TUGAS AKHIR - KI141502

RANCANG BANGUN SISTEM MONITORING PERANGKAT PUSAT DATA INSTITUT TEKNOLOGI SEPULUH NOPEMBER (ITS) SURABAYA DENGAN IMPLEMENTASI PUBLISH-SUBSCRIBE DAN TELEGRAM API

NAFIA RIZKY YOGAYANA
NRP 05111440000017

Dosen Pembimbing I
Royyana Muslim Ijtihadie, S.Kom, M.Kom, PhD

Dosen Pembimbing II
Henning Titi Ciptaningtyas, S.Kom, M.Kom.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR - KI141502

RANCANG BANGUN SISTEM MONITORING PERANGKAT PUSAT DATA INSTITUT TEKNOLOGI SEPULUH NOPEMBER (ITS) SURABAYA DENGAN IMPLEMENTASI PUBLISH-SUBSCRIBE DAN TELEGRAM API

NAFIA RIZKY YOGAYANA
NRP 05111440000017

Dosen Pembimbing I
Royyana Muslim Ijtihadie, S.Kom, M.Kom, PhD

Dosen Pembimbing II
Henning Titi Ciptaningtyas, S.Kom, M.Kom.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS - KI141502

DEVICE MONITORING SYSTEM DESIGN FOR DATA CENTER OF INSTITUT TEKNOLOGI SEPULUH NOPEMBER (ITS) SURABAYA USING PUBLISH-SUBSCRIBE AND TELEGRAM API IMPLEMENTATION

NAFIA RIZKY YOGAYANA
NRP 05111440000017

Supervisor I
Royyana Muslim Ijtihadie, S.Kom, M.Kom, PhD

Supervisor II
Henning Titi Ciptaningtyas, S.Kom, M.Kom.

Department of INFORMATICS
Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya, 2018

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

RANCANG BANGUN SISTEM MONITORING PERANGKAT PUSAT DATA INSTITUT TEKNOLOGI SEPULUH NOPEMBER (ITS) SURABAYA DENGAN IMPLEMENTASI PUBLISH-SUBSCRIBE DAN TELEGRAM API

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Arsitektur dan Jaringan Komputer
Program Studi S1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :
NAFIA RIZKY YOGAYANA
NRP: 05111440000017

Disetujui oleh Dosen Pembimbing Tugas Akhir

Royyana Muslim Ijtihadie, S.Kom, M.Kom, PhD ..... (Pembimbing 1)
NIP: 197708242006041001

Henning Titi Ciptaningtyas, S.Kom, M.Kom ..... (Pembimbing 2)
NIP: 198407082010122004

SURABAYA
Juni 2018

(Halaman ini sengaja dikosongkan)

RANCANG BANGUN SISTEM MONITORING PERANGKAT PUSAT DATA INSTITUT TEKNOLOGI SEPULUH NOPEMBER (ITS) SURABAYA DENGAN IMPLEMENTASI PUBLISH-SUBSCRIBE DAN TELEGRAM API

Nama	:	**NAFIA RIZKY YOGAYANA**
NRP	:	**05111440000017**
Jurusan	:	**Informatika FTIK**
Pembimbing I	:	**Royyana Muslim Ijtihadie, S.Kom, M.Kom, PhD**
Pembimbing II	:	**Henning Titi Ciptaningtyas, S.Kom, M.Kom.**

Abstrak

Semakin majunya teknologi, maka semakin banyak penggunaan perangkat berupa perangkat komputer maupun jaringan. Salah satu perangkat yang umum digunakan adalah server. Fungsi dari server ini adalah sebagai penyedia layanan dalam sebuah jaringan. Kemudian, pada saat sekarang ini penggunaan IoT (Internet of Things) pun semakin berkembang. Sebagai contoh adalah penggunaan single-board computer Raspberry Pi (Rpi) yang memiliki fungsi yang sama dengan server hanya saja berukuran mini. Rpi ini dapat digunakan sebagai server untuk berbagai macam sensor.

Pada Pusat Data Institut Teknologi Sepuluh Nopember (DPTSI ITS), tentu kebutuhan akan perangkat jaringan seperti server dan Rpi sangat besar. Semakin banyak jumlah perangkat yang digunakan, maka semakin banyak pula jumlah perangkat yang harus dipantau. Administrator mengalami kesulitan dalam memantau semua perangkat karena jumlahnya yang banyak tersebut. Oleh karena itu, dibutuhkan sebuah sistem yang dapat

memantau kinerja dari perangkat yang ada di DPTSI ITS.

Pada Tugas Akhir ini, dirancang sebuah aplikasi dalam bentuk web yang berfungsi untuk memantau seluruh server dan Rpi yang ada pada jaringan. Metode yang digunakan adalah publish-subscribe. Dengan metode ini, pengguna dapat memilih server mana yang ingin dipantau dan memilih informasi apa saja yang ingin didapat dari tiap server yang terpilih. Selain itu, pengguna juga dapat memilih Rpi mana yang ingin dipantau data sensornya sehingga lingkungan DPTSI ITS terpantau dengan baik.

Administrator dapat memantau satu atau lebih perangkat yang diinginkan saja. Ketika server mengalami masalah, maka sistem akan mengirimkan pesan kepada pemantau melalui Telegram. Hal ini diharapkan para administrator di DPTSI ITS tidak lagi mengalami kesulitan saat memantau perangkat yang ada.

***Kata-Kunci**: pemantauan, server, publish-subscribe, Raspberry Pi*

DEVICE MONITORING SYSTEM DESIGN FOR DATA CENTER OF INSTITUT TEKNOLOGI SEPULUH NOPEMBER (ITS) SURABAYA USING PUBLISH-SUBSCRIBE AND TELEGRAM API IMPLEMENTATION

Name	:	**NAFIA RIZKY YOGAYANA**
NRP	:	**05111440000017**
Major	:	**Informatics FTIK**
Supervisor I	:	**Royyana Muslim Ijtihadie, S.Kom, M.Kom, PhD**
Supervisor II	:	**Henning Titi Ciptaningtyas, S.Kom, M.Kom.**

Abstract

The more advanced technology, the more the use of devices in the form of computer and network devices. One of the commonly used devices is the server. The function of this server is as a service provider in a network. Then, at this moment the use of IoT (Internet of Things) is growing. An example is the use of a single-board computer Raspberry Pi (Rpi) which has the same functionality as the server is only a mini-sized. This Rpi can be used as a server for various sensors.

At Data Center Institute of Technology Sepuluh Nopember (DPTSI ITS), of course the need for network devices such as servers and Rpi is very large. The large number of devices used, the more the number of devices to monitor. Administrators have difficulty monitoring all devices due to their large numbers. Therefore, it needs a system that can monitor the performance of the existing devices in DPTSI ITS.

In this Final Project, designed an application in web form that serves to monitor the entire server and Rpi on the network. The

method used is publish-subscribe. With this method, the user can choose which server to monitor and choose what information you want to get from each server selected. In addition, users can also choose Rpi which want to monitor the sensor data so that DPTSI ITS environment monitored properly.

Administrators can monitor one or more of the desired devices only. When the server encounters a problem, the system will send a message to the monitor via Telegram. It is expected that the administrators in DPTSI ITS no longer have difficulty when monitoring the existing devices.

***Keywords**: monitoring, server, publish-subscribe, Raspberry Pi*

KATA PENGANTAR

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **Rancang Bangun Sistem *Monitoring* Perangkat Pusat Data Institut Teknologi Sepuluh Nopember (ITS) Surabaya dengan Implementasi *Publish-Subscribe* dan Telegram API**. Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang sangat baik bagi penulis. Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Departemen Informatika ITS. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari. Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT atas anugerahnya yang tidak terkira kepada penulis dan Nabi Muhammad SAW.
2. Bapak, Mama, Ghina dan keluarga Penulis yang selalu memberikan perhatian, dorongan dan kasih sayang yang menjadi semangat utama bagi diri Penulis sendiri baik selama Penulis menempuh masa perkuliahan maupun pengerjaan Tugas Akhir ini.
3. Bapak Royyana Muslim Ijtihadie, S.Kom., M.Kom., PhD. selaku Dosen Pembimbing yang telah banyak meluangkan waktu untuk memberikan ilmu, nasihat, motivasi, pandangan dan bimbingan kepada Penulis baik selama Penulis menempuh masa kuliah maupun selama pengerjaan Tugas Akhir ini.
4. Ibu Henning Titi C., S.Kom., M.Kom. selaku dosen

pembimbing yang telah memberikan ilmu, dan masukan kepada Penulis.

5. Seluruh tenaga pengajar dan karyawan Departemen Informatika ITS yang telah memberikan ilmu dan waktunya demi berlangsungnya kegiatan belajar mengajar di Departemen Informatika ITS.
6. Teman-teman, Kakak-kakak dan Adik-adik *administrator* Workshop Pemrograman 2 yang selalu menjadi teman untuk berbagi canda, tawa, dan dukungan.
7. Teman-teman, Kakak-kakak dan Adik-adik *administrator* Laboratorium Arsitektur dan Jaringan Komputer yang selalu menjadi teman untuk berbagi ilmu.
8. Mas John S. Peter yang telah memberikan semangat dan dukungan penuh selama Penulis mengerjakan Tugas Akhir ini di sela-sela kesibukannya.
9. Afif Ridho, Fourir Akbar, dan Fathoni Adi yang setia menyemangati, membantu, dan menemani Penulis selama pengerjaan Tugas Akhir ini.
10. Bebet, Raras, Rara, Ival, Aufar, Ical, Mila, dan teman-teman TC14 yang menjadi teman berbagi dan penyemangat Penulis.

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

Surabaya, Juni 2018

Nafia Rizky Yogayana

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I

PENDAHULUAN

Pada bab ini akan dijelaskan latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi dan sistematika penulisan Tugas Akhir.

1.1 Latar Belakang

Monitoring adalah suatu kegiatan yang meliputi observasi dan pengecekan terhadap kemajuan suatu proses atau kualitas dari suatu barang atau pekerjaan dan dilakukan secara berkala. Banyak hal yang bisa dimonitor, mulai dari barang, kesehatan, hingga pekerjaan. Untuk keperluan Tugas Akhir ini, *monitoring* dilakukan terhadap perangkat yang ada di Pusat Data Institut Teknologi Sepuluh Nopember (ITS) Surabaya, yakni server dan komputer *single-board*. Tujuan dari adanya *monitoring* ini adalah untuk memantau jaringan dan lingkungan di Pusat Data ITS.

Monitoring server adalah kegiatan memantau sebuah server dari segi kinerja perangkat keras, *traffic* lalu lintas data, dan masih banyak lagi. *Monitoring* server ini merupakan suatu pekerjaan yang sangat penting untuk dilakukan dalam memanajemen jaringan. *Monitoring* ini menjadi suatu titik yang menentukan apakah suatu layanan jaringan sudah berjalan dengan baik atau tidak.

Dalam suatu kegiatan *monitoring* server, tidak semua pengguna layanan bisa melakukan hal tersebut karena terkait dengan hak akses masing-masing. Dan dalam pelaksanaannya selama ini, *monitoring* server dilakukan secara manual dan tidak seragam. Hal ini yang menyebabkan administrator merasa kesulitan dalam mengatasi masalah jaringan yang terjadi.

Selain itu, lingkungan di Pusat Data ITS juga perlu dipantau. Amat sulit rasanya jika administrator harus memantau dari komputer *single-board* secara manual. Pekerjaan ini

membutuhkan ekstra waktu. Dan belum tentu juga administrator langsung mengetahui jika ada server yang *down* atau keadaan lain yang membutuhkan penanganan langsung.

