

**PERANCANGAN DAN IMPLEMENTASI SISTEM JARINGAN, BASIS DATA, DAN
GUI UNTUK PENGGUNA UMUM PADA SISTEM JARINGAN DETEKTOR GEMPA
DAN TSUNAMI *DECISION SUPPORT SYSTEM***

TUGAS AKHIR

Oleh
CHRISTOPORUS DEO PUTRATAMA
NIM: 13212008
Program Studi Teknik Elektro



**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2017**

**PERANCANGAN DAN IMPLEMENTASI SISTEM JARINGAN, BASIS DATA, DAN
GUI UNTUK PENGGUNA UMUM PADA SISTEM JARINGAN DETEKTOR GEMPA
DAN TSUNAMI *DECISION SUPPORT SYSTEM***

Oleh:

Christoporus Deo Putratama

Tugas Akhir ini telah diterima dan disahkan
sebagai persyaratan untuk memperoleh gelar

SARJANA TEKNIK

di

**PROGRAM STUDI TEKNIK ELEKTRO
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG**

Bandung, 12 Mei 2017

Disetujui oleh:

Pembimbing I,

Pembimbing II,

Dr. tech. Ary Setijadi Prihatmanto
NIP. 197208271997021003

Dr.Ir. Aciek Ida Wuryandari
NIP. 195410051982112001

ABSTRAK

PERANCANGAN DAN IMPLEMENTASI SISTEM JARINGAN, BASIS DATA, DAN GUI UNTUK PENGGUNA UMUM PADA SISTEM JARINGAN DETEKTOR GEMPA DAN TSUNAMI *DECISION SUPPORT SYSTEM*

Oleh

Christoporus Deo Putratama

NIM 13213008

PROGRAM STUDI TEKNIK ELEKTRO

Earthquake Catcher Network (ECN) merupakan hasil pengembangan dari sistem jaringan detektor gempa dan tsunami decision support system dengan menggunakan sensor pergerakan dan sensor penentuan lokasi yang terintegrasi jaringan internet sebagai alat bantu keputusan untuk melakukan evakuasi. ECN digunakan sebagai salah satu solusi untuk Sistem Peringatan Awal Gempa dan Tsunami yang murah. ECN terdiri dari tiga bagian, yaitu sensor gempa, sistem jaringan dan basis data, dan perangkat lunak pengolahan gempa. ECN dapat mendeteksi getaran yang ada di permukaan tanah. Data percepatan ini kemudian digabungkan dengan data lokasi sensor lalu dikirimkan melalui jaringan internet. Protokol pengiriman yang dilakukan harus dapat membedakan antar sensor dan juga dapat mengirimkan data seismik menuju server basis data maupun perangkat lunak pengolahan gempa, maupun menuju perangkat lunak GUI untuk penggunaan umum. Hasil pengolahan data akan dibandingkan dengan pengolahan data yang dilakukan oleh BMKG, dan USGS. Pada implementasinya, sebuah modul sensor gempa akan mengirimkan data seismik dalam format GeoJSON. Pengiriman data dilakukan dengan menggunakan melalui jaringan internet menuju Messaging Server RabbitMQ dan kemudian dilakukan konsumsi data oleh perangkat lunak server berbasis MySQL. Pengujian jaringan dilakukan dengan mengirimkan sebuah tanda waktu menuju RabbitMQ kemudian data tersebut dikonsumsi dengan menggunakan sebuah program. Program ini akan memberikan tanda waktu kedua lalu kedua tanda tsb akan dikirimkan ke basis data. Hasil yang diinginkan adalah data kedua waktu ini sama sehingga membuktikan bahwa pengiriman data dari sensor gempa menuju konsumen yang diinginkan berhasil.

Kata kunci: Earthquake Catcher Network, data seismik, jaringan, basis data, GUI

ABSTRACT

DESIGN AND IMPLEMENTATION OF NETWORK SYSTEMS, DATA BASES, AND GUI FOR GENERAL USE OF DETECTOR EQUIPMENT AND EQUIPMENT SYSTEMS DECISION SUPPORT SYSTEM

By

Christoporus Deo Putratama

NIM 13213008

ELECTRICAL ENGINEERING

Earthquake Catcher Network (ECN) is the result of the development of earthquake and tsunami network detector and decision support system by using the movement sensors and location locating sensors that being integrated by the Internet network as a decision tools for evacuation. ECN is used as one of the solutions of cheap Earthquake and Tsunami Early Warning System. ECN consists of three parts, namely earthquake sensors, network systems and databases, and earthquake processing software. ECN can detect vibrations that exist in the soil surface. Data of accelerations are being combined with sensor location data and then will be transmitted over the internet network. The delivery protocol must be able to distinguish between sensors and can also transmit seismic data to both database servers and earthquake processing software, as well as to GUI software for general use. The results of data processing will be compared with data processing conducted by BMKG, and USGS. In its implementation, an earthquake sensor module will transmit seismic data in GeoJSON format. Data transmission is done by using over the internet network to RabbitMQ Messaging Server and then performed data consumption by MySQL based server software. Network testing is done by sending a timestamp to RabbitMQ then the data is consumed using a program. This program will give a second timestamp then both marks will be sent to the database. The desired result is the same second time data so as to prove that the data transmission from the earthquake sensor to the desired customer succeeds.

