengan *server controller* menggunakan perintah ping. Berikut adalah hasil dari perintah ping untuk menunjukkan koneksi antara *server aplikasi pengguna* dengan *server controller* yang berhasil dibangun. Dibuktikan dengan tidak adanya pesan *request time out*. Gambar 4.4 dan 4.5 merupakan hasil tes koneksi dari masing-masing *server*:

1. Tes koneksi dari *server controller* ke *server aplikasi pengguna*



Gambar 4. 4 Hasil tes koneksi dari server controller ke server aplikasi pengguna

2. Tes koneksi dari *server aplikasi pengguna* ke *server controller*



Gambar 4. 5 Hasil tes koneksi dari server aplikasi pengguna ke server pengguna

4.2.2 Instalasi dan Konfigurasi MongoDB

Mongodb hanya di install pada *server controller* dengan *centos server* versi 6.5 sebagai *operating system.* Lagkah-langkah instalasi seperti pada lampiran 1. Setelah instalasi berhasil, kemudian dilakukan konfigurasi agar

mongodb bisa berjalan pada web server dengan menambahkan `extension=mongo.so` pada direktori `/etc/php5/apache2/php.ini`.

4.2.3 Insalasi dan Konfigurasi *Controller Unifi*

Pada penelitian ini, *versi controller unifi* yang digunakan adalah versi terbaru yang stabil yaitu versi 3.2.1. Paket instalasi *controller unifi* dapat di download di `http://dl.ubnt.com/unifi/3.2.1/UniFi.unix.zip`. Setelah paket terdownload selanjutnya dilakukan ekstrasi paket tersebut pada direktori `/opt`.

Setelah berhasil di ekstraksi kemudian di buat service untuk menjalankan *controller unifi* . Service yang di buat untuk melakukan *start, stop,* dan *restart controller unifi* dengan membuat secure shell pada direktori `/etc/init.d/UniFi`. Langkah terakhir adalah membuka port yang di gunakan oleh *controller unifi* agar bisa di jalankan melalui *web browser*, Diawali dengan membuka folder *iptables* pada `/etc/sysconfig/iptables` dan menambahkan aturan pada iptables. Instalasi dan konfigurasi *controller unifi* sebagaimana yang terlampir pada lampiran 2.

4.3 Implementasi Aplikasi

Pada penelitiaan ini ada 3 aplikasi yang dirancang yaitu aplikasi pemetaan data, aplikasi *web service*, dan aplikasi pengguna. Berikut ini merupakan implementasi dari masing-masing aplikasi.

4.3.1 Implementasi Aplikasi Pemetaan Data

Aplikasi pemetaan data di lakukan untuk memetakan data dari *database controller* ke *database web service*. *Database controller* merupakan database yang digunakan oleh *controller Unifi* untuk menyimpan semua aktifitas dari *access point*. Sedangkan *database web service* merupakan database hasil pemetaan data dari *database controller*. *Database web service* digunakan sebagai database dari *aplikasi web service*. Pemetaan data dilakukan karena tidak semua data dari *database controller* di butuhkan dan hanya beberapa data saja yang dibutuhkan oleh *web service*. Selain itu juga faktor keamanan dikarenakan adanya data yang seharusnya tidak di tunjukkan ke pengguna. Data yang yang dipetakan

dari *database server* ke *database web service* seperti halnya terlihat pada perancangan data.

Secara garis besar ada 3 *collection* dari *database controller* yang dilakukan pemetaan data ke *database web service*. *Collection* tersebut antara lain *collection cache_sta, device, dan stat_daily*. Pemetaan data *collection cache_sta* dipetakan ke *collection users, collection device* ke *collection access_point,* dan *collection stat_daily* ke *collection daily*. Aplikasi pemetaan data sebagaimana yang terlampir pada lampiran 4. Secara garis besar ada 4 urutan proses pembuatan aplikasi pemetaan data, antara lain :

4.3.1.1 Membuat Koneksi ke database

Langkah awal dalam pembuatan aplikasi pemetaan data adalah membuat koneksi ke masing-masing database. Lampiran 4a merupakan *script* yang digunakan untuk melakukan koneksi ke *database controller.* Sedangkan lampiran 4b merupakan *script* digunakan untuk melakukan koneksi ke *database web service*. Perbedaan *script* masing-masing database terletak pada baris 2-4 yaitu host, port, dan nama database.

4.3.1.2 Menentukan *Collection*

Setelah koneksi masing-masing database berhasil di buat. Selanjutnya menentukan *collection* sumber pada *database controller* dan *collection* yang akan dijadikan tujuan pemetaan data pada *database web service*. Gambar 4.6 merupakan *method* untuk menentukan collection. Nilai yang dikeluarkan dari *method* ini sebanyak data yang ada pada *collection*. Keluaran dari *method* ini masih belum bisa dibaca secara langsung. Diperlukan sebuah proses untuk memecah keluaran dari *method* untuk menampilkan data.

```
$db->nama_collection->find()
```

Gambar 4. 6 Menentukan Collection

4.3.1.3 Proses Pemetaan Data

Setelah *collection* berhasil di deskripsikan, kemudian di buat proses untuk pemetaan data. Pemetaan data dilakukan pada *collection* sumber pada *database controller* dengan *collection* tujuan yang ada di *database web service*. Salah satu

proses pemetaan data seperti pada lampiran 4c. Proses pemetaan data di mulai dari mengecek jumlah data yang ada pada *collection* tujuan. Jika data *collection* tujuan bernilai 0 seperti baris 14-15 maka semua data di *collection* sumber akan di insertkan ke *collection* tujuan yang ditunjukkan pada baris 59-67.

Sedangkan jika data *collection* tujuan tidak bernilai 0 akan di lakukan dua proses. Proses pertama dilakukan untuk *insert* data ke *collection* tujuan sejumlah data baru yang ada pada *collection* sumber yang ditunjukkan di baris 18-33. Proses kedua dilakukan untuk *remove* data pada *collection* tujuan sejumlah data yang dihapus oleh *collection* sumber yang ditunjukkan di baris 37-54. Proses *remove* data tidak dilakukan terhadap pemetaan data dari *collection stat_daily* ke *collection daily*. Hal ini karena data di *collection daily* digunakan untuk membuat grafik estimasi jumlah pengguna berdasarkan hari dan bulan.

4.3.1.4 Penjadwalan Aplikasi Pemetaan Data

Penjadwalan aplikasi pemetaan data dilakukan agar aplikasi dapat di jalankan secari berkala oleh *system*. Penjadwalan aplikasi pemetaan data memanfaatkan *daemon cron* yang telah di sediakan oleh *system operasi linux*. Penjadwalan aplikasi pemetaan data ada 2 perlakuan, pertama untuk memetakan data dari *collection cache_sta* ke *collection users* dan *collection device* ke *collections access_point* dilakukan penjadwalan selama 1 menit. Dilakukan selama waktu tersebut dilihat dari lamanya proses dalam pengambilan data sehingga tidak terjadi proses yang saling tumpang tindih. Kedua untuk memetakan data dari *collections stat_daily* ke *collection users_daily* dilakukan penjadwalan selama 1 hari. Dilakukan selama waktu tersebut karena dilihat dari data pada *collections stat_daily* akan melakukan update data selama 1 hari.

Langkah dalam melakukan cron di linux adalah buka terminal dan ketik *crontab –e* kemudian tambahkan baris seperti gambar 4.7.

Gambar 4. 7 Penjadwalan Aplikasi Pemetaan Data

4.3.2 Implementasi Aplikasi *Web Service*

Implementasi aplikasi *web service* sebagaimana yang terlampir pada lampiran 5. Secara garis besar terdapat 4 proses yaitu *generate library*, menentukan *function*, dan membuat WSDL *(Web Service Description Language)*. Berikut adalah masing-masing proses tersebut.

4.3.2.1 Generate Library

Sebelum membuat aplikasi dilakukan *generate library* nusoap untuk bahasa pemrogaman yang di pakai dalam implementasi SOAP *web service*. Nusoap berisikan class-class yang menyediakan *method* yang memungkinkan pengguna untuk mengirim dan menerima pesan format xml melalui *protocol* http. *Generate library* nusoap dilakukan dengan cara menyalin semua file nusoap ke dalam lokasi aplikasi *web service*. Dengan memanfaatkan library tersebut, aplikasi SOAP dapat di buat dengan bahasa pemrogaman PHP.

4.3.2.2 Menentukan *Function*

Setelah library nusoap telah di *generate*, kemudian di buat function untuk mendefinisikan layanan data yang diberikan oleh *web service*. Function ini nanti akan di panggil oleh aplikasi pengguna untuk layanan data yang diinginkan. Diawalai dari mendefinisikan *host, port,* dan nama database seperti pada lampiran 5a baris 2-4. Kemudian membuat mebuat koneksi dengan berdasarkan *host, port*, dan nama database yang di definisikan seperti pada lampiran 5a baris 7-9. Selanjutnya membuat function atau layanan data yang diambil dari database *web service*. Function yang di buat antara lain function *users, access point, admin, daily*, dan posisi ap.

Keluaran dari function berupa layanan data yang ada pada *collection* yang sudah di tentukan, seperti pada lampiran 5a baris 11. Pada pembuatan function ini

hanya function posisi ap yang menggunakan parameter. Sedangkan function yang lain tidak menggunakan parameter dalam pemangggilan data. Artinya saat pengguna memanggil function ini semua *item* data pada *collection* yang telah di tentukan langsung di keluarkan seperti pada lampiran 5a baris 14-25. Keluaran data dari function ini masih belum bisa dibaca secara langsung. Diperlukan sebuah *method* dari aplikasi pengguna untuk memanggil function yang di definisikan berdasarkan parameter yang telah di tentukan.

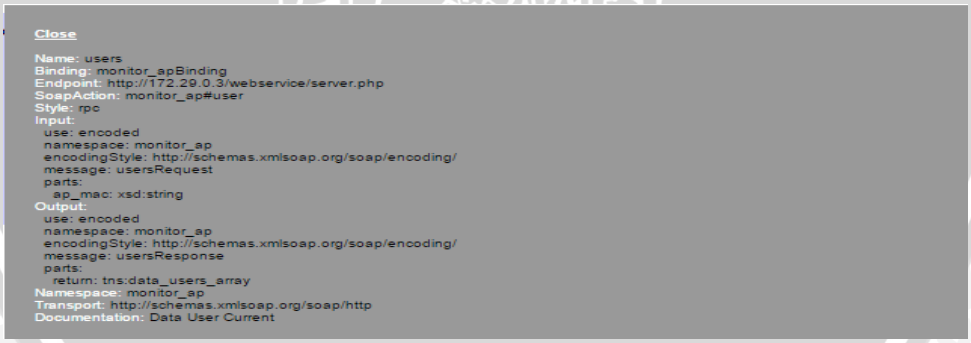
4.3.2.3 Membuat SOAP WSDL (Web Service Description Language)

Setelah layanan data sudah di definisiakan selanjutnya dilakukan pembuatan SOAP WSDL. WSDL digunakan untuk mendefinisikan semua *service, type data,* dan *detail pengoperasian* yang di berikan ke pengguna melalui protocol http. Pembuatan SOAP *web service* dengan WSDL bermanfaat ketika terjadi penambahan *service* yang baru atau perubahan detail pengoperasiannya pada web service. Dengan adanya WSDL pengguna dapat segera menyesuaikan dengan memanfaatkan data informasi dari WSDL.

Pembuatan SOAP WSDL sebagaimana yang dilampirkan pada lampiran 5b. Diawali dengan memanggil *library nusoap* dan layanan data yang di buat di lampiran 5a. Layanan data ini yang akan di deskripsikan secara detail pada WSDL. Pada lampiran 5b baris 4-9 digunakan untuk mendefinisikan SOAP WSDL. Setelah data berhasil di deskripsikan dengan WSDL, dapat di lihat hasil semua *service* yang di sediakan oleh *web service* dengan mengakses URL pada web browser yang mengacu pada file wsdl. Pada penelitian ini soap wsdl berada pada alamat *http://172.29.0.3/webservice/server.php* , jika tidak ada kesalahan dalam deskripsikan semua service pada WSDL maka akan tampil seperti gambar 4.8.