Berdasarkan uraian di atas maka dapat dilihat bahwa dibutuhkan sebuah sistem untuk menangani masalah *monitoring* perangkat yang ada di Pusat Data ITS ini. Dengan adanya sistem yang seragam, diharapkan dapat membantu para administrator dalam memantau server dan komputer *single-board* dengan lebih mudah dan efisien.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana cara membangun sistem administrasi *monitoring* untuk pemantauan server?
2. Bagaimana cara mengimplementasikan *publish-subscribe* pada monitoring server?
3. Bagaimana cara membangun *agent* yang melaporkan keadaan server ke pengguna?
4. Bagaimana cara mengambil informasi penggunaan CPU, memori, *bandwidth*, dan hal-hal lain yang terkait pada server?
5. Bagaimana cara mengimplementasikan *publish-subscribe* pada komputer papan tunggal (*single board computer*) untuk pemantauan lingkungan Pusat Data ITS?
6. Bagaimana cara mengintegrasikan antara *agent* ke Telegram untuk mengirim notifikasi mengenai keadaan lingkungan atau server di Pusat Data ITS yang membutuhkan penanganan langsung?

1.3 Batasan Masalah

Permasalahan yang dibahas pada Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Sistem hanya melakukan *monitoring* pada Linux Server dan Raspberry Pi sebagai komputer *single-board* untuk membaca sensor.
2. Sistem ini diimplementasikan di Pusat Data Institut Teknologi Sepuluh Nopember (ITS) Surabaya.

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

1. Membuat sebuah sistem *monitoring* untuk perangkat di Pusat Data ITS berupa server dan komputer *single-board* yang seragam untuk seluruh pengguna.
2. Mengimplementasikan metode *publish-subscribe* pada sistem *monitoring* server dan komputer *single-board* di Pusat Data ITS.

1.5 Manfaat

Manfaat dari Tugas Akhir ini adalah sebagai berikut:

1. Membangun sebuah sistem administrator *monitoring* server dan komputer *single-board* agar memudahkan para admin dalam memantau server yang ada.
2. Membangun sistem administrator untuk memantau lingkungan Pusat Data ITS dengan sensor yang dihubungkan ke Raspberry Pi.
3. Memonitoring server dan yang hanya ingin dimonitoring dengan cara memilih saluran server yang ada.
4. Membangun *agent* dengan sistem *publish-subscribe* agar dapat melaporkan keadaan server ke pengguna.

5. Membangun *agent* untuk diletakkan di komputer *single-board* agar bisa melaporkan keadaan lingkungan Pusat Data ITS ke pengguna dengan mengimplementasikan *publish-subscribe*.
6. Mengetahui informasi seperti penggunaan CPU, memori, *bandwidth*, dan hal lain yang terkait pada server dengan adanya sistem *monitoring* server ini.
7. Memberitahu pengguna tentang keadaan server atau lingkungan Pusat Data ITS yang butuh perhatian langsung melalui aplikasi Telegram.

1.6 Metodologi

Metodologi yang digunakan pada pengerjaan Tugas Akhir ini adalah sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir yang berisi gagasan untuk menyelesaikan permasalahan di Pusat Data ITS Surabaya.

Proposal tugas akhir ini berisi tentang deskripsi pendahuluan dari tugas akhir yang akan dibuat. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya usulan tugas akhir, rumusan masalah yang diangkat, batasan masalah untuk tugas akhir, tujuan dari pembuatan tugas akhir, dan manfaat dari hasil pembuatan tugas akhir. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan tugas akhir. Sub bab metodologi berisi penjelasan mengenai tahapan penyusunan tugas akhir mulai dari penyusunan proposal hingga penyusunan buku tugas akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal pengerjaan tugas akhir.

1.6.2 Studi literatur

Studi literatur merupakan langkah yang dilakukan untuk mendukung dan memastikan setiap tahap pengerjaan tugas akhir sesuai dengan standar dan konsep yang berlaku. Pada tahap studi literatur ini, akan dilakukan studi mendalam mengenai *monitoring* server, Raspberry Pi, dan *Publish-Subscribe*. Adapun literatur yang dijadikan sumber berasal dari paper, buku, materi perkuliahan, forum serta artikel dari internet.

1.6.3 Desain dan Perancangan Sistem

Pada tahap ini dilakukan desain dan perancangan sistem *monitoring* perangkat di Pusat Data ITS Surabaya.

Aktor dari aplikasi ini adalah administrator yang akan melakukan *monitoring* server dan komputer *single-board*. Admin memilih server atau perangkat mana yang ingin ia pantau. Dari permintaan tersebut, maka sistem akan memberikan informasi perangkat terpilih kepada pengguna yang memilih perangkat tersebut.

Tahap ini merupakan tahap yang paling penting dimana bentuk awal aplikasi yang akan diimplementasikan didefinisikan. Pada tahapan ini dibuat desain dari arsitektur sistem.

1.6.4 Implementasi Sistem

Implementasi merupakan tahap membangun dan mengimplementasikan rancangan sistem yang telah dibuat. Pada tahapan ini merealisasikan apa yang telah didesain dan dirancang pada tahapan sebelumnya, sehingga menjadi sebuah sistem yang sesuai dengan apa yang telah direncanakan.

1.6.5 Uji Coba dan Evaluasi

Pada tahapan ini dilakukan uji coba terhadap sistem yang telah dibuat. Pengujian dan evaluasi akan dilakukan dengan melihat kesesuaian dengan perencanaan. Selain itu, tahap ini juga akan melakukan uji performa sistem untuk mengetahui efisiensi penggunaan sumber daya serta evaluasi berdasarkan hasil uji performa tersebut.

Pengujian dalam aplikasi ini akan dilakakukan dalam beberapa cara, antara lain:

- Pengujian pada keberhasilan dalam mengambil informasi dari tiap-tiap server (*monitoring* server).
- Pengujian pada keberhasilan dalam pengambilan informasi dari komputer *single-board* untuk memantau lingkungan Pusat Data ITS.
- Pengujian ini berfokus pada ketepatan informasi dari hasil *monitoring* perangkat dan diberikan (*publish*) ke pengguna yang meminta (*subscribe*).

1.6.6 Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan buku Tugas Akhir yang berisi dokumentasi hasil pengerjaan Tugas Akhir.

1.7 Sistematika Penulisan

Berikut adalah sistematika penulisan buku Tugas Akhir ini:

1. BABI: PENDAHULUAN
   Bab ini berisi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi dan sistematika penulisan Tugas Akhir.
2. BAB II: DASAR TEORI
   Bab ini berisi dasar teori mengenai permasalahan dan algoritma penyelesaian yang digunakan dalam Tugas

Akhir

3. BAB III: DESAIN
   Bab ini berisi desain algoritma dan struktur data yang digunakan dalam penyelesaian permasalahan.
4. BAB IV: IMPLEMENTASI
   Bab ini berisi implementasi berdasarkan desain algortima yang telah dilakukan pada tahap desain.
5. BAB V: UJI COBA DAN EVALUASI
   Bab ini berisi uji coba dan evaluasi dari hasil implementasi yang telah dilakukan pada tahap implementasi.
6. BAB VI: PENUTUP
   Bab ini berisi kesimpulan dan saran yang didapat dari hasil uji coba yang telah dilakukan.

(Halaman ini sengaja dikosongkan)

BAB II

DASAR TEORI

Pada bab ini akan dijelaskan mengenai dasar teori yang menjadi dasar pengerjaan Tugas Akhir ini.

2.1 Deskripsi Umum

Pada subbab ini akan dijelaskan mengenai deskripsi-deskripsi umum yang terdapat pada Tugas Akhir ini.

2.1.1 *Publish-Subscribe*

Secara umum, maksud dari *publish-subscribe* adalah susunan dari sekumpulan node yang didistribusikan melalui jaringan komunikasi. Klien dari sistem ini dibagi menjadi dua peran yaitu *publisher* dan *subscriber*.[1]

Publisher adalah yang penyedia informasi dan memberikannya kepada yang meminta informasi tersebut. Sedangkan *subscriber* bertindak sebagai konsumen informasi tersebut. Dua klien ini tidak diharuskan untuk berkomunikasi secara langsung di antara mereka sendiri yang namun agak terpisah: interaksi terjadi melalui nodes sistem *publish-subscribe*, yang mengkoordinasikan diri mereka untuk mengarahkan informasi dari penerbit ke pelanggan. Pengunaan *publisher* dan *subscriber* ini dapat memungkinkan skalabilitas yang lebih besar dan topologi jaringan yang lebih dinamis.

Terdapat dua jenis *publish-subscribe*, yaitu *topic based publish-subscribe* dan *content based publish-subscribe*.[2] *Topic based publish-subscribe* merupakan pola pengiriman pesan *publish-subscribe* dimana penyedia informasi menyampaikan pesan ke konsumen berdasarkan topik yang dipilihnya. *Content based publish-subscribe* merupakan pola pengiriman pesan *publish-subscribe* dimana penyedia informasi menyampaikan pesan ke konsumen berdasarkan isi dari pesan yang ada. Pada

umumnya pola pengiriman pesan *publish-subscribe* merupakan *topic based*.

2.1.2 *Websocket*

Websocket merupakan protokol komunikasi yang memungkinkan komunikasi dua arah antara *web client* dengan *server*.[3] Penerapan komunikasi dua arah ini dapat digambarkan oleh pengguna telepon. Kedua orang yang sedang bertelepon dapat berbicara dan mendengarkan secara bersamaan dan *real-time*. Protokol *websocket* memungkinkan komunikasi dua arah antara klien dengan *server* secara *real-time*.

Pada umumnya *web client* mendapatkan satu kali tanggapan dari *server* untuk setiap satu permintaan. Alur tersebut kurang tepat apabila digunakan untuk menerapkan pola pengiriman pesan *publish-subscribe* dimana *subscriber* biasanya akan menunggu terus menerus terhadap pesan yang dikirimkan oleh *publisher*. Sedangkan *publisher* mungkin saja mengirimkan pesan lagi setelah 5 menit. Maka dari itu *websocket* digunakan untuk menampilkan secara *real-time* pesan yang diperoleh dari *publisher* walaupun harus menunggu dalam jangka waktu yang tidak pasti.

Untuk membuat koneksi *websocket*, klien harus mengirimkan *request* HTTP kepada *server*. Setelah itu protokol akan diubah menjadi protokol *Websocket*, lalu *server* akan mengenali tipe *request* berdasarkan *header* pada HTTP. Protokol akan diupgrade menjadi *Websocket* apabila diminta. Kemudian klien dan server akan memulai komunikasi *full-duplex*, yang berarti klien dan server dapat bertukar data kapanpun sampai salah satu dari klien atau server menutup koneksi tersebut.

2.1.3 Rabbitmq

Rabbitmq merupakan perantara (broker) pada pertukaran pesan.[4] Rabbitmq juga biasa disebut *message-oriented middleware*. Rabbitmq mendukung pola pengiriman pesan *publish-subscribe*.

Pada arsitektur pola pengiriman pesan *publish-subscribe* yang terdapat di rabbitmq akan sering ditemui istilah-istilah seperti *exchange* dan *queue*. *Exchange* dapat diartikan sebagai persimpangan, sedangkan *queue* dapat diartikan sebagai tempat penyimpanan. Rabbitmq menggunakan istilah *producer* untuk pihak yang membuat pesan untuk nantinya dikirim ke broker dan *consumer* untuk pihak yang nantinya akan menerima pesan dari broker yang dibuat oleh *producer*. *Producer* pada pola pengiriman pesan *publish-subscribe* dikenal dengan istilah *publisher* sedangkan *consumer* dikenal dengan istilah *subscriber*.

Pada arsitektur pola pengiriman pesan *publish-subscribe* pesan yang dibuat oleh *text*publisher mungkin saja dikonsumsi oleh beberapa *subscriber*. *Publisher* nantinya akan mengirim pesan ke suatu *exchange* yang berada pada broker, akan tetapi *exchange* tidak dapat menyimpan pesan. Ketika suatu *exchange* yang dikirimi pesan tidak memiliki *queue* yang tersambung, maka pesan tersebut akan hilang. Pada kasus ini masing-masing *subscriber* akan membuat *queue* yang berbeda untuk menyimpan pesan yang dibuat oleh *publisher*. Oleh rabbitmq pesan yang sudah pernah dipakai (*consume*) akan langsung dihapus dari *queue*.

