PENGEMBANGAN OTENTIKASI TOKEN DAN MANAJEMEN USER UNTUK SISTEM *SMART AGRICULTURE*

Pra Pendadaran

untuk memenuhi sebagian persyaratan
mencapai derajat Sarjana S-2

Program Studi S2 Teknik Elektro
Konsentrasi Teknologi Informasi
Departemen Teknik Elektro dan Teknologi Informasi

diajukan oleh
Arif Setiawan
13/356785/PTK/9213

Kepada

PROGRAM PASCASARJANA
FAKULTAS TEKNIK
UNIVERSITAS GADJAH MADA
YOGYAKARTA
2017

PERNYATAAN

Dengan ini saya menyatakan bahwa:

1. Tesis ini tidak mengandung karya yang diajukan untuk memperoleh gelar kesarjanaan di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak mengandung karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

2. Informasi dan materi tesis yang terkait hak milik, hak intelektual dan paten merupakan milik bersama antara tiga pihak yaitu penulis, dosen pembimbing dan Universitas Gadjah Mada. Dalam hal penggunaan informasi dan materi tesis terkait paten maka akan diskusikan lebih lanjut untuk mendapatkan persetujuan dari ketiga pihak tersebut diatas.

Yogyakarta, 16 Mei 2017

Arif Setiawan

PRAKATA

Puji syukur ke hadirat Allah SWT yang telah melimpahkan rahmat dan barokah-Nya sehingga penulis dapat menyelesaikan tesis dengan judul "Pengembangan Otentikasi Token dan Manajemen User untuk Sistem *Smart Agriculture* ". Laporan tesis ini disusun untuk memenuhi salah satu syarat dalam memperoleh gelar *Master of Engineering (M.Eng.)* pada Program Studi S2 Teknik Elektro Fakultas Teknik Universitas Gadjah Mada Yogyakarta.

Dalam melakukan penelitian dan penyusunan laporan tesis ini penulis telah mendapatkan banyak dukungan dan bantuan dari berbagai pihak. Penulis mengucapkan terima kasih yang tak terhingga kepada:

1. I Wayan Mustika, S.T., M.Eng., Ph.D selaku dosen pembimbing utama, dan Teguh Bharata Adji, S.T., M.T., M.Eng., Ph.D selaku dosen pembimbing pendamping,yang telah dengan penuh kesabaran dan ketulusan memberikan ilmu dan bimbingan terbaik kepada penulis.
2. Dr. Eng. Suharyanto, S.T., M.Eng selaku Ketua Departemen Teknik Elektro dan Teknologi Informasi dan Dr. Eng. Ir. Risanuri Hidayat, M.Sc. selaku Ketua Program Studi S2 Teknik Elektro Fakultas Teknik Universitas Gadjah Mada yang memberikan izin kepada penulis untuk belajar.
3. Para Dosen Program Studi S2 Teknik Elektro Fakultas Teknik Universitas Gadjah Mada yang telah memberikan bekal ilmu kepada penulis.
4. Para Karyawan/wati Program Studi S2 Teknik Elektro Fakultas Teknik Universitas Gadjah Mada yang telah membantu penulis dalam proses belajar.

Penulis menyadari sepenuhnya bahwa laporan tesis ini masih jauh dari sempurna, untuk itu semua jenis saran, kritik dan masukan yang bersifat membangun sangat penulis harapkan. Akhir kata, semoga tulisan ini dapat memberikan manfaat dan memberikan wawasan tambahan bagi para pembaca dan khususnya bagi penulis sendiri.

Yogyakarta, 12 Mei 2017

Arif Setiawan

ARTI LAMBANG DAN SINGKATAN

6LowPAN	= IPv6 Over Low Power Area Network
API	= Application Programming Interface
CSRF	= Cross-Site Request Forgery
HMM	= Health Monitoring Management
HTTP	= Hypertext Transfer Protocol
IoT	= Internet Of Things
JSON	= JavaScript Object Notation
MITM	= Man in the Middle
REST	= Representational State Transfer
RFID	= Radio Frequency Identification
ROA	= Resource Oriented Architecture
SOA	= Service Oriented Architecture
SOAP	= Simple Object Access Protocol
URI	= Uniform Resource Identifiers
XSS	= Cross-Site Scripting

ABSTRACT

Nowadays, The Internet of Things (IoT) sensor which can be set to collect data every minute and second make the IoT be the first choice for systems that require continuous monitoring. One of the most common data communication methods applied to IoT is Representational State Transfer (REST). REST has the advantage of being able to support different standard devices. However, the security aspect has not been a major concern in this topic. In this research, REST authentication system will be developed by using token based mechanism on smart agriculture system.. In every request that occurs, the client must include a registered token for the request can be served. In addition, user management will also be implemented based on user roles. Access from user to resource will be limited based on their role to improve the security of data. This research show thaty, the created system can improve the security aspects of IoT applications based on REST communication especially on smart agriculture system.

Keyword: REST, web service, internet of things, authentication, role based access control

INTISARI

Mudahnya sensor *Internet of Things* (IoT) yang dapat diatur untuk mengumpulkan data setiap menit bahkan detik membuat penggunaan IoT menjadi pilihan utama untuk sistem yang membutuhkan monitoring secara terus-menerus. Salah satu metode komunikasi data yang paling umum diterapkan pada IoT adalah Representational State Transfer (REST). REST memiliki keunggulan mampu mendukung perangkat-perangkat yang berbeda standar. Namun, di beberapa penelitian aspek keamanan belum menjadi topik perhatian utama. Di dalam penelitian ini, akan dikembangkan otentikasi REST dengan menggunakan token pada sistem *smart agriculture*. Mekanismenya dalam setiap request yang terjadi, klien harus menyertakan token yang terdaftar agar request tersebut dilayani. Selain itu, akan diterapkan juga manajemen user berdasar peran pengguna. Akses dari pengguna ke *resource* akan dibatasi berdasar perannya untuk meningkatkan keamanan privacy data. Hasil dari penelitian ini, sistem yang dibuat mampu meningkatkan aspek keamanan pada aplikasi IoT yang menggunakan metode komunikasi REST khususnya pada sistem *smart agriculture.*

Kata kunci – REST, *web service, internet of things*, otentikasi, *role based access control*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Internet Of Things (IoT) menjadi salah satu teknologi yang ramai diperbincangkan saat ini. Kemajuannya yang pesat dan penerapannya yang bisa digunakan di semua bidang menjadikan IoT menjadi salah satu teknologi yang paling berkembang. Menurut hasil studi dari Gartner [1], perusahaan riset dan teknologi dari Amerika Serikat. Pada tahun 2017 ini akan ada 1,5 miliar perangkat baru yang terhubung ke internet. Jumlah tersebut akan meningkat hingga 20 miliar perangkat pada tahun 2020. Perangkat IoT sendiri dapat dibedakan menjadi 3 kategori, yaitu *Wearables*, Perangkat *Smart Home* dan Perangkat M2M *(Machine to Machine)*.

Wearables merupakan perangkat yang selalu dibawa oleh pengguna. Biasanya terhubung melalui koneksi *Bluetooth* ke perangkat seluler yang kemudian tersambung ke Internet. Perangkat di kategori ini termasuk jam pintar dan fitness band. Perangkat *Smart Home* merupakan perangkat yang bisa ditemukan di dalam rumah. Perangkat ini meliputi *motion sensor*, pintu dan saklar otomatis hingga oven. Kategori ketiga yaitu M2M merupakan perangkat yang langsung terhubung ke jaringan seperti mobil yang mampu memberitahu lokasinya saat terjadi kecelakaan, atau sebuah kulkas yang mampu memesan sendiri ketika stok buah yang disimpannya habis.

Dengan semakin banyaknya perangkat IoT yang terhubung secara *online*, pekerjaan manusia tentu akan terbantu. Namun dilain pihak, juga akan menimbulkan masalah baru ketika perangkat-perangkat yang terhubung tersebut memiliki tingkat keamanan yang rendah. Perangkat IoT yang dikuasai *hacker* dapat diubah menjadi *botnet*. *Botnet* ini mampu dikendalikan dari jarak jauh oleh *hacker* untuk melakukan serangan *Distributed Denial Of Service Attacks* (DDOS) ke jaringan tertentu. Pada akhir tahun 2016 kemarin, Dyn sebuah perusahaan provider

Domain Operated System (DNS) mengalami serangan DDOS pada server mereka. Hal ini mengakibatkan situs-situs besar yang menggunakan layanan DSN Dyn seperti Amazon, Airbnb, CNN, Netflix dan Spotify tidak dapat diakses oleh pengguna. Setelah dilakukan investigasi menyeluruh, ditemukan bahwa serangan tersebut dilakukan lebih dari 100.000 perangkat IoT yang terkena malware Mirai *botnet* [2].

Berkaca dari hal diatas, sisi keamanan dari IoT perlu ditingkatkan agar kasus tersebut tidak terulang kembali. Baik dari sisi perangkat itu sendiri, komunikasi data ataupun dari sisi *middleware*. *Middleware* merupakan penghubung antar komponen dalam IoT sehingga setiap komponen tersebut dapat berkomunikasi. Dalam implementasinya, *middleware* bisa diterapkan dalam bentuk *Application Programming Interface* (API). API merupakan penghubung antara bagian *backend* dan *frontend*. Di pengembangan aplikasi berbasis IoT, API menjadi sistem penghubung antara sensor dengan basis data, ataupun basis data dengan antarmuka aplikasi. Penggunaan API diimplementasikan dalam bentuk *web service*. Generasi pertama *web service* yang diperkenalkan adalah *Simple Object Acces Protoco*l (SOAP) namun karena perkembangan perangkat IoT yang semakin banyak dan SOAP tidak mampu menghandle perangkat yang berbeda standar maka penggunaan SOAP mulai ditinggalkan.

Representational State Transfer (REST) menjadi pengembangan selanjutnya dari *web service*. REST web service / RESTful memiliki keunggulan mampu mendukung perangkat-perangkat yang berbeda standar karena menggunakan basis *Resource Oriented Architecture* (ROA) [3]. REST sendiri memiliki 4 attribute yaitu :

1. *Addresssability*

   Semua *resource* akan diimplementasikan menggunakan *Uniform Resource Identifiers* (URI). Setiap *resource* tersebut akan memiliki alamat URI sendiri. Ketika alamat URI dipanggil dia akan mengembalikan respon dalam bentuk JSON atau XML.

2. *Connectedness*

   Resource yang ada dalam REST harus memiliki relasi dengan *resource* yang lain agar dapat dipresentasikan melalui URI.

3. *Homogeneus Interface*

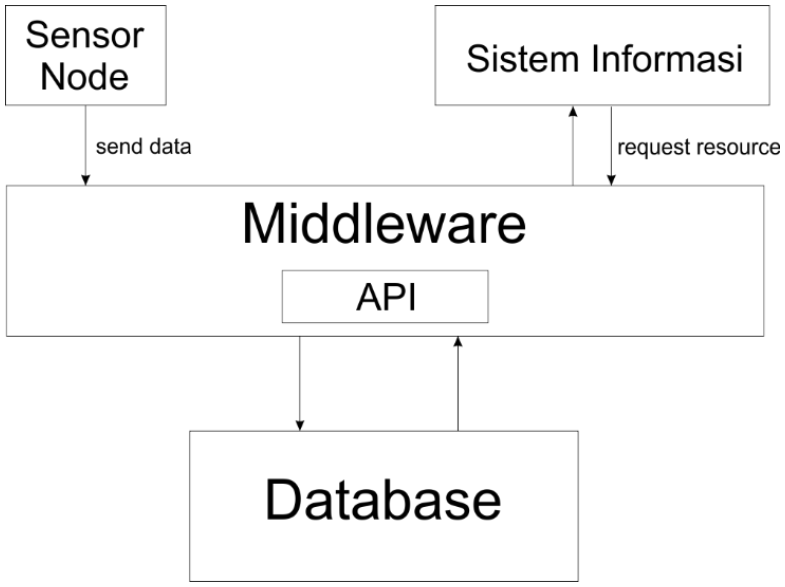
   Resource akan dipanggil menggunakan 4 metode HTTP yaitu GET, PUT, POST dan DELETE dengan 2 tambahan metode yaitu HEAD dan OPTION. HEAD digunakan untuk menunjukkan metadata sedang OPTION digunakan untuk memeriksa metode yang ada

4. *Statelessness*

   *Statele*ss menunjukkan bahwa server tidak menyimpan data dari klien dari setiap koneksi yang terbentuk.

Protokol REST ini banyak digunakan sebagai standar komunikasi data pada sistem-sistem IoT antara lain pada bidang *smart agriculture*. Salah satu bidang *smart agriculture* yang menggunakan RESTful adalah Smart Farm[4]. Pada Smart Farm, perangkat IoT digunakan sebagai alat monitoring pada perkebunan kelapa sawit yang dapat diakses melalui http://smartcity.wg.ugm.ac.id/webapp/smartfarm/index.php. Ilustrasi sistem ini dapat dilihat pada Gambar 1.1. Ada 4 bagian utama didalam sistem ini yaitu : Sensor Node, Gateway, *Back End* dan *Front End.* Sensor node akan memonitoring kondisi lingkungan dan mengambil data berupa suhu udara, suhu tanah, kelembapan tanah, kelembapan udara, intensitas cahaya matahari, curah hujan, arah angin, kecepatan angin, ketinggian air dan kelembapan daun tanaman kelapa sawit. Data-data tersebut akan dikirimkan secara nirkabel melalui gateway yang akan disimpan kedalam *Back End* atau database. Agar data dari database tersebut dapat divisualisasikan dengan baik maka perlu adanya layanan *Front End* atau Sistem Informasi. Untuk menjembatani antara *Back End* dan *Front End* tersebut maka diperlukan pengembangan API menggunakan protokol REST. API didalam sistem Smart Farm ini memiliki 2 fungsi yaitu sebagai jalur penghubung antara sensor node dengan database dan penghubung antara database dengan Sistem informasi.