Gambar 4. 8 Aplikasi Web Service dengan WSDL

SOAP akan melakukan generate HTML untuk menampilkan semua detail pengoperasian *web service* yang diberi nama *monitor_ap*, sekaligus menampilkan semua *service* yang disediakan. Pada gambar 4.8 merupakan detail pengoperasian yang terdapat beberapa item *service* yang di berikan oleh *web service* yaitu *users*, *access point*, *admin*, *daily*, dan posisi ap. Untuk melihat detail informasi pada item service, klik link pada *service* maka akan terlihat seperti gambar 4.9.



Gambar 4. 9 Detail Operasi Pada item Service

Kemudian untuk menampilkan dokumen WSDL , klik link wsdl pada web service *monitor_ap* sehingga akan tampil seperti gambar 4.10. Pada gambar 4.10 merupakan isi dokumen dari WSDL yang di generate oleh SOAP yang memberikan detail informasi mengenai *service-service* yang telah di buat.

172.29.0.3/webservice/server.php?wsdl

This XML file does not appear to have any style information associated with it. The document tree is shown below.


```
<definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="http://localhost/mongodb/server.php" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://localhost/mongodb/server.php">
 <types>
  <xsd:schema targetNamespace="http://localhost/mongodb/server.php">
   <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/"/>
   <xsd:complexType name="data_users">
    <xsd:all>
     <xsd:element name="mac" type="xsd:string"/>
     <xsd:element name="ip" type="xsd:string"/>
     <xsd:element name="ap_mac" type="xsd:string"/>
    </xsd:all>
   </xsd:complexType>
   <xsd:complexType name="data_users_array">
    <xsd:complexContent>
     <xsd:restriction base="SOAP-ENC:Array">
      <xsd:attribute ref="SOAP-ENC:arrayType" wsdl:arrayType="tns:data_users[]"/>
     </xsd:restriction>
    </xsd:complexContent>
   </xsd:complexType>
   <xsd:complexType name="theAccessPoint">
    <xsd:all>
     <xsd:element name="mac" type="xsd:string"/>
     <xsd:element name="name" type="xsd:string"/>
     <xsd:element name="x" type="xsd:string"/>
     <xsd:element name="y" type="xsd:string"/>
    </xsd:all>
   </xsd:complexType>
   <xsd:complexType name="theAccessPointArray">
    <xsd:complexContent>
     <xsd:restriction base="SOAP-ENC:Array">
      <xsd:attribute ref="SOAP-ENC:arrayType" wsdl:arrayType="tns:theAccessPoint[]"/>
     </xsd:restriction>
    </xsd:complexContent>
   </xsd:complexType>
   <xsd:complexType name="data_admin">
    <xsd:all>
     <xsd:element name="name" type="xsd:string"/>
     <xsd:element name="x_password" type="xsd:string"/>
    </xsd:all>
   </xsd:complexType>
   <xsd:complexType name="data_admin_array">
    <xsd:complexContent>
     <xsd:restriction base="SOAP-ENC:Array">
      <xsd:attribute ref="SOAP-ENC:arrayType" wsdl:arrayType="tns:data_admin[]"/>
     </xsd:restriction>
    </xsd:complexContent>
   </xsd:complexType>
   <xsd:complexType name="data_users_daily">
    <xsd:all>
     <xsd:element name="ap" type="xsd:string"/>
     <xsd:element name="datetime" type="xsd:string"/>
     <xsd:element name="num_sta" type="xsd:string"/>
    </xsd:all>
   </xsd:complexType>
   <xsd:complexType name="data_users_daily_array">
    <xsd:complexContent>
     <xsd:restriction base="SOAP-ENC:Array">
      <xsd:attribute ref="SOAP-ENC:arrayType" wsdl:arrayType="tns:data_users_daily[]"/>
     </xsd:restriction>
    </xsd:complexContent>
   </xsd:complexType>
   <xsd:complexType name="posisi_ap">
    <xsd:all>
     <xsd:element name="ap_mac" type="xsd:string"/>
     <xsd:element name="x" type="xsd:string"/>
     <xsd:element name="y" type="xsd:string"/>
     <xsd:element name="name" type="xsd:string"/>
    </xsd:all>
   </xsd:complexType>
   <xsd:complexType name="posisi_ap_array">
    <xsd:complexContent>
     <xsd:restriction base="SOAP-ENC:Array">
      <xsd:attribute ref="SOAP-ENC:arrayType" wsdl:arrayType="tns:posisi_ap[]"/>
     </xsd:restriction>
    </xsd:complexContent>
   </xsd:complexType>
  </xsd:schema>
 </types>
 <message name="usersRequest">
  <part name="name" type="xsd:string"/>
 </message>
 <message name="usersResponse">
  <part name="return" type="tns:data_users_array"/>
 </message>
 <message name="access_pointRequest">
  <part name="mac" type="xsd:string"/>
 </message>
 <message name="access_pointResponse">
  <part name="return" type="tns:theAccessPointArray"/>
 </message>
 <message name="adminRequest">
  <part name="name" type="xsd:string"/>
 </message>
 <message name="adminResponse">
  <part name="return" type="tns:data_admin_array"/>
 </message>
 <message name="dailyRequest">
  <part name="ap" type="xsd:string"/>
 </message>
 <message name="dailyResponse">
  <part name="return" type="tns:data_users_daily_array"/>
 </message>
 <message name="posisi_apRequest">
  <part name="new_data" type="xsd:string"/>
  <part name="x" type="xsd:string"/>
  <part name="y" type="xsd:string"/>
  <part name="name" type="xsd:string"/>
 </message>
 <message name="posisi_apResponse">
  <part name="return" type="tns:theDeviceArray"/>
 </message>
 <portType name="monitor_apPortType">
  <operation name="users">
   <documentation>Data User Current</documentation>
   <input message="tns:usersRequest"/>
   <output message="tns:usersResponse"/>
  </operation>
  <operation name="access_point">
   <documentation>Data Access Point</documentation>
   <input message="tns:access_pointRequest"/>
   <output message="tns:access_pointResponse"/>
  </operation>
  <operation name="admin">
   <documentation>Data Admin</documentation>
   <input message="tns:adminRequest"/>
   <output message="tns:adminResponse"/>
  </operation>
  <operation name="daily">
   <documentation>Data Users Daily</documentation>
   <input message="tns:dailyRequest"/>
   <output message="tns:dailyResponse"/>
```

Gambar 4. 10 Dokumen WSDL

4.3.3 Implementasi Aplikasi Pengguna

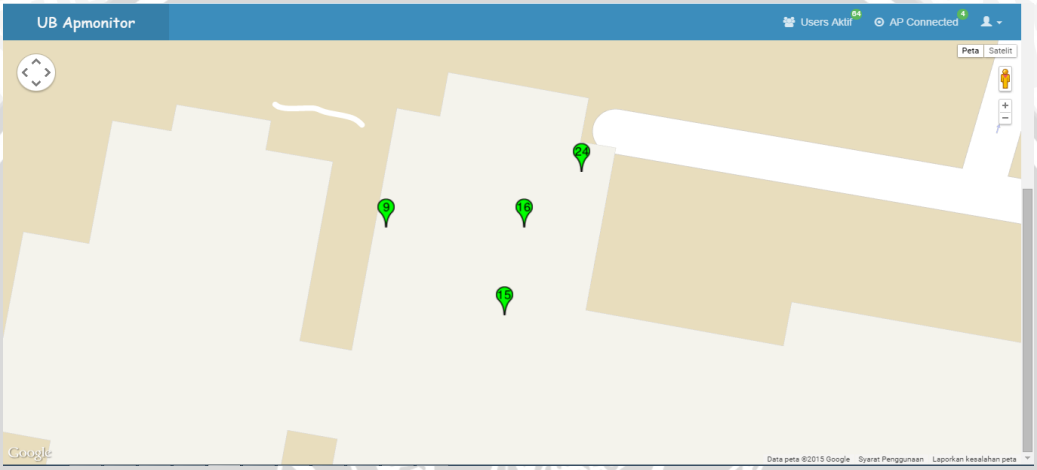
Aplikasi pengguna memberikan informasi dalm bentuk antar muka ke pengguna. Informasi tersebut antara lain lokasi access point, jumlah pengguna pada sebuah *access point*, *access point* yang di jadikan tujuan koneksi oleh pengguna, dan estimasi jumlah pengguna berdasarkan waktu yaitu hari dan bulan. Aplikasi pengguna akan meminta *service* yang mengacu pada URL dari aplikasi *web service* yaitu *http://172.29.0.3/webservice/server.php?wsdl*. Kemudian untuk mendapatkan data yang diinginkan, aplikasi pengguna akan memanggil *service* yang telah di sediakan oleh *web service*, seperti pada gambar 4.11 . *Script* lengkap implementasi aplikasi pengguna sebagaimana yang terlampir di lampiran 6.

```
$service =('http://172.29.0.3/webservice/server.php?wsdl');
$client = new nusoap_client ($service);
$client->call('nama_service');
```

Gambar 4. 11 Request Service aplikasi pengguna

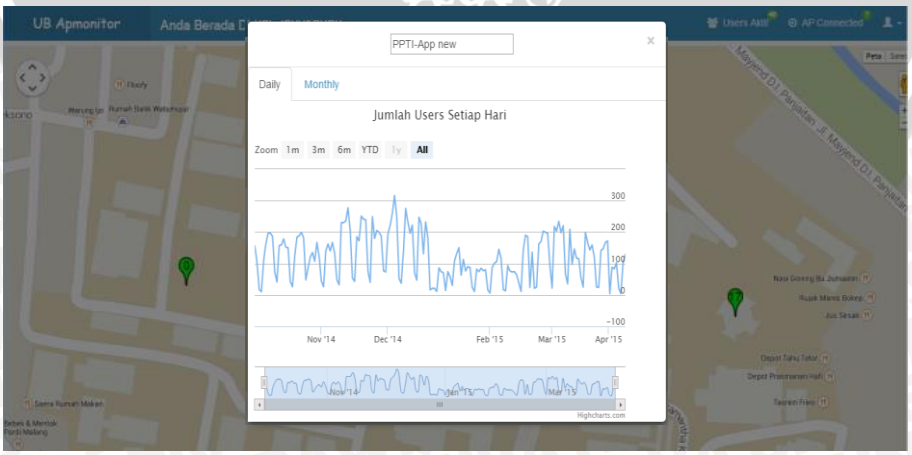
Selanjutnya di buat tampilan antar muka untuk aplikasi pengguna. Gambar 4.12 merupakan tampilan utama dari aplikasi pengguna yang berisikan informasi

access point. Pada tampilan utama pengguna akan mendapatkan informasi mengenai lokasi *access point* yang di tunjukkan oleh marker di map universitas brawijaya. Selain itu juga informasi jumlah pengguna yang terhubung ke sebuah *access point*. Jika pengguna terhubung ke sebuah *access point* yang ada di map, aplikasi akan menunjukkan *access point* yang digunakan oleh pengguna dengan cara *marker* melakukan aksi. Untuk menampilkan halaman utama aplikasi pengguna dengan mengakses pada *http://172.21.3.54/apmonitor*.



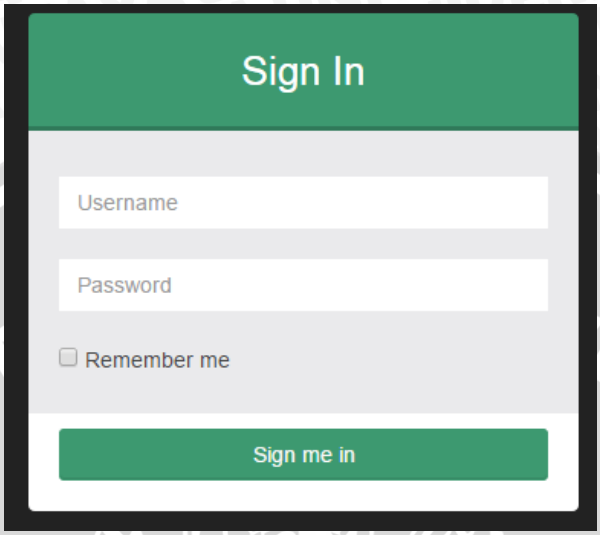
Gambar 4. 12 Tampilan Utama Aplikasi Pengguna

Gambar 4.13 merupakan tampilan dari grafik jumlah pengguna yang terhubung pada sebuah *access point* berdasarkan waktu hari dan bulan. Pengguna bisa mendapatkan informasi ini dengan melakukan klik salah satu *access point* yang di tunjukkan oleh *marker*.