2.1.4 Nagios

Nagios adalah perangkat lunak open source yang menyediakan alat pemantauan yang sempurna yang dapat membantu untuk memantau seluruh protokol yang aktif dan perangkat jaringan yang terhubung dengan topologi.[5] Nagios

juga merupakan sistem pemantauan yang paling yang paling populer yang cocok dengan hampir semua distribusi linux. Aplikasi ini memiliki banyak *plugin* tambahan yang dikembangkan oleh pengguna maupun yang terdapat langsung pada awal pengaturan. Peralatan pemantauan yang berbasis Nagios juga tersedia, seperti sensor yang dirancang untuk beroperasi bersama Nagios. Karena fleksibilitas dari rancangan perangkat lunak yang menggunakan arsitektur *plugin*, layanan pengecekan untuk aplikasi yang pustakanya sudah ditentukan dapat digunakan. Di dalam Nagios terdapat beberapa *plugin* lain, seperti script tambahan yang dapat dikostumisasi dan dapat digunakan pada Nagios. Nagios juga mampu untuk menyediakan grafik yang komperhensif dan bersifat *real-time* dan analisis tren.

2.1.5 NRPE

NRPE adalah *plugin* dari Nagios yang memungkinkan pengguna untuk menjalankan Nagios secara *remote* atau jarak jauh.[6] Alasan utama untuk memasang NRPE ini adalah untuk memungkinkan Nagios melakukan pemantauan sumber daya ”lokal” (seperti beban CPU, penggunaan memori, dan lain-lain) pada mesin jarak jauh. Agen NRPE harus diinstal pada *host* yang ingin dipantau.

Pada NRPE kita bisa melakukan pengecekan pada *disk* dan *load* untuk sumber daya dan servis lokal. Sedangkan untuk *remote services* di *host* lain dapat dilakukan `check_http` dan `check_ftp`.

2.1.6 Raspberry Pi

Raspberry Pi adalah salah satu jenis *Single-Board Computer* (SBC) yang bisa digunakan dalam sistem pemantauan. Raspberry Pi merupakan SBC yang memiliki *low-power* dengan ukuran sebesar kartu kredit yang dirilis pada tahun 2012. Alat ini

bisa dibilang merupakan alat yang paling populer yang dikembangkan oleh Universitas Cambridge UK untuk kepentingan pendidikan komputasi. [7]

Raspberry Pi ini memiliki RAM sebesar 1 GB, 900 Mhz *quad-core* ARM Cortex A7 CPU, 10/100 Ethernet, dan dihargai sebesar $35.00. Alat ini dapat digunakan di sistem operasi Linux dan Windows. Selain itu, alat ini banyak diadaptasi untuk berbagai macam aplikasi seperti sensor jaringan nirkabel, robotika, dan UAV.[8]

2.1.7 Telegram

Telegram adalah aplikasi untuk *chatting* yang fokus pada kecepatan dan keamanan.[9] Aplikasi ini tidak berbayar atau gratis. Kelebihan dari Telegram adalah kita dapat membuka akun kita di banyak perangkat secara bersamaan karena pesan akan disinkronisasi dengan baik di sejumlah ponsel, tablet, atau komputer kita.

Telegram juga memiliki API yang memperbolehkan pengguna untuk membuat klien Telegram yang disesuaikan dengan pengguna tersebut. API ini gratis untuk siapa saja yang ingin membuat aplikasi Telegram di *platform* ini. Telegram sudah menyediakan *open source code* untuk aplikasi Telegram yang sudah ada agar pengguna bisa melihat bagaimana segala sesuatunya bekerja di sini. [9]

(Halaman ini sengaja dikosongkan)

BAB III

DESAIN DAN PERANCANGAN SISTEM

Pada bab ini akan dibahas tentang dasar perancangan sistem yang akan dibuat. Secara khusus akan dibahas mengenai deskripsi umum sistem, perancangan skenario dan arsitektur sistem.

3.1 Deskripsi Umum Sistem

Tugas akhir ini disusun untuk menangani masalah *monitoring* server dan *computer single-board* dengan menggunakan pola *publish-subscribe*. Pola ini dipilih agar pengguna dapat memilih perangkat mana yang ingin dipantau sehingga tidak semua perangkat terpantau oleh semua orang.

Proses *monitoring* ini diawali dengan pengambilan informasi tiap-tiap server menggunakan SNMP *monitoring tools*. Untuk komputer *single-board* diberikan *agent* untuk mengambil data hasil pemantauan. Kemudian pengguna, atau di sini dapat disebutkan sebagai administrator jaringan, dapat memilih saluran (*subscribe*) *server* atau komputer *single-board* mana yang ingin ia pantau melalui sebuah *agent* yang berupa *web-socket*.

Kemudian, *agent* yang berupa web yang responsif ini akan melanjutkan permintaan ke sebuah *middleware*. *Middleware* yang telah terpola *publish-subscribe* ini akan memproses permintaan tersebut. Kemudian perangkat yang terpilih akan memberikan informasinya (*publish*) melalui *middleware* dan diteruskan kembali ke *agent* berupa notifikasi. Notifikasi ini dapat dilihat oleh admin. Selain itu, *agent* juga mengirimkan notifikasi mengenai keadaan lingkungan dan *server* Pusat Data ITS yang butuh penanganan langsung, seperti *server time out*, suhu ruangan meningkat tajam, ke Telegram. Sehingga administrator bisa langsung bertindak cepat.

3.2 Arsitektur Sistem

Pada subbab ini, dibahas mengenai tahap analisis dan kebutuhan bisnis dan desain dari sistem yang akan dibangun.

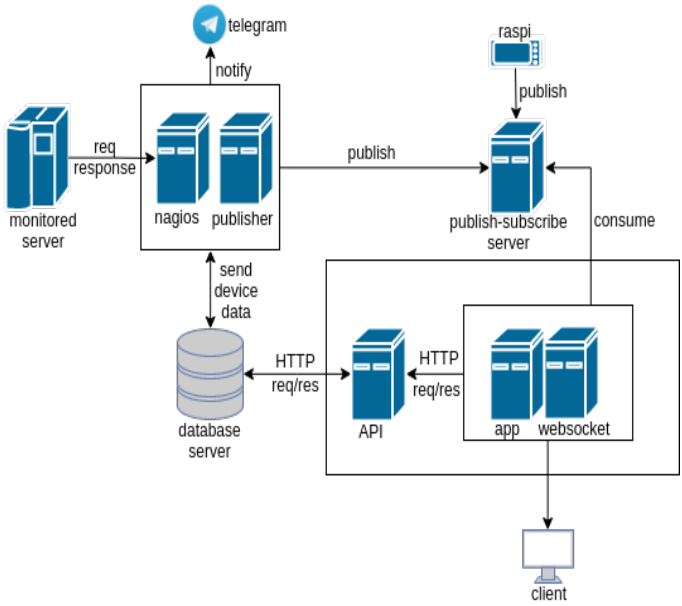
3.2.1 Desain Umum Sistem

Sistem *monitoring* perangkat ini merupakan sebuah sistem yang berfungsi sebagai pemantau *server* dan kondisi lingkungan di DPTSI ITS. Sistem ini menggunakan pola *publish-subscribe* dimana pengguna harus berlangganan (*subscribe*) ke suatu perangkat yang memuat data hasil pemantauan.

Sistem ini terdiri dari 3 (tiga) *server* yaitu *webserver*, *database server*, dan *publish-subscribe server*. Pada *webserver* terdapat aplikasi dan *websocket*. Kemudian terdapat pula REST API yang berfungsi sebagai transaktor data dari *webserver* ke *database server*.

Pengguna yang melakukan fitur selain memantau perangkat akan mengirimkan permintaan (*request*) HTTP ke REST API. REST API kemudian menyalurkan permintaan ke *database server*. Setelah itu REST API akan mengirimkan respon (*response*) ke aplikasi.

Untuk fitur pemantauan server, maka aplikasi akan mengakses *websocket*. *Websocket* ini bertugas untuk mengakses *publish-subscribe server* dimana data yang dihasilkan dari pemantauan oleh Nagios disimpan oleh *publish-subscribe server* ini. Penjelasan dari desain umum arsitektur sistem akan ditampilkan pada Gambar 3.1.



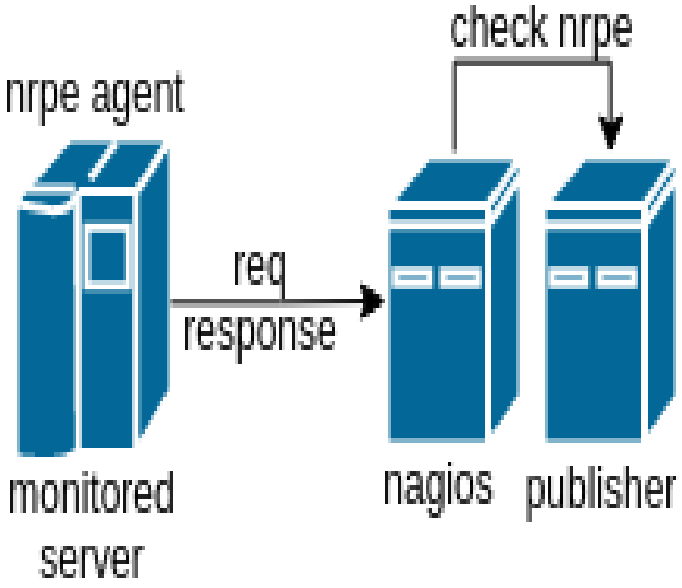
Gambar 3.1: Desain Umum Sistem

3.2.2 Desain *Publisher Server*

Pada *publisher server*, Nagios dipasang untuk memantau keadaan kinerja dari suatu *server*. Kemudian, dipasang pula NRPE untuk mengeksekusi *plugin* Nagios dari jauh. Alasan utama menggunakan NRPE ini adalah memungkinkan Nagios untuk memantau sumber daya ”lokal” (seperti beban CPU, penggunaan memori, dan sebagainya). NRPE ini diaplikasikan dengan check-nrpe.

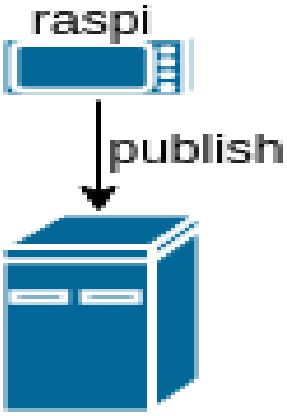
Setiap *server* yang dipantau dimasukkan ke sebuah *thread* baru agar dapat berjalan secara paralel. Kemudian, pada *thread* tersebut setiap *server* terkait akan diperiksa kinerjanya dengan NRPE. Hasil dari pemeriksaan dikirim ke *publish-subscribe server* melalui sebuah *exchange* yang telah diikat dengan sebuah

message queue yang sudah diinisiasi sebelumnya. Desain dari *publisher server* digambarkan pada Gambar 3.2



Gambar 3.2: Desain Publisher Server

Sedangkan, untuk *publisher* pada Raspberry Pi memiliki cara yang hampir sama dengan *publisher* untuk Nagios. Perbedaannya hanya terletak pada cara pengambilan data dari sensor yang kemudian dikirim ke *publisher*. Desain dari *publisher server* untuk Raspberry Pi digambarkan pada Gambar 3.3.



Gambar 3.3: Desain Raspberry Pi dengan *Publisher*

Lalu, pada sistem ini juga dapat mengirimkan pesan jika *server* yang dipantau mengalami *error* ke *subscriber* melalui aplikasi Telegram.