Keywords: Earthquake Catcher Network, seismic data, network, database, GUI

KATA PENGANTAR

Puji syukur penulis panjatkan ke Tuhan Yang Maha Esa, yang atas rahmat dan karunia-Nya penulis dapat menyelesaikan penulisan dan pengerjaan tugas akhir ini. Selama mengerjakan tugas akhir ini, penulis mendapat bantuan dan dukungan dari berbagai pihak. Untuk itu, penulis ingin mengucapkan terima kasih kepada:

- ✓ Ayah dan Ibuku, Hadrianus Krisna Subranta dan Therecia Ratna, yang selalu membimbing, mendidik, mengajarkan kedisiplinan tanpa kenal lelah, memberikan dukungan dan mendoakan penulis hingga bisa seperti sekarang ini, serta adikku tercinta, Robertus Dewa Brata Krisantoaji, yang setiap hari selalu belajar dengan tekun dan memberikan semangat juang bagi penulis, semoga sukses belajarnya dan mendapatkan cita-cita yang diinginkan.
- ✓ Bapak Dr. tech. Ary Setijadi Prihatmanto, selaku Dosen Pembimbing pertama yang telah membimbing penulis serta memberikan arahan dalam pengerjaan tugas akhir ini;
- ✓ Ibu Dr.Ir. Aciek Ida Wuryandari, selaku Dosen Pembimbing kedua yang telah membimbing penulis serta memberikan arahan dalam pengerjaan tugas akhir ini;
- ✓ Bapak Hendy Irawan S.T., M.T, selaku Asisten Dosen Pembimbing yang telah membimbing penulis terutama dalam memberikan arahan dalam keilmuan komputer;
- ✓ Bapak Ridwan Suhud S.T selaku Rekan dalam bekerja yang telah membantu melancarkan proses konsultasi kepada badan yang berwenang.
- ✓ Ibu Dr. Lina Handayani selaku Peneliti di Lembaga Ilmu Pengetahuan Indonesia yang telah memberikan konsultasi dan juga fasilitas pengujian tugas akhir ini.
- ✓ Bramantio Yuwono dan Kevin Shidqi Prakoso selaku satu tim seperjuangan bersama dengan penulis yang tidak kenal lelah menyelesaikan setiap tantangan yang hadir dalam pengerjaan tugas akhir ini, selalu memberikan motivasi dan semangat satu sama lain, semoga bisa bertemu lagi dengan cerita kesuksesan masing – masing;
- ✓ Teman-teman seperjuangan di ruang riset mandiri tempat penulis mengerjakan tugas akhir ini yang selalu memberikan keceriaan, dukungan, dan motivasi untuk pantang menyerah;
- ✓ Yosua Sepri Andasto, Brian Benyamin Sialagan, dan Muhammad Aldo Aditya Nugroho yang sudah membantu penulis dalam menyusun video dokumentasi tugas akhir elektro ini;

- ✓ Seluruh teman-teman Teknik Elektro ITB angkatan 2013, serta seluruh teman-teman HME ITB;
- ✓ dan semua pihak yang membantu, yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa pengerjaan tugas akhir ini tidaklah sempurna. Karena itu kritik dan saran sangat diharapkan oleh penulis.

Akhir kata, semoga buku tugas akhir ini dapat bermanfaat bagi para pembaca.

Bandung, 12 Mei 2017

Penulis

Christoporus Deo Putatama, 13212008

DAFTAR ISI

ABSTRAK	II
ABSTRACT.....	III
KATA PENGANTAR.....	IV
DAFTAR ISI.....	VI
DAFTAR GAMBAR.....	VII
DAFTAR SINGKATAN DAN LAMBANG	VIII
BAB I PENDAHULUAN.....	1
I.1 LATAR BELAKANG	1
I.2 PERUMUSAN MASALAH.....	2
I.3 TUJUAN DAN MANFAAT PENELITIAN	2
I.4 BATASAN MASALAH	3
I.5 METODOLOGI	3
I.6 SISTEMATIKA PENULISAN	3
BAB II TINJAUAN PUSTAKA	5
II.1 RABBITMQ MESSAGING SERVER	5
II.2 BASIS DATA MYSQL.....	5
II.3 GRAPHICAL USER INTERFACE	6
BAB III PERANCANGAN	7
III.1 MESSAGING SERVER PADA RABBITMQ	7
III.2 BASIS DATA SERVER MYSQL	9
III.3 GRAPHICS USER INTERFACE UNTUK PENGGUNA UMUM.....	10
BAB IV IMPLEMENTASI DAN PENGUJIAN	12
IV.1 IMPLEMENTASI MESSAGING SERVER, BASIS DATA, DAN GUI UNTUK PENGGUNA UMUM	12
IV.1.1 Pendahuluan	12
IV.1.2 Struktur Implementasi.....	12
IV.1.3 Environment.....	13
IV.1.4 Implementasi Messaging Server RabbitMQ	13
IV.1.5 Implementasi Basis data MySQL.....	19
IV.1.6 Implementasi GUI untuk pengguna umum	23
IV.2 PENGUJIAN MESSAGING SERVER DAN BASIS DATA	24
IV.2.1 Pengujian Messaging Server RabbitMQ	25
IV.2.2 Pengujian Basis data MySQL.....	26
IV.2.3 Pengujian Latency Jaringan Sistem.....	28
BAB V KESIMPULAN	30
V.1 KESIMPULAN	30
V.2 SARAN.....	30
DAFTAR PUSTAKA.....	31
LAMPIRAN.....	32