Di dalam sistem Smart Farm ini, REST sudah dikembangkan secara baik namun masih memiliki beberapa kekurangan karena dalam pengembangan sistemnya aspek keamanan data belum menjadi prioritas. Sebagai contoh, belum adanya otentikasi *user* saat mengakses API tersebut, sehingga jika alamat URI dari API tersebut diketahui maka dapat diakses oleh semua orang. Masalah yang lain yaitu setiap *user* bisa melihat semua data dalam basis data. Baik itu data dari sensor node publik maupun data sensor node milik pengguna lain. Untuk itu, penelitian ini bertujuan untuk membangun layanan otentikasi dan layanan manajemen *user* untuk mengatur dan membatasi hak akses dari *user* itu sendiri.



Gambar 1.1 Konsep sistem Smart Farm

1.2 Perumusan masalah

Berdasar latar belakang diatas maka maka dirumuskan masalah sebagai berikut :

1. Data API pada system Smart Farm masih bisa diakses oleh semua *user* selama user mengetahui alamat URI
2. Perlunya penambahan metode otentikasi didalam protokol REST yang digunakan

3. Belum ada pembagian user berdasarkan peran dari pengguna

1.3 Keaslian penelitian

Penelitian - penelitian di bidang *Internet of Things* yang menggunakan arsitektur metode komunikasi REST sudah banyak dipublikasikan. Di bidang kesehatan ada *Health Monitoring Management* (HMM) yang digunakan untuk memantau tanda-tanda vital pasien [5], Di bidang property, IoT sudah diterapkan pada *Smart Home* untuk melakukan kontrol dan pemantauan alat-alat rumah tangga [6] dan juga sebagai pemantau suhu dan temperature [7]. Di bidang pertanian, ada Smart Farm yang digunakan untuk memantau kondisi perkebunan kelapa sawit [4].

Namun dari beberapa penelitian yang menggunakan REST sebagai metode komunikasinya, sebagian besar masih fokus dalam pemanfaatan metode tersebut sebagai penghubung antara sensor dengan aktuator maupun sensor dengan database. Faktor keamanan pada penelitian-penelitian tersebut masih belum menjadi perhatian. Dalam Tabel 1.1 dibawah ini, beberapa penelitian yang berhubungan dengan keamanan penggunakan metode komunikasi REST antara lain:

Tabel 1.1 Tabel Keaslian Penelitian

No	**Judul**	**Pengarang**	**Metode Otentikasi**
1	HTTP Authentication : Basic dan Digest Access Authentication (1999)	Lawrence et al[8]	Username dan Password
2	An Extended Username Token-based Approach for REST-style Web Service Security Authentication	Dunlu Peng et al[9]	Username dan Password + Hash

	(2009)		
3	A token-based user authentication mechanism for data exchange in RESTful API (2015)	Xiang-Wen Huang et al [10]	Token
4	A Method for Secure RESTful Web Service (2015)	Sungchul Lee et al[3]	Token dan ID Based Encryption
5	Study on Access Permission Control for the Web of Things (2015)	Se Won Oh et al[11]	Manajemen Akses Kontrol berdasar Web of Thing
6	Security Analysis and proposal of new Access Control model in the Internet of Thing (2015)	Ouaddah et al [12]	Manajemen Akses Kontrol berdasar Organization Based Access Control (OrBAC)

Dari Tabel 1.1 diketahui penggunaan *username* menjadi salah satu metode otentikasi pada REST. Namun dalam perkembangannya metode tersebut mulai ditinggalkan dan beralih ke penggunaan Token. Manajemen hak akses juga sudah mulai diperkenalkan dalam sebuah peneltian namun belum secara spesifik membahas implementasinya didalam aplikasi IoT.

Pada penelitian ini akan digunakan 2 metode untuk meningkatkan keamanan metode komunikasi REST yaitu penggunaan token sebagai otentikasi

serta pembatasan hak akses *resource* berdasar dari peran user. Penerapan kedua metode tersebut dalam bidang IoT berserta tantangan-tantangannya akan menjadi bahasan utama dalam penelitian ini. Hasil dari penelitian ini diharapkan dapat memberikan kontribusi dalam bidang keamanan *Internet of Things,* serta memberi arah pengembangan aplikasi-aplikasi *Internet of Things* di masa mendatang.

1.4 Tujuan Penelitian

Penelitian ini bertujuan :

1. Mengembangkan metode otentikasi pada API Smart Farm berdasar mekanisme token
2. Membuat manajemen user berdasar peran user terhadap *resource* dalam sistem Smart Farm

1.5 Batasan Masalah

Penelitian ini memiliki batasan masalah sebagai berikut :

1. Difokuskan pada sistem API yang telah ada
2. Pengembangan otentikasi menggunakan mekanisme token
3. Layanan hak ases user berdasar peran pengguna dalam mengakses *resource* dibatasi pada *resource sensor node*
4. API penghubung sensor node dengan database tidak menjadi perhatian
5. Hardware dan *sensor node* yang digunakan tidak menjadi perhatian dalam penelitian ini
6. Perancangan sistem informasi dan *user interface* tidak menjadi perhatian dalam penelitian ini

1.6 Manfaat Penelitian

Dengan penelitian ini, maka diharapkan akan memberikan manfaat antara lain :

1. Sistem Smart Farm yang telah dibuat akan lebih aman karena hanya user yang tervalidasi yang memiliki hak akses
2. Sistem mendukung multi user dan memiliki pembagian user, setiap user memiliki hak akses sendiri terhadap nodes yang dimiliki
3. Sistem keamanan API yang dibuat dapat dikembangkan lagi untuk penelitian selanjutnya

BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Metode komunikasi *Web Service* sudah semakin banyak digunakan sekarang ini. Terutama dengan semakin banyaknya peralatan-peralatan yang berbasis IoT membuat penerapan *Web Service* menjadi semakin masif. *Simple Object Access Protocol* (SOAP) merupakan generasi pertama web service yang diperkenalkan. Namun karena perkembangan perangkat IoT yang semakin banyak dan SOAP tidak mampu menghandle perangkat yang berbeda standar maka penggunaan SOAP mulai ditinggalkan. Pengembangan web service selanjutnya yaitu *Representational State Transfer* (REST). REST ini memiliki keunggulan mampu menghandle perangkat-perangkat yang berbeda standar karena dalam komunikasinya menggunakan metode HTTP umum yaitu GET, PUT, POST, DELETE, HEAD dan OPTION.

Penerapan metode REST juga sudah banyak digunakan pada penelitian - penelitian di bidang Internet of Things. Di bidang kesehatan ada *Health Monitoring Management* (HMM) yang digunakan untuk memantau tanda-tanda vital pasien [5], Di bidang property, IoT sudah diterapkan pada *Smart Home* untuk melakukan kontrol dan pemantauan alat-alat rumah tangga [6] dan juga sebagai pemantau suhu dan temperature [7]. Di bidang pertanian, ada Smart Farm yang digunakan untuk memantau kondisi perkebunan kelapa sawit [4]. Namun dari beberapa penelitian yang menggunakan REST sebagai metode komunikasinya, sebagian besar masih fokus dalam pemanfaatan metode tersebut sebagai penghubung antara sensor dengan aktuator maupun sensor dengan basis data. Faktor keamanan pada penelitian-penelitian tersebut masih belum menjadi perhatian.

Karena menggunakan dasar HTTP sebagai komunikasinya, maka metode REST memiliki kerentanan yang sama dengan beberapa aplikasi web yang lain. Menurut Femke [13], beberapa serangan yang bisa dilakukan terhadap komunikasi REST, yaitu :

1. *SQL Injection*

   Metode komunikasi REST sangat bergantung pada komunikasi HTTP. Jika tidak ada mekanisme untuk melakukan validasi pada data yang masuk, maka sistem akan rentan terhadap serangan *SQL Injection*

2. *Serangan Man-in-the-Middle* (MITM)

   Serangan yang terjadi ketika penyerang membuat sebuah koneksi baru kepada user, membuat user seolah-seolah sedang berkomunikasi dengan server yang asli.

3. *Replay Attack*

   Serangan yang terjadi pada jaringan dimana penyerang mengambil informasi yang bersifat rahasia seperti otentifikasi , lalu penyerang menggunakan informasi tersebut untuk pura-pura menjadi klien yang ter-otentifikasi

4. *Spoofing*

   Serangan dimana penyerang berpura-pura menjadi host yang dapat dipercaya pada suatu jaringan. Teknik ini dapat digunakan oleh penyerang untuk memalsukan data yang diminta oleh klien

5. *Cross-Site Scripting* (XSS) dan *Cross-Site Request Forgery* (CSRF)

   XSS dan CSRF merupakan 2 serangan yang sama-sama ditujukan melalui browser kepada klien. Serangan ini mampu mencuri otentikasi dari klien atau memanipulasi konten yang dikirim dari server

REST merupakan metode komunikasi yang *stateless* dimana setiap request yang terjadi bersifat independen sehingga server harus melalukan otentikasi klien setiap kali request terjadi. *Stateless* juga berarti tidak ada session yang disimpan selama otentikasi dilakukan. Hal ini mengakibatkan otentikasi berdasar protokol HTTP menjadi tidak memadai. Beberapa penelitian mulai dilakukan untuk menemukan metode yang relevan untuk menghindari serangan-serangan terhadap komunikasi REST seperti diatas.

Penelitian pertama menggunakan metode HTTP *Basic Authentication*[8]. Metode ini menggunakan server HTTP untuk melakukan otentikasi terhadap Web Browser. Ketika klien melakukan request terhadap resource maka server akan meminta identitas dari klien tersebut. Identitas ini berupa *username* dan *password*. Ketika klien memberikan identitas yang sesuai maka server akan memberikan response berupa *resource* yang diminta. Namun metode ini memiliki kelemahan yaitu tidak ada enkripsi terhadap request yang dilakukan.

Metode HTTP *Basic Authentication* kemudian dikembangkan menjadi HTTP *Digest Authentication*. Proses otentikasi yang dilakukan sama dengan *Basic Authentication* namun mekanisme yang dilakukan lebih komplek. Ketika server meminta identitas dari klien, maka klien akan memberikan *username* dan *password* yang ditambah dengan *hash string* seperti MD5 [9]. Proses otentikasi pada HTTP *Digest Authentication* seperti berikut. Pertama klien akan melakukan request kepada server dan server akan memberikan *nonce* (kata acak) kepada klien. Kemudian proses kedua, klien akan menggabungkan *username, pasword* dan *nonce* tersebut untuk membuat *hash*. *Hash* tersebut akan dikirim kembali dari klien ke server. Oleh server, *hash* dari akan dibandingkan dengan *hash* yang dibuat sendiri oleh server berdasar *username* dan *password* klien. Jika hash tersebut memiliki nilai yang sama maka klien akan diberikan akses terhadap *resource*. Metode HTTP *Digest Authentication* ini lebih aman jika dibandingkan dengan HTTP *Basic Authentication* namun memiliki kelemahan karena penyerang dapat melakukan serangan MITM [3].

Pengembangan selanjutnya yaitu penggunaan token sebagai otentikasi. Metode ini diklaim lebih aman dari 2 metode sebelumnya karena tidak menggunakan *username* dan *password*. Proses otentikasi pada metode ini yaitu sebagai berikut. Pertama, klien melakukan request kepada server dengan menggunakan *username* dan *password*. Server akan memberikan response berupa token. Token ini akan digunakan oleh klien setiap melakukan request *resource* kepada server. Token bersifat acak dan tidak berelasi apapun dengan data klien yang ada sehingga lebih aman.

Pengembangan metode ini dilanjutkan oleh Huang et al [10] dengan penambahan *timestamp* di token pada setiap request yang terjadi. Token yang ditambahkan *timestamp* disebut *disposable* token. Dengan penambahan *timestamp*, maka token yang digunakan hanya berlaku dalam waktu tertentu. Sehingga mengurangi terjadinya resiko serangan. Kelemahan dari metode ini yaitu klien dan server harus membuat *disposable* token setiap request baru sehingga akan membebani resource dari sistem. Metode ini tidak cocok diterapkan pada sistem IoT karena sebagian besar perangkat IoT memiliki spesifikasi yang rendah.

Metode lain dikembangkan oleh Lee et al [3]. Metode ini merupakan pengembangan token yang dipadukan dengan otentikasi berdasar *Identity-Based Encryption* yang dicetuskan oleh Boneh dan Franklin [14]. Pada metode ini terdapat 4 tahap yaitu *Setup, Extract, Encrypt* dan *Decrypt*. Semua otentikasi dan otorisasi klien akan dilakukan melalui public dan private key yang digenerate oleh *Public Key Generator* (PKG). Dengan metode ini maka server tidak memerlukan session ID atau *username* dan *password* dari klien. Namun metode ini masih sebatas konsep dan belum diimplementasikan dalam aplikasi IoT.

Pengembangan lain dari otentikasi klien dalam metode komunikasi REST yaitu otorisasi klien untuk melakukan request terhadap *resource* yang ada. Permasalahan keamanan dan privacy data menjadi fokus utama pada hal ini. Beberapa penelitian yang membahas tentang kontrol akses klien terhadap resource URI sudah diusulkan seperti WOT *Access Control* [11] dan SmartOrBAC [12]. WOT *Access Control* menerapkan sistem yang tersebar (*Decentralized Access Control*) dimana setiap request yang terjadi akan memberikan *response permission resource* yang boleh atau tidak diakses. Sedangkan SmartOrBAC merupakan kebalikannya dengan menerapkan sistem kontrol yang terpusat. SmartOrBAC membuat aturan berisi list-list klien yang memiliki permission untuk melakukan request terhadap resource di server. Metode tersebar dan terpusat ini masih menjadi perdebatan dikalangan peneliti dalam hal penerapan ke dalam sistem IoT sehingga masih diperlukan penelitian lebih lanjut.