3.2.3 Desain *Publish-Subscribe Server*

Publish-subscribe server adalah sebuah *server* yang berfungsi sebagai wadah untuk menampung pesan dari *publisher* yang mengirimkan data hasil pemantauan oleh Nagios. Pada *server* ini terpasang Rabbitmq sebagai *message broker*. Seluruh pesan yang dikirimkan oleh publisher ditampung ke *publish -subscribe server* melalui sebuah *exchange* yang diikat dengan sebuah *queue*. Setelah itu, *server* akan menyimpan pesan dalam *queue* tersebut hingga ada konsumen, yakni *websocket server*, yang meminta data tersebut untuk dikirimkan.

Secara umum, desain dari *Publish-Subscribe Server* dapat dilihat pada Gambar 3.4.

Gambar 3.4: Desain Publish-Subscribe Server

3.2.4 Desain Webserver

Pada server ini, terdapat *application server* dan *websocket*. *Websocket* digunakan untuk mengambil data yang dikirimkan oleh *publisher*. Istilahnya, *websocket* ini bertindak sebagai konsumen oleh klien. Data yang diterima sifatnya *realtime* yang berarti merupakan data-data terbaru.

Konsumen akan membuat sebuah *queue* dengan nama yang ditentukan oleh klien. Nama dari *queue* tersebut ditentukan dengan membuat *string* UUID versi 4 secara acak. Setelah *queue* berhasil dibuat, konsumen membuat *exchange*. Jumlah *exchange* yang terbuat sama banyaknya dengan jumlah *server* yang terdaftar di sistem. Penamaan *exchange* disesuaikan dengan ID masing-masing *server* yang juga berformat UUID versi 4.

Pembuatan *queue* dan *exchange* ini dilakukan jika *queue* dan *exchange* belum terdaftar pada *publish-subscribe server*. Jika sudah terdaftar, maka tidak dilakukan pembuatan *queue* dan *exchange* lagi. Setelah *queue* dan *exchange* terbuat, maka dilakukan pengikatan (*binding*) *queue* oleh *exchange*. *Queue* dapat diikat oleh satu atau lebih *exchange*.

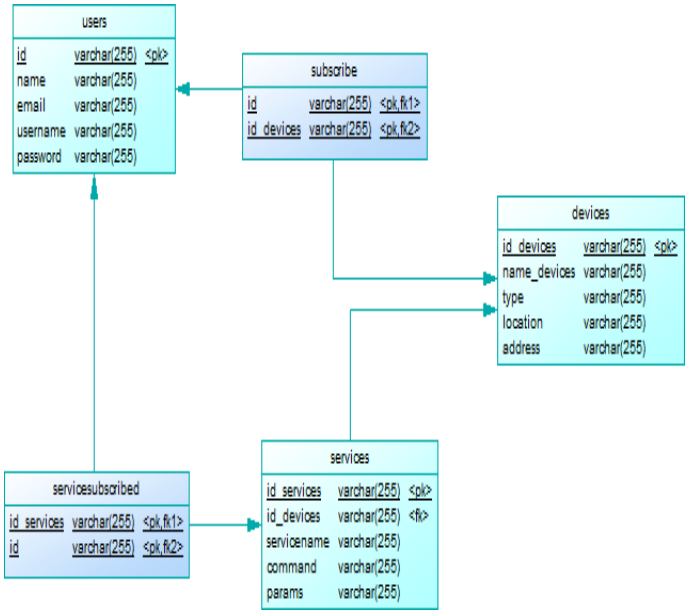
Selain itu, semua fitur selain memantau *server* diaplikasikan di *application server*. Fitur-fitur tersebut adalah:

- memasukkan data perangkat beserta *service*-nya,
- mengubah data perangkat,
- menghapus perangkat,
- melakukan *subscribe* pada perangkat yang dipilih,

- melakukan *unsubscribe*

3.2.5 Desain *Database Server*

Desain *database* pada sistem ini adalah seperti yang digambarkan pada Gambar 3.5 di bawah ini.



Gambar 3.5: Physical Data Model

Terdapat 5 (lima) tabel pada *database* ini. 3 (tiga) tabel di antaranya adalah tabel berentitas kuat dan merupakan tabel utama pada sistem ini, yaitu: users, devices, dan services. 2 (dua) tabel lainnya merupakan tabel hasil dari relasi *many-to-many*, yaitu tabel subscribe dan servicesubscribe.

Tabel subscribe ini digunakan untuk menyimpan data pengguna yang melakukan *subscribe* ke suatu *server*. Dan pengguna juga dapat melakukan *subscribe* ke *service* yang

dimiliki oleh *server*. *Service* ini berguna untuk mengetahui informasi apa saja yang ada pada tiap perangkat. Tiap *service* memiliki *command* yang berbeda.

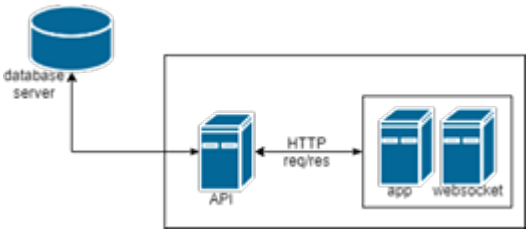
3.2.6 Desain REST API

REST API merupakan implementasi dari API (*Application Programming Interface*). REST (*Representional State Transfer*) sendiri adalah suatu arsitektur metode komunikasi yang menggunakan protokol HTTP untuk pertukaran data. Metode ini sering diterapkan dalam pengembangan aplikasi. Tujuan dari REST API yang diterapkan pada pengembangan aplikasi adalah agar sistem memiliki performa yang cepat namun berjalan dengan baik. Sistem juga menjadi mudah dikembangkan terutama dalam hal pertukaran dan komunikasi data.

URL pada API biasa disebut *endpoint*. Pada sistem ini, terdapat beberapa *endpoint* yang mana masing-masing *endpoint* memiliki *endpoint-endpoint* lagi.

Application server mengirimkan HTTP *request* ke API. Kemudian, API akan melakukan transaksi dengan *database*. Setelah mendapatkan data, maka API akan mengirimkan HTTP *response* ke *application server*.

Desain dari REST API dapat dilihat pada Gambar 3.6.



Gambar 3.6: Desain Umum Sistem

BAB IV

IMPLEMENTASI

Pada bab ini dijelaskan mengenai implementasi *publish-subscribe* pada rancang bangun sistem *monitoring* perangkat di DPTSI ITS.

4.1 Lingkungan Implementasi

Lingkungan implementasi dalam pembuatan Tugas Akhir ini meliputi perangkat keras dan perangkat lunak yang digunakan untuk rancang bangun sistem *monitoring* perangkat di DPTSI ITS dengan implementasi *publish-subscribe* adalah sebagai berikut:

1. Perangkat Keras:
    - *Processor* Intel(R) Core(TM) i5-4210U CPU @1.70GHz 64-bit dengan memori RAM 4 GB.
2. Perangkat Lunak:
    - Sistem operasi Ubuntu 14.04 LTS 64 bit.
    - Rabbitmq, digunakan sebagai broker pada *publish-subscribe*.
    - Flask, digunakan untuk membuat aplikasi sistem.
    - Nagios, digunakan untuk memantau server.
    - MySQL, digunakan untuk membangun *database* pada sistem.
    - Telegram, digunakan sebagai media penerimaan notifikasi dari sistem.
    - *Text editor* Sublime Text 3.
    - TeXstudio, digunakan untuk menyusun buku tugas akhir ini.

4.2 Implementasi *Publisher Server*

Publisher server merupakan *server* yang berfungsi untuk mengambil data pada perangkat (*server*) dan mengirimkannya

menuju *publish-subscribe server*. *Publisher server* menggunakan *plugin* NRPE pada Nagios sebagai tambahan untuk memantau sumber daya ”lokal” (seperti beban CPU, penggunaan memori, dan sebagainya). Untuk itu kita perlu memasang Nagios terlebih dahulu pada *server* agar bisa melakukan pengambilan data pemantauan *server*.

Sedangkan pada Raspberry Pi, *publisher* dapat langsung dipasangkan setelah sebelumnya memasang Adafruit DHT untuk pembacaan data dari sensor.

Setelah data berhasil dikumpulkan, data hasil pemantauan pada tiap *server* dikirimkan menuju *publish-subscribe server* melalui *thread* yang berbeda. Proses ini dinamakan *multithreading*.

Kemudian, pada *publisher* yang terhubung Nagios ditambahkan paket dari Telegram. Fungsinya adalah untuk mengirimkan pesan jika data dari Nagios yang dikirimkan ke *publisher* mengalami *error*. Pengguna yang berlangganan *service server* tersebut akan diberikan pesan melalui aplikasi Telegram.

4.2.1 Instalasi Nagios pada *Server*

Instalasi dilakukan dengan mengunduh Nagios terlebih dahulu pada laman di bawah ini `https://assets.nagios.com/downloads/nagioscore/releases/nagios-4.3.4.tar.gz`. Setelah itu, dilakukan prosedur penginstalan Nagios di *server*.

4.2.2 *Script* NRPE untuk Pengambilan Data dari Nagios

Kode Sumber 4.1 ini menunjukkan *script* dari *[plugin]* NRPE untuk megambil data dari hasil pemantauan oleh Nagios.

```
$ /usr/local/nagios/libexec/check_nrpe –H <
    alamat_server> –c <nama_service> [’
    nama_command’]
```

Kode Sumber 4.1: Perintah Mengumpulkan Data Perangkat dengan NRPE

4.2.3 *Pseudocode* untuk Pengambilan Data Hasil Sensor

Kode Sumber 4.2 ini menunjukkan *pseudocode* untuk pembacaan data hasil dari sensor pada Raspberry Pi.

```
while True
        get humidity, temperature pin
        show data humidity, temperature
```

Kode Sumber 4.2: *Pseudocode* Untuk Mengambil Data Sensor

4.2.4 *Pseudocode* Mengirimkan Pesan *Error* dari *Publisher* ke *Subscriber*

Kode Sumber 4.3 ini menunjukkan *pseudocode* untuk mengirimkan pesan ke *subscriber* melalui aplikasi Telegram jika ada data dari Nagios yang bermasalah.

```
for device is error
        get chat_id from user
        send message
```

Kode Sumber 4.3: *Pseudocode* Untuk Mengambil Data Sensor

4.3 Implementasi *Publish-Subscribe Server*

Pada *publish-subscribe server*, dipasang aplikasi RabbitMQ. RabbitMQ menerima seluruh data yang dikirim oleh *publisher*. Setelah itu, RabbitMQ menyimpan dan menunggu hingga ada

konsumen yang meminta data tersebut pada RabbitMQ . Data yang dikirimkan harus sesuai dengan apa yang diminta oleh konsumen, tidak boleh berbeda dari kriteria yang diminta.

Instalasi RabbitMQ dapat dilihat melalui halaman web resminya, `https://www.rabbitmq.com/install-debian.html`. Langkah instalasi diawali dengan penginstalan erlang, sebuah bahasa pemrograman yang dibutuhkan untuk menjalankan RabbitMQ. Setelah itu barulah dipasang `rabbitmq-server`.

Setelah RabbitMQ *server* terpasang, selanjutnya dilakukan konfigurasi untuk mengaktifkan akses ke RabbitMQ Management Console, web admin milik RabbitMQ. Hal ini dilakukan agar mudah untuk melakukan manajemen data, *user*, dan lain-lain. RabbitMQ sudah menyediakan *plugin* agar web admin dapat langsung digunakan. Hanya dengan menjalankan perintah `sudo rabbitmq-plugins enable rabbitmq_management`, web admin RabbitMQ dapat dijalankan.

4.4 Implementasi *Webserver*

Pada *webserver*, terdapat *application server* dan *websocket. Websocket* berfungsi untuk meminta data dari *publish-subscribe server*. Websocket disambungkan dengan suatu *endpoint* pada *application server*, sehingga ketika klien mengakses *endpoint* pada aplikasi, `javascript` pada halaman tersebut akan menyambungkan ke halaman pada *websocket*.