DAFTAR GAMBAR

Gambar III-1 skematik topologi messaging server di RabbitMQ yang paling sederhana.	8
Gambar III-2 pengiriman pesan dari satu queue ke consumer yang berbeda.	8
Gambar III-3 skematik topologi yang digunakan dalam proyek	8
Gambar III-4 Flowchart dari program konversi pesan dari RabbitMQ menuju MySQL	10
Gambar III-5 Gambar Tampilan-tampilan yang akan dibuat pada GUI untuk pengguna umum.	11
Gambar IV-1 Gambar fitur gratis CloudAMQP yang dapat digunakan untuk implementasi tugas akhir	14
Gambar IV-2 Gambar pendaftaran instance pada CloudAMQP dengan memilih server yang akan digunakan	14
Gambar IV-3 Gambar daftar instance yang dapat dibuat di CloudAMQP	15
Gambar IV-4 Gambar manajemen CloudAMQP pada sebuah instance	15
Gambar IV-5 Gambar konfigurasi queues pada CloudAMQP	16
Gambar IV-6 daftar queues pada CloudAMQP	16
Gambar IV-7 daftar exchanges pada CloudAMQP	17
Gambar IV-8 binding queue dengan exchange menggunakan routing key	17
Gambar IV-9 proses binding exchanges dengan queues	18
Gambar IV-10 tampilan GUI untuk pengguna umum yang dapat mengamati akuisisi data pada sensor gempa	23
Gambar IV-11 hasil dari pengujian pengiriman pesan yang dilakukan oleh serial monitor IDE Arduino	25
Gambar IV-12 hasil pengujian yang dilakukan oleh program console C# pada proses pemerolehan data dari RabbitMQ	27
Gambar IV-13 tampilan basis data seismik pada perangkat lunak MySQL	27
Gambar IV-14 hasil pengiriman data yang dilakukan oleh IDE Python untuk proses pengujian latency data	28
Gambar IV-15 grafik latency data, dimana garis oranye adalah ketepatan waktu data sampai ke server, dan garis biru adalah batas toleransi data untuk tidak terlambat	29
Gambar IV-16 persentase kemungkinan data mengalami latency yang tidak dapat ditoleransi ..	29

DAFTAR SINGKATAN DAN LAMBANG

AMQP	Advanced Messaging Queue Protocol
BMKG	Badan Meteorologi, Klimatogi, dan Geofisika
ECN	Earthquake Catcher Netwirk
EWS	Early Warning System
GUI	Graphical User Interface
IDE	Integrated Development Environment
ITB	Institut Teknologi Bandung
JSON	JavaScript Object Notation
PPTIK	Pusat Penelitian Teknologi, Informasi dan Komunikasi
QCN	Quake Catcher Network
SQL	Structured Query Language
URL	Uniform Resource Locator

BAB I

PENDAHULUAN

I.1 Latar Belakang

Secara geografis, Indonesia terletak pada pertemuan 3 lempeng tektonik utama dunia yang bergerak relatif saling mendesak satu dengan lainnya. Hal ini menyebabkan Indonesia menjadi negara yang di dalamnya rawan terjadi bencana, khususnya bencana gempa bumi. Berdasarkan data dari Badan Meteorologi Klimatologi dan Geofisika (BMKG), di wilayah Indonesia dapat dideteksi sekitar 4000 gempa bumi terjadi pertahun, sedangkan gempa bumi berkekuatan di atas 5,5 SR dan gempa bumi yang dapat dirasakan oleh manusia, terjadi rata-rata sekitar 70–100 kali per tahun, dan gempa bumi tektonik yang menimbulkan kerusakan terjadi antara 1–2 kali per tahun. Sejak tahun 1991 sampai dengan 2011 tercatat telah terjadi 186 kali gempabumi tektonik yang merusak. Gempa bumi dengan magnitudo besar (7 SR atau lebih) dengan kedalaman yang dangkal di bawah laut, dapat menimbulkan tsunami karena adanya perubahan ketinggian kolom air dalam waktu singkat. Oleh karena itu berdasarkan fakta-fakta di atas, kesiapan dan ketanggapan terhadap kondisi darurat dan bencana khususnya bencana gempa bumi dan tsunami menjadi sesuatu yang mendesak bagi negara dan masyarakat Indonesia.

Salah satu solusi yang dapat diupayakan dalam rangka meningkatkan kemampuan negara dan masyarakat Indonesia dalam menghadapi kondisi darurat kebencanaan adalah dengan membangun Sistem Peringatan Dini (Early Warning System – EWS) khususnya untuk bencana gempa bumi dan tsunami. Supaya EWS dapat terlaksana, pembangunan infrastruktur deteksi gempa yang efektif dan responsif menjadi urgensi yang utama. Hal ini dikarenakan jumlah seismometer yang ada di Indonesia tergolong sedikit untuk mencakup luas wilayah Indonesia. Selain itu, pengambilan keputusan untuk melakukan diseminasi juga harus didukung data dengan tingkat kepercayaan yang tinggi, dan itu bisa dilakukan ketika semakin banyak detektor gempa yang terpasang di Indonesia.

Earthquake Catcher Network (ECN) merupakan pengembangan teknologi yang dibuat dari teknologi yang sebelumnya dibuat oleh Stanford University dengan judul Quake Catcher Network (QCN). Teknologi detektor gempa yang murah dan diintegrasikan dengan jaringan internet memungkinkan peneliti untuk memperoleh data seismik dengan lebih akurat dan juga dapat

meningkatkan performa diseminasi dan EWS itu sendiri. Dalam tulisan ini, penulis akan berfokus pada sistem jaringan dan basis data yang menjadi jembatan bagi sensor gempa untuk dapat mengirimkan data seismik yang dibutuhkan oleh peneliti dan juga dapat diolah sebagai perangkat lunak untuk pengguna umum. Sistem jaringan ini melibatkan koneksi dari sensor menuju server pengirim pesan dan juga server basis data.

I.2 Perumusan Masalah

Berdasarkan latar belakang di atas, maka masalah yang akan diangkat oleh penulis dalam tugas akhir ini adalah sebagai berikut:

- Bagaimana membuat konfigurasi untuk koneksi dari sensor gempa menuju server pengirim pesan?
- Bagaimana prosedur dan algoritma yang digunakan untuk dapat melakukan konsumsi data dari server pengirim pesan?
- Bagaimana algoritma yang digunakan untuk dapat mengirimkan data menuju server basis data?
- Bagaimana membuat perangkat lunak yang dapat melakukan monitoring data terhadap sensor gempa yang sedang bekerja?

I.3 Tujuan dan Manfaat Penelitian

Tugas akhir ini memiliki tujuan sebagai berikut:

- Konfigurasi koneksi sensor gempa menuju server pengirim pesan benar
- Ada prosedur dan algoritma yang dibuat dan digunakan untuk dapat melakukan proses koneksi dan konsumsi data dari server pengirim pesan
- Algoritma yang digunakan untuk mengirimkan data menuju server basis data telah dibuat dan dapat diimplementasikan
- Pembuatan GUI untuk penggunaan umum yang dapat melakukan monitoring terhadap sensor gempa yang sedang bekerja.