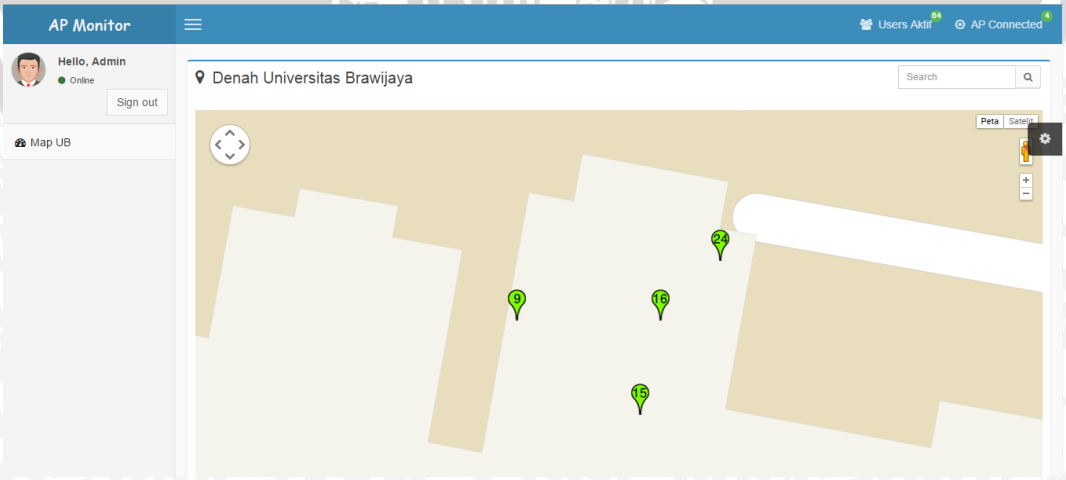
Gambar 4. 13 Grafik Jumlah Pengguna

Gambar 4.14 merupakan *form login*, dimana hanya pengguna admin yang bisa login. Form login digunakan untuk proses *otentikasi* admin untuk masuk ke halaman admin dengan melakukan inputan *username* dan *password*.



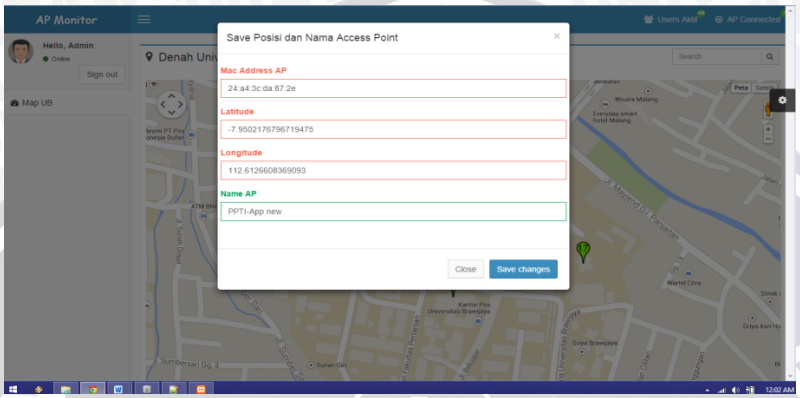
Gambar 4. 14 Tampilan Login Admin

Setelah login berhasil dilakukan, maka akan tampil halaman admin seperti yang ditunjukkan oleh gambar 4.15. Halaman admin digunakan untuk menentukan lokasi *access point* di map dengan cara *drag marker*. Halaman admin di perlukan karena data lokasi access point yang di dapatkan dari database *access point* data berupa data map *static*.



Gambar 4. 15 Halaman Admin

Gambar 4.16 merupakan antar muka admin melakukan update lokasi *access point* di map. Admin melakukan update lokasi *access point* di map dengan cara *drag marker*. Kemudian akan muncul *form* untuk *save* lokasi dari *access point* dan untuk melakukan *save* dengan cara klik *save* pada *form*.



Gambar 4. 16 Tampilan Update Lokasi Access Point

BAB V
PENGUJIAN DAN ANALISA HASIL

Pada bab ini menjelaskan proses pengujian sistem yang telah dibuat, pengambilan data, analisa kemampuan *web service*, dan analisa kemampuan dari sistem yang telah dibuat.

5.1 Pengujian Sistem

Pengujian sistem meliputi pengujian aplikasi pemetaan data, pengujian peforma sistem dengan JMeter, pengujian *web service*, dan pengujian Validasi. Berikut ini adalah penjelasan dari masing-masing pengujian.

5.1.1 Pengujian Aplikasi Pemetaan Data

Pengujian aplikasi pemetaan data dilakukan beberapa tahapan pengujian, antara lain :

- Menguji rata-rata *request time* pemrosesan aplikasi pemetaan data. Pengujian ini bertujuan untuk menentukan waktu penjadwalan yang tepat untuk aplikasi pemetaan data dalam memproses data menggunakan Jmeter.
- Menguji aplikasi pemetaan sesuai dengan jadwal yang di buat pada *cron*. Pembuktian penjadwalan sudah berjalan sesuai dengan waktu yang telah di tentukan dapat di lihat di *log cron* di server controller dengan membuka folder */var/log/cron*. Berdasarkan proses penjadwalan tersebut aplikasi bisa mendapatkan sejumlah data.

5.1.2 Pengujian Performansi Sistem dengan Apache Jmeter

Pengujian performansi sistem dengan menggunakan aplikasi Apache jMeter berdasarkan file aplikasi pengguna. Aplikasi Jmeter akan mengirimkan sejumlah *request* sesuai dengan *jumlah thread* yang sudah ditentukan. Hasil tiap proses ditampilkan dalam bentuk tabel maupun tree. Hasil yang didapat berupa *response time, throughput,* dan *latency* yang ketika *request* datang dan diproses.

5.1.3 Pengujian Web Service

Pengujian web service dilakukan uji fungsionalitas dan peformansi web service.

- Pengujian Fungsionalitas

  Pengujian fungsionalitas *web service* untuk membuktikan bahwa fungsi yang di sediakan oleh *web service* mampu menangani *request* dan memberikan *response* data. Pengujian *web service* menggunakan SOAPUI yang melakukan pengujian berdasarkan file WSDL.

- Pengujian Performansi

  Pengujian peformansi *web service* menggunakan JMeter berdasarkan file WSDL. Test plan dibuat dengan menambahkan *Thread Group* dengan thread 100, 200, 300 berturut-turut dengan *Ramp-up Period* 1 dan *Loop Count* 1 .

5.1.4 Pengujian Validasi Data

Pengujian validasai dilakukan dengan dua sekenario. Skenario pertama dilakukan pengujian validasi jumlah pengguna yang terhubung ke *access point* dengan *output* jumlah pengguna yang terhubung ke *access point* yang di tampilkan oleh sistem. Skenario kedua dilakukan pengujian validasi lokasi *access* point yang dijadikan koneksi oleh pengguna dengan output yang di tampilkan oleh sistem.

- Pengujian validasi jumlah pengguna

  Pengujian ini dilakukan untuk memastikan jumlah pengguna yang terhubung ke *access point* di gedung PPTI sesuai dengan *output* jumlah pengguna yang terhubung ke *access point* yang di tampilkan oleh sistem. Pengujian ini menggunakan 4 buah *handphone* sebagai pengguna, dimana masing-masing device akan dilakukan koneksi ke *access point* dengan memilih *ssid wifi-ub*. Penentuan tujuan access point dilakukan dengan cara mendekatkan *device* dengan sebuah *access point* untuk mendapatkan sinyal paling kuat sehingga *device* terhubung ke *access point* tersebut. Kemudian dilakukan pengecekan pada tampilan sistem, jika sistem menampilkan jumlah pengguna sesuai

dengan jumlah *device* yang terhubung ke *access point* maka sistem valid jika sebaliknya maka sistem tidak valid.

- Pengujian validasi lokasi pengguna

  Pengujian ini dilakukan untuk memastikan lokasi *access point* yang dijadikan koneksi oleh pengguna pada kondisi sebenarnya di gedung PPTI sesuai dengan output lokasi pengguna pada *access point* yang di tampilkan oleh sistem. Pengujian ini menggunakan 1 buah *handphone* sebagai pengguna, dimana device akan dilakukan koneksi ke *access point* dengan memilih *ssid wifi-ub*. Penentuan tujuan *access point* dilakukan dengan cara mendekatkan device dengan sebuah *access point* untuk mendapatkan sinyal yang kuat sehingga *device* terhubung ke *access point* tersebut. Kemudian dilakukan pengecekan pada tampilan sistem, jika sistem menampilkan lokasi *access point* sesuai dengan lokasi *device* pada *access point* maka sistem dikatakan valid jika sebaliknya maka sistem dikatakan tidak valid.

5.2 Analisa Hasil

5.2.1 Analisa Hasil Pengujian Aplikasi Pemetaan Data

- Analisa Hasil Pengujian *request time* aplikasi pemetaan data

  Berikut ini adalah hasil pengujian yang dilakukan untuk menentukan lama waktu penjadwalan aplikasi pemetaan data.

Pengujian	Request Time (ms)
1	183
2	91
3	92
4	85
5	84
6	82
7	82
8	81
9	81
10	81
Rata-rata	**94**

Tabel 5. 1 Pengujian *Request Time* Data Pengguna dan *Access Point*

Berdasarkan tabel 5.1 hasil *request time* yang dihasilkan aplikasi pemetaan data terhadap data pengguna dan *access point* yang terdapat pada file */var/www/html/PetaData/index.php* dilakukan sebanyak 10 kali dengan banyak data pengguna 50 dan access point sebanyak 4 buah. Berdasarkan data request time diperoleh rata-rata pemrosesan selama 94 ms, dimana request time kurang dari satu menit. Sehingga berdasarkan rata-rata request time 94 ms penjadwalan untuk mendapatkan data pengguna dan *access point* pada file */var/www/html/PetaData/index.php* dilakukan selama 1 menit.

Pengujian	Request Time (ms)
1	19747
2	15684
3	16132
4	16716
5	16213
6	16506
7	15649
8	15744
9	19814
10	16668
Rata-rata	**16787**

Tabel 5. 2 Pengujian Request Time Jumlah Pengguna Setiap Hari

Tabel 5.2 merupakan hasil request time yang dihasilkan aplikasi pemetaan data terhadap data jumlah pengguna setiap hari yang terdapat pada file */var/www/html/PetaData/daily.php* dilakukan sebanyak 10 kali. Berdasarkan data request time diperoleh rata-rata pemrosesan selama 16784 ms. Jumlah pengguna yang terhubung ke access point setiap hari di lakukan update data selama satu hari maka penjadwalan juga di lakukan setiap satu hari sekali yaitu pukul 00.00.

- Analisa Hasil Pengujian Penjadwalan Aplikasi Pemetaan Data

Berdasarkan gambar 5.1 merupakan *log cront* yang dihasilkan oleh aplikasi pemetaan data. Dilakukan 2 penjadwalan pada aplikasi pemetaan data. Penjadwalan pertama untuk memetakan data collection *cache_sta* (data pengguna) dan *device* (data *access point*) dilakukan setiap 1 menit sekali dimana *cron* memproses file */var/www/html/PetaData/index.php*.

Penjadwalan kedua untuk mendapatkan jumlah pengguna yang terhubung ke access point setiap hari dilakuakan setiap hari pukul 00.00 dimana cron memproses file */var/www/html/PetaData/daily.php*.