Websocket pada sistem ini menggunakan node.js, sebuah perangkat lunak yang didesain untuk mengembangkan aplikasi berbasis web. Node.js menggunakan bahasa pemrograman JavaScript.

```
socket.on('disconnect', function() {
```

```
$("#idlogs").text("Disconnect");
$("#idlogs").parent().parent().attr('class', 'block-content block
    -content-full bg-danger');
$('.nrperesult').each(function() {
$(this).html("")
});
});
```

Kode Sumber 4.4: Inisiasi Komunikasi Websocket dengan Klien dan Publish-Subscribe Server Tidak Terkonesi

Kode Sumber 4.4 menunjukkan jika kondisi komunikasi antara *websocket* dengan klien dan *publish-subscribe server* tidak tersambung.

```
socket.on('connect', function() {
$("#idlogs").text("Connected");
$("#idlogs").parent().parent().attr('class', 'block-content block
    -content-full bg-success');
{% if monitor.subscribing %}
var deviceid = [];
$('.device').each(function( index ) {
deviceid.push($( this ).attr('id'));
});
socket.emit('startRabbit', {'data': deviceid, 'id': uuid4() });
{% endif %}
});
```

Kode Sumber 4.5: Inisiasi Komunikasi Websocket dengan Klien dan Publish-Subscribe Server Terkoneksi

Kode Sumber 4.5 menunjukkan jika kondisi komunikasi antara *websocket* dengan klien dan *publish-subscribe server* telah tersambung. Kemudian, klien mengirimkan ID dari perangkat (*server*) yang telah dipilih oleh pengguna untuk menjadikannya nama *exchange*. UUID versi 4 dibuat untuk menamai *queue* yang baru.

Jika koneksi sudah tersambung dan ID perangkat untuk penamaan *exchange* serta UUID versi 4 untuk penamaan *queue* sudah dikirim oleh klien, maka selanjutnya *server* akan memproses pembuatan *exchange* dan *queue* baru. Kode

websocket server saat pembuatan *exchange* dan *queue* lalu menyalurkan data pada klien dapat dilihat pada Kode Sumber 4.6

```
rabbitMqConnection.createChannel().then(function(ch) {
consumerChannel = ch
var q;
function allDone(notAborted, arr) {
var okok = ch.prefetch(1)
okok = okok.then(() => {
return ch.consume(q, logMessage, {noAck: true})
.then(() => {
console.log(' [*] Waiting for '+msg['data']+'.');
})
})
}
forEach(msg['data'], function (item, index, arr) {
var done = this.async()
var ok = ch.assertExchange(item, 'fanout', {durable: false});
ok = ok.then(function() {
return ch.assertQueue(msg['id'], {exclusive: false});
});
ok = ok.then(function(qok) {
return ch.bindQueue(qok.queue, item, '').then(function() {
q = qok.queue
return qok.queue;
});
});
ok = ok.then(function() {
done()
})
}, allDone)
```

Kode Sumber 4.6: Pembuatan Exchange dan Queue Baru pada Websocket

4.5 Implementasi REST API

Fungsi utama dari REST API untuk sistem ini adalah untuk menyimpan data pengguna yang melakukan *subscribe* pada perangkat beserta *service*-nya. REST API dibuat dengan menggunakan *framework* Flask dan ORM Database peewee.

REST API menggunakan protokol HTTP dalam pengaksesannya. Sehingga, terdapat URL API yang dapat

disebut sebagai *endpoint*. Tabel 4.1 akan menguraikan ada *endpoint* apa saja pada REST API di sistem ini.

Tabel 4.1: Daftar Endpoint pada REST API

No	***Endpoint***	**Metode**	**Aksi**
1	/register	POST	Membuat data baru pada tabel user di database
2	/login	POST	Mengambil data pada tabel user dan mencocokkannya dengan JSON yang dikirimkan lewat *body*. setelah data username dan password cocok, lalu dibuatkan sebuah token JWT.
3	/logout	POST	Memasukkan token JWT yang terdaftar pada server kedalam daftar hitam agar token tidak dapat digunakan lagi.

Tabel 4.1: Daftar Endpoint pada REST API

No	*Endpoint*	Metode	Aksi
4	/users	GET	Menampilkan seluruh data user yang terdaftar pada sistem
5	/users/<string:username>	GET	Menampilkan data user berdasarkan username yang tertulis pada URL
6	/devices/create	POST	Membuat data baru pada tabel devices di database
7	/devices/edit/<string:id>	PUT	Mengubah data pada tabel devices di database yang ID nya sama dengan ID yang ada pada URL.
8	/devices/delete	DELETE	Menghapus data pada tabel devices di database yang ID nya tertulis pada *body* yang bertipe JSON.

Tabel 4.1: Daftar Endpoint pada REST API

No	*Endpoint*	Metode	Aksi
9	/devices	GET	Menampilkan seluruh data perangkat yang terdaftar pada sistem
10	/devices/<string:id>	GET	Menampilkan data user berdasarkan username yang tertulis pada URL
11	/service/create	POST	Membuat data baru pada tabel service
12	/service/edit	POST	Mengubah data pada tabel service di database yang ID nya tertulis pada *body* yang bertipe JSON.
13	/service/delete	POST	Menghapus data pada tabel service di database yang ID nya tertulis pada *body* yang bertipe JSON.

Tabel 4.1: Daftar Endpoint pada REST API

No	***Endpoint***	**Metode**	**Aksi**
14	/subscribe/devices	POST	Membuat data baru pada tabel subscribe
15	/unsubscribe/devices	POST	Menghapus data pada tabel subscribe di database yang ID nya tertulis pada *body* yang bertipe JSON.
16	/subscribe/service	POST	Membuat data baru pada tabel servicesubscribed
17	/unsubscribe/service	POST	Menghapus data pada tabel servicesubscribed di database yang ID nya tertulis pada *body* yang bertipe JSON.

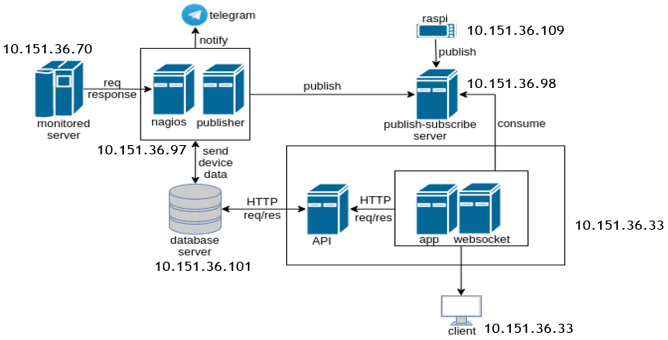
BAB V

PENGUJIAN DAN EVALUASI

Pada bab ini akan dibahas uji coba dan evaluasi dari sistem yang telah dibuat. Sistem akan diuji coba fungsionalitas dan performanya dengan menjalankan skenario uji coba yang sudah ditentukan. Uji coba dilakukan untuk mengetahui hasil dari sistem ini sehingga dapat menjawab rumusan masalah pada tugas akhir ini.

5.1 Lingkungan Uji Coba

Lingkungan pengujian menggunakan komponen-komponen yang terdiri dari: satu *server publisher*, satu *server publish-subscribe*, satu *server* aplikasi dan API, satu *server database*, dan satu komputer penguji. Pengujian dilakukan di Laboratoriom Arsitektur dan Jaringan Komputer Departemen Informatika ITS. Denah lingkungan uji coba dapat dilihat pada Gambar 5.1.



Gambar 5.1: Denah Uji Coba

Spesifikasi untuk setiap komponen yang digunakan ditunjukkan pada Tabel 5.2 untuk *publisher server*, Tabel 5.3 untuk *publish-subscribe server*, Tabel 5.4 untuk *web server* dan

API, Tabel 5.5 untuk *database server*, dan Tabel 5.6 untuk komputer penguji.

1. ***Publisher Server (Nagios)***

Tabel 5.1: *Server* Untuk *Publisher*

Perangkat Keras	Processor Intel(R) Core(TM) i5-2120 CPU @ 3.30GHz
	RAM 8GB
	Hard disk 500GB
Perangkat Lunak	Ubuntu 16.04 64 bit
	Nagios
Konfigurasi Jaringan	IP address : 10.151.36.97
	Netmask : 255.255.255.0
	Gateway : 10.151.36.1

2. ***Publisher Server (Raspberry Pi)***

Tabel 5.2: *Server Raspberry Pi* Untuk *Publisher*

Perangkat Keras	Raspberry Pi 2 QUAD Core Broadcom BCM2836 CPU
	RAM 1GB
Perangkat Lunak	Ubuntu Mate 64 bit
Konfigurasi Jaringan	IP address : 10.151.36.109
	Netmask : 255.255.255.0
	Gateway : 10.151.36.1

3. ***Publish-Subscribe Server***

Tabel 5.3: *Server* Untuk *Publish-Subscribe*

Perangkat Keras	Processor Intel(R) Core(TM) i5-2120 CPU @ 3.30GHz

	RAM 8GB
	Hard disk 500GB
Perangkat Lunak	Ubuntu 16.04 64 bit
	RabbitMQ
Konfigurasi Jaringan	IP address : 10.151.36.98
	Netmask : 255.255.255.0
	Gateway : 10.151.36.1

4. ***Web Server* dan API**

Tabel 5.4: *Server* Untuk Aplikasi dan API

Perangkat Keras	Processor Intel(R) Core(TM) i5-2120 CPU @ 3.30GHz
	RAM 8GB
	Hard disk 500GB
Perangkat Lunak	Ubuntu 14.04 64 bit
	RabbitMQ
	Flask
	NodeJS
Konfigurasi Jaringan	IP address : 10.151.36.33
	Netmask : 255.255.255.0
	Gateway : 10.151.36.1

5. ***Database Server***

Tabel 5.5: *Server* Untuk *Database*

Perangkat Keras	Processor Intel(R) Core(TM) i5-2120 CPU @ 3.30GHz
	RAM 8GB
	Hard disk 500GB
Perangkat Lunak	Ubuntu 16.04 64 bit
	MySQL Server

Konfigurasi Jaringan	IP address : 10.151.36.101
	Netmask : 255.255.255.0
	Gateway : 10.151.36.1

6. **Komputer Penguji**

Tabel 5.6: Komputer Penguji

Perangkat Keras	Processor Intel(R) Core(TM) i5-2120 CPU @ 3.30GHz
	RAM 8GB
	Hard disk 500GB
Perangkat Lunak	Ubuntu 14.04 64 bit
	Google Chrome
Konfigurasi Jaringan	IP address : 10.151.36.33
	Netmask : 255.255.255.0
	Gateway : 10.151.36.1

5.2 Skenario Uji Coba

Uji coba akan dilakukan untuk mengetahui keberhasilan sistem yang telah dibangun. Skenario pengujian dibedakan menjadi 2 bagian, yaitu:

- **Uji Fungsionalitas**
  Pengujian ini didasarkan pada fungsionalitas yang disajikan sistem.
- **Uji Performa**
  Pengujian ini untuk menguji ketahanan sistem terhadap sejumlah permintaan ke aplikasi secara bersamaan.

5.2.1 Skenario Uji Coba Fungsionalitas

Uji coba fungsionalitas dilakukan dengan cara menjalankan sistem yang telah dibangun dan melakukan pengujian terhadap fitur yang telah dibuat. Uji coba fungsionalitas akan berfungsi untuk memastikan sistem sudah memenuhi kebutuhan.

5.2.1.1 Uji Pengguna Dapat Melihat Data Perangkat

Pengujian ini dilakukan untuk mengetahui apakah pengguna dapat melihat data perangkat yang telah terdaftar di sistem.