2.2 Landasan Teori

Pada bagian ini akan dijelaskan teori-teori yang mendasari penelitian ini seperti konsep *Internet of Things*, *Middleware*, *Web Service*, REST, JSON, ROA dan Slim Framework

2.2.1 Internet of Things

Internet of things merupakan kata yang dipopulerkan pertama kali oleh Kevin Ashton pada 1999 [15]. Kevin Ashton merupakan pekerja di divisi Supply Chain Optimization pada perusahaan Procter & Gamble. Pada awalnya dia ingin menarik perhatian manajemen dengan teknologi *Radio Frequency Identification* (RFID). Namun karena pada masa itu internet sedang menjadi trend maka dia menamai teknologi RFID tersebut dengan nama *"Internet of Things"*.

Definisi dari IoT sendiri menurut McKinsey[16] adalah kumpulan sensor dan aktuator yang terpasang pada benda fisik yang saling terhubung baik melalui jaringan kabel atau nirkabel ke internet. Konsep IoT mulai mencapai popularitasnya pada tahun 2010. Ketika aplikasi Google StreetView diketahui tidak hanya menyimpan foto dari jalan-jalan di kota namun juga data wifi milik orang-orang sekitar. Pada tahun itu juga pemerintah Cina mengumumkan akan menggunakan teknologi IoT untuk strategi pemerintahannya selama 5 tahun kedepan. Menurut hasil studi dari Gartner [1], perusahaan riset dan teknologi dari Amerika Serikat. Pada tahun 2017 ini akan ada 1,5 miliar perangkat baru yang terhubung ke internet. Jumlah tersebut akan meningkat hingga 20 miliar perangkat pada tahun 2020.

Pada awal mulanya teknologi IoT terbatas pada RFID saja dan terbatas untuk digunakan pada perusahaan-perusahaan besar. Sekarang ini, teknologi IoT sudah hampir digunakan pada berbagai bidang. Berikut ini beberapa aplikasi teknologi IoT yang ada di sekitar kita menurut Knud[15] :

1. *Smart Home*

   Smart Home atau *Home Automation* berisi berbagai perangkat seperti pemanas ruangan, alarm kebakaran, smart tv, lampu , AC hingga stop

kontak yang mampu menyampaikan berapa penggunaan listrik harian.

2. *Wearables*

   Wearables merupakan perangkat yang mampu dibawa oleh pemiliknya. Bentuk *wearables* tidak terbatas pada jam tangan saja. Pada perkembangannya perangkat wearables meliputi kacamata pintar, pengukur jalan / pedometer hingga pengukur waktu tidur

3. *Smart City*

   Smart City meliputi berbagai hal seperti pengaturan lalu lintas, manajemen sampah, manajemen transportasi publik dan hal-hal yang menyangkut keamanan penduduk didalamnya seperti *panic button* dan aplikasi lainnya

4. *Smart Grid*

   Smart grid merupakan jaringan listrik yang mampu mengintegrasikan aksi-aksi atau kegiatan dari pengguna dengan tujuan agar lebih efisian dan ekonomis

5. *Connected Car*

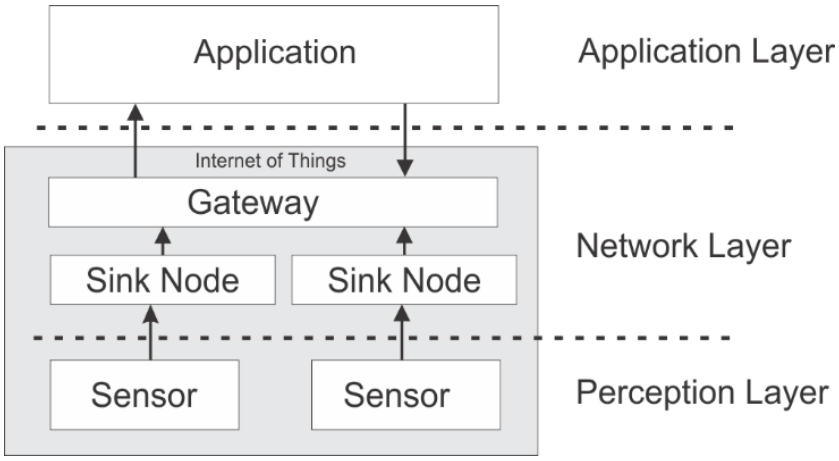
   Teknologi mobil pintar sekarang ini tidak hanya sistem entertainment yang bagus. Konsep mobil yang mampu menyopir sendiri sudah bukan hal mustahil lagi. Saat ini mobil sudah mampu berkomunikasi dengan mobil lain, terhubung dengan peta dan gps secara online serta mampu mengenali kondisi lalu lintas

6. *Connected Health*

   Monitoring kondisi pasien secara online tidak hanya memudahkan dokter dalam memantau pasiennya. Ketika pasien mengalami kondisi darurat dan harus dilakukan operasi tindak lanjut, maka dokter tidak perlu melakukan pemeriksaan ulang karena semua kondisi pasien sudah terekam didalam sistem

7. *Smart Farming*

   Pemantauan stok bahan pangan hingga monitoring kondisi perkebunan menjadikan tema smart farming menjadi topik menarik bagi penelitian Internet of Things



Gambar 2.1 Layer dalam *Internet of Things* [17]

Berdasar pada Gambar 2.1, IoT sendiri dapat dibedakan menjadi beberapa layer [17] . Layer pertama merupakan *Perception Layer* dimana informasi dan data dikumpulkan. Teknik untuk proses pengumpulan data ini bermacam-macam seperti *two dimentional code*, RFID dan perangkat sensor. *Two dimentional code* merupakan penggunaan gambar untuk merepresentasikan data. Contohnya adalah penggunaan barcode pada kemasan produk. RFID menggunakan gelombang magnet untuk mengirim data, sedangkan perangkat sensor sendiri sangat beraneka macam sesuai dengan fungsinya. *Perception Layer* ini bertanggungjawab mengubah data yang telah dikumpulkan dari sensor menjadi sinyal yang dapat dikirim melalui *network*.

Setelah data tersebut dikumpulkan, kemudian data ditransfer melalui *network layer*. *Network layer* ini yang bertanggung jawab untuk mengirim data dari satu host ke host yang lain. Teknologi yang digunakan bermacam-macam seperti ZigBee, Z-Wave dan 6LoWPAN. Zigbee dan Z-Wave [18] merupakan protokol komunikasi yang mendukung standar jaringan mesh dan transfer data tingkat rendah. Zigbee sering digunakan dalam jaringan sensor nirkabel sedangkan Z-Wave

lebih sering digunakan dalam produk elektronik seperti TV, remot kontrol dan *home automation*. Teknologi 6LoWPAN atau kepanjangan dari IPv6 *Over Low Power Area Network* merupakan protokol komunikasi berbasis IP versi 6 yang mendukung konsumsi daya rendah. Penggunaan jaringan 6LoWPAN sangat dianjurkan jika *sensor node* yang digunakan sudah mendukung komunikasi melalui IP.

Terakhir data dari *network layer* tersebut masuk ke *application layer. Application layer* ini yang bertanggungjawab mengolah data tersebut menjadi informasi yang berguna bagi pengguna. *Application layer* ini tidak termasuk dalam arsitektur IoT namun *layer* ini yang paling bertanggung jawab sebagai *end point* data yang dikumpulkan oleh sensor dari *perception layer*.

2.2.2 Middleware

Perangkat IoT memiliki tipe yang beragam dan sebagian besar tidak memiliki standar yang sama. Disinilah fungsi *middleware* berada. *Middleware* merupakan sebuah layer yang berfungsi untuk menghubungkan 2 layer didalam aplikasi. Biasanya layer aplikasi dengan layer teknologi [19]. Sedang pengertian dari *middleware* sendiri adalah lapisan perangkat lunak yang berada diantara sistem operasi dengan aplikasi. Berfungsi untuk menyediakan solusi untuk masalah seperti *heterogenitas, interoperability*, kehandalan dan keamanan [20]. Didalam perspektif IoT, *Middleware* digunakan untuk menghubungkan layer fisik (sensor & aktuator) dengan layer diatasnya yaitu layer aplikasi.

Penggunaan *middleware* membuat developer fokus pada tugas programming yang dikerjakannya tanpa memikirkan bagaimana harus menyeragamkan standar dari sensor-sensor yang digunakan. Arsitektur *middleware* pada IoT dibedakan menjadi 3 tipe [21]. Yang pertama *middleware* berbasis *service-based solution*. Arsitektur ini menggunakan standar *Service Oriented Architecture* (SOA) [22]. Penggunaan standar SOA membuat developer mampu mengubah sistem yang besar dan komplek menjadi sebuah sistem yang lebih sederhana berdasar service yang ditangani. Contoh *middleware* tipe pertama ini adalah Hydra system [23].

Tipe yang kedua adalah *middleware* berbasis *cloud. Middleware* ini membatasi jumlah pengguna dan jumlah perangkat yang dapat digunakan. Namun memiliki keunggulan, memberikan kemudahan bagi pengguna untuk menghubungkan, mengumpulkan dan mencari informasi dari data yang dikumpulkan oleh sensor selama pengguna tersebut terhubung ke internet. Contoh dari middleware berbasis *cloud* adalah Xively [24].

Tipe *middleware* yang ketiga yaitu *actor based framework*. Pada middleware ini, perangkat IoT yang digunakan diumpamakan sebagai aktor. Dengan menjadi aktor maka perangkat IoT tersebut dapat digunakan dan didistribusikan kepada siapa saja di dalam suatu jaringan. Contoh dari *middleware* tipe ketiga ini adalah Calvin [25].

Penggunaaan *middleware* didalam sistem berbasis IoT memiliki karakteristik tersendiri, yaitu :

1. *Scalability*

   Middleware dalam IoT harus memiliki tingkat skalabilitas yang tinggi seiring dengan semakin banyaknya perangkat IoT yang beredar saat ini.

2. *Realtime*

   Sebagian besar aplikasi IoT merupakan sistem monitoring dimana keadaan suatu objek harus dapat terpantau secara realtime

3. *Reliability*

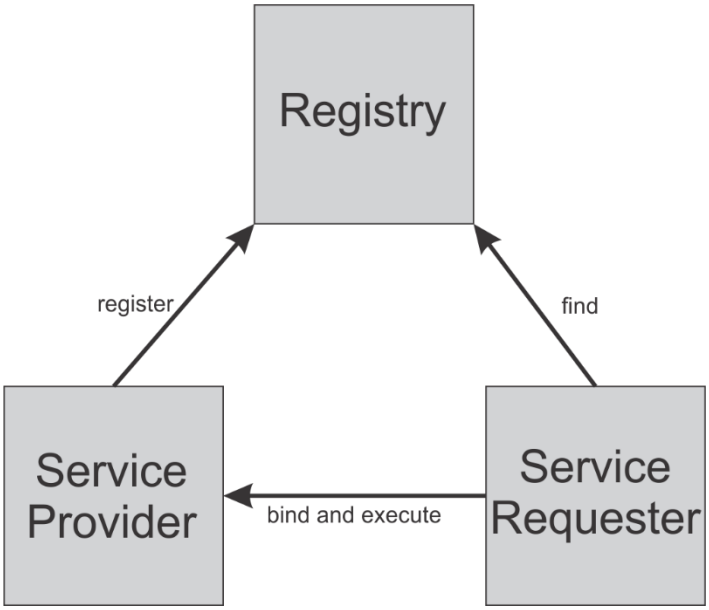
   Setiap komponen atau service yang menggunakan middleware harus dipastikan tersedia dan tetap beroperasi dalam jangka waktu tertentu selama sistem IoT tersebut berfungsi.

4. *Availability*

   Sebuah *middleware* harus mampu beroperasi selama kurun waktu tertentu, bahkan terus menerus jika seandainya diperlukan.

2.2.3 Web Service

Web service merupakan standar yang digunakan untuk melakukan pertukaran data antar aplikasi atau sistem. Tidak semua aplikasi ditulis pada bahasa pemrograman yang sama atau berjalan pada platform yang sama maka perlu dilakukan standarisasi pertukaran data. Contoh dari implementasi *web service* adalah SOAP dan REST.



Gambar 2.2 Ilustrasi SOAP [26]

SOAP merupakan generasi pertama *web service* yang diperkenalkan. SOAP terdiri dari 3 entitas yang saling berhubungan yaitu *service provider*, *service registry* dan *service requester* [26]. Ilustrasi dari ketiga entitas ini dapat dilihat pada Gambar 2.2. *Service provider* berfungsi membuat *service* SOAP berbasis web dan mempublikasikan *service* tersebut ke dalam *service registry*. *Service requester* kemudian mencari layanan service yang tersedia dengan menghubungi *service registry*. Jika layanan tersedia maka *service registry* akan mengirimkan service description kepada *service requestor*. Komunikasi yang terjadi pada 3 entitas tersebut menggunakan XML dan *Web Service Description Language* (WSDL). WSDL menyediakan informasi web service dan bagimana penggunaan metode service tersebut. Karena perkembangan perangkat IoT yang semakin banyak dan

SOAP tidak mampu melayani perangkat yang berbeda standar maka penggunaan SOAP mulai ditinggalkan..

Pengembangan web service selanjutnya yaitu *Representational State Transfer* atau disingkat REST. REST ini memiliki keunggulan mampu menghandle perangkat-perangkat yang berbeda standar karena dalam komunikasinya menggunakan metode HTTP.

2.2.4 Representational State Transfer (REST)

REST pertama kali diusulkan oleh Fielding [27] pada disertasinya. REST merupakan sebuah desain software arsitektur untuk membangun sistem terdistribusi yang memiliki kemampuan skalabilitas yang tinggi. Didalam REST, data set dan objek ditangani oleh aplikasi klien – server dengan dimodelkan menjadi *resource.* Prinsip dari REST adalah :

1. Unified Resource Identifier (URI)

   Setiap resource yang dimiliki dalam REST diakses oleh klien melalui URI

2. *Uniform Interface*

   Setiap interaksi yang dilakukan oleh URI menganut 4 model HTTP yaitu GET, PUT, DELETE dan POST

3. *Self Descriptive Message*

   Setiap message atau komunikasi harus berisi informasi

4. *Stateless*

   Setiap request yang dilakukan bersifat independen dan tidak terkait dengan request sebelumnya.