Berdasarkan gambar 5.1 penjadwalan berjalan sesuai dengan jadwal yang telah di tentukan yaitu aplikasi akan melakukan proses 1 menit sekali pada file */var/www/html/PetaData/index.php* dan satu hari sekali setiap pukul 00:00 pada file */var/www/html/PetaData/daily.php*. Hal ini dapat dibuktikan dengan tidak adanya pesan *failed* pada *log cron.*

```
Mar 29 23:58:02 vm-unifi3 CROND[30641]: (root) CMD (php /var/www/html/PetaData/index.php)
Mar 29 23:59:01 vm-unifi3 CROND[30649]: (root) CMD (php /var/www/html/PetaData/index.php)
Mar 30 00:00:01 vm-unifi3 CROND[30659]: (root) CMD (php /var/www/html/PetaData/index.php)
Mar 30 00:00:01 vm-unifi3 CROND[30660]: (root) CMD (php /var/www/html/PetaData/daily.php)
```

Gambar 5. 1 Log Cron Aplikasi Pemetaan Data

5.2.2 Analisa Hasil Pengujian Performansi Sistem dengan Jmeter

Pengujian performansi sistem dilakukan dengan melakukan percobaan sebanyak lima kali berturut-turut dengan perlakuan yang sama. Hal ini dimaksudkan untuk mengetahui perubahan yang dihasilkan pada *response time, troughput, dan latency* sistem. Dimana tiap hasil nantinya di buat rata-rata untuk mendapatkan nilai yang terbaik. Pengujian dilakukan dengan mengirimkan *HTTP request* ke alamat http://172.21.3.54/apmonitor dengan jumlah *thread* sebesar 100, waktu *rump-up* 1 detik dan jumlah perulangan sebanyak 1 kali. Tabel 5.3 merupakan hasil pengujian *response Time*, *throughput*, dan *latency* dari sistem dengan menggunakan *jmeter*.

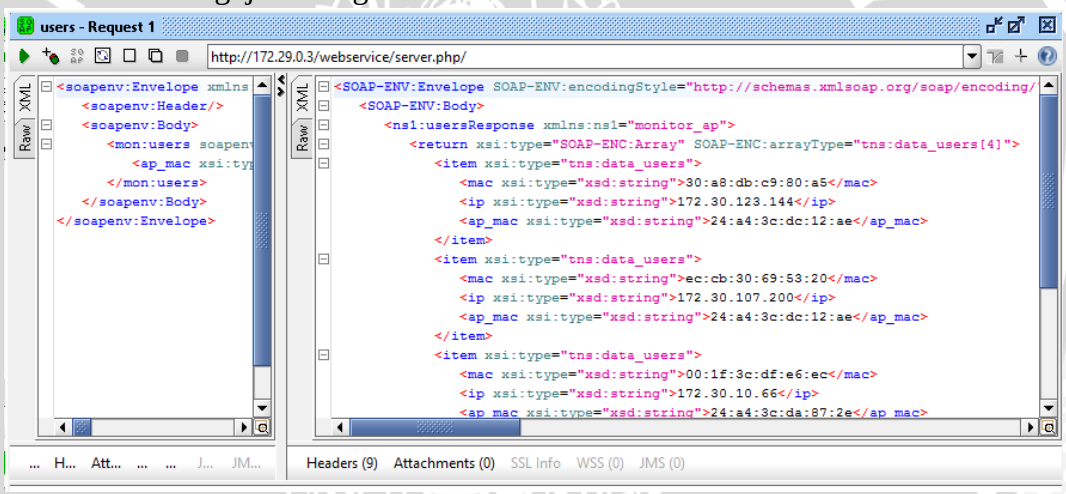
Keterangan	Response Time (ms))	Throughput (Kb/Second)	Latency (ms)
Uji 1	28249	56,69	2191
Uji 2	29326	56,31	2401
Uji 3	29138	56,89	2365
Uji 4	25317	57,85	2142
Uji 5	27835	54,54	2240
Rata-rata	27973	56,45	2267

Tabel 5. 3 Hasil Response Time, Throughput, dan Latency Pengujian Sistem

Hasil pengujian *response time*, *throughput*, dan *latency* sistem dapat di lihat table 5.3 bahwa *response time* terendah terdapat pada pengujian ke 4 dengan nilai 25317 ms dan *response time* tertinggi terdapat pada pengujian ke 2 dengan *response time* 29326 ms. *Throughput* terendah ditunjukkan pada pengujian ke 5 sebesar 54,54 Kbps dan *throughput* tertinggi ditunjukkan pada pengujian ke 4 sebesar 57,85 Kbps. *Latency* terendah terdapat pada pengujian ke 4 dengan nilai 2142 ms dan *latency* tertinggi terdapat pada pengujian ke 2 dengan nilai 2401 ms. Berdasarkan 5 pengujian tersebut di dapatkan nilai rata-rata Kemudian *response time* sebesar 27973 ms, *throughput* sebesar 56,45 Kbps, dan *latency* sebesar 2267ms. Nilai rata-rata ini merupakan nilai terbaik yang di hasilkan oleh sistem.

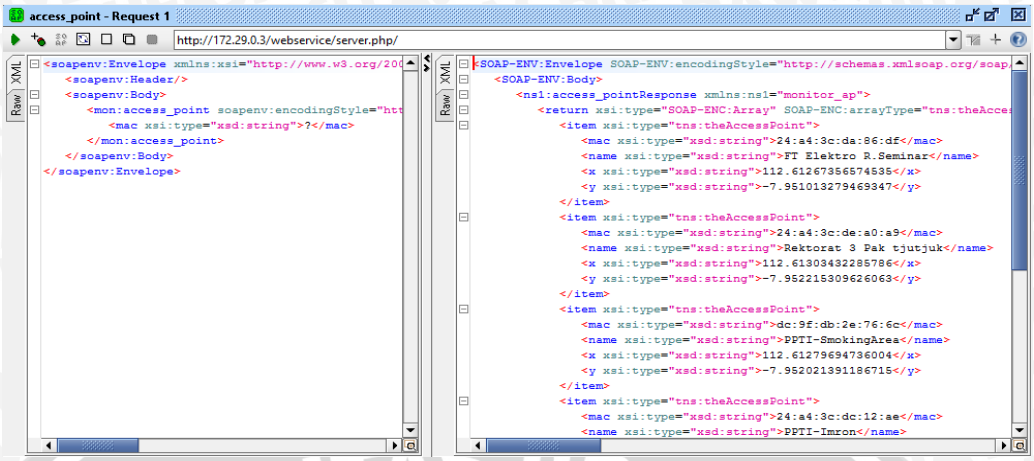
5.2.3 Analisa Hasil Pengujian Aplikasi Web Service

- Analisa Hasil Pengujian Fungsionalitas



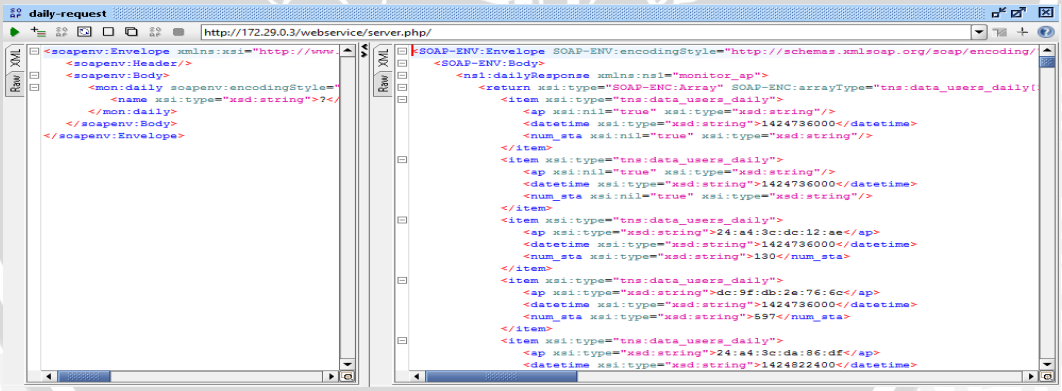
Gambar 5. 2 Hasil Request dan Response Fungsi Users

Berdasarkan gambar 5.2 hasil *request* dan *response* yang di hasilkan dari fungsi *users*. Pengguna akan melakukan *request* dengan memanggil fungsi *users*, kemudian *web service* akan memberikan *response* ke pengguna dengan mengirimkan sejumlah data *users* yaitu *mac address users*, IP *address users*, dan *mac address access point*. Hasil *request* dan *response* fungsi *users* dapat berjalan dengan baik hal ini dibuktikan dengan tidak adanya pesan *error*.



Gambar 5. 3 Hasil Request dan Response Fungsi Access Point

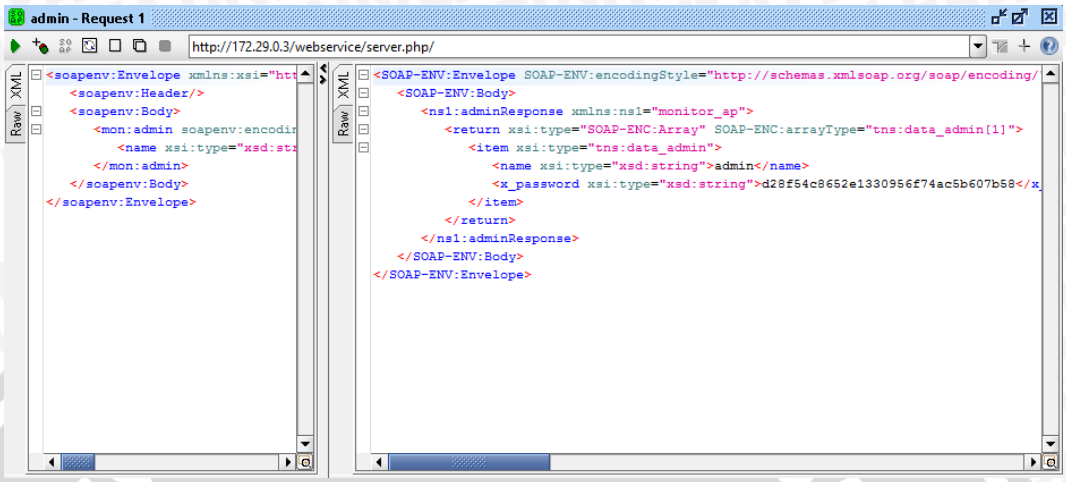
Berdasarkan gambar 5.3 hasil *request* dan *response* yang di hasilkan dari fungsi *access point*. Pengguna akan melakukan *request* dengan memanggil fungsi *access point*, kemudian *web service* akan memberikan *response* ke pengguna dengan mengirimkan sejumlah data *access point* yaitu *mac address access point*, nama *access point*, dan koordinat *access point* (x dan y). Hasil *request* dan *response* fungsi *access point* dapat berjalan dengan baik hal ini dibuktikan dengan tidak adanya pesan *error*.



Gambar 5. 4 Hasil Request dan Response Fungsi Daily

Berdasarkan gambar 5.4 hasil *request* dan *response* yang di hasilkan dari fungsi *daily*. Pengguna akan melakukan *request* dengan memanggil fungsi *daily*, kemudian *web service* akan memberikan *response* ke pengguna dengan mengirimkan sejumlah data *daily* yaitu *mac address access point*, waktu (datetime), dan jumlah pengguna (num_sta).

Hasil *request* dan *response* fungsi *daily* dapat berjalan dengan baik hal ini dibuktikan dengan tidak adanya pesan *error*.



Gambar 5. 5 Hasil Request dan Response Pada Fungsi Admin

Berdasarkan gambar 5.5 hasil dari *request* dan *response* yang di hasilkan dari fungsi *admin*. Pengguna akan melakukan *request* dengan memanggil fungsi *admin*, kemudian *web service* akan memberikan *response* ke pengguna dengan mengirimkan sejumlah data *admin* yaitu *username* dan *password.* Hasil *request* dan *response* fungsi *admin* dapat berjalan dengan baik hal ini dibuktikan dengan tidak adanya pesan *error*.