I.4 Batasan Masalah

Batasan masalah yang penulis rumuskan dalam tugas akhir ini adalah sebagai berikut:

- Pembuatan jaringan dibatasi hanya untuk sekitar 6 buah sensor gempa
- Data seismik yang dikirimkan oleh sensor gempa berupa data diskrit dan bukan data analog
- Tidak dilakukan pembahasan teori frekuensi jaringan pada implementasi sistem jaringan
- Tidak dilakukan analisa magnituda, episentrum dan hiposentrum pada GUI untuk pengguna umum

I.5 Metodologi

Metode penelitian yang dilakukan dalam mengerjakan tugas akhir ini adalah sebagai berikut.

1. Penentuan topik
2. Studi pustaka
3. Penentuan spesifikasi
4. Perancangan sistem
5. Implementasi sistem rancangan
6. Pengujian dan perbaikan sistem

I.6 Sistematika Penulisan

Penulis membagi laporan menjadi lima bab dengan urutan sebagai berikut:

BAB I PENDAHULUAN

Pada bab ini dibahas mengenai latar belakang, perumusan masalah, tujuan dan manfaat penelitian, batasan masalah, metodologi pengerjaan, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Pada bab ini dijabarkan dasar-dasar teori yang menjadi acuan dasar penulis dalam merancang dan mengimplementasikan sistem jaringan dan basis data pada ECN

BAB III PERANCANGAN

Dalam bab ini dijelaskan mengenai perancangan sistem jaringan dan basis data pada ECN menggunakan RabbitMQ dan MySQL

BAB IV IMPLEMENTASI DAN PENGUJIAN

Bab ini menjelaskan hasil pengujian keterlambatan pengiriman dalam sistem jaringan dan basis data pada ECN

BAB V KESIMPULAN DAN SARAN

Pada bab ini dijabarkan mengenai kesimpulan yang didapatkan pada tugas akhir ini, dan saran untuk pengembangan sistem selanjutnya.

BAB II

TINJAUAN PUSTAKA

II.1 RabbitMQ Messaging Server

RabbitMQ merupakan sebuah perangkat lunak messaging broker open source yang mengimplementasikan protokol AMQP atau *Advanced Message Queueing Protocol*. RabbitMQ dapat dijalankan dengan menggunakan bahasa pemrograman Erlang. RabbitMQ mempunyai fitur seperti mengatur pesan secara asinkron, dapat diintegrasikan dengan berbagai program dan bahasa pemrograman seperti Java, .NET, PHP, Python, JavaScript, Ruby, dll.

RabbitMQ dipilih pada proyek ini karena dapat mengimplementasikan AMQP secara open source, dapat melakukan pengelompokan data dengan mudah karena adanya bahasa pemrograman Erlang, dan juga RabbitMQ lebih dapat diandalkan dan tahan terhadap kerusakan apabila dibandingkan dengan broker pesan lain seperti ActiveMQ, ZeroMQ, dan Apache Qpid.

II.2 Basis data MySQL

MySQL adalah sistem manajemen basis data Open Source SQL terpopuler, yang dikembangkan, didistribusikan, dan didukung oleh Oracle Corporation. MySQL adalah sistem manajemen basis data (Database). Supaya dapat menambah, mengakses, dan mengolah data yang tersimpan dalam basis data komputer, diperlukan sistem manajemen basis data seperti MySQL Server. Komputer merupakan alat yang sangat handal dalam menangani data yang besar, sehingga sistem manajemen basis data memainkan peran yang penting. Basis data MySQL bersifat relasional. Basis data relasional menyimpan data dalam tabel terpisah daripada memasukkan semua data ke dalam satu gudang besar. Struktur basis data disusun menjadi file fisik yang dioptimalkan untuk kecepatan. Model logis, dengan objek seperti basis data, tabel, tampilan, baris, dan kolom, menawarkan lingkungan pemrograman yang fleksibel.

Bagian SQL dari "MySQL" adalah singkatan dari "Structured Query Language". SQL adalah bahasa standar yang paling umum digunakan untuk mengakses basis data. SQL dapat dimasukkan secara langsung, tergantung pada lingkungan pemrograman. SQL dapat dimasukkan ke dalam kode yang ditulis dalam bahasa lain, atau menggunakan API khusus. MySQL Server pada awalnya dikembangkan untuk menangani basis data besar jauh lebih cepat daripada solusi yang ada dan

telah berhasil digunakan di lingkungan produksi yang sangat menuntut selama beberapa tahun. Meskipun dalam perkembangan yang konstan, MySQL Server saat ini menawarkan seperangkat fungsi yang kaya dan berguna. Konektivitas, kecepatan, dan keamanannya membuat MySQL Server sangat cocok untuk mengakses basis data di Internet. Perangkat Lunak Basis Data MySQL adalah sistem klien / server yang terdiri dari server SQL multi-threaded yang mendukung berbagai ujung yang berbeda, beberapa program klien dan perpustakaan yang berbeda, alat administratif, dan berbagai antarmuka pemrograman aplikasi (API).

II.3 Graphical User Interface

Graphical User Interface (GUI) merupakan salah satu jenis antarmuka yang dibuat untuk pengguna yang dapat melakukan interaksi dengan alat-alat elektronik melalui tombol grafis dan indikator visual. Tindakan yang bisa dilakukan di dalam GUI biasanya dilakukan melalui manipulasi langsung elemen grafis. Pada tugas akhir kali ini, bahasa pemrograman yang digunakan untuk membuat GUI adalah visual C#. Penggunaan visual C# ditujukan untuk membuat sebuah halaman Form yang dapat didesain untuk dijadikan antarmuka, seperti menambah tombol kontrol, menampilkan sebuah grafik secara realtime dan juga menampilkan teks secara interaktif.