REST membuat pengembangan sistem menjadi lebih mudah karena dalam aplikasinya menggunakan 4 operasi standar HTTP yaitu POST, GET, PUT dan DELETE. POST digunakan untuk membuat resource baru. PUT digunakan untuk melakukan update terhadap resource sedangkan DELETE digunakan untuk

menghapus resource. Pengembangan API yang menggunakan metode REST sering disebut RESTful API. Bila dibandingkan dengan metode komunikasi *web service* sebelumnya yaitu SOAP. RESTful API memiliki keunggulan seperti [28]:

1. Menggunakan XML dan JSON untuk mengirim dan menerima data. Dimana XML dan JSON lebih mudah dibaca oleh manusia.
2. Mudah dipanggil melalui URL Path / URI
3. Transfer hanya dilakukan melalui HTTP, sedangkan SOAP menggunakan banyak protokol seperti FTP dan SMTP
4. Bandwith yang digunakan lebih rendah
5. Jika terjadi perubahan kode pada *web service* tidak perlu melakukan konfigurasi ulang pada klien
6. Mendukung semua perangkat IoT selama perangkat tersebut memiliki komunikasi HTTP

Didalam sistem IoT, sensor mendapatkan informasi dari objek dan mengirim informasi tersebut dalam bentuk MIME. Akan lebih mudah jika informasi tersebut dapat diakses dalam bentuk *resource* melalui web. Penggunaan REST memungkinkan hal ini terjadi dengan membuat URI sesuai dengan *resource* yang diperlukan.

2.2.5 Resource Oriented Architectures (ROA)

Konsep utama dari ROA adalah *resource* [29]. Didalam arsitektur ini, sebuah r*esource* memiliki nama dan alamat yang direpresentasikan dengan sebuah URI. URI memiliki cara tersendiri untuk mengidentifikasi suatu *resource*, terlepas dari jenis dan tipe *resource* tersebut. Aspek penting dalam penggunaan ROA adalah setiap *resource* harus memiliki URI yang berbeda. Dengan URI yang berbeda maka klien dapat melakukan *request resource* dengan mengacu pada representasi dari URI tersebut.

REST merupakan salah satu contoh teknologi yang menggunakan basis arsitektur ROA. ROA sendiri memiliki 4 properties yaitu :

1. *Addresssability*

   Semua resource akan diimplementasikan menggunakan Uniform Resource Identifiers (URI). Setiap resource tersebut akan memiliki alamat URI sendiri. Ketika alamat URI dipanggil dia akan mengembalikan respon dalam bentuk JSON atau XML.

2. *Connectedness*

   Resource yang ada dalam REST harus memiliki relasi dengan resource yang lain agar dapat dipresentasikan melalui URI.

3. *Homogeneus Interface*

   Resource akan dipanggil menggunakan 4 metode HTTP yaitu GET, PUT, POST dan DELETE dengan 2 tambahan metode yaitu HEAD dan OPTION. HEAD digunakan untuk menunjukkan Metadata sedang OPTION digunakan untuk memeriksa metode yang ada

4. *Statelessness*

   *Stateles*s menunjukkan bahwa server tidak menyimpan data dari klien dari setiap koneksi yang terbentuk.

2.2.6 JavaScript Object Notation (JSON)

Sistem REST API dalam membuat dan melakukan request data membutuhkan sebuah standar format pertukaran data. Format yang paling banyak digunakan saat ini yaitu XML dan JSON. XML digunakan oleh SOAP sebagai standar pertukaran data, sedangkan REST API mendukung keduanya, XML dan JSON. Namun JSON memiliki popularitas yang lebih tinggi karena format penulisannya lebih mudah dibaca baik oleh manusia maupun komputer.

Walaupun format JSON ini dibuat berdasarkan Standar dari ECMA-262 [30], JSON merupakan format teks yang tidak bergantung dengan bahasa

pemrograman apapun sehingga menjadikan JSON bahasa pertukaran data terpopuler saat ini.

JSON terdiri dari 2 struktur, yang pertama merupakan kumpulan dari pasangan nama atau nilai. Pada bahasa pemrograman hal ini disebut sebagai object, record, struct atau array. Sedang struktur kedua adalah daftar nilai yang terurutkan. Sering diimplementasikan dalam bentuk array atau list. Gambar 2.3 berikut ini merupakan contoh format JSON

```
{
     "namaFakultas": "Fakultas Teknik UGM",
     "namaDepartemen": "Departemen Teknik Elektro",
     "alamat": {
         "namaJalan": " Jl. Grafika No.2 Kampus UGM",
         "kota": "Sleman",
         "provinsi": "Yogyakarta",
         "kodePos": 11111
     },
     "nomerTelepon": " (0274) 547506"

 }
```

Gambar 2.3 Contoh Format JSON

2.2.7 Slim Framework

Slim merupakan sebuah *framework* PHP yang dibuat oleh Josh Lockhart pada akhir 2010 [31]. Menurut Josh Lockhart, Slim merupakan sebuah *micro framework* yang membantu developer untuk membangun aplikasi Web atau API. Dikatakan micro karena Slim hanya fokus pada kebutuhan pokok yang diperlukan dalam suatu aplikasi web seperti : menerima HTTP *request,* mengirimkan *request* tersebut ke code yang sesuai dan mengembalikan HTTP response.

Micro Framework digunakan untuk membuat aplikasi web skala kecil untuk tujuan khusus dengan tingkat kompleksitas yang rendah, seperti dalam pembuatan sebuah API. Akan lebih mudah dan cepat jika menggunakan *micro framework*

daripada *full stack framework* seperti laravel atau codeigniter.

Slim framework memiliki beberapa fitur utama [32], seperti :

1. *HTTP Router*
2. *Middleware*
3. *Dependency Support*
4. *PSR-7 Support*

2.3 Hipotesis

Penelitian ini bertujuan untuk memperkuat aspek keamanan pada sistem smart farm yang telah dibuat pada penelitian sebelumnya. Pada sistem smart farm tersebut aspek keamanan belum menjadi perhatian utama sehingga rawan akan terjadinya serangan atau pencurian data.

Dengan penambahan metode otentikasi token pada API Smart Farm tersebut maka akses terhadap *resource* dapat difilter. *Middleware* akan melakukan validasi terhadap semua request yang masuk. Jika tidak terdapat token yang tervalidasi maka server akan menolak request tersebut.

Selain itu, belum adanya pengaturan hak akses user dalam mengakses resource menjadi hal lain yang perlu diperhatikan. Keamanan dan *privacy* data menjadi perhatian utama dalam hal ini. Pembatasan user akan dilakukan berdasar peran user tersebut didalam sistem.

BAB III
METODE PENELITIAN

3.1 Alat dan Bahan Penelitian

3.1.1 Alat Penelitian

Alat – alat yang digunakan dalam penelitian ini antara lain :

1. *Notebook* dengan sistem operasi Windows 10 , *processor* Dual Core @ 2 Ghz, memori 4 GB dan *hardisk* 320 GB.
2. Bahasa pemrograman PHP.
3. *Slim Framework, framework* yang digunakan untuk membuat sistem REST.
4. Atom sebagai *text editor* dalam mengembangkan sistem.
5. XAMPP, perangkat lunak server yang berisi *service* Apache, MySQL dan FTP.
6. PHPMyadmin digunakan sebagai interface dalam manajemen *database.*
7. Postman digunakan sebagai aplikasi testing dalam melakukan *request resource* URI

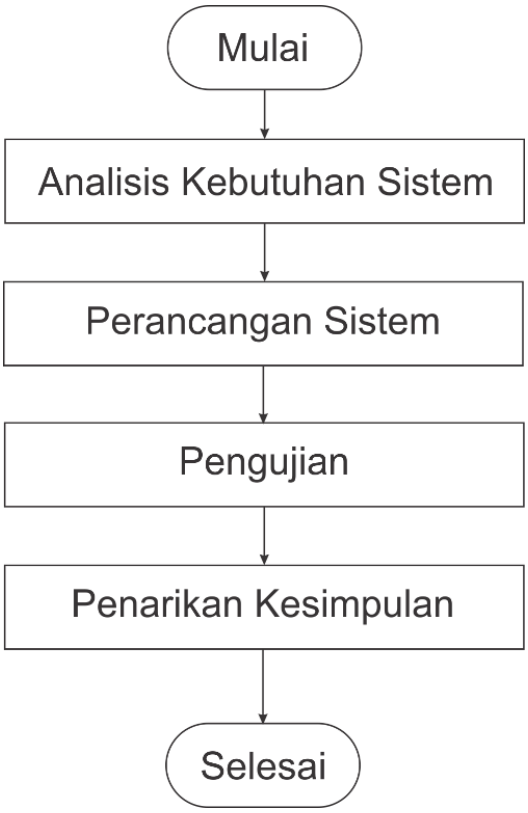
3.1.2 Bahan Penelitian

Bahan yang digunakan dalam penelitian ini adalah API sistem smartfarm beserta *resource* data simulasi dari sensor yang telah dibuat pada penelitian sebelumnya. Metode pengambilan bahan penelitian dilakukan dengan melakukan tatap muka secara langsung dengan peneliti sebelumnya.

Dengan melakukan wawancara secara langsung dengan peneliti akan diketahui secara mendetail proses bisnis yang ada pada sistem tersebut. Selain itu dapat diketahui pula masalah-masalah yang ada dalam proses pengembangan sebelumnya.

3.2 Alur Penelitian

Alur penelitian bertujuan untuk menjelaskan langkah-langkah atau tahap dalam mengembangkan sistem ini. Tahap-tahap dalam melakukan penelitian ini yaitu analisis kebutuhan sistem, perancangan sistem, pengujian dan penarikan kesimpulan. Tahapan tersebut dapat diilustrasikan pada Gambar 3.1.



Gambar 3.1 Alur Penelitian

Tahapan dalam penelitian ini dapat dijelaskan sebagai berikut :

1. Analisis kebutuhan sistem

   Sebelum sistem mulai dikembangkan, dilakukan terlebih dahulu analisis kebutuhan sistem. Tahap ini digunakan untuk memahami lebih dalam permasalahan yang dihadapi. Kompleksitas sistem API yang sudah ada , data-data yang diperlukan, keluaran dari sistem serta cara evaluasi sistem

2. Perancangan Sistem

Tahap ini dilakukan dengan cara merancang metode otentikasi berbasis token yang sesuai dengan sistem API yang telah ada. Perancangan manajemen user dalam melakukan request terhadap resource juga dibahas dalam tahap ini

3. Pengujian

   Setelah sistem dikembangkan maka akan dilakukan pengujian untuk menilai apakah sistem sudah sesuai dengan kebutuhan sistem yang telah ditentukan pada tahap sebelumnya.

4. Penarikan Kesimpulan

   Tahap ini merupakan tahap akhir yang bertujuan untuk mengevaluasi sistem yang dibangun. Hasil yang dari dari evaluasi dapat dilakukan sebagai pertimbangan dalam mengembangkan aplikasi selanjutnya

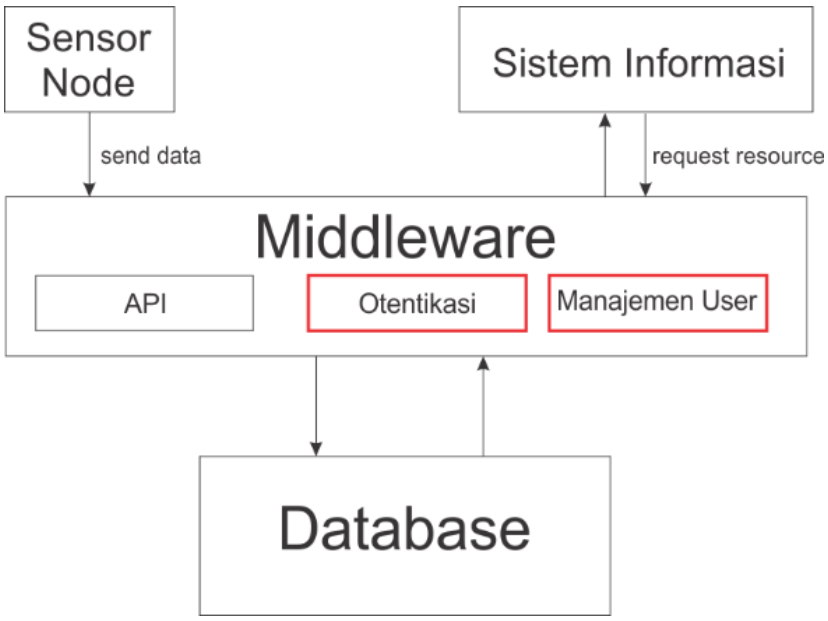
3.2.1 Analisis Kebutuhan Sistem

Sistem API smart farm dikembangkan oleh Anisa [4] dalam penelitiannya untuk memudahkan pemantauan kondisi perkebunan kelapa sawit. Parameter yang dipantau merupakan kondisi lingkungan yang berpengaruh secara langsung atau tidak langsung dalam budidaya kelapa sawit tersebut. Sistem yang dikembangkan memiliki beberapa fungsi layanan API seperti fungsi untuk menerima data dari sensor *node*, data *resource* dari pemilik lahan, data *resource* perangkat yang digunakan dan lain - lain.

Pengembangan API ini bertujuan untuk memudahkan *developer* dalam membangun sistem informasi tanpa harus memikirkan sistem *backend* seperti perancangan database dan cara untuk mengaksesnya.

Dalam penelitian ini, sistem API tersebut akan dikembangkan lagi dengan melakukan penambahan fungsi otentikasi dan manajemen user. Ilustrasi pengembangan fungsi yang baru dapat dilihat pada Gambar 3.2. Pada gambar tersebut dapat dilihat gambaran besar sistem API pada smart farm. Sedangkan

pengembangan penelitian ini fokus pada kotak merah yang berisi fungsi otentikasi dan manajemen user.