- Analisa Hasil Pengujian Performansi

  Pengujian performansis web service dilakukan dengan melakukan percobaan sebanyak tiga kali berturut-turut dengan perlakuan yang sama. Hal ini dimaksudkan untuk mengetahui perubahan yang dihasilkan pada *response time, troughput, dan latency* sistem. Pengujian dilakukan dengan mengirimkan *HTTP request* ke alamat WSDL yaitu *http://172.29.0.3/webservice/server.php?wsdl* dengan jumlah *thread* sebesar 100, 200, dan 300 waktu *rump-up* 1 detik dan jumlah perulangan sebanyak 1 kali. Tabel 5.4 merupakan hasil pengujian *response Time, throughput,* dan *latency* dari web service dengan menggunakan *jmeter*.

Jumlah Threads	Average (ms)	Throughput (Kb/Second)	Latency (ms)
100	2436	23,4	2425
200	2465	22,5	2460
300	2517	21,5	2510

Tabel 5. 4 Hasil Pengujian Performansi Web Service

Hasil pengujian *response time*, *throughput*, dan *latency* sistem dapat di lihat table 5.4 bahwa meningkatnya jumlah pengguna 2 atau 3 kali tidak mempengaruhi sistem secara signifikan. Terbukti untuk jumlah threads 100 rata-rata response yang dihasilkan sebesar 2436 ms, throughput 23,4 Kbps , dan latency 2425. Jumlah threads 200 rata-rata response time yang dihasikan 2465, throughput 22,5 Kbps, dan latency 2460 ms. Jumlah threads 300 rata-rata response time yang dihasilkan 2517 ms, throughput 22,5 Kbps, dan latency 2460 ms. Berdasarkan peningkatan jumlah threads dari 100 ke 200 didapatkan peningkatan rata-rata response time web service sebesar 29 ms, berkurangnya throughput sebesar 0,9 Kbps, meningkatnya latency sebesar 35 ms dan berdasarkan peningkatan jumlah threads dari 200 ke 300 didapatkan peningkatan rata-rata request time web service sebesar 52 ms, menurunnya throughput sebesar 2 Kbps, dan meningkatnya latency sebesar 2510 ms.

5.2.4 Analisa Hasil Pengujian Validasi Data

- Analisa Hasil Validasi Jumlah Pengguna

  Berikut ini merupakan tabel hasil dari pengujian validasi jumlah pengguna pada *access point* dan *output* jumlah pengguna yang ditampilkan oleh sistem.

NO	Jumlah Pengguna pada Access Point				Output Jumlah Pengguna pada Sistem				Hasil (Valid/Tidak Valid)
	SMO	IMR	HEL	APP	SMO	IMR	HEL	APP	
1	4	0	0	0	4	0	0	0	Valid
2	0	4	0	0	0	4	0	0	Valid
3	0	0	4	0	0	0	4	0	Valid

4	0	0	0	4	0	0	0	4	Valid
5	1	1	1	1	1	1	1	1	Valid
6	2	2	0	0	2	2	0	0	Valid
7	3	1	0	0	3	1	0	0	Valid
8	2	1	1	0	2	1	1	0	Valid

Tabel 5.4 Hasil Pengujian Validasi Jumlah Pengguna pada *Access Point* dengan *Output* Sistem

Keterangan Access Point :

1. SMO = PPTI-Smoking Area
2. IMR = PPTI-Imron
3. HEL = PPTI-Helpdesk
4. APP = PPTI-App New

Berdasarkan tabel 5.4 hasil pengujian validasi jumlah pengguna yang terhubung ke 4 *access point* di gedung PPTI dengan *output* yang ditampilkan sistem didapatkan hasil yang valid. Hal ini terbukti dengan 8 kali uji hasil yang didapatkan antara jumlah pengguna yang terhubung ke masing-masing *access point* di gedung PPTI sama dengan *output* jumlah pengguna yang di tampilkan oleh sistem pada masing-masing *access point*.

- Analisa Hasil Validasi Lokasi Pengguna

  Berikut ini merupakan tabel hasil validasi pengujian lokasi *access point* yang dijadikan koneksi oleh pengguna pada kondisi sebenarnya di gedung PPTI dengan output lokasi pengguna pada *access point* yang di tampilkan oleh sistem.

NO	**Lokasi Pengguna pada Access Point**				**Output Lokasi Pengguna pada Sistem**				**Hasil (Valid/Tidak Valid)**
	SMO	**IMR**	**HEL**	**APP**	**SMO**	**IMR**	**HEL**	**APP**	
1	V	-	-	-	V	-	-	-	Valid
2	-	V	-	-	-	V	-	-	Valid
3	-	-	V	-	-	-	V	-	Valid

4	-	-	-	V	-	-	-	V	Valid

Tabel 5.5 Hasil Pengujian Validasi Lokasi Pengguna pada *Access Point* dengan *Output* Sistem

Keterangan Access Point :

1. SMO = PPTI-Smoking Area
2. IMR = PPTI-Imron
3. HEL = PPTI-Helpdesk
4. APP = PPTI-App New

Berdasarkan tabel 5.5 hasil validasi pengujian lokasi access point yang dijadikan koneksi oleh pengguna pada kondisi sebenarnya di gedung PPTI dengan output lokasi pengguna pada access point yang di tampilkan oleh sistem di dapatkan hasil yang valid. Hal ini terbukti dengan 4 kali uji ketika *device (Handphone)* di dekatkan pada access point PPTI-Smoking Area untuk mendapatkan sinyal yang kuat dan output lokasi pengguna yang ditampilkan oleh sistem berada pada access point PPTI-Smoking Area. Perlakuan yang sama juga dilakukan dengan mendekatkan device untuk mendapatkan sinyal yang kuat pada access point PPTI-Imron, PPTI-Helpdesk, PPTI-APP New dan output yang dihasilkan oleh sistem menunjukkan hasil lokasi *access point* yang sama dengan lokasi yang di dekati oleh *device*.

BAB VI
PENUTUP

6.1 Kesimpulan

Berdasarkan hasil penelitian dan analisa yang sudah dilakukan, dapat diambil kesimpulan sebagai berikut :

1. SOAP *(Simple Object Access Protocol) web service* dapat digunakan sebagai penyedia layanan ke aplikasi pengguna dan aplikasi pengguna mampu memberikan informasi ke pengguna berupa lokasi *access point*, jumlah pengguna *access point*, dan lokasi pengguna berdasarkan waktu *real time*, hari, dan bulan.
2. Rata-rata *request time* yang di hasilkan oleh aplikasi pemetaan data untuk memetakan data *access point* dan pengguna dari database controller ke database *web service* sebesar 94 ms. Berdasarkan hal ini waktu penjadwalan *cron* diatur selama 1 menit yang merupakan waktu terkecil *cron*.

   Semua fungsi layanan yang di sediakan *web service* berfungsi dengan baik. Terbukti tidak adanya pesan error pada *response web service*. Hasil pengujian performansi *web service* menunjukkan bahwa meningkatnya jumlah pengguna tidak mempengaruhi sistem secara signifikan. Terbukti untuk jumlah threads 100 rata-rata response yang dihasilkan sebesar 2436 ms, throughput 23,4 Kbps , dan latency 2425. Jumlah threads 200 rata-rata response time yang dihasikan 2465, throughput 22,5 Kbps, dan latency 2460 ms. Jumlah threads 300 rata-rata response time yang dihasilkan 2517 ms, throughput 22,5 Kbps, dan latency 2460 ms.

   Hasil pengujian sistem menunjukkan rata-rata *request time* yang di hasilkan dari aplikasi yang sudah dibuat adalah 27973 ms, *throughput* sebesar 56,45 KBps, dan *latency* sebesar 2267 ms.

   Hasil pengujian validasi jumlah pengguna, lokasi *access point*, dan lokasi pengguna menunjukkan bahwa sistem mampu memberikan informasi ke pengguna secara tepat dan akurat.

6.2 Saran

Saran yang diberikan untuk pengembangan selanjutnya pada web *monitoring access point* berbasis *simple object access protocol* (SOAP) *web service* pada infrastruktur unifi antar lain :

1. Untuk pengembangan selanjutnya, perlu adanya penambahan parameter *trafic* dari *access point* untuk mendapatkan informasi kesibukan dari *access point*.
2. Untuk penelitian selanjutnya, perlu membandingan peforma *web service* antara SOAP dengan REST sehingga didapatkan hasil peforma web service yang lebih baik.

DAFTAR PUSTAKA

[BIT-14] Bits, 2014. *Brawijaya Information Technology Services,* http://bits.ub.ac.id/, [7 Juli 2014]

[WID-13] Widyaningsih, Bekti.2013.*"Optimasi Area Cakupan Jaringan Nirkabel Dalam Ruangan ".* Skripsi : Informatika/Ilmu Komputer Universitas Brawijaya.

[SUR-13] Surotih, I.B. 2013. "*Perancangan Infrastruktur Managemen Bandwidth Berdasarkan Kategori User".* Skripsi : Informatika/Ilmu Komputer Universitas Brawijaya.

[WIF-14] Wifi.2014. *Who we are (Wi-Fi Aliance).* http://www.wi-fi.org/who-we-are, [7 Juli 2014]

[RAB-08] Rabbit, 2008. *An Introduction to Wi-Fi.* Digital International, Inc.,New York.

[ROS-03] Roshan, P dan Leary, J. 2003. *802.11 Wireless LAN Fundamental*.Cisco Press, Indiana.

[BOO-04] Booth, David, dkk.2004. *Web Services Architecture*. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/, [25 Juli 2014]

[QUI-10] Quin, Liam. 2010. *XML Essentials.* http://www.w3.org/standards/xml/core, [25 Juli 2014

[PUT-10] Putra,K.A.B. 2010. "*Analisa Perbandingan Perfomansi XML-RPC dan JSON-RPC dengan Studi Kasus Web Service"*. Skripsi : Informatika Institut Teknologi Telkom.

[WIL-10] Wilson, Jean. 2010. *Using WSDL Generator and SOAP*. Redwood City. Oracle

[W3C-14] w3c. 2014. *Web Service Tutorial*. http://www.w3schools.com/webservices/default.asp, [7 Agustus 2014]

[JAN-13] Jane,M.L. 2013. *PHP Web Service* . Sebastopol CA. O'Reilly Media

[UBN-14] Ubnt. 2014. UniFI Enterprise WIFI System , http://www.ubnt.com/unifi/, [7 Juli 2014]

[CHO-10] Chodorow, Kristina.2010 . *Powerful and Scalable Data Storage MongoDB The Difinitive Guide.* United States of America. O'Reilly Media

[MID-12] Midgley, Julian T J, 2012, Autobench , http://www.xenoclast.org/autobench/, [7 September 2014]

[APA-13] Apache, 2013, *Apache JMeter™*, https://jmeter.apache.org/, [7 Juli 2014]

[KRE-01] Kreger, Heather. 2001. *Web Services Conceptual Architecture.* New York. IBM.