BAB III

PERANCANGAN

III.1 Messaging Server Pada RabbitMQ

Pada jaringan yang terdapat dalam Sistem Jaringan Detektor Gempa dan Tsunami *Decision Support System*, standar data yang digunakan dalam pengiriman secara aktual adalah dalam bentuk sebuah pesan singkat yang memiliki isi yang padat dan jelas. Sebagai contoh, sebuah protokol pesan singkat untuk memberitakan terjadinya sebuah gempa dan/atau tsunami adalah sebagai berikut:

```
Tsunami Warning in BENGKULU, Eq Mag:7.0RS, 09-Dec-09 15:52:59 UTC,  
Loc:4.64S/101.11E,Dep:10km::BMKG
```

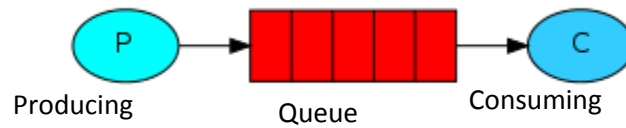
Dapat dilihat bahwa dalam mengirimkan pesan singkat maka diperlukan susunan aturan yang dapat dimengerti oleh sistem. Pada proyek ini, protokol komunikasi yang digunakan adalah berbasis Advanced Messaging Queueing Protocol (AMQP). Protokol ini dipilih karena mempunyai beberapa fitur yang sesuai dengan sistem yang digunakan. AMQP diterapkan oleh sebuah vendor komunikasi RabbitMQ sehingga akan digunakan RabbitMQ dengan metode AMQP sebagai protokol komunikasi sistem. RabbitMQ sendiri merupakan broker pesan, dimana sebuah pesan yang dikirimkan dari pengirim menuju tujuan akan difasilitasi oleh RabbitMQ. Protokol pesan dapat dimodifikasi sehingga:

- Pengiriman pesan dapat diurutkan dalam sebuah antrian baris;
- Melakukan penerbitan pesan ke seluruh bagian sistem jaringan;
- Melakukan langganan pesan oleh pengguna pada broker;
- Melakukan penghubungan dan penyaringan pesan;
- Seleksi pesan yang akan diperoleh;
- Kendali jarak jauh pada jaringan dengan metode *remote*.

Fitur-fitur diatas dapat menguntungkan karena tidak semua data yang diperoleh merupakan informasi yang penting dan besar kemungkinan pada sistem bawa sebuah pesan dapat terkena

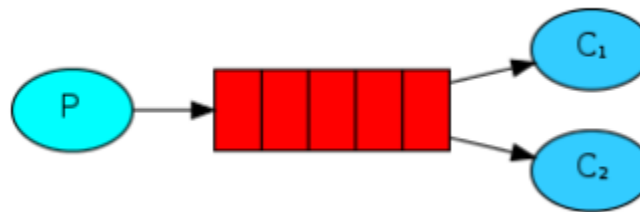
distorsi akibat *bug* pada sistem. Selain itu mudah untuk mengendalikan sebuah sistem jaringan yang besar dengan protokol pesan AMQP ini.

Diagram blok dari sistem protokol komunikasi RabbitMQ adalah sebagai berikut:



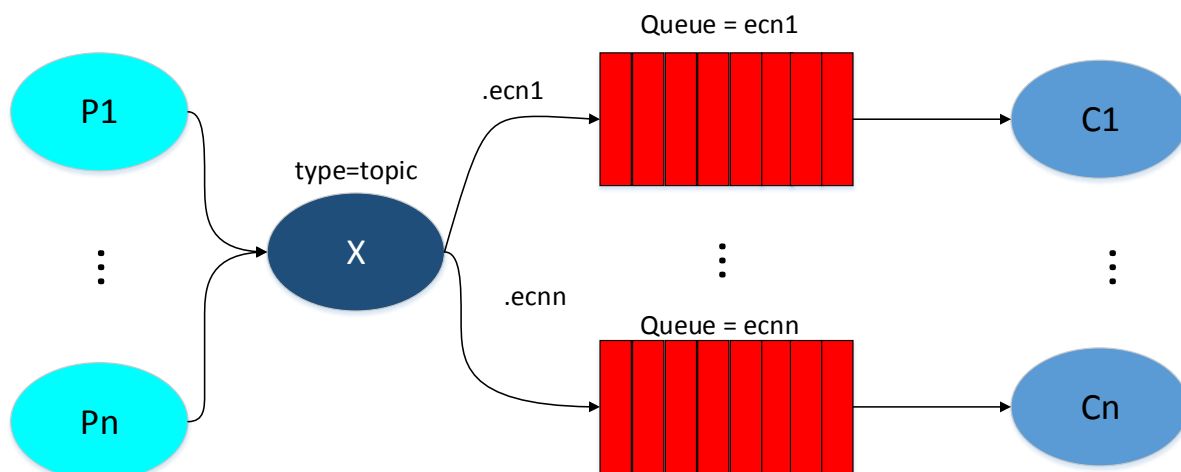
Gambar III-1 skematik topologi messaging server di RabbitMQ yang paling sederhana.

Gambar diagram blok diatas dari sistem paling sederhana dari RabbitMQ, dimana ada sebuah komponen sistem yang menghasilkan sebuah pesan, kemudian dimasukkan ke dalam antrian dan selanjutnya dikirimkan kepada konsumen. Selanjutnya akan disertakan diagram blok dari fitur RabbitMQ berupa penerbitan pesan pada beberapa konsumen.



Gambar III-2 pengiriman pesan dari satu queue ke consumer yang berbeda.