Gambar 3.2 Layanan yang akan dikembangkan dalam API smartfarm

Layanan otentikasi merupakan layanan yang digunakan untuk melakukan *filtering* terhadap request yang masuk ke *middleware*. Proses request terjadi ketika *developer* melakukan akses terhadap URI yang mewakili *resource* pada API smartfarm. API sistem smartfarm tersebut menggunakan menggunakan metode komunikasi REST. REST merupakan metode komunikasi yang stateless dimana setiap request yang terjadi bersifat independen sehingga server harus melakukan otentikasi klien setiap kali request terjadi. Stateless juga berarti tidak ada session yang disimpan selama otentikasi dilakukan. Hal ini mengakibatkan otentikasi berdasar protokol HTTP seperti penggunaan *username* dan *password* menjadi tidak memadai.

Pada sistem sebelumnya, *middleware* akan memproses semua request yang masuk dan mengembalikan resource yang diminta dalam bentuk JSON. Pada sistem yang akan dikembangkan, fungsi otentikasi akan bekerja dengan melakukan pengecekan *header* pada *request* yang masuk. Jika terdapat token yang sesuai pada

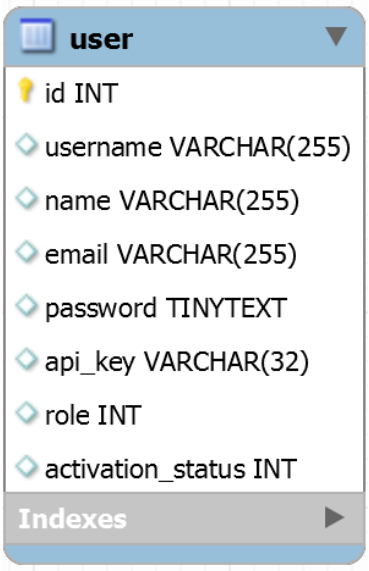
database *user* maka *request* akan diproses. Namun jika tidak terdapat token atau terdapat token tapi tidak sesuai dengan database maka *request* akan distop.

Sedangkan layanan manajemen user akan bekerja mengatur hak akses user untuk melihat *resource* pada database. Sistem yang dikembangkan nantinya mendukung fungsi *multiuser* dimana setiap *user* akan memiliki *sensor node* sendiri. *User* dapat melakukan tindakan seperti menambah *sensor node*, menghapus *sensor node* serta memperbarui *sensor node*. Pada sistem informasi yang telah dibuat, semua *sensor node* yang ada dalam database ditampilkan di halaman utama tanpa ada *filter* terhadap kepemilikan sensor node tersebut. Sehingga dibutuhkan layanan manajemen user agar nantinya user hanya dapat melakukan tindakan terhadap *sensor node* yang menjadi miliknya sendiri.

3.2.2 Perancangan Sistem

Dalam sistem ini akan dibangun 2 layanan utama yaitu layanan otentikasi dan manajemen user. Layanan otentikasi akan memerlukan atribut *api key* sebagai token sedangkan layanan manajemen user memerlukan atribut *role id* sebagai penanda peran dari user. Untuk mendukung 2 layanan tersebut maka diperlukan rancangan tabel user yang baru untuk menyimpan data token yaitu *api key* dan dan data peran dari *user* yaitu *role*. Rancangan tabel *user* yang baru dapat dilihat pada Gambar 3.3

Tabel *user* tersebut digunakan untuk menyimpan data informasi *user* seperti *username, name, email* dan *password* dengan tambahan 2 atribut baru yaitu *api key* untuk menyimpan token dan *role* untuk menyimpan id peran pengguna dari *user*. Atribut *api key* memiliki value yang akan digenerate secara acak oleh sistem sedangkan *role* memiliki 2 value yang sudah ditentukan yaitu 0 untuk admin dan 1 untuk user biasa.



Gambar 3.3 Rancangan tabel user

Token yang berfungsi sebagai validasi user saat mengakses URI didapatkan saat user melakukan registrasi pertama kali. Untuk itu perlu dikembangkan perancangan URI untuk mengelola proses ini. Perancangan URI untuk layanan manajemen user dapat dilihat pada Tabel 3.1. Pada saat melakukan *registrasi*, *user* perlu memasukkan data seperti *username*, *name*, *email* dan *password* dengan mengakses URI /register.

Tabel 3.1 Rancangan URI untuk layanan manajemen user

No	Layanan	Rancangan URI
1	Pendaftaran User	/register
2	Login User	/login
3	Request Token Baru	/apikey/:id
4	Aktivasi User	/activate/:id

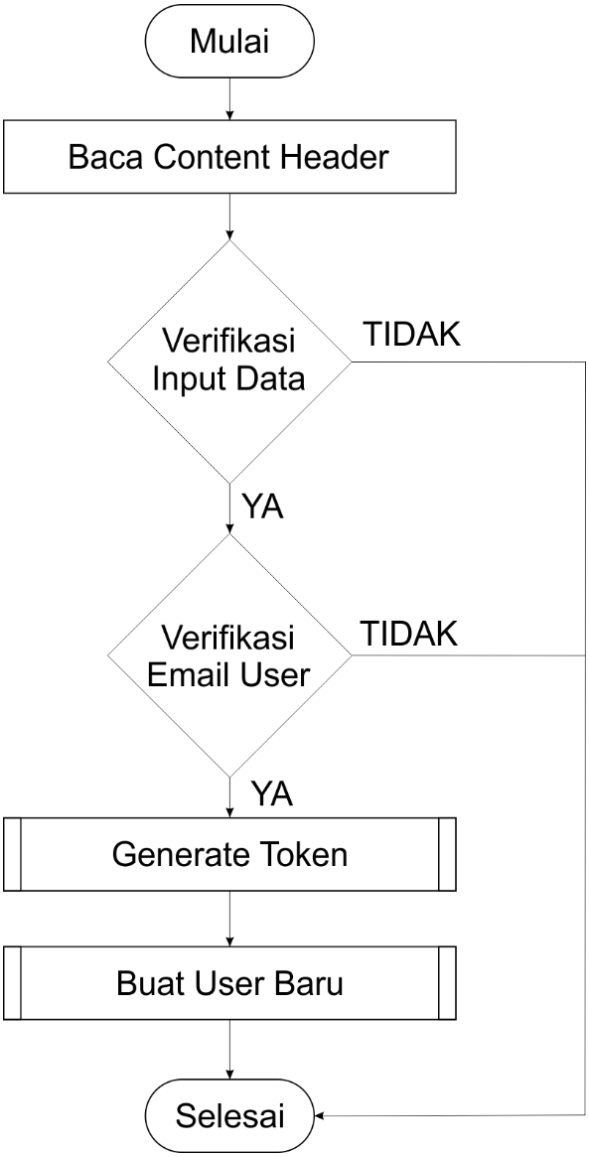
Selain proses pendaftaran dirancang juga URI untuk melakukan proses *login*, pada proses *login* ini diperlukan data *login* yaitu *email* dan *password* dari *user*. Jika data *login* sesuai dengan *database* maka akan dikembalikan informasi data *user*

seperti *name, username* dan token dalam format JSON. Jika data tidak sesuai dan login gagal maka akan dikembalikan pesan error.

Token yang didapatkan saat proses pendaftaran bersifat rahasia dan hanya user tersebut yang tahu. Jika token tersebut tersebar maka data *sensor node* dari user pemilik token tersebut dapat diakses oleh orang lain. Untuk itu perlu layanan *reques*t token baru. Layanan ini diperlukan jika token *user* telah tersebar dan *user* perlu memperbarui token miliknya. *Request* token baru dapat dilakukan dengan mengakses URI /apikey dengan menyertakan id dari user tersebut.

Setiap user yang telah melakukan registrasi memiliki status yang belum teraktivasi. Didalam tabel user sendiri terdapat atribut activation_status, atribut ini memiliki value yang telah ditentukan yaitu 0 untuk user yang belum teraktivasi dan 1 untuk user yang telah aktif. Proses aktivasi user ini dilakukan dengan mengakses URI /activate dengan menyertakan id dari user tersebut. Layanan ini hanya dapat diakses oleh user yang berstatus admin. Sehingga perlu otentikasi token admin sebelum proses ini dilakukan.

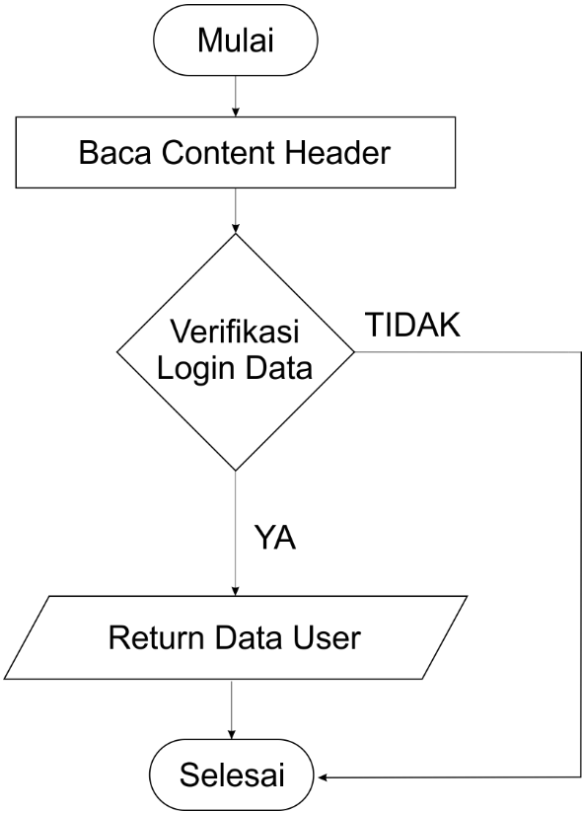
Untuk mempermudah proses yang terjadi pada layanan pendaftaran diatas maka dirancang urutan-urutan dalam diagram alir yang dapat dilihat pada Gambar 3.4. Saat proses pendaftaran dimulai, akan dilakukan proses pembacaan pada *header* (Baca Content Header), Sistem akan melakukan verifikasi *content* pada *header* tersebut apakah terdapat *value* untuk data *username, name, email* dan *password* (Verifikasi Input Data). Jika value tersebut lengkap maka proses akan berlanjut ke verifikasi *email* (Verifikasi Email User). Verifikasi *email* diperlukan agar dalam sistem tersebut tidak ada user yang memiliki *email* yang sama. Jika verifikasi *email* sukses maka sistem akan membuat token secara acak (Generate Token) dan melakukan proses *insert* ke *database* untuk membuat user baru (Buat User Baru).



Gambar 3.4 Diagram alir proses pendaftaran user

Proses selanjutnya setelah *user* selesai melakukan pendaftaran adalah proses login. Dalam proses ini user mengirimkan data berupa *email* dan *password* saat melakukan *request*. Diagram alir proses login ini dapat dilihat pada Gambar 3.5. Saat proses login dimulai akan dilakukan proses pembacaan pada header (Baca Content Header), kemudian sistem akan melakukan verifikasi apakah *email* dan *password* yang disertakan di dalam request sudah sesuai dengan data didalam tabel user (Verifikasi Login Data). Jika tidak sesuai maka proses akan berhenti dan sistem

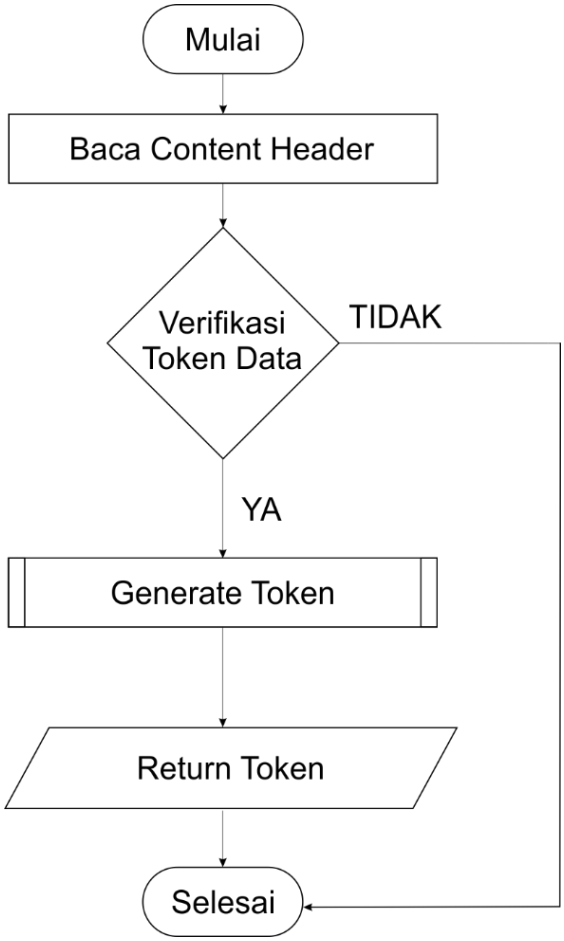
akan mengembalikan pesan error. Jika email dan password valid maka sistem akan mengembalikan data user (Return Data User). Data user ini berupa data nama, *username, api key, activation status* dan role id.