Pengujian dilakukan setelah pengguna melakukan *login*. Uji fungsionalitas pengguna dapat melihat data perangkat yang telah terdaftar di sistem dijelaskan pada Tabel 5.7.

Tabel 5.7: Skenario Uji Pengguna Dapat Melihat Data Perangkat

No	Uji Coba	Hasil Harapan
1	Pengguna menekan tombol menu ’Device Management’.	Pengguna dapat melihat perangkat apa saja yang terdaftar di sistem.

5.2.1.2 Uji Pengguna Dapat Melihat Detail Data Perangkat

Pengujian ini dilakukan untuk mengetahui apakah pengguna dapat melihat rincian data dari suatu perangkat.

Uji fungsionalitas pengguna dapat melihat detail data perangkat yang telah terdaftar di sistem dijelaskan pada Tabel 5.8.

Tabel 5.8: Skenario Uji Pengguna Dapat Melihat Detail Data Perangkat

No	Uji Coba	Hasil Harapan
1	Pengguna menekan tombol menu 'Info' yang ada pada tiap perangkat.	Pengguna dapat melihat detail data dari suatu perangkat.

5.2.1.3 Uji Pengguna Dapat Mengubah Data Perangkat

Pengujian ini dilakukan untuk mengetahui apakah pengguna dapat mengubah data dari suatu perangkat.

Uji fungsionalitas pengguna dapat mengubah data perangkat dijelaskan pada Tabel 5.9

Tabel 5.9: Skenario Uji Pengguna Dapat Mengubah Data Perangkat

No	Uji Coba	Hasil Harapan
1	Pengguna menekan tombol menu 'Ubah' yang ada pada tiap perangkat.	Sistem menampilkan *form* yang sudah berisi detail data dari perangkat.
2	Pengguna mengubah data perangkat sesuai kebutuhan.	Sistem menampilkan data yang sudah diubah oleh pengguna sebelumnya menggantikan data yang lama.

5.2.1.4 Uji Pengguna Dapat Menghapus Data Perangkat

Pengujian ini dilakukan untuk mengetahui apakah pengguna dapat menghapus data perangkat dari sistem.

Uji fungsionalitas pengguna dapat menghapus data perangkat dari sistem dijelaskan pada Tabel 5.10.

Tabel 5.10: Skenario Uji Pengguna Dapat Menghapus Data Perangkat

No	Uji Coba	Hasil Harapan
1	Pengguna menekan tombol menu ’Hapus’ yang tersedia pada tiap perangkat.	Data perangkat berhasil dihapus.

5.2.1.5 Uji Pengguna Dapat Melakukan *Subscribe* pada Perangkat

Pengujian ini dilakukan untuk mengetahui apakah pengguna dapat melakukan *subscribe* pada perangkat yang dipilih.

Uji fungsionalitas pengguna dapat melakukan *subscribe* pada perangkat yang dipilih dijelaskan pada Tabel 5.11.

Tabel 5.11: Skenario Uji Pengguna Dapat Melakukan *Subscribe* pada Perangkat

No	Uji Coba	Hasil Harapan
1	Pengguna menekan tombol ’Subscribe’ yang tersedia pada tiap perangkat.	Pengguna berhasil melakukan *subscribe* pada perangkat yang telah dipilih ditandai dengan berubahnya tombol ’Subscribe’ menjadi ’Unsubscribe’.

5.2.1.6 Uji Pengguna Dapat Menghentikan *Subscribe* (*Unsubscribe*) pada Perangkat

Pengujian ini dilakukan untuk mengetahui apakah pengguna dapat berhenti langganan dari perangkat yang dipilih sebelumnya.

Uji fungsionalitas pengguna dapat melakukan *unsubscribe* pada perangkat yang dipilih sebelumnya dijelaskan pada Tabel

5.12.

Tabel 5.12: Skenario Uji Pengguna Dapat Menghentikan *Subscribe* (*Unsubscribe*) pada Perangkat

No	Uji Coba	Hasil Harapan
1	Pengguna menekan tombol 'Unsubscribe' yang tersedia pada tiap perangkat yang sebelumnya di-*subscribe*.	Pengguna berhasil melakukan *unsubscribe* pada perangkat yang ditandai dengan berubahnya tombol 'Unsubscribe' menjadi 'Subscribe' kembali.

5.2.1.7 Uji Pengguna Dapat Melakukan *Monitoring* atau Pemantauan pada Perangkat yang Telah Di-*Subscribe*

Pengujian ini dilakukan untuk mengetahui apakah pengguna dapat memantau perangkat yang di-*subscribe* sebelumnya.

Uji fungsionalitas pengguna dapat melakukan *monitoring* atau pemantauan pada perangkat yang di-*subscribe* sebelumnya dijelaskan pada Tabel 5.13.

Tabel 5.13: Skenario Uji Pengguna Dapat Melakukan *Monitoring* atau Pemantauan pada Perangkat yang Telah Di-*Subscribe*

No	Uji Coba	Hasil Harapan
1	Pengguna menekan tombol menu 'Monitor' yang tersedia di sistem.	Pengguna dapat melihat kondisi perangkat yang di-*subscribe*.

5.2.2 Skenario Uji Coba Performa

Uji performa dilakukan dengan dua skenario, yaitu uji performa REST API dan uji performa *publish-subscribe*. Uji coba REST API dilakukan dengan menggunakan satu buah komputer untuk melakukan akses secara bersamaan ke REST API menggunakan bantuan aplikasi JMeter, sebuah aplikasi pengujian untuk menganalisa dan menghitung performa dari suatu servis.

Sedangkan untuk uji performa *publish-subscribe* dilakukan dengan cara menghitung waktu pengiriman data. Data dikirim oleh publisher yang berada pada server dengan alamat IP 10.151.36.97 menuju konsumen yang berada pada alamat IP 10.151.36.33.

5.2.2.1 Uji Performa REST API

Pengujian dilakukan menggunakan bantuan aplikasi JMeter untuk mengukur jumlah waktu yang diperlukan oleh REST API untuk menyelesaikan *request* dari komputer penguji.

5.2.2.2 Uji Performa *Publish-Subscribe*

Pengujian dilakukan dengan mengukur jumlah waktu yang diperlukan *publisher* dalam mengirim data dan diterima oleh *subscriber*.

5.3 Hasil Uji Coba dan Evaluasi

Berikut dijelaskan hasil uji coba dan evaluasi berdasarkan skenario yang telah dijelaskan pada subbab 5.2.

5.3.1 Hasil Uji Fungsionalitas

Berikut dijelaskan hasil pengujian fungsionalitas pada sistem yang dibangun.

5.3.1.1 Uji Pengguna Dapat Melihat Data Perangkat

Pengujian dilakukan sesuai dengan skenario yang dijelaskan pada subbab 5.2.1.1 dan pada Tabel 5.7. Hasil pengujian pengguna melihat data perangkat yang telah terdaftar di sistem dapat dilihat pada Tabel 5.14

Tabel 5.14: Hasil Uji Coba *Client* dapat Mengakses Internet

No	Uji Coba	Hasil
1	Pengguna menekan tombol menu 'Device Management'.	Berhasil

Sesuai dengan skenario uji coba yang diberikan pada Tabel 5.7, hasil uji coba menunjukkan skenario berhasil dilakukan dan sistem menampilkan daftar perangkat yang ada di sistem.

5.3.1.2 Uji Pengguna Dapat Melihat Detail Data Perangkat

Pengujian dilakukan sesuai dengan skenario yang dijelaskan pada subbab 5.2.1.2 dan pada Tabel 5.8. Hasil pengujian pengguna melihat detail dari data perangkat yang telah terdaftar di sistem dapat dilihat pada Tabel 5.15

Tabel 5.15: Hasil Uji Pengguna Melihat Detail Data Perangkat

No	Uji Coba	Hasil
1	Pengguna menekan tombol 'Info' yang ada pada setiap perangkat.	Berhasil

Sesuai dengan skenario uji coba yang diberikan pada Tabel 5.8, hasil uji coba menunjukkan skenario berhasil dilakukan dan sistem menunjukkan data perangkat secara rinci.

5.3.1.3 Uji Pengguna Dapat Mengubah Data Perangkat

Pengujian dilakukan sesuai dengan skenario yang dijelaskan pada subbab 5.2.1.3 dan pada Tabel 5.9. Hasil pengujian pengguna mengubah data perangkat dapat dilihat pada Tabel 5.16

Tabel 5.16: Hasil Uji Pengguna Mengubah Data Perangkat

No	Uji Coba	Hasil
1	Pengguna menekan tombol 'Ubah' yang ada pada setiap perangkat.	Berhasil
2	Pengguna mengubah data perangkat sesuai kebutuhan.	Berhasil

Sesuai dengan skenario uji coba yang diberikan pada Tabel 5.9, hasil uji coba menunjukkan skenario berhasil dilakukan dan data terubah.

5.3.1.4 Uji Pengguna Dapat Menghapus Data Perangkat

Pengujian dilakukan sesuai dengan skenario yang dijelaskan pada subbab 5.2.1.4 dan pada Tabel 5.10. Hasil pengujian pengguna menghapus data perangkat dapat dilihat pada Tabel 5.17

Tabel 5.17: Hasil Uji Pengguna Menghapus Data Perangkat

No	Uji Coba	Hasil
1	Pengguna menekan tombol 'Hapus' yang tersedia pada setiap perangkat.	Berhasil

Sesuai dengan skenario uji coba yang diberikan pada Tabel 5.10, hasil uji coba menunjukkan skenario berhasil dilakukan dan data terhapus.

5.3.1.5 Uji Pengguna Dapat Melakukan *Subscribe* pada Perangkat

Pengujian dilakukan sesuai dengan skenario yang dijelaskan pada subbab 5.2.1.5 dan pada Tabel 5.11. Hasil pengujian pengguna melakukan *subscribe* pada perangkat yang diinginkan dapat dilihat pada Tabel 5.18

Tabel 5.18: Hasil Uji Pengguna Dapat Melakukan *Subscribe* pada Perangkat

No	Uji Coba	Hasil
1	Pengguna menekan tombol 'Subscribe' yang tersedia pada setiap perangkat.	Berhasil

Sesuai dengan skenario uji coba yang diberikan pada Tabel 5.11, hasil uji coba menunjukkan skenario berhasil dilakukan dan perangkat berhasil di-*subscribe*.

5.3.1.6 Uji Pengguna Dapat Menghentikan (*Unsubscribe*) pada Perangkat

Pengujian dilakukan sesuai dengan skenario yang dijelaskan pada subbab 5.2.1.6 dan pada Tabel 5.12. Hasil pengujian pengguna melakukan penghentian berlangganan *unsubscribe* pada perangkat yang sebelumnya telah di-*subscribe* dapat dilihat pada Tabel 5.19

Tabel 5.19: Hasil Uji Pengguna Dapat Menghentikan (*Unsubscribe*) pada Perangkat

No	Uji Coba	Hasil
1	Pengguna menekan tombol 'Unsubscribe' yang tersedia pada setiap perangkat yang sebelumnya telah di-*subscribe*.	Berhasil

Sesuai dengan skenario uji coba yang diberikan pada Tabel 5.12, hasil uji coba menunjukkan skenario berhasil dilakukan dan perangkat berhasil di-*unsubscribe*.

5.3.1.7 Uji Pengguna Dapat Melakukan *Monitoring* atau Pemantauan pada Perangkat yang Telah Di-*Subscribe*

Pengujian dilakukan sesuai dengan skenario yang dijelaskan pada subbab 5.2.1.7 dan pada Tabel 5.13. Hasil pengujian pengguna melakukan pemantauan pada perangkat yang telah di-*subscribe* dapat dilihat pada Tabel 5.20

Tabel 5.20: Hasil Uji Pengguna Dapat Melakukan *Monitoring* atau Pemantauan pada Perangkat yang Telah Di-*Subscribe*

No	Uji Coba	Hasil
1	Pengguna menekan tombol menu 'Monitor' yang tersedia pada sistem.	Berhasil

Sesuai dengan skenario uji coba yang diberikan pada Tabel 5.13, hasil uji coba menunjukkan skenario berhasil dilakukan.