Pada Sistem Jaringan Detektor Gempa dan Tsunami *Decision Support System*, topologi koneksi ke *Messaging Server* RabbitMQ adalah sebagai berikut:



Gambar III-3 skematik topologi yang digunakan dalam proyek

Dimana $P_1 \dots P_n$ adalah sensor pertama hingga ke -n, kemudian sensor akan mengirimkan pesan menuju *exchange* dengan type: topic. Pesan-pesan ini kemudian akan dikirimkan menuju *queue* dari yang pertama hingga ke-n untuk disimpan sesuai dengan jalur *routing* key .ecn masing-masing. C_1 hingga C_n merupakan *consumer* yaitu klien yang membutuhkan data dari RabbitMQ. Konsumer ini dapat berupa klien langsung ataupun sebuah server basis data.

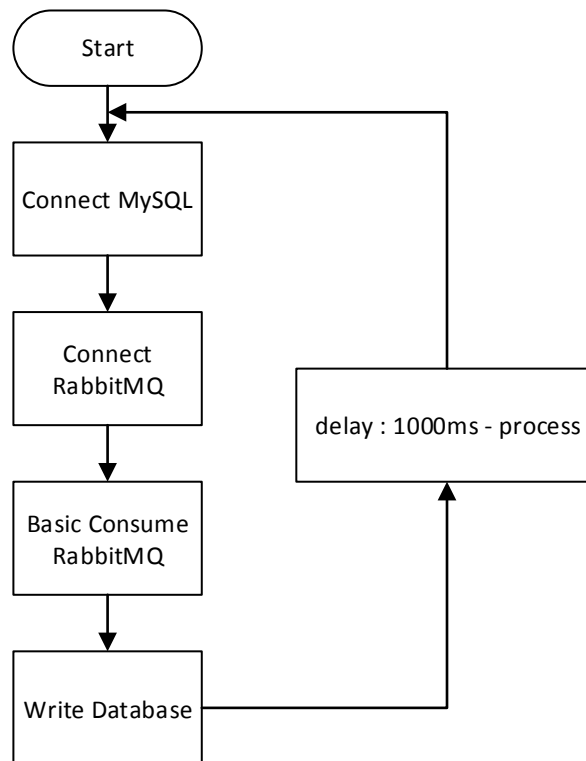
III.2 Basis data Server MySQL

Pada server terdapat media penyimpanan data berupa basis data MySQL. Basis data MySQL dibuat untuk menyimpan data GeoJSON atau parameter-parameter yang dibutuhkan untuk analisa gempa. Penggunaan MySQL umum digunakan dalam mengimplementasikan basis data sehingga desain server pada proyek ini menggunakan MySQL. Oleh karena itu, akan dibuat kolom-kolom yang sesuai dengan data GeoJSON.

Data-data tersebut meliputi:

- a. Nama sensor + *Timestamp*
- b. *Time Zone*
- c. Interval waktu pengiriman
- d. Garis Lintang
- e. Garis Bujur
- f. Percepatan sumbu x
- g. Percepatan sumbu y
- h. Percepatan sumbu z

Nama sensor beserta *timestamp* akan menjadi data yang unik sehingga bisa dijadikan *primary key* untuk pengisian *query* pada *basis data* MySQL. Desain program untuk dapat melakukan penyimpanan pesan menuju Basis data adalah sebagai berikut:



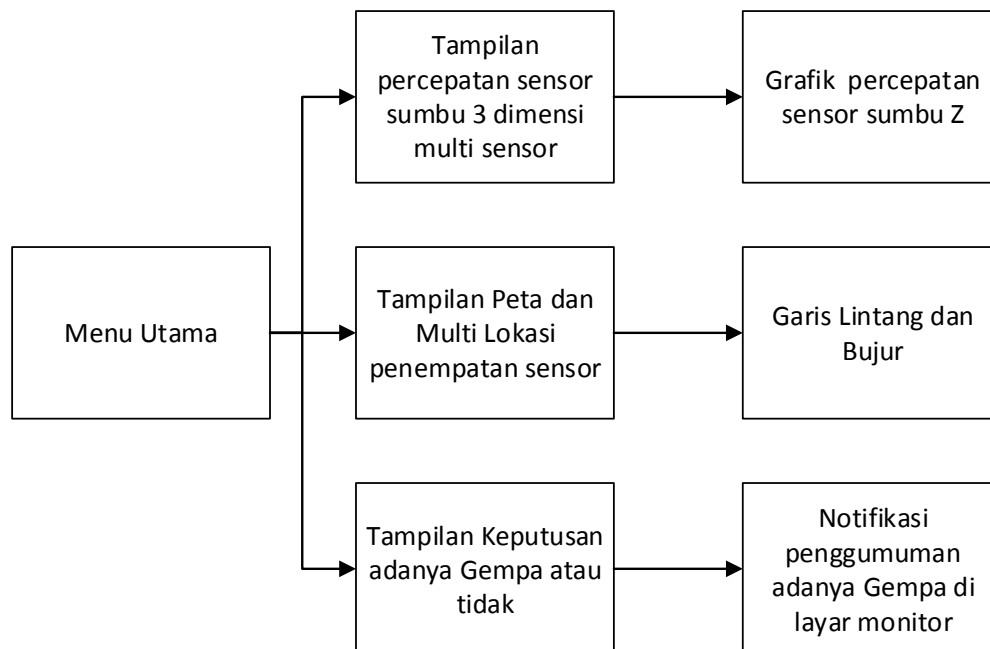
Gambar III-4 Flowchart dari program konversi pesan dari RabbitMQ menuju MySQL

III.3 Graphics User Interface untuk Pengguna Umum

GUI yang akan dibuat berfungsi untuk memberitahu pengguna akan adanya potensi gempa atau tidak dari pengolahan data secara cepat ketika terhubung dengan jaringan ECN. Pembuatan GUI memerlukan *development tools* yang sesuai dengan kebutuhan perancangan. Pada saat ini terdapat berbagai macam *development tools* yang dapat digunakan dalam pengembangan GUI berbasis sistem operasi Microsoft Windows, namun harus dibatasi oleh Bahasa yang digunakan sesuai dengan kemampuan pengembang. Saat ini pengembang mampu menggunakan bahasa C# dalam pembuatan GUI. Sebagai contoh, program pembuatan GUI dengan bahasa C# adalah seperti Microsoft Visual Studio, dan Xamarin Studio. Selain dengan bahasa C#, ada perangkat lunak NetBeans dengan bahasa pemrograman Java yang dapat digunakan dalam pembuatan GUI. Berikut ini tabel perbandingan ketiga perangkat lunak tersebut untuk mempertimbangkan dengan perangkat lunak apa yang akan digunakan untuk melakukan pembuatan GUI khusus untuk pengguna umum.