Gambar 3.5 Diagram alir proses login user

Token yang digunakan oleh user untuk melakukan *request resource* bersifat rahasia. Jika token tersebut tersebar maka orang yang tidak memiliki hak dapat menggunakan token tersebut untuk melihat data dari sensor node milik user lain. Lebih bahaya lagi jika token yang tersebar merupakan token milik user yang memiliki role sebagai admin. Untuk itu layanan *request* token baru ini diperlukan. Diagram alir proses *request* token ini dapat dilihat pada Gambar 3.6. Proses dimulai dengan pengecekan header dari *request* yang terjadi (Baca Content Header). Kemudian sistem akan melakukan verifikasi apakah terdapat token dalam request tersebut (Verifikasi Token Data). Jika token yang ada valid maka sistem akan menjalankan proses untuk membuat token baru sesuai dengan id user yang

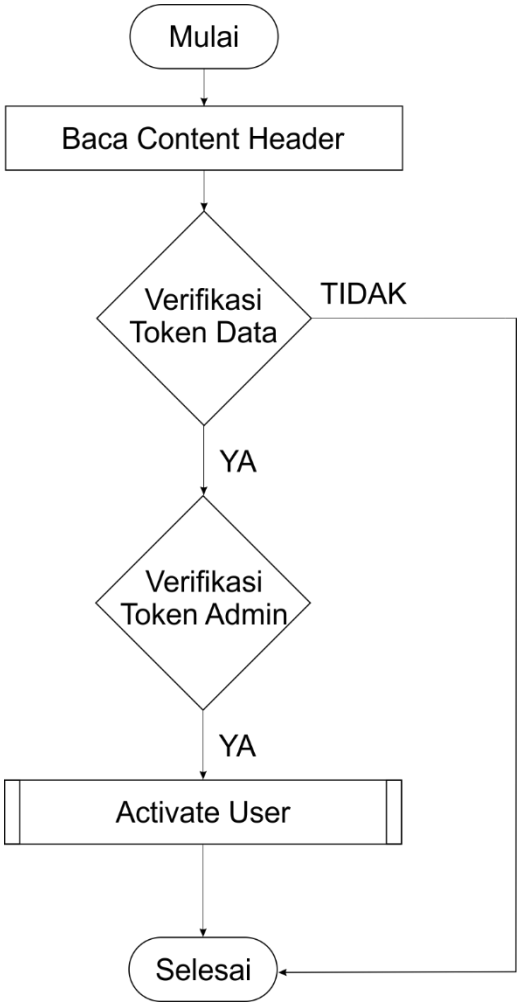
melakukan request (Generate Token). Jika proses berhasil maka sistem akan mengeluarkan output berupa data token baru (Return Token).



Gambar 3.6 Diagram Alir proses request token baru

Setiap *user* yang telah melakukan *registrasi* memiliki status yang belum aktif. Proses aktivasi user ini hanya dapat dilakukan oleh admin. Sehingga diperlukan otentikasi token admin didalam proses ini. Proses aktivasi user dimulai dengan sistem membaca header dari *request* yang terjadi (Baca Content Header). Kemudian sistem akan melakukan verifikasi apakah terdapat token didalam request tersebut (Verifikasi Token Data). Jika token tersebut valid maka dilakukan proses verifikasi kedua untuk menentukan token tersebut milik admin atau user biasa (Verifikasi Token Admin). Jika token tersebut milik admin maka sistem akan

menjalankan proses aktivasi user sesuai dengan id user yang disertakan didalam request (Activate User).

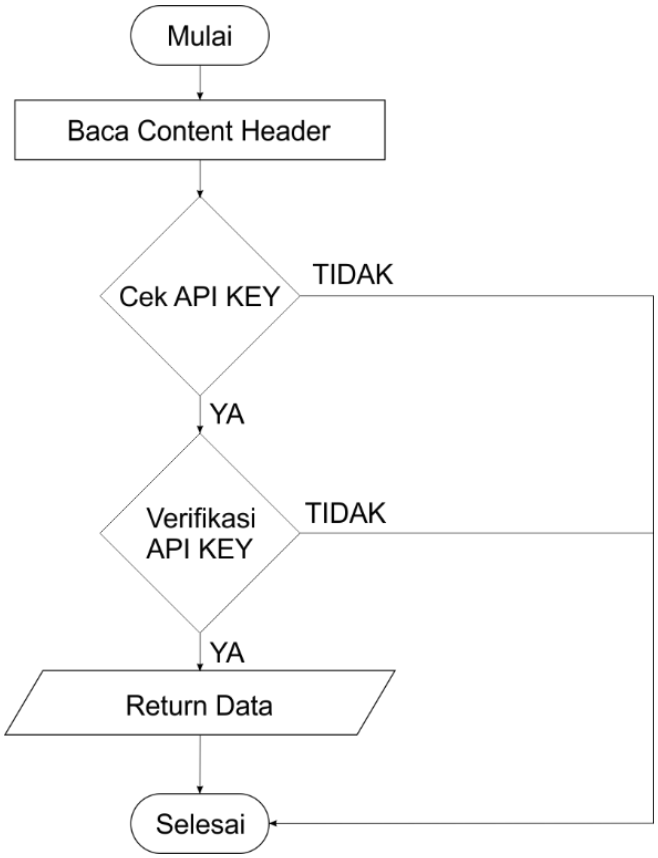


Gambar 3.7 Diagram Alir proses aktivasi user

Setelah user mendapatkan token yang diperoleh dari proses login maka dalam setiap melakukan request terhadap resource harus menyertakan token tersebut. Sistem akan menolak request yang tidak memiliki token yang valid. Proses verifikasi token dapat diilustrasikan dalam Gambar 3.8

Request resource terjadi saat user melakukan akses terhadap URI yang mewakili *resource* yang dituju. Request dilakukan dengan beberapa metode seperti GET, PUT, POST dan DELETE. Sistem akan membaca *header* dari *request* tersebut untuk memastikan terdapat *value* token disana (Cek API Key). Jika tidak

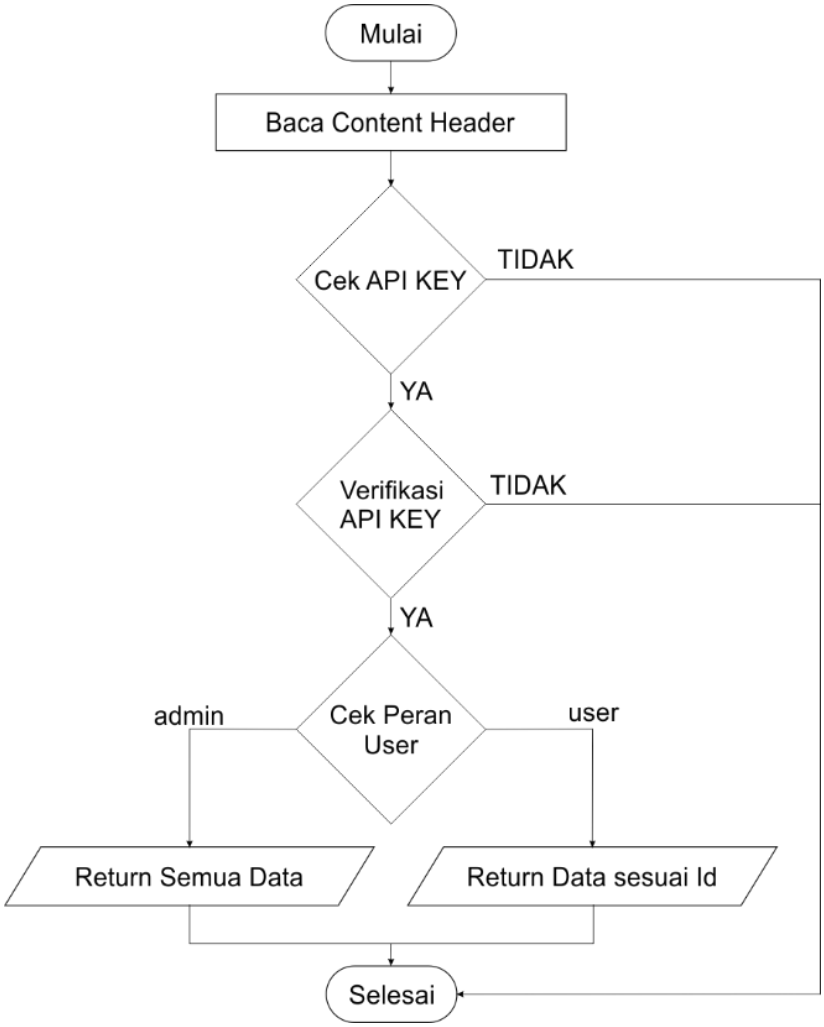
terdapat *value* token pada *header* maka *request* akan langsung ditolak. Namun jika terdapat *value* token maka proses akan berlanjut ke verifikasi token. Token yang berasal dari *request user* akan dicocokkan dengan token yang berada di *database* (Verifikasi API Key). Jika token tersebut ada dalam *database* maka sistem akan memproses *request* tersebut dan mengembalikan *resource* yang dituju dalam bentuk JSON (Return Data).



Gambar 3.8 Diagram alir proses request dengan token

Layanan manajemen *user* memiliki fungsi untuk mengelola sensor node yang akan ditampilkan ke halaman utama pada sistem informasi. *Sensor node* sendiri memiliki 2 atribut yaitu *public* dan *private*. *Sensor node* yang memiliki atribut *public* akan dapat dilihat oleh semua pengguna. Sedangkan *sensor node* yang bersifat *private* hanya dapat dilihat oleh admin sistem dan pemilik *sensor node* tersebut. Untuk membedakan peran dari user maka dibutuhkan *value role* yang sudah dibuat pada perancangan *database*. Jika admin maka *role* akan memiliki

value bernilai 0, sedangkan jika pengguna adalah user maka *value* akan bernilai 1. Proses request terhadap *sensor node* dapat dijelaskan pada Gambar 3.9



Gambar 3.9 Diagram alir untuk *request sensor node*

Seperti *request* pada *resource* yang lain, sistem akan melakukan verifikasi token terlebih dahulu pada *header* (Verifikasi API Key). Jika token sudah terverifikasi maka sistem akan melakukan proses tambahan untuk melihat peran user yang melakukan request tersebut (Cek Peran User). Pertama sistem akan mencari *value role* didalam database berdasar token yang masuk. Jika value bernilai 0 maka peran yang melakukan request tersebut adalah admin dan sistem akan mengembalikan data berupa semua *sensor node* baik yang memiliki atribut *private* maupun *sensor node* dengan atribut *public* (Return Semua Data). Jika value bernilai 1, maka peran dari orang yang melakukan request tersebut adalah *user* biasa. Sistem

kemudian akan mencari *id* dari user tersebut dan mengembalikan data semua sensor node yang sesuai dengan *id* dari user tersebut ditambah dengan data dari *sensor node* yang bersifat *public* (Return Data Sesuai Id).

3.2.3 Pengujian Sistem

Setelah dilakukan proses perancangan maka tahap selanjutnya adalah proses pengujian atau *testing*. Pengujian yang dimaksud adalah untuk menguji perangkat lunak tersebut apakah sudah dapat memenuhi kebutuhan proses bisnis dari pengguna atau belum.

Pengujian ini bertujuan untuk melakukan verifikasi dan validasi terhadap fungsi yang telah ditentukan dalam tahap analisis kebutuhan sistem. Verifikasi bertujuan untuk menjamin perangkat lunak tersebut sudah sesuai dengan fungsinya, sedangkan validasi bertujuan untuk memastikan perangkat lunak sudah dapat memenuhi kebutuhan pengguna.

Pada penelitian ini dilakukan pengujian dengan metode *Black Box testing*[33]. *Black Box Testing* adalah pengujian yang dilakukan dengan hanya mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari perangkat lunak tersebut. Dengan kata lain, input dengan output yang dihasilkan dari perangkat lunak diharapkan sudah sesuai dengan kebutuhan pengguna.

Pengujian *Black Box testing* [34] memiliki beberapa level yaitu *Integration, Functional, System, Acceptance, Beta* dan *Regression*. Pada penelitian ini akan difokuskan pada *functional testing*. *Functional testing* akan menjamin fungsionalitas dari perangkat lunak yang diuji sudah sesuai dengan kebutuhan dari pengguna. Fungsi yang diuji pada *testing* ini adalah fungsi pendaftaran user baru, fungsi *login user*, fungsi *request* token baru, fungsi otentikasi token dan fungsi *request sensor node.*

BAB IV
HASIL DAN PEMBAHASAN

4.1 Pengembangan Sistem

Pada tahap pengembangan ini, akan dilakukan pemetaan URI untuk mempermudah *developer* dalam mengakses layanan yang disediakan. URI merupakan representasi dari *resource* yang ditulis dalam bentuk link seperti alamat suatu website. Dalam pengembangan API ini ditentukan 5 layanan utama yaitu pendaftaran *user*, *login* u*ser*, *request* token baru, aktivasi user dan layanan untuk melihat *sensor node* berdasar peran dari user. Setiap layanan tersebut memiliki URI sendiri dengan method yang berbeda-beda. Selain itu juga ditentukan *content* apa yang diperlukan saat mengakses URI. *Conten*t dapat berupa *header* atau *body* sesuai dengan layanan yang ditentukan. Untuk menjalankan fungsi otentikasi maka didalam *header* diperlukan *key* berupa *Authorization* dengan isi *value* berupa token dari user. Sedangkan untuk layanan *login user*, akan memerlukan *content* berupa *email* dan *password.* Penjelasan lebih lengkap tentang pemetaan layanan URI dapat dilihat pada Tabel 4.1

Tabel 4.1 Rancangan URI untuk layanan API

No	Layanan	Rancangan URI	Method	Content
1	Pendaftaran User	/register	POST	Username, name, email, password
2	Login User	/login	POST	Email, password
3	Request Token Baru	/apikey/:id	GET	Id, Authorization : token
4	Aktivasi User	/activate/:id	GET	Id, Authorization : token

5	Melihat sensor node	/shownodes	GET	Authorization : token

Pembuatan token dilakukan saat *user* melakukan pendaftaran pertama kali. Token ini bersifat rahasia dan tidak boleh ada kaitannya dengan data user agar tidak mudah ditebak. Untuk membuat token ini maka digunakan fungsi *bin2hex*. *Bin2hex* sendiri merupakan bagian fungsi dari *Cryptographically Secure Pseudo-random Number Generator* (CSPRNG) [35] yang dipekenalkan sejak PHP versi 7. Fungsi ini digunakan untuk membuat kode kriptografi secara acak. Fungsi ini bekerja dengan cara, pertama fungsi akan membangkitkan kode acak bernilai 16 *byte.* Kemudian kode acak tersebut diubah ke bentuk *hexadecimal* dengan panjang 32 karakter. Kode yang dihasilkan lebih aman jika dibandingkan dengan fungsi MD5 yang sudah *deprecated.* Potongan kode pembuatan token dapat dilihat pada Gambar 4.1 .

```
private function generateApiKey() {
      return bin2hex(random_bytes(16));
}
```