5.3.2 Hasil Uji Performa

Seperti yang sudah dijelaskan pada subbab 5.2 pengujian performa dilakukan dengan menggunakan sebuah komputer yang

berperan sebagai klien untuk melakukan *monitoring* terhadap perangkat (*server*).

Pengujian performa REST API dilakukan dengan menghitung respon waktu dari sistem dan persentase keberhasilan sistem dalam menangani sejumlah permintaan dalam waktu yang bersamaan. Pengujian dilakukan dengan bantuan JMeter. Pada JMeter dilakukan skenario mulai dari login, membuat data perangkat baru, mengubah data perangkat yang sudah ada, dan menghapus data perangkat. Skenario dilakukan sebanyak 3 (tiga) kali perulangan dengan menjalankan beberapa *thread* secara bersamaan.

Untuk pengujian performa pada *publish-subscribe*, dilakukan perhitungan pada respon waktu yang dibutuhkan untuk mengirimkan data dari *publisher* ke *subscriber*. Caranya adalah dengan menghitung selisih waktu dikirimkannya data dari *publisher* dengan waktu sampainya data di *subscriber*.

5.3.2.1 Hasil Uji Coba Performa pada REST API

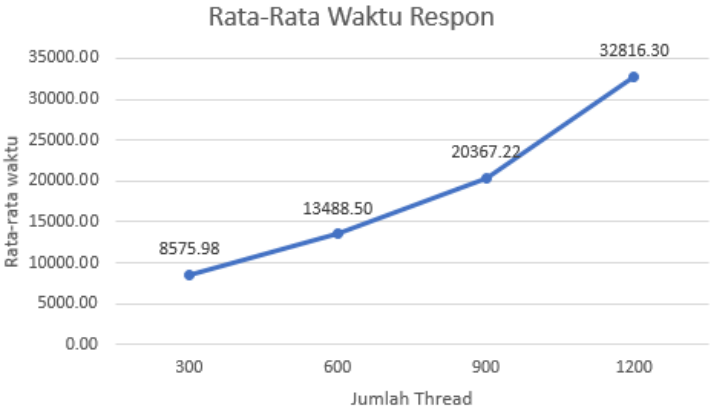
Hasil pengujian performa REST API pada rata-rata waktu respon dalam menjalankan sejumlah permintaan pada waktu yang bersamaan dapat dilihat pada Tabel 5.21 dan direpresentasikan pada grafik yang dapat dilihat pada Gambar 5.2

Tabel 5.21: Hasil Uji Coba Mengetahui Rata-Rata Waktu Respon REST API

Thread	**Jumlah *Request***	**Rata-Rata Waktu Respon (ms)**
300	4500	8575.98
600	9000	13488.50
900	13500	20367.22

Tabel 5.21: Hasil Uji Coba Mengetahui Rata-Rata Waktu Respon REST API

Thread		**Jumlah *Request* Rata-Rata Waktu Respon (ms)**
1.200	18000	32816.30

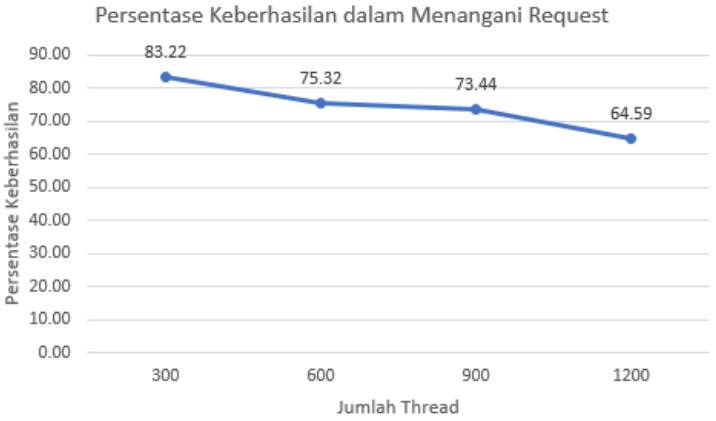


Gambar 5.2: Grafik Waktu Respon pada REST API

Hasil pengujian performa REST API dalam menangani sejumlah permintaan dalam waktu yang bersamaan dapat dilihat pada Tabel 5.22 dan direpresentasikan pada grafik yang dapat dilihat pada Gambar 5.3

Tabel 5.22: Hasil Uji Coba Performa REST API dalam Penanganan *Request*

Thread	**Jumlah *Request***	***Request* Berhasil**	**Persentase Keberhasilan**
300	4500	3745	83,22%
600	9000	6779	75,32%
900	13500	9915	73,44%
1200	18000	11626	64,59%



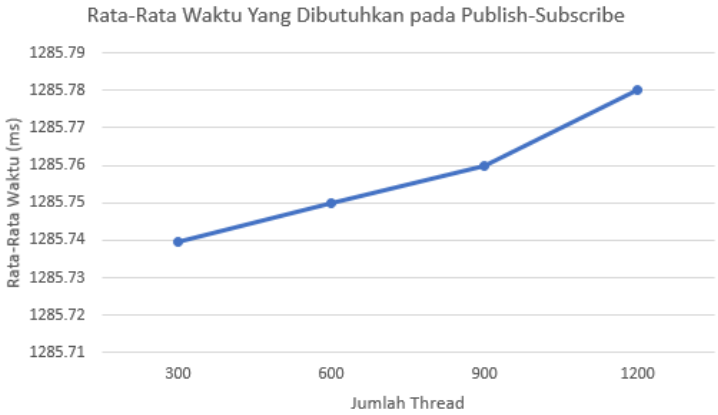
Gambar 5.3: Grafik Persentase Keberhasilan Penanganan *Thread*

5.3.2.2 Hasil Uji Coba Performa pada *Publish-Subscribe*

Hasil pengujian performa pada *publish-subscribe* dilakukan dengan mencari rata-rata waktu respon yang dibutuhkan setiap *publisher* mengirim data hingga diterima oleh *subscriber*. Hasil pengujian dapat dilihat pada Tabel 5.23 dan direpresentasikan pada grafik yang dapat dilihat pada Gambar 5.4

Tabel 5.23: Hasil Uji Performa pada *Publish-Subscribe*

Thread	**Rata-Rata Waktu Respon (ms)**
300	1285.74
600	1285.75
900	1285.76
1.200	1285.78



Gambar 5.4: Grafik Waktu Respon *Publish-Subscribe Request*

5.3.3 Evaluasi

Dari data hasil uji coba yang didapat, dapat disimpulkan bahwa semakin banyak *request* yang ditangani dalam waktu yang bersamaan, maka semakin lama juga waktu yang dibutuhkan untuk menangani sebuah *request*. Begitu pula dengan keberhasilan dalam menangani *request*. Keberhasilan dalam menangani *request* dipengaruhi oleh banyaknya jumlah *request* yang dilakukan secara bersamaan.

Sedangkan untuk *publish-subscribe* sendiri tidak mengalami perbedaan waktu yang signifikan jika jumlah thread semakin banyak. Hal ini menunjukkan bahwa jumlah *thread* tidak terlalu berpengaruh pada kinerja *publish-subscribe* sendiri.

Lalu, alasan utama mengapa menggunakan arsitektur *publish-subscribe* adalah agar administrator tidak harus memantau semua perangkat (*server* atau Raspberry Pi) dan servis, namun dapat memilih perangkat mana dan servis apa yang ingin dipantau. Dibandingkan dengan cara konvensional, yakni membuka Nagios satu per satu secara manual, akan lebih mudah jika melihat data langsung dari halaman pemantauan.

BAB VI

PENUTUP

Bab ini membahas kesimpulan yang dapat diambil dari tujuan pembuatan sistem dan hubungannya dengan hasil uji coba dan evaluasi yang telah dilakukan. Selain itu, terdapat beberapa saran yang bisa dijadikan acuan untuk melakukan pengembangan dan penelitian lebih lanjut.

6.1 Kesimpulan

Dari proses perancangan, implementasi dan pengujian terhadap sistem, dapat diambil beberapa kesimpulan berikut:

1. Sistem *monitoring* untuk pemantauan *server* berhasil dibuat dan dijalankan.
2. *Publish-subscribe* yang digunakan pada sistem diimplementasikan dengan meletakkan *publisher* pada Nagios *server*, yakni Nagios yang berfungsi untuk melakukan pemantauan, dan membuat *publish-subscribe server* yang terintegrasi dengan Rabbitmq.
3. Sistem ini dipasangi *publisher* di Nagios *server* yang kemudian terhubung dengan *publish-subscribe server*. Kemudian, klien meminta informasi (*consume*) lewat aplikasi yang terhubung dengan *websocket* ke *publish-subscribe server*.
4. Sistem dapat mengambil informasi terkait pada *server* menggunakan protokol check_nrpe yang sudah diatur pada Nagios *server* maupun yang dipantau.
5. Sistem berhasil mengimplementasikan *publish-subscribe* pada Raspberry Pi guna memantau lingkungan DPTSI ITS.
6. Sistem berhasil mengintegrasikan antara *publisher agent* ke Telegram untuk mengirimkan pesan saat *server* sedang bermasalah.
7. Hasil dari uji coba fungsionalitas menunjukkan bahwa aplikasi dapat berjalan dengan baik ditandai dengan

keberhasilan uji coba semua *endpoint*.

8. Hasil dari uji coba performa REST API yang dilakukan dengan menggunakan JMeter menunjukkan bahwa kecepatan menangani *request* dipengaruhi oleh banyaknya jumlah *request* yang dilakukan secara bersamaan. Hal ini dibuktikan dengan hasil uji performa REST API dimana *request* yang harus ditangani berjumlah 300, 600, 900, dan 1200 klien dalam waktu yang bersamaan. Setelah dilakukan pengujian, data dari hasil uji dihitung untuk mendapatkan rata-rata waktu REST API dalam menangani request. Berikut ini adalah hasil dari uji coba performa waktu penanganan *request* oleh REST API:
    - pada 300 *thread* yang ditangani, total ada 4500 *request* dan rata-rata waktu yang dibutuhkan untuk menangani *request* adalah 8578,98 ms.
    - pada 600 *thread* yang ditangani, total ada 9000 *request* dan rata-rata waktu yang dibutuhkan untuk menangani *request* adalah 13488,50 ms.
    - pada 900 *thread* yang ditangani, total ada 13500 *request* dan rata-rata waktu yang dibutuhkan untuk menangani *request* adalah 20367,22 ms.
    - pada 1200 *thread* yang ditangani, total ada 18000 *request* dan rata-rata waktu yang dibutuhkan untuk menangani *request* adalah 32816,30 ms.

    Hal ini menunjukkan bahwa semakin banyak *request* yang ditangani dalam waktu yang bersamaan, maka semakin lama juga waktu yang dibutuhkan untuk menangani sebuah *request*.
9. Hasil dari uji coba performa REST API yang dilakukan dengan menggunakan JMeter menunjukkan bahwa keberhasilan dalam menangani *request* dipengaruhi oleh banyaknya jumlah *request* yang dilakukan secara bersamaan. Hal ini dibuktikan dengan hasil uji performa

REST API dimana *request* yang harus ditangani berjumlah 300, 600, 900, dan 1200 klien dalam waktu yang bersamaan. Setelah dilakukan pengujian, data dari hasil uji dihitung untuk mendapatkan jumlah *request* yang berhasil ditangani. Berikut ini adalah hasil dari uji coba performa keberhasilan dalam menangani *request* oleh REST API:

- pada 300 *thread* yang ditangani, total ada 4500 *request* dan persentase keberhasilannya mencapai 83.22
- pada 600 *thread* yang ditangani, total ada 9000 *request* dan persentase keberhasilannya mencapai 75.32
- pada 900 *thread* yang ditangani, total ada 13500 *request* dan persentase keberhasilannya mencapai 73.44
- pada 1200 *thread* yang ditangani, total ada 18000 *request* dan persentase keberhasilannya mencapai 64.59

6.2 Saran

Berikut beberapa saran yang diberikan untuk pengembangan lebih lanjut:

1. Sistem dapat dikembangkan dengan menggunakan *server* yang memiliki performa yang lebih baik lagi agar sistem dapat berjalan dengan lebih optimal.
2. Sistem dapat menyimpan hasil pemantauan ke *database* dengan desain *database* yang lebih efisien, tidak hanya disimpan pada *log*. Gunanya adalah agar bisa melakukan kilas balik pada data hasil pemantauan.