[HEN-14] Heng, Christopher.2014. *How to Create a Cron Job.* http://www.thesitewizard.com/general/set-cron-job.shtml, [7 Juli 2014]

LAMPIRAN

Lampiran 1. Instalasi MongoDB

1. Buka terminal di linux dan masuk pada file `/etc/yum.repos.d/mongodb.repo` dan tambahkan baris seperti di bawah ini untuk link download modul mongodb.

```
[mongodb]
name=MongoDB Repository
baseurl=http://downloads-
distro.mongodb.org/repo/redhat/os/x86_64/
gpgcheck=0
enabled=1
```

2. Install mongodb dengan perintah seperti di bawah ini.

```
# yum install -y mongo-10gen mongo-10gen-server
```

3. Jalankan mongodb dengan perintah seperti di bawah ini

```
# service mongodb start
```

4. Tambahkan perintah seperti di bawah ini agar mongodb dapat terus berjalan otomatis saat booting.

```
# chkconfig mongod on
```

Lampiran 2. Instalasi dan konfigurasi Controller UniFi

a. Buka terminal linux dan install mongodb
b. Kemudian install Java JRE dengan perintah seperti di bawah ini
   `Wget http://download.oracle.com/otn-pub/java/jdk/7u51-b13/jre-7u51-linux-x64.rpm`
   Kemudian install dengan perintah `# rpm -ivh jre-7u51-linux-x64.rpm`
c. Kemudian download Unifi controller versi 3.2.1 dengan perintah seperti di bawah ini
   `#wget http://dl.ubnt.com/unifi/3.2.1/UniFi.unix.zip`
d. Setelah berhasil di download dalam bentuk zip kemudian extrak dan pindahkan ke folder */opt* dengan perintah `#unzip -q UniFi.unix.zip -d /opt`
e. Kemudian membuat simlink mongdb ke folder instalasi unifi perintah sebagai berikut.
   ```
   # cd /opt/UniFi/bin/
   # sudo ln -fs /usr/bin/mongod mongod
   ```
f. Kemudian membuat service untuk menjalankan unifi, dengan perintah seperti di bawah ini
   `# vi /etc/init.d/UniFi`

   Kemudian tambahkan script shell seperti di bawah ini

```
#!/bin/bash
ctrl_start(){
    #nohup java -jar JarFile.jar >myLogFile
    2>>myErrorFile&
    java -jar /opt/UniFi/lib/ace.jar start &
}
ctrl_stop(){
     java -jar /opt/UniFi/lib/ace.jar stop &
     }

ctrl_restart(){
     ctrl_stop
     sleep 1
     ctrl_start
     }
     case "$1" in
     start)
     echo -n "starting UniFi system"
     ctrl_start
     echo " service started"
     ;;
```

```
        stop)
        echo -n "stopping UniFi system"
        ctrl_stop
        echo " service stopped"
        ;;
        restart)
        echo -n "restarting UniFi system"
        ctrl_restart
        echo "service restarted"
        ;;
        *)
        echo "usage: service UniFi
        {start|stop|restart}"
        ;;
        esac
exit 0
```

Kemudian membuat layanan agar script diatas bisa di eksekusi dengan perintah seperti di bawah ini.

```
# chmod +x /etc/init.d/UniFi
```

Kemudian menambahkan `# chkconfig UniFi on` agar bisa di jalankan secara otomatis saat booting.

g. Kemudian membuka port dengan iptables agar unifi bisa di jalankan melalui http, di awali dengan membuka folder iptables dan menambahkan aturan pada iptables perintahnya sebagai berikut ini

```
# vi /etc/sysconfig/iptables
```

Kemudian menambahkan aturan untuk membuka tcp port 8443,8080,8843, dan 27117 seperti perintah di bawah ini

```
# -A INPUT -m state --state NEW -m tcp -p tcp --dport 8080 -j ACCEPT
```

Terahir menambahkan aturan untuk membuka udp port 3478 dengan perintah di bawah ini

```
# -A INPUT -m state --state NEW -m udp -p udp --dport 3478 -j ACCEPT
```

Kemudian simpan dan restast iptables dengan perintah `# service iptables restart` jika tidak muncul error berarti aturan tadi berjalan dengan baik.

Lampiran 3. Instalasi dan konfigurasi web server

a. *Server controller*.

1. Buka terminal linux.
2. Ketikkan perintah seperti di bawah ini.
   ```
   #yum install httpd
   ```

Dikarenakan pada penelitian ini menggunakan bahasa pemrograman PHP pada aplikasi pemetaan data dan *aplikasi web service* yang akan di jalankan pada *server controller* maka diperlukan penambahan installasi modul PHP. Berikut perintah untuk installasi modul PHP.

```
#yum install php php5 php-pear php5-xcache
```

Setelah *web server* dan modul PHP telah terinstall pada linux, selanjutnya dilakukan konfigurasi agar *web server* dapat berjalan dengan baik. Berikut tahapan untuk melakukan konfigurasi *web server:*

1. Buka file pada */etc/httpd/conf/httpd.conf* dan tambahkan baris seperti di bawah ini untuk menentukan alamat server dengan menambahkan IP dari *server controller* sehingga aplikasi bisa di jalankan oleh komputer lain.
   ```
   #ServerName 172.29.0.3:80
   ```
2. Jalankan *web server* seperti perintah di bawah ini.
   ```
   #service httpd start
   ```
3. Kemudian agar web server dapat terus berjalan otomatis saat booting, ketikkan perintah seperti di bawah ini.
   ```
   # chkconfig --levels 235 httpd on
   ```

b. Server aplikasi pengguna.

1. Buka terminal linux
2. Ketikkan perintah seperti di bawah ini.
   ```
   #sudo apt-get install apache2 php5 libapache2-mod-php5
   ```
   Perintah diatas digunakan untuk *install web server* dengan paket program apache dengan nama apache2 untuk linux distro Ubuntu, selain itu di tambahkan modul PHP dikarenakan server ini nanti akan menjalankan aplikasi pengguna dengan bahasa pemrogaman PHP.
3. Setelah itu jalankan apache dengan perintah seperti di bawah ini
   ```
   #service apache2 start
   ```

Setelah instalasi selesai kemudian jalankan aplikasi pengguna, langkahnya seperti di bawah ini.

1. Pindahkan folder aplikasi program ke direktori /var/www dengan perintah seperti di bawah ini.
   ```
   #mv -R apmonitor/ /var/www/html/
   ```
2. Masuk pada direktori /var/www untuk melihat program yang di pindahkan hasil dengan perintah.

```
#cd /var/www/html
```

3. Setelah program berhasil di pindahkan kemudian ubah permisi aplikasi pengguna yang baru saja dipindahkan, agar file program dapat terbaca dan dapat dieksekusi dengan perintah.
   ```
   #chmod -R 777 /apmonitor
   ```
4. Kemudian buka aplikasi pada browser dengan alamat `http://172.21.3.39`

Lampiran 4. Aplikasi Pemetaan Data

a. db_server.php

```
1.  <?php
2.  $dbhost = 'localhost';
3.  $dbport = '27117';
4.  $dbname = 'ace';
5.
6.  try{
7.  $mongo = new Mongo("mongodb://$dbhost:$dbport");
8.  $databases = $mongo->$dbname;
9.  }
10. catch(MongoConnectionEcxeption $e)
11. {
12.   die($e->getMessage());
13. }
14. ?>
```

b. db_webservice.php

```
1.    <?php
2.    // Config
3.    $dbhost = '172.29.0.3';
4.    $dbport = '27017';
5.    $dbname = 'db_webservice';
6.
7.    try{
8.    $mongo = new Mongo("mongodb://$dbhost:$dbport");
9.    $db = $mongo->$dbname;
10.   }
11.   catch(MongoConnectionEcxeption $e)
12.   {
13.         die($e->getMessage());
14.   }
15.   ?>
```

c. index.php

```
1.    <?php
2.    include_once 'db_server.php';
3.    include_once 'db_webservice.php';
4.    try {
5.    $col_ap = $db->access_point;
6.    $cursor_access_point = $col_ap->find();
7.    $col_device = $databases->device;
```

```
8.   $cursor_dev = $col_device->find();
9.   $col_user = $db->users;
10.  $cursor_users= $col_user->find();
11.  $col_sta = $databases->cache_sta;
12.  $cursor_sta = $col_sta->find();
13.
14.  $num = $cursor_access_point->count();
15.  if ($num > 0){
16.  foreach ($cursor_dev as $obj_device) {
17.  $status = 0;
18.  foreach ($cursor_access_point as
19.  $obj_access_point) {
20.  if ($obj_device['mac'] ==
21.  $obj_access_point['mac']){
22.  $status = 1;
23.  break;
24.  }
25.  }
26.  if($status == 0){
27.  $insert_device = array(
28.  'mac' => $obj_device['mac'],
29.  'name' => $obj_device['name'],
30.  'x' => $obj_device['x'],
31.  'y' => $obj_device['y']
32.  );
33.  $col_ap->insert($insert_device);
34.  }
35.  }
36.
37.  foreach ($cursor_access_point as
38.  $obj_access_point) {
39.  $status = 0;
40.  foreach ($cursor_dev as $obj_device) {
41.  if ($obj_device['mac'] ==
42.  $obj_access_point['mac']){
43.  $status = 1;
44.  break;
45.  }
46.  }
47.  if($status == 0){
48.  $data_lokasi = array(
49.  'mac' => $obj_access_point['mac'],
50.  'name' => $obj_access_point['name'],
51.  'x' => $obj_access_point['x'],
52.  'y' => $obj_access_point['y']
53.  );
54.  $col_ap->remove( $data_lokasi );
55.  }
```

```
56.  }
57.  }
58.  else {
59.  foreach ($cursor_dev as $obj)
60.  {
61.  $access_point = array(
62.  'mac' => $obj['mac'],
63.  'name' => $obj['name'],
64.  'x' => $obj['x'],
65.  'y' => $obj['y']
66.  );
67.  $col_ap->insert($access_point);
68.  }
69.  }
70.  $jml_obj_users = $cursor_users->count();
71.  if ($jml_obj_users > 0 ){
72.  foreach ($cursor_sta as $obj ){
73.  $status = 0;
74.  foreach ($cursor_users as $value) {
75.  if(($obj['mac'] == $value['mac']) &&
76.  ($obj['ap_mac'] == $value['ap_mac'])) {
77.  $status = 1;
78.  break;
79.  }
80.  }
81.  if ($status == 0){
82.  $users = array(
83.  'mac' => $obj['mac'],
84.  'ap_mac' => $obj['ap_mac'],
85.  'ip' => $obj['ip']
86.  );
87.  $col_user->insert($users);
88.  }
89.  }
90.  foreach ($cursor_users as $value) {
91.  $status = 0;
92.  foreach ($cursor_sta as $obj) {
93.  if(($obj['mac'] == $value ['mac']) &&
94.  ($obj['ap_mac'] == $value['ap_mac'])){
95.  $status = 1;
96.  break;
97.  }
98.  }
99.  if($status == 0){
100. $users = array(
101. 'mac' => $value['mac'],
102. 'ap_mac' => $value['ap_mac'],
103. 'ip' => $value['ip']
```

```
104. );
105. $col_user->remove($users);
106. }
107. }
108. }
109. else{
110. foreach ($cursor_sta as $obj ){
111. $users = array(
112. 'mac' => $obj['mac'],
113. 'ap_mac' => $obj['ap_mac'],
114. 'ip' => $obj['ip']
115. );
116. $col_user->insert($users);
117. }
118. }
119. }
120. catch ( MongoConnectionException $e )
121. {
122. echo $e->getMessage();
123. }
124. catch ( MongoException $e )
125. {
126. echo $e->getMessage();
127. }
128. ?>
```

d. daily.php

```
1.  <?php
2.  include_once 'db_server.php';
3.  include_once 'db_webservice.php';
4.  try{
5.  $col_users_daily = $db->users_daily;
6.  $cursor_users_daily = $col_users_daily->find();
7.  $col_stat_daily = $databases->stat_daily;
8.  $cursor_stat_daily = $col_stat_daily->find();
9.  $jml_obj_daily = $cursor_users_daily->count();
10. if ($jml_obj_daily > 0){
11. foreach ($cursor_stat_daily as $obj_stat_daily) {
12. $status = 0;
13. foreach ($cursor_users_daily as $value) {
14. if (($obj_stat_daily['ap'] == $value ['ap']) &&
15. ($obj_stat_daily['datetime'] ==
16. $value['datetime'])){
17. $status = 1;
18. break;
19. }
```

```
20. }
21. if ( $status == 0){
22. $users_daily = array(
23. 'ap' => $obj_stat_daily['ap'],
24. 'datetime' => $obj_stat_daily['datetime'],
25. 'num_sta' => $obj_stat_daily['num_sta']
26. );
27. $col_users_daily->insert ($users_daily);
28. }
29. }
30. }
31. else{
32. foreach ($cursor_stat_daily as $obj_stat_daily) {
33. $users_daily = array(
34. 'ap' => $obj_stat_daily['ap'],
35. 'datetime' => $obj_stat_daily['datetime'],
36. 'num_sta' => $obj_stat_daily['num_sta']
37. );
38. $col_users_daily->insert ($users_daily);
39. }
40. }
41. }
42. catch ( MongoConnectionException $e )
43. {
44. echo $e->getMessage();
45. }
46. catch ( MongoException $e )
47. {
48. echo $e->getMessage();
49. }
50. ?>
```