Microsoft Visual Studio	Xamarin Studio	Netbeans
(-) Hanya dapat digunakan pada OS Microsoft Windows	(+) Dapat digunakan di Microsoft Windows, Linux, dan Mac OS	(+) Dapat digunakan di Microsoft Windows, Linux, dan Mac OS
(+) Ada versi yang tidak berbayar (Community Edition)	(-) Berbayar	(+) Tidak berbayar
(+) Menggunakan bahasa C#	(+) Menggunakan bahasa C#	(-) Menggunakan bahasa Java

Setelah dilakukan perbandingan dan pertimbangan antara kelebihan dan kekurangan masing-masing perangkat lunak untuk pembuatan GUI, maka diputuskan untuk menggunakan Microsoft Visual Studio. Hal ini disebabkan oleh penggunaan bahasa C# yang lebih familiar digunakan oleh pengembang. Apabila menggunakan Netbeans, maka pengembang harus mengerti penggunaan bahasa Java dan hal ini memakan waktu proses pengerjaan. Kemudian setelah melakukan penentuan perangkat lunak pengembangan GUI, maka selanjutnya akan dibahas tentang fungsi yang akan ditampilkan oleh GUI. Fungsi yang akan dibuat adalah sebagai berikut:



Gambar III-5 Gambar Tampilan-tampilan yang akan dibuat pada GUI untuk pengguna umum

BAB IV

IMPLEMENTASI DAN PENGUJIAN

IV.1 Implementasi Messaging Server, Basis data, dan GUI untuk pengguna umum

IV.1.1 Pendahuluan

Pengiriman pesan dilakukan dari sensor gempa menuju messaging server. Protokol yang digunakan oleh sensor untuk mengirim pesan adalah MQTT, sedangkan pesan diterima oleh messaging server RabbitMQ dengan protocol AMQP. Hal ini dapat dilakukan dengan menggunakan format pengiriman pesan berupa file dengan ekstensi JSON sehingga kedua protocol tersebut dapat saling berkomunikasi satu sama lain. Kemudian pesan yang sudah tersimpan dalam RabbitMQ akan di consume menggunakan console C# untuk kemudian dimasukkan ke dalam basis data MySQL.

IV.1.2 Struktur Implementasi

Implementasi pengiriman pesan menuju messaging server RabbitMQ dilakukan berdasarkan perancangan produk yang sudah dilakukan. Sebuah sensor gempa akan mengirimkan data dengan format JSON untuk dimasukkan ke dalam sebuah queue pada RabbitMQ. Namun untuk dapat memasukkan pesan dari sensor yang identik ke dalam sebuah queue, maka perlu dilakukan beberapa tahap implementasi, yaitu adalah sebagai berikut:

- a. menentukan protokol pengiriman pesan yang dilakukan oleh sensor gempa;
- b. pembuatan akun server RabbitMQ umum (CloudAMQP) di internet atau akun server RabbitMQ local yang telah disediakan;
- c. pengaturan server dengan melakukan konfigurasi queue, topic dan exchange melalui browser desktop sehingga pesan yang diterima mempunyai channel tersendiri.

Kemudian setelah itu, untuk dapat memasukkan data dari RabbitMQ menuju MySQL, maka dilakukan beberapa tahap implementasi, yaitu:

- a. membuat program yang dapat melakukan consume pada server RabbitMQ yang berkesesuaian (dalam bahasa C#, maupun Python);

- b. pengaturan kecepatan consume di dalam program hingga 1 pesan / detik supaya memiliki time stamp yang teratur;
- c. pengiriman data dari program menuju MySQL untuk disimpan di dalam basis data.

IV.1.3 Environment

Perangkat keras dan piranti lunak yang dibutuhkan dalam mengimplementasikan RabbitMQ server adalah sebagai berikut:

- a. perangkat keras Laptop Asus X450JE sebagai komputer untuk menjalankan lokal RabbitMQ dan juga sebagai tempat Microsoft Visual Studio 2013 dan MySQL dijalankan.
- b. perangkat lunak Microsoft Visual Studio 2013 sebagai perangkat lunak IDE untuk mengembangkan GUI dan program yang berfungsi sebagai consumer RabbitMQ.
- c. perangkat keras Raspberry Pi 3 sebagai mini komputer untuk menjalankan lokal RabbitMQ
- d. perangkat lunak Python 3.5 sebagai perangkat lunak IDE yang digunakan pada Raspberry Pi 3 yang berfungsi sebagai pengirim pesan maupun consumer pada RabbitMQ
- e. perangkat keras Node MCU Amica sebagai mikrokontroller sensor gempa untuk mengirimkan pesan geospasial menuju server RabbitMQ
- f. perangkat lunak Arduino IDE 1.8.1 sebagai perangkat lunak IDE untuk mengembangkan algoritma pengiriman pesan geospasial menuju RabbitMQ
- g. JSON sebagai library dan syntax pengiriman pesan antara sensor gempa dengan RabbitMQ dan juga antara RabbitMQ dengan consumer