(Halaman ini sengaja dikosongkan)

DAFTAR PUSTAKA

[1] A. Corsaro., L. Querzoni., S. Scipioni., S. T. Piergiovanni., dan A. Virgillito, "Quality of Service in Publish/Subscribe Middleware," 2016.

[2] P. T. Eugster, P. A. Felber, Guerraoui., Rachid, Kermarrec., dan Anne-Marie, "The Many Faces of Publish/Subscribe," Jun. 2013, hal. 114–131.

[3] F. I. dan M. A, "Web Socket Protocol," Dec. 2011.

[4] "Rabbitmq," 2 Januari 2018. [Daring]. Tersedia pada: https://www.rabbitmq.com/features.html. [Diakses: 2 Januari 2018].

[5] "Nagios," 2 Januari 2018. [Daring]. Tersedia pada: https://www.nagios.org/. [Diakses: 2 Januari 2018].

[6] "Nrpe Documentation," 4 Februari 2018. [Daring]. Tersedia pada: https://assets.nagios.com/downloads/nagioscore/docs/nrpe/NRPE.pdf. [Diakses: 4 Februari 2018].

[7] E. N. Wijatsongko., A. E. Putra., dan B. N. Prastowo, "Sistem Pemantauan Ruangan dengan Server Raspberry Pi," Jul. 2015, hal. 65–6.

[8] Matthews., S. J, R. W. Blaine., dan A. F. Brantley, "Evaluating Single Board Computer Clusters for Cyber Operations," Mar. 2016.

[9] "Telegram," 4 Januari 2018. [Daring]. Tersedia pada: http://telegram.org/. [Diakses: 4 Januari 2018].

(Halaman ini sengaja dikosongkan)

LAMPIRAN A

INSTALASI PERANGKAT LUNAK

Instalasi Pustaka Python

Dalam pengembangan sistem ini, digunakan berbagai pustaka pendukung. Pustaka pendukung yang digunakan merupakan pustaka untuk bahasa pemrograman Python. Berikut adalah daftar pustaka yang digunakan dan cara pemasangannya:

- Python Pip
  $ `sudo apt-get install python-pip`
- Python Dev
  $ `sudo apt-get install python-dev`
- Setuptools
  $ `sudo apt-get install python-setuptools`
- MySQLd
  $ `sudo apt-get install python-mysqldb`
- peewee
  $ `pip install peewee`

Pemasangan kerangka kerja Flask

Dalam pengembangan sistem ini, digunakan Flask karena Flask merupakan kerangka kerja yang menggunakan bahasa pemrograman Python. Untuk memasang Flask, jalankan perintah `sudo pip install flask`.

Pemasangan perangkat lunak RabbitMQ

Dalam pengembangan sistem ini, digunakan RabbitMQ sebagai *message broker* dalam menjalankan *publish-subscribe*. Untuk memasang RabbitMQ, jalankan perintah:

- Install Erlang
  $ `wget https://packages.erlang-solutions.com/erlang-solutions_1.0_all.deb`
  $ `sudo dpkg -i erlang-solutions_1.0_all.deb`

```
$ sudo apt-get update
$ sudo apt-get install erlang erlang-nox
```

- Install RabbitMQ-Server

```
$   echo 'deb http://www.rabbitmq.com/debian/
testing main' | sudo tee
/etc/apt/sources.list.d/rabbitmq.list
$ wget -O- https://www.rabbitmq.com/rabbitmq-
release-signing-key.asc | sudo apt-key add -
$ sudo apt-get update
$ sudo apt-get install rabbitmq-server
```

- Manage RabbitMQ Service `$ sudo update-rc.d rabbitmq-server defaults`

```
$ sudo service rabbitmq-server start
$ sudo service rabbitmq-server stop
```

- Create Admin User in RabbitMQ `$ sudo rabbitmqctl add_user admin password`

```
$        sudo rabbitmqctl set_user_tags admin
administrator
$ sudo rabbitmqctl set_permissions -p / admin
".*" ".*" ".*"
```

- Setup RabbitMQ Web Management Console `$ sudo rabbitmq-plugins enable rabbitmq_management`

Pemasangan perangkat lunak Nagios

Dalam pengembangan sistem ini, digunakan Nagios untuk memantau *server*. Untuk memasang Nagios *server*, jalankan perintah instalasi berikut ini :

- Install Nagios

```
$ sudo useradd nagios $ sudo groupadd nagcmd $
sudo usermod -a -G nagcmd nagios
$ sudo apt-get update
```

```
$        sudo apt-get install build-essential
libgd2-xpm-dev openssl libssl-dev unzip
$ cd
$                                    curl -L -O
https://assets.nagios.com/downloads/nagioscore/releases
4.3.4.tar.gz
$ tar zxf nagios-*.tar.gz
$ cd nagios-*
$      ./configure --with-nagios-group=nagios
--with-command-group=nagcmd
$ make all
$ sudo make install
$ sudo make install-commandmode
$ sudo make install-init
$ sudo make install-config
$             sudo /usr/bin/install -c -m 644
sample-config/httpd.conf
/etc/apache2/sites-available/nagios.conf
$ sudo usermod -G nagcmd www-data
```

- Installing the check_nrpe Plugin
```
$ cd
$                                    curl -L -O
https://github.com/NagiosEnterprises/nrpe/releases/down
3.2.1/nrpe-3.2.1.tar.gz
$ tar zxf nrpe-*.tar.gz
$ cd nrpe-*
$ ./configure
$ make check_nrpe
$ sudo make install-plugin
```

Selanjutnya, pada *server* yang ingin dipantau jalankan langkah-langkah di bawah ini:

- Install NRPE
```
$ sudo useradd nagios
$ sudo apt-get update
$        sudo apt-get install build-essential
```

```
libgd2-xpm-dev openssl libssl-dev unzip
$ cd
$                                    curl -L -O
http://nagios-plugins.org/download/nagios-
plugins-2.2.1.tar.gz
$ tar zxf nagios-plugins-*.tar.gz
$ cd nagios-plugins-*
$       ./configure --with-nagios-user=nagios
--with-nagios-group=nagios --with-openssl
$ make
$ sudo make install
$ cd
$                                    curl -L -O
https://github.com/NagiosEnterprises/nrpe/releases/downlo
3.2.1/nrpe-3.2.1.tar.gz
$ tar zxf nrpe-*.tar.gz
$ cd nrpe-*
$           ./configure --enable-command-args
--with-nagios-user=nagios
--with-nagios-group=nagios
--with-ssl=/usr/bin/openssl
--with-ssl-lib=/usr/lib/x86_64-linux-gnu
$ make all
$ sudo make install
$ sudo make install-init
$ sudo make install-config
```

Pemasangan perangkat lunak Telegram

Dalam pengembangan sistem ini, digunakan Telegram sebagai tempat untuk mendapatkan pesan dari *publisher* jika *server* terjadi masalah. Untuk memasang Telegram agar dapat

dimasukkan ke dalam *publisher*, jalankan perintah `pip install python-telegram-bot --upgrade`.

Instalasi pada Raspberry Pi

Pada Raspberry Pi, guna mendapatkan data dari sensor maka dilakukan instalasi Adafruit DHT11, yaitu sebuah Python *library*. Berikut ini adalah langkah-langkahnya:

```
$ git clone https://github.com/adafruit/Adafruit_Python_DHT.git
$ cd Adafruit_Python_DHT
$ sudo apt-get install build-essential python-dev
$ sudo python setup.py install
```

(Halaman ini sengaja dikosongkan)

LAMPIRAN B

KONFIGURASI

Konfigurasi Nagios

Pada Nagios *server* dilakukan konfigurasi yang sifatnya opsional. Konfigurasi ini terkait dengan perizinan akses dari ip lain. Konfigurasi dapat dilakukan di `/etc/apache2/sites-available/nagios.conf`. Selalu lakukan `sudo systemctl restart apache2` jika melakukan perubahan pada konfigurasi yang telah disebutkan di atas.

Lalu, kita membuat berkas konfigurasi `nagios.service` dengan membuka

$ `sudo nano /etc/systemd/system/nagios.service`

Isi dari berkas konfigurasi dapat dilihat pada Kode Sumber 2.1 berikut.

```
[Unit]
Description=Nagios
BindTo=network.target

[Install]
WantedBy=multi-user.target

[Service]
Type=simple
User=nagios
Group=nagios
ExecStart=/usr/local/nagios/bin/nagios /usr
    /local/nagios/etc/nagios.cfg
```

Kode Sumber 2.1: Isi Berkas app

Lalu, nyalakan Nagios dengan menjalankan perintah di bawah ini:

$ `sudo systemctl enable /etc/systemd/system/nagios.service`

```
$ sudo systemctl start nagios
```

Setelah semua dilakukan, kita dapat mengakses *web interface* Nagios dengan membuka `http://nagios_server_public_ip/nagios` di peramban.

Kemudian pada *host* yang ingin dipantau, setelah dilakukan penginstalan maka dilanjutkan dengan mengkonfigurasi di berkas `/usr/local/nagios/etc/nrpe.cfg`. Konfigurasi yang dilakukan dapat dilihat di Kode Sumber 2.2 berikut ini.

```
...
allowed_hosts=127.0.0.1,::1,
    your_nagios_server_private_ip
...
server_address=monitored_server_private_ip
...
command[check_vda1]=/usr/lib/nagios/plugins
    /check_disk -w 20% -c 10% -p /dev/vda1
...
```

Kode Sumber 2.2: Isi Berkas nrpe.cfg

Setelah dilakukan konfigurasi, jalankan perintah `sudo systemctl restart nrpe.service`.

BIODATA PENULIS

Nafia Rizky Yogayana, akrab dipanggil Nafia, lahir pada tanggal 9 Juli 1996 di Batam. Penulis merupakan seorang mahasiswa yang sedang menempuh studi di Departemen Informatika Institut Teknologi Sepuluh Nopember. Memiliki beberapa hobi antara lain fotografi, membaca, dan mendengarkan musik. Pernah menjadi asisten dosen pada mata kuliah Jaringan Komputer dan Analisis dan Perancangan Sistem Informasi (APSI) pada tahun ajaran 2016/2017. Lalu, juga pernah menjadi asisten dosen pada mata kuliah Manajemen Proyek Perangkat Lunak dan Keamanan Informasi Jaringan pada tahun ajaran 2017/2018. Penulis juga pernah menjadi asisten dosen Pendidikan Informatika dan Komputer Terapan (PIKTI) ITS pada tahun ajaran 2017/2018. Selama menempuh pendidikan di kampus, penulis juga aktif dalam organisasi kemahasiswaan, antara lain sebagai Staff Departemen Media Informasi Himpunan Mahasiswa Teknik Computer-Informatika pada tahun ajaran 2015/2016 dan 2016/2017. Penulis juga aktif dalam kepanitiaan Schematics, antara lain sebagai Koordinator 3 Biro Web dan Kesekretariatan pada tahun 2015 dan Badan Pengurus Harian (BPH) di biro yang sama pada tahun 2016.Penulis juga merupakan salah satu administrator aktif pada Workshop Pemrograman 2 di Departemen Informatika ITS.