Lampiran 5. Aplikasi *Web Service*

a. function.php

```
<?php
 $dbhost = "172.29.0.3";
 $dbport = "27017";
 $dbname = "db_webservice";

function users() {
global $dbhost, $dbname, $dbport;
if ($conn = new
MongoClient("mongodb://$dbhost:$dbport")) {
if ($db = $conn->selectDB($dbname)) {
$collection = new MongoCollection($db, 'users');
$result = $collection->find();
$rows = array();
foreach ($result as $row) {
$ip = '';
if (array_key_exists("ip", $row)) {
$ip = $row["ip"];
}
$rows[] = array(
'mac'=>$row["mac"],
'ip'=>$ip,
'ap_mac' => $row["ap_mac"]
);
}
return $rows;
}
else
{
return new soap_fault('Database Server', '',
'Koneksi ke database gagal!', '');
}
}
}

function access_point() {
global $dbhost, $dbname, $dbport;
if ($conn = new
MongoClient("mongodb://$dbhost:$dbport")) {
if ($db = $conn->selectDB($dbname)) {
$collection = new MongoCollection($db,
'access_point');
$result = $collection->find();
$rows = array();
foreach ($result as $row) {
$rows[] = array(
'mac'=>$row["mac"],
'name'=>$row["name"],
'x'=>$row["x"],
```

```
49.  'y'=>$row["y"]
50.  );
51.  }
52.  return $rows;
53.  }
54.  else
55.  {
56.  return new soap_fault('Database Server', '',
57.  'Koneksi ke database gagal!', '');
58.  }
59.  }
60.  }
61.
62.  function admin() {
63.  global $dbhost, $dbname, $dbport;
64.  if ($conn = new
65.  MongoClient("mongodb://$dbhost:$dbport")) {
66.  if ($db = $conn->selectDB($dbname)) {
67.  $collection = new MongoCollection($db, 'admin');
68.  $result = $collection->find();
69.  $rows = array();
70.  foreach ($result as $row) {
71.  $rows[] = array (
72.  'name' => $row['name'],
73.  'x_password' => $row['x_password']
74.  );
75.  }
76.  return $rows;
77.  }
78.  else
79.  {
80.  return new soap_fault('Database Server', '',
81.  'Koneksi ke database gagal!', '');
82.  }
83.  }
84.  }
85.
86.  function daily() {
87.  global $dbhost, $dbname, $dbport;
88.  if ($conn = new
89.  MongoClient("mongodb://$dbhost:$dbport")) {
90.  if ($db = $conn->selectDB($dbname)) {
91.  $collection = new
92.  MongoCollection($db,'users_daily');
93.  $result = $collection->find();
94.  $rows = array();
95.  foreach ($result as $row) {
96.  $rows[] = array (
97.  'ap' => $row['ap'],
98.  'datetime' => $row['datetime']->sec,
99.  'num_sta' => $row['num_sta']
100. );
```

```
101. }
102. return $rows;
103. }
104. else
105. {
106. return new soap_fault('Database Server', '',
107. 'Koneksi ke database gagal!', '');
108. }
109. }
110. }
111.
112. function posisi_ap($ap_mac,$x,$y,$name){
113. global $dbhost, $dbname, $dbport;
114. if ($conn = new
115. MongoClient("mongodb://$dbhost:$dbport")) {
116. if ($db = $conn->selectDB($dbname)) {
117. $collection = new
118. MongoCollection($db,'access_point');
119. $newdata = array ('$set' => array (
120. "name"=>$name,
121. "x"=>$y,
122. "y"=>$x));
123. $retval = $collection->update(
124. array("mac" => $ap_mac), $newdata
125. );
126. return $retval;
127. }
128. else
129. {
130. return new soap_fault('Database Server', '',
131. 'Koneksi ke database gagal!', '');
132. }
133. }
134. }
```

b. server.php

```
1.  <?php
2.  require_once('./lib/nusoap.php');
3.  require_once ('function.php');
4.  $namespace =
5.  "http://172.29.0.3/webservice/server.php";
6.  $server = new soap_server;
7.  $server->soap_defencoding = 'UTF-8';
8.  $server->configureWSDL('monitor_ap', $namespace);
9.  $server->wsdl->schemaTargetNamespace = $namespace;
10.
11. $server->wsdl->addComplexType(
12.         'data_users',
13.         'complexType',
```

```
14.         'struct',
15.         'all',
16.         '',
17.         array(
18.  'mac'=>array('name'=>'mac','type'=>'xsd:string'),
19.  'ip' =>array('name'=>'ip','type' => 'xsd:string'),
20.  'ap_mac'=>array('name'=>'ap_mac','type'=>
21.  'xsd:string')
22.  )
23.  );
24.  $server->wsdl->addComplexType(
25.         'data_users_array',
26.         'complexType',
27.         'array',
28.         '',
29.         'SOAP-ENC:Array',
30.         array(),
31.         array(
32.         array('ref'=>'SOAP-ENC:arrayType',
33.         'wsdl:arrayType'=>'tns:data_users[]'
34.         )
35.         ),
36.         'tns:data_users'
37.         );
38.  $server->register(
39.         'users',
40.         array('name' => 'xsd:string'),
41.         array('return'=>'tns:data_users_array'),
42.         'monitor_ap',
43.         'monitor_ap#user',
44.         'rpc',
45.         'encoded',
46.         'Data User Current'
47.         );
48.  $server->wsdl->addComplexType(
49.         'theAccessPoint',
50.         'complexType',
51.         'struct',
52.         'all',
53.         '',
54.   array(
55.   'mac'=>array('name'=>'mac','type'=> 'xsd:string'),
56.   'name'=>array('name'=>'name','type'=>
57.   'xsd:string'),
58.   'x' => array('name' => 'x', 'type' =>
59.   'xsd:double'),
60.   'y' => array('name' => 'y', 'type' =>
61.   'xsd:double')
62.  )
```

```
63. );
64. $server->wsdl->addComplexType(
65.         'theAccessPointArray',
66.         'complexType',
67.         'array',
68.         '',
69.         'SOAP-ENC:Array',
70.         array(),
71.         array(
72.         array('ref'=>'SOAP-ENC:arrayType',
73.         'wsdl:arrayType'=>'tns:theAccessPoint[]'
74.         )
75.         ),
76.         'tns:theAccessPoint'
77.         );
78. $server->register(
79.         'access_point',
80.         array('mac' => 'xsd:string'),
81.         array('return' =>
82.         'tns:theAccessPointArray'),
83.         'monitor_ap',
84.         'monitor_ap#AccessPoint',
85.         'rpc',
86.         'encoded',
87.         'Data Access Point'
88.         );
89. $server->wsdl->addComplexType(
90.         'data_admin',
91.         'complexType',
92.         'struct',
93.         'all',
94.         '',
95.         array(
96.         'name' => array('name' => 'name', 'type' =>
97.  'xsd:string'),
98.  'x_password' => array('name' => 'x_password',
99.  'type' => 'xsd:string')
100. )
101. );
102. $server->wsdl->addComplexType(
103.         'data_admin_array',
104.         'complexType',
105.         'array',
106.         '',
107.         'SOAP-ENC:Array',
108.         array(),
109.         array(
110.         array('ref'=>'SOAP-ENC:arrayType',
111.         'wsdl:arrayType'=>'tns:data_admin[]'
```

```
112.         )
113.         ),
114.         'tns:data_admin'
115.         );
116. $server->register('admin',
117.         array('name' => 'xsd:string'),
118.         array('return'=>'tns:data_admin_array'),
119.         'monitor_ap',
120.         'monitor_ap#admin',
121.         'rpc',
122.         'encoded',
123.         'Data Admin'
124.         );
125.
126. $server->wsdl->addComplexType(
127.         'data_users_daily',
128.         'complexType',
129.         'struct',
130.         'all',
131.         '',
132.         array(
133.         'ap' => array('name' => 'ap', 'type' =>
134.             'xsd:string'),
135.         'datetime' => array('name' => 'datetime',
136.             'type' => 'xsd:date'),
137.         array('name' => 'num_sta', 'type' =>
138.             'xsd:int')
139.         )
140.         );
141. $server->wsdl->addComplexType(
142.       'data_users_daily_array',
143.       'complexType',
144.       'array',
145.       '',
146.       'SOAP-ENC:Array',
147.       array(),
148.       array(
149.       array('ref'=>'SOAP-ENC:arrayType',
150.       'wsdl:arrayType'=>'tns:data_users_daily[]'
151.       )
152.       ),
153.     'tns:data_users_daily'
154.     );
155.     $server->register('daily',
156.     array('name' => 'xsd:string'),
157.     array('return'=>'tns:data_users_daily_array'),
158.     'monitor_ap',
159.     'monitor_ap#users_daily',
160.     'rpc',
```

```
161.     'encoded',
162.     'Data Users Daily'
163.     );
164.
165.$server->wsdl->addComplexType(
166.       'posisi_ap',
167.       'complexType',
168.       'struct',
169.       'all',
170.       '',
171.       array(
172. 'ap_mac' => array('name' => 'ap_mac', 'type' =>
173. 'xsd:string'),
174. 'x' => array('name' => 'x', 'type' =>
175. 'xsd:double'),
176. 'y' => array('name' => 'y', 'type' =>
177. 'xsd:doublr'),
178. 'name' => array('name' => 'name', 'type' =>
179. 'xsd:double'),
180. )
181. );
182. $server->wsdl->addComplexType(
183.       'posisi_ap_array',
184.       'complexType',
185.       'array',
186.       '',
187.       'SOAP-ENC:Array',
188.       array(),
189.       array(
190.       array('ref'=>'SOAP-ENC:arrayType',
191.       'wsdl:arrayType'=>'tns:posisi_ap[]'
192.       )
193.       ),
194.       'tns:posisi_ap'
195.       );
196. $server->register('posisi_ap',
197.       array('new_data' => 'xsd:string',
198.       'x' => 'xsd:double',
199.       'y' => 'xsd:double',
200.       'name' => 'xsd:string'
201.       ),
202. array('return'=>'tns:theDeviceArray'),
203.       'monitor_ap',
204.       'monitor_ap#update_posisi',
205.       'rpc',
206.       'encoded',
207.       'Update Posisi AP'
208.       );
209.
```

```
$GLOBALS['HTTP_RAW_POST_DATA'] = file_get_contents
('php://input');
$HTTP_RAW_POST_DATA =
$GLOBALS['HTTP_RAW_POST_DATA'];
$server->service($HTTP_RAW_POST_DATA);
exit();
?>
```