Gambar 4.1 Fungsi pembuatan Token

Untuk menjalankan layanan otentikasi token maka diperlukan pembuatan *route middleware*. *Route Middleware* akan dijalankan pertama kali saat *request* terjadi. Menurut dokumentasi manual dari *Slim Framework* [36] pembuatan *route middleware* dapat dilakukan dengan menuliskan potongan kode \Slim\Route pada saat mendefinisikan suatu fungsi. Pada layanan ini akan dikembangkan *function* dengan nama *authenticate* yang berguna untuk melakukan validasi token saat *request* terjadi. Contoh penerapan kode *route middleware* pada *function authenticate* dapat dilihat pada Gambar 4.2.

```
function authenticate(\Slim\Route $route) {
```

Gambar 4.2 Contoh pembuatan route middleware

Function authenticate sendiri berisi kode untuk melakukan validasi terhadap token yang disertakan didalam *header request.* Jika *header* memiliki key *Authorization* dengan *value* berisi token yang sesuai dengan data token *use*r di *database* maka *function* ini akan memberikan kembalian bernilai *true.* Sedangkan jika tidak ada *key Authorization* pada *header* atau *valu*e token tidak sesuai dengan token pada *database* maka function ini akan mengembalikan pesan *error*. Potongan kode *function authenticate* dapat dilihat pada Gambar 4.3

```
if (isset($headers['Authorization'])) {
    $db = new FunctionDB();
    $api_key = $headers['Authorization'];

    if (!$db->CekApiKey($api_key)) {

        $response["message"] = "Api Key Salah";
        echoRespnse(401, $response);
        $app->stop();
    } else {

        return true;

    }
} else {

    $response["message"] = "Api Key tidak ditemukan";
    echoRespnse(400, $response);
    $app->stop();
```

Gambar 4.3 Potongan kode *function authenticate*

Fungsi *authenticate* ini kemudian akan diterapkan pada setiap URI yang mewakili *resource* didalam sistem API ini. Penerapan *function authenticate* sendiri dapat dilihat pada Gambar 4.4.

```
                              function1    function2
$app->get('/alerts', 'authenticate','getAlerts');
```

Gambar 4.4 Penerapan *fungsi authenticate* untuk validasi token

Pada gambar tersebut *function authenticate* diterapkan pada URI yang digunakan untuk mengambil data dari tabel *alerts*. Function 1 merupakan *function authenticate* yang merupakan fungsi untuk otentikasi token, sedangkan function 2 merupakan fungsi yang digunakan untuk mewakili *resource* dari tabel *alerts*. Ketika URI /alerts dipanggil maka *function 1* dan *function 2* akan dijalankan secara berurutan. Jika otentikasi token gagal dan function 1 mengembalikan nilai false maka aplikasi akan mengembalikan pesan error dan function 2 tidak akan diproses. Namun jika function 1 mengembalikan nilai *true* maka function 2 akan dipanggil dan sistem akan mengembalikan *resource* yang diwakili function 2 tersebut.

Layanan manajemen *user* diperlukan agar tidak sembarang *user* bisa melihat semua *sensor node* yang berada didalam sistem. *Sensor node* sendiri memiliki 2 atribut yaitu *public* dan *private*. *Sensor node public* dapat diakses oleh semua user, sedangkan *sensor node* yang memiliki atribut *private* hanya bisa diakses oleh user yang memiliki *sensor node* tersebut. Untuk memfasilitasi layanan ini, maka perlu dibuat URI baru untuk melakukan *request* terhadap *sensor node*. URI baru tersebut dapat dilihat pada Gambar 4.5

```
$app->get('/shownodes', 'authenticate','showNode');
```

Gambar 4.5 URI dan function untuk *request sensor node*

Fungsi *request sensor node* ini diawali dengan proses validasi token dengan memanggil fungsi *authenticate*, jika validasi *true* maka akan dipanggil fungsi *showNode* yang merupakan fungsi untuk melakukan validasi peran dari user. Potongan kode dari fungsi *showNode()* dapat dilihat pada Gambar 4.6

Fungsi *showNode* akan membaca token yang ada pada *header request*. Setelah token ditemukan maka akan dimasukan kedalam parameter $api_key. Kemudian proses akan berlanjut dengan memanggil fungsi *getUserRoleId()* dengan mengirim parameter $api_key. Fungsi ini merupakan fungsi yang digunakan untuk mencari value dari *id* dan *role* berdasar token yang ada. Jika value *role* bernilai 0 maka peran dari pengguna tersebut adalah admin, akan dijalankan fungsi untuk

memanggil semua *sensor node*. Sedangkan jika value *role* bernilai selain 0 maka peran dari pengguna adalah user, akan dijalankan fungsi untuk memanggil *sensor node* yang menjadi milik dari user tersebut ditambah dengan *sensor node* yang memiliki atribut *public*.

```
function ShowNode() {
    $app = \Slim\Slim::getInstance();
    $app->contentType('application/json');

    $db = new functionDB();
    $headers = apache_request_headers();

    $api_key = $headers['Authorization'];
    $user = $db->getUserRoleId($api_key);

    if ($user["role"]==0){
      // akan menjalankan fungsi untuk admin

    } else {
      // akan menjalankan fungsi untuk user
    }
```

Gambar 4.6 Potongan kode fungsi showNode()

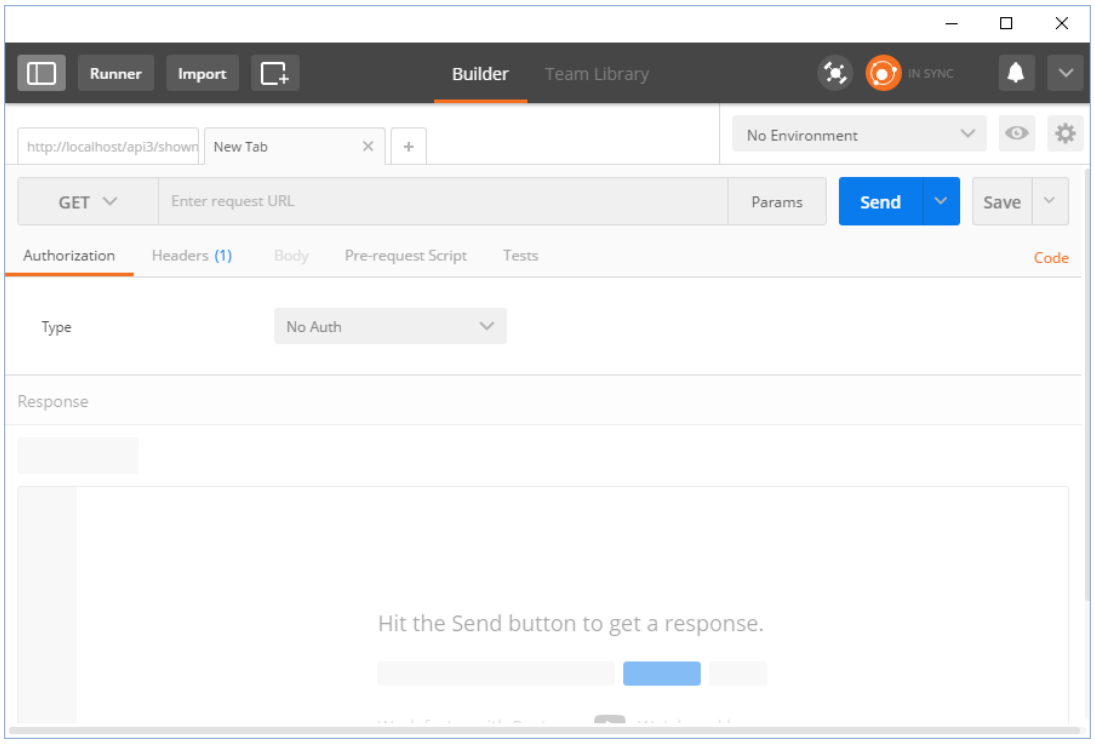
4.2 Uji Fungsionalitas

Pada tahap uji fungsionalitas ini akan dijelaskan bagaimana proses pengujian dan hasil dari pengujian yang telah dilakukan. Pengujian ini dilakukan untuk melakukan validasi dan verifikasi apakah layanan yang dikembangkan telah sesuai dengan *requirement* dalam tahap analisis kebutuhan sistem.

4.2.1 Persiapan Pengujian

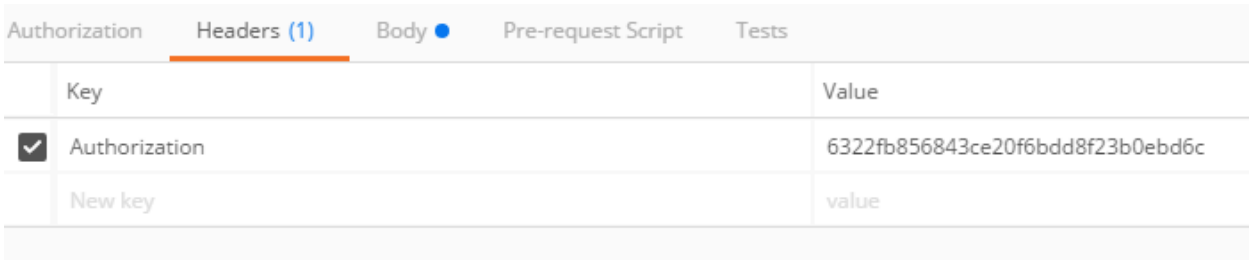
Pada tahap persiapan pengujian ini akan dijelaskan sekilas tentang aplikasi yang digunakan untuk melakukan pengujian. Pada saat penelitian ini dikerjakan, aplikasi sistem informasi untuk layanan yang dikembangkan belum dibuat

sehinggga untuk proses pengujian akan menggunakan aplikasi REST Client yang sudah ada. Aplikasi yang digunakan adalah Postman. Postman merupakan aplikasi native dan plugin dari google chrome yang berguna untuk melakukan uji coba terhadap REST API. Antarmuka aplikasi ini dapat dilihat pada Gambar 4.7.



Gambar 4.7 Antarmuka Aplikasi Postman

Untuk menjalankan aplikasi ini, pertama tentukan metode HTTP yang digunakan apakah POST, GET, PUT atau DELETE. Jika method yang dipilih adalah POST maka sertakan data yang diminta pada tab *Body*. Jika ingin menyertakan token saat melakukan request maka terlebih dahulu pilih tab *headers*. Masukkan *Key* berupa *Authorization* dan *Value* berupa token key seperti pada Gambar 4.8. Langkah kedua, isikan URI yang dituju pada kolom Request URL. Kemudian klik Send. Jika proses berhasil maka aplikasi akan menampilkan response dari URI yang dituju pada kolom dibawahnya.



Gambar 4.8 Request URI pada aplikasi Postman dengan menyertakan token

4.2.2 Hasil Pengujian

Menggunakan metode *black box testing*, layanan yang akan diuji pada tahapan ini adalah layanan pendaftaran user, layanan login user, layanan request token baru, layanan aktivasi user, layanan otentikasi token dan layanan manajemen user

1. Pengujian Layanan Pendaftaran User

Pada layanan pendaftaran user ini terdapat 4 skenario yang diuji. URI yang dilakukan tes adalah /register dengan metode POST. Data yang diperlukan pada body adalah nama, *username*, *password* dan email.

Dari hasil pengujian pada tabel 4.2 dapat diambil kesimpulan telah berhasil dikembangkan layanan pendaftaran user dengan validitas 100%.

Tabel 4.2 Hasil Pengujian Layanan Pendaftaran User

No	Skenario Pengujian	Hasil yang diharapkan	Hasil Pengujian
1	Melakukan request registrasi dengan menyertakan data yang diperlukan pada body (name, *username*, *password* dan email)	API akan menerima data dan mengembalikan pesan sukses bahwa user telah terdaftar	Valid
2	Melakukan request registrasi dengan tidak menyertakan data apapun	API akan mengembalikan pesan	Valid

		error berisi informasi data yang diperlukan	
3	Melakukan request registrasi dengan menyertakan sebagian data	API akan mengembalikan pesan error berisi informasi data yang kurang	Valid
4	Melakukan request registrasi dengan menyertakan data yang diperlukan pada body (*name, username, password* dan email) namun dengan email yang sudah pernah digunakan	API akan mengembalikan pesan error berisi informasi bahwa email sudah terdaftar	Valid

2. Pengujian Layanan Login User

Pada layanan *login* user ini terdapat 4 skenario yang diuji. URI yang dilakukan tes adalah /login dengan metode POST. Data yang diperlukan adalah email dan *password*.

Dari hasil pengujian pada tabel 4.3 dapat diambil kesimpulan telah berhasil dikembangkan layanan login user dengan validitas 100%.

Tabel 4.3 Hasil Pengujian Layanan Login User

No	Skenario Pengujian	Hasil yang diharapkan	Hasil Pengujian
1	Melakukan request login dengan menyertakan data yang sesuai saat registrasi (*email* dan *password*)	API akan menerima data dan mengembalikan informasi berupa data user	Valid

2	Melakukan request login dengan tidak menyertakan data apapun	API akan mengembalikan pesan error berisi informasi data yang diperlukan	Valid
3	Melakukan request login dengan menyertakan data sebagian	API akan mengembalikan pesan error berisi informasi data yang kurang	Valid
4	Melakukan request login dengan menggunakan *email* atau *password* yang salah	API akan mengembalikan pesan error berisi informasi bahwa email dan password yang digunakan salah	Valid

3. Pengujian Layanan Request Token

Pada layanan *request* token ini terdapat 4 skenario yang diuji. URI yang dilakukan tes adalah /apikey dengan metode GET. Pada pengujian ini harus menyertakan id dari user agar token yang diubah tidak tertukar dengan user yang lain. Jika user yang melakukan request memiliki id 3, maka alamat URI yang dituju menjadi /apikey/3.

Dari hasil pengujian pada tabel 4.4 dapat diambil kesimpulan telah berhasil dikembangkan layanan request token dengan validitas 100%.