Lampiran 6. Aplikasi Pengguna

a. apmonitor\application\controllers\client.php

```
1.  <?php
2.  class Client extends CI_Controller {
3.  function index (){
4.  $service_wsdl =
5.  ('http://172.29.0.3/webservice/server.php?wsdl');
6.  $client = new nusoap_client ($service_wsdl);
7.  $client->soap_defencoding="UTF-8";
8.  $response_ap = $client->call('access_point');
9.  $response_user = $client->call('users');
10. $response_users_daily = $client->call('daily');
11. if ($client->fault){
12. echo "Code: { $client->faultcode }<br>";
13. echo "String: { $client->faultstring }";
14. }
15. else{
16. $this->load->library('Googlemaps');
17. $config['center'] = '112.615935, -7.956008';
18. $config['zoom'] = 'auto';
19. $config['minzoom']=16;
20. $config['maxzoom']=21;
21. $config['zoomControlPosition']='RIGHT';
22. $config['zoomControlStyle']='SMALL';
23. $config['cluster'] = TRUE;
24. $config['geocodeCaching'] = TRUE;
25. $config['minifyJS'] = TRUE;
26. $config['cluster'] = FALSE;
27. $this->googlemaps->initialize($config);
28.
29. $ip ='';
30. if (getenv("HTTP_CLIENT_IP"))
31. $ip = getenv("HTTP_CLIENT_IP");
32. else if(getenv("HTTP_X_FORWARDED_FOR"))
33. $ip = getenv("HTTP_X_FORWARDED_FOR");
34. else if(getenv("REMOTE_ADDR"))
35. $ip = getenv("REMOTE_ADDR");
36. else
37. $ip = "UNKNOWN";
38. foreach ($response_ap as $value) {
39. $marker = array();
40. $marker['position'] = $value['y'].','.$value['x'];
41. $marker['animation'] = 'DROP';
42.
43. $array_hari = array();
44. foreach ($response_users_daily as $value_daily) {
```

```
45.   if($value_daily['ap'] == $value['mac']){
46.   $array_hari[] = array($value_daily['datetime']*
47.   1000,$value_daily['num_sta']+0);
48.   }
49.   }
50.   $hari = array();
51.   foreach ($array_hari as $key => $row) {
52.   $hari[$key]  = $row[0];
53.   }
54.   array_multisort($hari,SORT_ASC,$array_hari);
55.   $month = array();
56.   foreach ($array_hari as $key => $row) {
57.   if(!array_key_exists(date('Y',
58.   $row[0]/1000)."_",$month)){
59.   $month[date('Y', $row[0]/1000)."_"] =
60.   array(0,0,0,0,0,0,0,0,0,0,0,0);
61.   }
62.   $month[date('Y', $row[0]/1000)."_"][date('m',
63.   $row[0]/1000)-1] += $row[1];
64.   }
65.   $marker['onclick'] ='
66.   document.getElementById("name").value =
      "'.$value['name'].'";
67.   createDaily('.json_encode($array_hari).');
68.   createHourly('.json_encode($array_jam).');
69.   createMonthly('.json_encode($month).');
70.   ';
71.   $jumlah = 0;
72.   foreach ($response_user as $jml) {
73.        if($jml['ap_mac']==$value['mac']){
74.        $jumlah++;
75.        if(array_key_exists("ip", $jml) &&
76.   $jml['ip']==$ip){
77.   $marker['animation'] = 'BOUNCE';
78.   $noip = '';
79.   }
80.   else{
81.   $noip = 'Anda Berada Di Luar Jangkauan';
82.   }
83.   }
84.   }
85.   $marker['icon'] =
86.   'http://chart.apis.google.com/chart?chst=d_map_pin_
87.   letter&chld='.$jumlah.'|00FF00|000000|';
88.   $this->googlemaps->add_marker($marker);
89.   }
90.   $data['map'] = $this->googlemaps->create_map();
91.   $this->googlemaps->initialize($config);
```

```
92.   $data['judul'] = 'UB Apmonitor';
93.   $data['noip'] = $noip;
94.   $data['count_user']= count($response_user);
95.   $data['response_ap'] = $response_ap;
96.   $data['resonse_users_daily'] =
97.   $response_users_daily ;
98.   $this->load->view('header_menu',$data);
99.   $this->load->view('dashboard',$data);
100.  }
101.  }
102.
103.  function login(){
104.      $this->load->view('login');
105.  }
106.  }
107.  ?>
```

b. apmonitor\application\controllers\admin.php

```
1.    <?php
2.    Class Admin extends CI_Controller{
3.    function __construct() {
4.       parent::__construct();
5.       $this->load->library('session');
6.       if (!$this->session->userdata('IS_LOGGED_IN')) {
7.       redirect('client/login');
8.    }
9.    }
10.   function index(){
11.   $service_wsdl =
12.   ('http://172.29.0.3/webservice/server.php?wsdl');
13.   $client = new nusoap_client ($service_wsdl);
14.   $response_device = $client->call('access_point');
15.   $response_user = $client->call('users');
16.   if ($client->fault){
17.   echo "Code: { $client->faultcode }<br>";
18.   echo "String: { $client->faultstring }";
19.   }
20.   else
21.   {
22.   $this->load->library('Googlemaps');
23.   $config['center'] = '112.615935, -7.956008';
24.   $config['zoom'] = 'auto';
25.   $config['minzoom']=16;
26.   $config['maxzoom']=21;
```

```
27.  $config['zoomControlPosition']='RIGHT';
28.  $config['zoomControlStyle']='SMALL';
29.  $config['geocodeCaching'] = TRUE;
30.  $config['minifyJS'] = TRUE;
31.  $config['cluster'] = FALSE;
32.  $this->googlemaps->initialize($config);
33.  foreach ($response_device as $value) {
34.  $marker = array();
35.  $marker['position'] = $value['y'].','.$value['x'];
36.  $marker['draggable'] = TRUE;
37.  $marker['ondragend'] = '
38.  document.getElementById("x").value =
39.  event.latLng.lat();
40.  document.getElementById("y").value =
41.  event.latLng.lng();
42.  document.getElementById("mac").value =
43.  "'.$value['mac'].'";
44.  document.getElementById("name").value =
45.  "'.$value['name'].'";
46.  simpanPosisi();
47.  ';
48.  $marker['animation'] = 'DROP';
49.  $jumlah = 0;
50.  foreach ($response_user as $jml) {
51.  if($jml['ap_mac']==$value['mac'])
52.  $jumlah++;
53.  }
54.  $marker['infowindow_content'] = $value['name'];
55.  $marker['icon'] =
56.  'http://chart.apis.google.com/chart?chst=d_map_pin_
57.  letter&chld='.$jumlah.'|7FFF00|000000|';
58.  $this->googlemaps->add_marker($marker);
59.  }
60.  $data ['judul'] = 'AP Monitor';
61.  $data['count_user']= count($response_user);
62.  $data['count_aktif']= count($response_device);
63.  $data['map'] = $this->googlemaps->create_map();
64.  $this->load->view('header_menu_admin', $data);
65.  $this->load->view('admin');
66.  $this->load->view('footer');
67.  }
68.  }
69.
70.  function logout() {
71.  $this->load->library('session');
```

```
72.   $this->session->unset_userdata('IS_LOGGED_IN');
73.   $this->session->sess_destroy();
74.   session_destroy();
75.   redirect('client');
76.   }
77.   }
78.   ?>
```

c. apmonitor\application\views\admin.php

```
1.     <script type="text/javascript">
2.           function simpanPosisi(){
3.                       $("#myModal").modal();
4.           }
5.     </script>
6.
7.    <?php echo $map['js']; ?>
8.    <aside class="right-side">
9.    <section class="content">
10.     <div class="row">
11.       <section class="col-xs-12 connectedSortable">
12.     <div class="box box-primary">
13.     <div class="box-header">
14.     <i class="fa fa-map-marker"></i>
15.     <h3 class="box-title">
16.         Denah Universitas Brawijaya
17.     </h3>
18.     <div class="box-tools">
19.     <div class="input-group">
20.     <input type="text" name="table_search"
21.     class="form-control input-sm pull-right"
22.     id="search" style="width: 150px;"
23.     placeholder="Search"/>
24.     <div class="input-group-btn">
25.    <button class="btn btn-sm btn-default"><i
26.    class="fa fa-search"></i></button>
27.    </div>
28.    </div>
29.    </div>
30.    </div>
31.    <div class="box-body no-paddi">
32.    <?php echo $map['html']; ?>
33.   </div>
34.   </div>
35.   </section>
36.   </div>
37.   <div class="modal fade" id="myModal" tabindex="-1"
38.   role="dialog" aria-labelledby="myModalLabel" aria-
```

```
39.   hidden="true">
40.
41.   <form class="form-horizontal" role="form"
42.   action="<?php echo
43.   base_url("proses_save_posisi")?>" method="post">
44.   <div class="modal-dialog">
45.      <div class="modal-content">
46.      <div class="modal-header">
47.      <button type="button" class="close" data-
48.      dismiss="modal"><span aria-
49.      hidden="true">&times;</span><span class="sr-
50.      only">Close</span></button>
51.    <h4 class="modal-title" id="myModalLabel">Save
52.    Posisi dan Nama Access Point</h4>
53.    </div>
54.    <div class="modal-body">
55.    <div class="form-group has-error">
56.    <label for="mac">Mac Address AP</label>
57.    <input type="text" class="form-control" id="mac"
58.    name="mac" placeholder="Mac Addr">
59.    </div>
60.    <div class="form-group has-error">
61.    <label for="x">Latitude</label>
62.    <input type="text" class="form-control" id="x"
63.    name="x" placeholder="Latitude">
64.    </div>
65.    <div class="form-group has-error">
66.    <label for="x">Longitude</label>
67.    <input type="text" class="form-control" id="y"
68.    name="y" placeholder="Langtitude">
69.    </div>
70.    <div class="form-group has-success">
71.    <label for="name">Name AP</label>
72.    <input type="text" class="form-control" id="name"
73.    name="name" placeholder="Name Acces Point">
74.   </div></div>
75.   <div class="modal-footer">
76.   <button type="button" class="btn btn-default" data-
77.   dismiss="modal">Close</button>
78.   <input type="submit" class="btn btn-primary"
79.   value="Save changes">
80.   </div></div></div></form></div></section>
81.   </aside>
```

d. apmonitor\application\views\dashboard.php

```
1.    <?php echo $map['js']; ?>
2.    <section class="col-lg-12">
3.    <div class="row">
```

```
4.    <section>
5.    <?php echo $map['html']; ?>
6.    </section>
7.    </div>
8.    <div class="modal fade" id="myModal" tabindex="-1"
9.    role="dialog" aria-labelledby="myModalLabel" aria-
10.   hidden="true">
11.   <div class="modal-dialog">
12.   <div class="modal-content">
13.   <div class="modal-header">
14.   <button type="button" class="close" data-
15.   dismiss="modal" aria-label="Close"><span aria-
16.   hidden="true">&times;</span></button>
17.   <center><input disabled type="" class="modal-title"
18.   id="name" readonly></center>
19.   </div><div>
20.   <ul class="nav nav-tabs" role="tablist" id="myTab">
21.   <li role="presentation"><a href="#daily" aria-
22.   controls="daily" role="tab" data-
23.   toggle="tab">Daily</a></li>
24.   <li role="presentation"><a href="#monthly" aria-
25.   controls="monthly" role="tab" data-
26.   toggle="tab">Monthly</a></li>
27.   </ul>
28.   <div class="tab-content">
29.   <div role="tabpanel" class="tab-pane" id="daily"
30.   style="min-width: 550px; height: 400px; margin: 0
31.   auto"></div>
32.   <div role="tabpanel" class="tab-pane" id="monthly"
33.   style="min-width: 550px; height: 400px; margin: 0
34.   auto"></div>
35.   </div></div>
36.   <div id="daily" style="min-width: 480px; height:
37.   400px; margin: 0 auto"></div>
38.   </div></div></div>
39.   </section><body>
40.   </html>
41.
42.   <script>
43.         $(function () {
44.         $('#myTab a:last').tab('show')
45.    })
46.   </script>
47.
48.   <script type="text/javascript">
49.   function createDaily(datas) {
50.   $('#daily').highcharts('StockChart', {
51.         rangeSelector : {
```

```
            selected : 0,
            inputEnabled: $('#daily').width() > 480
                },
                title : {
                   text : 'Jumlah Users Setiap Hari'
                },
                series : [{
                  name : 'Users',
                  data : datas,
                  tooltip: {
                    valueDecimals: 0
                  }
              }]
            });
          $('#myModal').modal('show');
      }
  </script>

  <script>
  function createMonthly(month) {
  var parsing = [];
   $.each(month,function(index, value){
         parsing.push({"name":index,"data":value});
   });
      $('#monthly').highcharts({

          title: {
              text: 'Jumlah Users Setiap Bulan',
              x: -20 //center
          },
   xAxis: {
   categories: ['Jan', 'Feb', 'Mar', 'Apr', 'May',
   'Jun',
   'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
   },
   yAxis: {
     title: {
     text: 'Jumlah Pengguna'
     },
     plotLines: [{
     value: 0,
     width: 1,
     color: '#808080'
     }]
     },
    legend: {
    layout: 'vertical',
    align: 'right',
```

```
100.    verticalAlign: 'middle',
101.    borderWidth: 0
102.    },
103.    series:parsing
104.    });
105.  }
106.  </script>
```