Tabel 4.4 Hasil Pengujian Layanan Request Token

No	Skenario Pengujian	Hasil yang diharapkan	Hasil Pengujian
1	Melakukan *request* token baru dengan menyertakan id user yang sesuai	API akan menerima data dan mengembalikan	Valid

		informasi berupa token baru	
2	Melakukan *request* token baru dengan menyertakan id user yang tidak ada dalam database	API akan mengembalikan pesan error berisi informasi id user tidak ditemukan	Valid
3	Melakukan *request* dengan menggunakan token lama setelah mendapatkan token baru	API akan mengembalikan pesan error berisi informasi token salah	Valid
4	Melakukan *request* dengan menggunakan token baru yang telah didapatkan	API akan mengembalikan pesan sukses	Valid

4. Pengujian Layanan Aktivasi User

Pada layanan aktivasi user terdapat 3 skenario yang diuji. URI yang digunakan adalah /activate dengan metode GET. Pada pengujian ini harus disertakan id dari user yang akan dilakukan aktivasi. Jika user yang akan diaktivasi memiliki id 2, maka alamat URI menjadi /activate/2. Selain itu hanya user yang memiliki peran sebagai admin yang berhak melakukan proses aktivasi ini.

Dari hasil pengujian pada tabel 4.5 dapat diambil kesimpulan telah berhasil dikembangkan layanan aktivasi user dengan validitas 100%.

Tabel 4.5 Hasil Pengujian Layanan Aktivasi User

No	Skenario Pengujian	Hasil yang diharapkan	Hasil Pengujian
1	Melakukan aktivasi user dengan menyertakan id user yang sesuai dengan token admin	API akan menerima data dan mengembalikan pesan sukses	Valid

2	Melakukan aktivasi user dengan menyertakan id user yang sesuai namun dengan menggunakan token user biasa	API akan mengembalikan pesan error berisi informasi token harus admin	Valid
3	Melakukan aktivasi user dengan menyertakan id user yang tidak ada didalam	API akan mengembalikan pesan error berisi informasi id salah	Valid

5. Pengujian Layanan Otentikasi Token

Pada layanan otentikasi token ini terdapat 3 skenario yang diuji. URI yang dilakukan tes adalah URI smart farm yang telah diimplementasikan dengan otentikasi token. Metode yang digunakan adalah GET, PUT, dan DELETE.

Dari hasil pengujian pada tabel 4.6 dapat diambil kesimpulan telah berhasil dikembangkan layanan otentikasi token dengan validitas 100%.

Tabel 4.6 Hasil Pengujian Layanan Otentikasi Token

No	Skenario Pengujian	Hasil yang diharapkan	Hasil Pengujian
1	Melakukan *request* pada URI dengan menyertakan token yang sesuai	API akan memproses data *resource* yang dituju pada URI tersebut	Valid
2	Melakukan *request* pada URI dengan tidak menyertakan token	API akan mengembalikan pesan error berisi informasi bahwa token tidak ditemukan dan tidak memproses *request*	Valid

3	Melakukan *request* pada URI dengan menyertakan token yang salah	API akan mengembalikan pesan error berisi informasi bahwa token salah dan tidak memproses *request*	Valid

6. Pengujian Layanan Manajemen User

Pada layanan manajemen user ini terdapat 3 skenario yang diuji. URI yang dilakukan tes adalah URI /shownodes dengan metode GET. Pada pengujian ini, dalam setiap request harus menyertakan token yang valid. Peran dari pengguna dan data dari sensor node akan ditentukan sesuai dengan token yang disertakan.

Dari hasil pengujian pada tabel 4.7 dapat diambil kesimpulan telah berhasil dikembangkan layanan otentikasi manajemen user dengan validitas 100%.

Tabel 4.7 Hasil Pengujian Layanan Manajemen User

No	Skenario Pengujian	Hasil yang diharapkan	Hasil Pengujian
1	Melakukan *request show nodes* dengan menggunakan token admin	API akan mengembalikan data semua *sensor node*	Valid
2	Melakukan *request show nodes* dengan menggunakan token user yang memiliki *sensor node*	API akan mengembalikan data *sensor node private* dan *public* milik user tersebut ditambah dengan data sensor node lainnya yang bersifat *public*	Valid
3	Melakukan *request show nodes* dengan menggunakan token user	API akan mengembalikan data	Valid

	yang tidak memiliki sensor node	sensor node yang bersifat *public*	

4.3 Kelebihan dan Kelemahan Sistem

Secara umum layanan otentikasi dan manajemen user yang diusulkan didalam penelitian ini dapat dikembangkan untuk meningkatkan keamanan dan *privacy* data pada API smart farm. Beberapa kelebihan yang dimiliki dari layanan yang telah dikembangkan antara lain :

1. API yang dikembangkan mengukung *multi user*
2. Sistem API dapat menentukan peran dari pengguna, apakah admin atau user biasa
3. Hanya user yang tervalidasi yang dapat melakukan request terhadap *resource*

Sedangkan kelemahan dari sistem ini adalah sebagai berikut :

1. Peran pengguna masih terbatas 2 peran yaitu admin dan user
2. Jika token milik pengguna tersebar, maka pengguna harus melakukan request token baru
3. Manajemen user yang diterapkan masih terbatas untuk *request resource sensor node*

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah dilakukan penelitian dengan tema pengembangan otentikasi token dan manajemen user untuk sistem *internet of things* berbasis *restful* ini maka dapat diambil kesimpulan beberapa point sebagai berikut :

1. Telah berhasil dikembangkan layanan otentikasi token dan manajemen user dengan tambahan 6 layanan yaitu layanan pendaftaran, layanan login user, layanan request token, aktivasi user, layanan otentikasi token dan layanan manajemen user.
2. Berdasar hasil uji fungsionalitas dengan menggunakan 21 skenario, sistem telah terbukti berhasil memenuhi identifikasi kebutuhan yang telah ditentukan pada tahap analisis kebutuhan.
3. Keamanan dari API menjadi lebih baik karena penggunaan otentikasi token membuat akses *resource* melalui URI menjadi terbatas, hanya user yang terdaftar dan memiliki token yang dapat melakukan request
4. Pengembangan manajemen user berdasarkan peran telah berhasil membatasi hak akses user. Setiap user memiliki hak akses sendiri terhadap *sensor nodes* yang dimiliki

5.2 Saran

Saran yang dapat penulis berikan berdasarkan pada beberapa penelitian yang telah dilakukan adalah sebagai berikut :

1. User harus melalukan request token secara manual jika token yang dimiliki diketahui publik. Perlu dikembangkan otentikasi token yang lebih baik seperti penggunaan publik key dan secret key atau teknologi otentikasi seperti Oauth atau OpenID

2. Pembatasan manajemen user didalam penelitian ini masih dibatasi untuk akses terhadap *request sensor node*, perlu dikembangkan pembatasan untuk metode request seperti GET , POST, PUT dan DELETE

DAFTAR PUSTAKA

[1] "Gartner Says 6.4 Billion Connected 'Things' Will Be in Use in 2016, Up 30 Percent From 2015." [Online]. Available: http://www.gartner.com/newsroom/id/3165317. [Accessed: 20-Mar-2017].

[2] "DDoS attack that disrupted internet was largest of its kind in history, experts say | Technology | The Guardian." [Online]. Available: https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet. [Accessed: 20-Mar-2017].

[3] S. Lee, J. Y. Jo, and Y. Kim, "A Method for secure RESTful web service," *2015 IEEE/ACIS 14th Int. Conf. Comput. Inf. Sci. ICIS 2015 - Proc.*, pp. 77–81, 2015.

[4] A. Azis, "Pengembangan Restful API Untuk Mendukung Sistem Pemantauan Perkebunan Kelapa Sawit," Skripsi, Jurusan Teknologi Informasi, Departemen Teknik Elektro dan Teknologi Informasi. Universitas Gadjah Mada. 2017.

[5] S. R. Singh, J. Jayasuriya, C. Zhou, and M. Motani, "A RESTful web networking framework for vital sign monitoring," in *IEEE International Conference on Communications*, 2015, vol. 2015–Septe, pp. 524–529.

[6] S. Kim, J. Y. Hong, S. Kim, S. H. Kim, J. H. Kim, and J. Chun, "Restful Design and Implementation of Smart Appliances for Smart Home," in *Proceedings - 2014 IEEE International Conference on Ubiquitous Intelligence and Computing, 2014 IEEE International Conference on Autonomic and Trusted Computing, 2014 IEEE International Conference on Scalable Computing and Communications and Associated Sy*, 2015, pp. 717–722.

[7] E. Communications and B. S. Preethi, "Cloud Integrated Temperature Sensor Using Restful Web Services," *Int. J. Comput. Sci. Eng. Commun.*, vol. 3, no.

3, pp. 1103–1107, 2015.

[8] S. Lawrence and L. Stewart, *HTTP Authentication : Basic dan Digest Access Authentication*. The Internet Society. 1999.

[9] D. Peng, C. Li, and H. Huo, "An extended UsernameToken-based approach for REST-style Web Service Security Authentication," *Comput. Sci. Inf. Technol. 2009. ICCSIT 2009. 2nd IEEE Int. Conf.*, pp. 582–586, 2009.

[10] X. W. Huang, C. Y. Hsieh, C. H. Wu, and Y. C. Cheng, "A token-based user authentication mechanism for data exchange in RESTful API," in *Proceedings - 2015 18th International Conference on Network-Based Information Systems, NBiS 2015*, 2015, pp. 601–606.

[11] S. W. Oh and H. S. Kim, "Study on access permission control for the Web of Things," *Int. Conf. Adv. Commun. Technol. ICACT*, vol. 2015–Augus, no. 1, pp. 574–580, 2015.

[12] A. Ouaddah, I. Bouij-Pasquier, A. Abou Elkalam, and A. Ait Ouahman, "Security analysis and proposal of new access control model in the Internet of Thing," in *2015 International Conference on Electrical and Information Technologies (ICEIT)*, 2015, pp. 30–35.

[13] F. De Backere, B. Hanssens, R. Heynssens, R. Houthooft, A. Zuliani, S. Verstichel, B. Dhoedt, and F. De Turck, "Design of a security mechanism for RESTful web service communication through mobile clients," in *IEEE/IFIP NOMS 2014 - IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World*, 2014.

[14] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing," *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003.

[15] A. Knud and L. Lueth, "IoT basics : Getting started with the Internet of Things IoT Analytics IoT basics : Getting started with the Internet of Things," *IoT Analytic*, no. March, pp. 0–9, 2015.

[16] A. M. James Manyika, Michael Chui, Jacques Bughin, Richard Dobbs, Peter

Bisson, “Disruptive technologies: Advances that will transform life, business, and the global economy | McKinsey & Company.” [Online]. Available: http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/disruptive-technologies. [Accessed: 17-Apr-2017].

[17] J. Tan and S. G. M. Koo, “A Survey of Technologies in Internet of Things,” *2014 IEEE Int. Conf. Distrib. Comput. Sens. Syst.*, pp. 269–274, 2014.

[18] G. Omojokun, “A Survey of ZigBee Wireless Sensor Network Technology: Topology, Applications and Challenges,” *Int. J. Comput. Appl.*, vol. 130, no. 9, pp. 47–55, 2015.

[19] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Cla, “Middleware for internet of things: A survey,” *IEEE Internet Things J.*, vol. 3, no. 1, pp. 70–95, 2016.

[20] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, “Context aware computing for the internet of things: A survey,” *IEEE Commun. Surv. Tutorials*, vol. 16, no. 1, pp. 414–454, 2014.

[21] V. M. S. N. A. H. H. Ngu M. Gutierrez and Q. Z. Sheng, “IoT middleware: a survey on issue and enabling technologies,” *IEEE Internet Things J.*, vol. 4, no. 1, pp. 1–20, 2016.

[22] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Comput. Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[23] M. Eisenhauer, P. Rosengren, and P. Antolin, “A development platform for integrating wireless devices and sensors into Ambient Intelligence systems,” in *2009 6th IEEE Annual Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks Workshops, SECON Workshops 2009*, 2009, vol. 0, no. c, pp. 1–3.

[24] “IoT Platform for Connected Devices| Xively by LogMeIn.” [Online]. Available: https://www.xively.com/. [Accessed: 17-May-2017].

[25] P. Persson and O. Angelsmark, “Calvin – Merging Cloud and IoT,” *Procedia*

Comput. Sci., vol. 52, no. Ant, pp. 210–217, 2015.

[26] S. Mumbaikar and P. Padiya, "Web Services Based On SOAP and REST Principles," *Int. J. Sci. Res. Publ.*, vol. 3, no. 5, pp. 3–6, 2013.

[27] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," *DISSERTATION*, vol. 54, p. 162, 2000.

[28] K. Wagh and R. Thool, "A Comparative Study of SOAP Vs REST Web Services Provisioning Techniques for Mobile Host," *J. Inf. Eng. Appl.*, vol. 2, no. 5, pp. 12–16, 2012.

[29] R. Lucchi, M. Millot, and C. Elfers, "Resource Oriented Architecture and REST," *Assess. impact advantages INSPIRE, Ispra Eur. Communities*, pp. 5–13, 2008.

[30] ECMA International, "Standard ECMA-262 ECMAScript® 2016 Language Specification," p. 586, 2016.

[31] "A Comprehensive Interview About Slim The Micro PHP Framework." [Online]. Available: http://7php.com/slim-php-framework-interview/. [Accessed: 18-Apr-2017].

[32] "Documentation - Slim Framework." [Online]. Available: https://www.slimframework.com/docs/. [Accessed: 18-Apr-2017].

[33] L. Williams, "Testing Overview and Black-Box Testing Techniques," *Int. Conf. Softw. Eng. 2007*, pp. 35–59, 2006.

[34] M. S. Mustaqbal, R. F. Firdaus, and H. Rahmadi, "Pengujian Aplikasi Menggunakan Black Box Testing Boundary Value Analysis," *J. Ilm. Teknol. Inf. Terap.*, vol. I, no. 3, pp. 31–36, 2015.

[35] "PHP: CSPRNG - Manual." [Online]. Available: http://php.net/manual/en/book.csprng.php. [Accessed: 27-Apr-2017].

[36] "Middleware - Slim Framework v2." [Online]. Available: http://docs.slimframework.com/routing/middleware/. [Accessed: 09-May-2017].