IV.1.4 Implementasi Messaging Server RabbitMQ

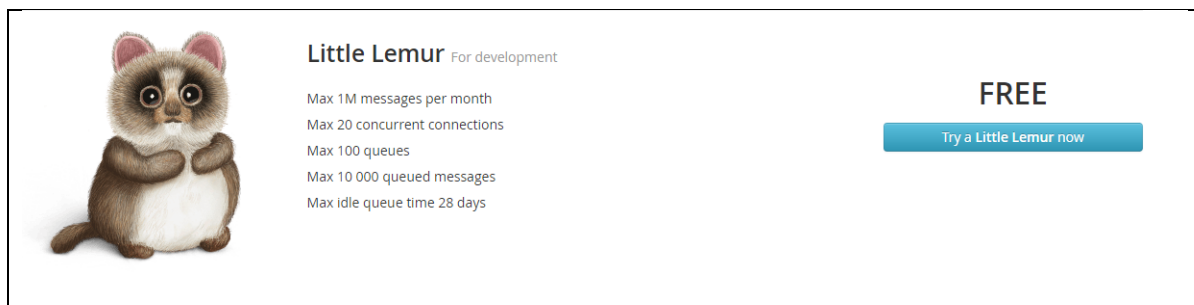
Penggunaan RabbitMQ pada sebuah devais bisa dilakukan dengan melakukan instalasi dan konfigurasi pada platform tertentu. Dalam proyek ini, platform yang digunakan adalah dengan Windows, Linux, maupun Arduino. Selain menggunakan platform, dapat juga menggunakan RabbitMQ umum yang terdapat pada internet dengan menggunakan jasa CloudAMQP. Rincian implementasi RabbitMQ pada semua devais yang digunakan adalah sebagai berikut:

IV.1.4.1 Implementasi RabbitMQ dengan menggunakan CloudAMQP

CloudAMQP digunakan sebagai messaging server online. Supaya dapat menggunakan CloudAMQP, dilakukan pendaftaran akun terlebih dahulu. Pendaftaran CloudAMQP dapat dilakukan dengan membuka tautan dibawah ini pada browser internet.

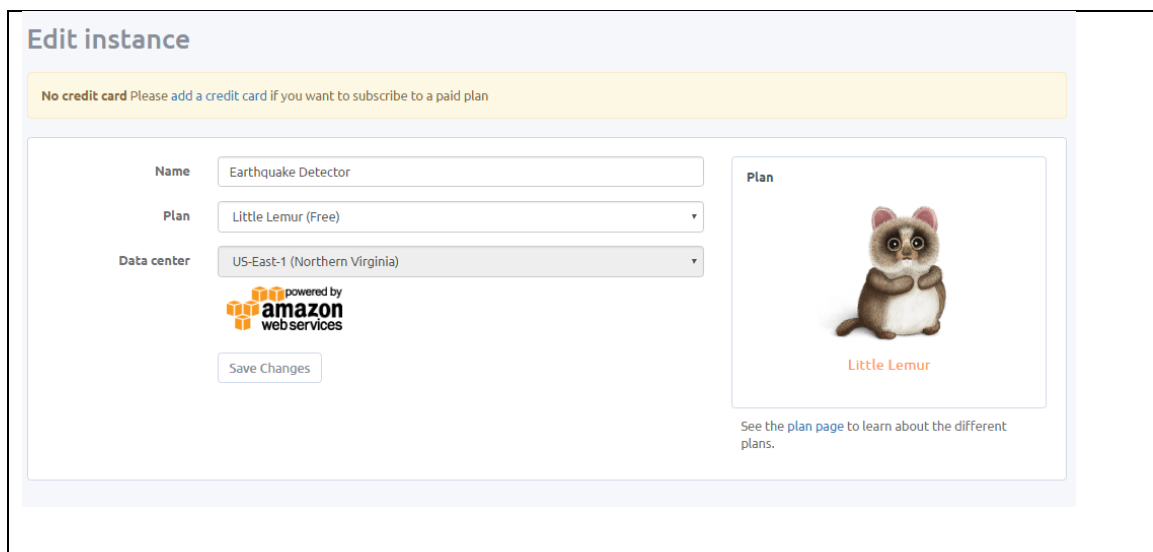
<https://www.cloudamqp.com/plans.html>

Apabila menelaah halaman web ini, disediakan beberapa plan untuk membuat sebuah instance pada CloudAMQP. Proyek ini menggunakan plan Little Lemur karena tidak berbayar. Klik tombol “Try Little Lemur now” untuk menggunakan plan ini.



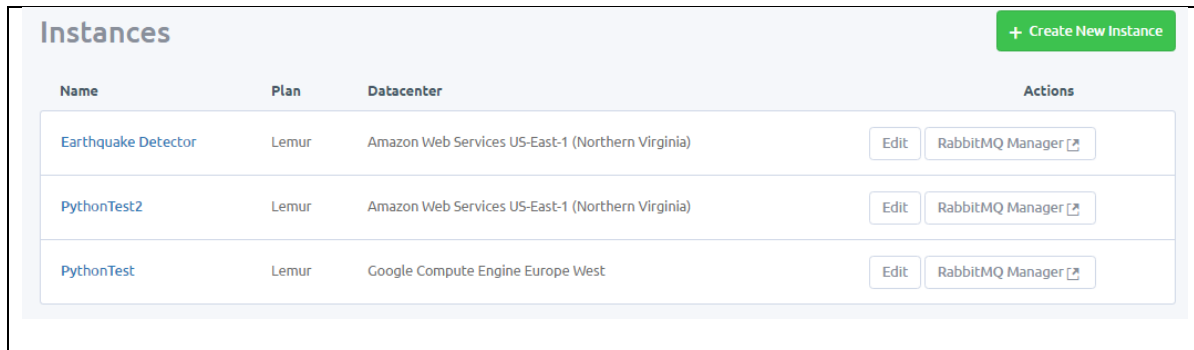
Gambar IV-1 Gambar fitur gratis CloudAMQP yang dapat digunakan untuk implementasi tugas akhir

Setelah melakukan pendaftaran, maka akan terbuka halaman web untuk membuat sebuah instance sesuai dengan plan yang dipilih. Masukkan nama Instance yang diinginkan, plan yang digunakan, dan data center. Data center dapat dipilih dengan bebas namun disarankan dekat dengan negara tempat mengakses server agar dapat mengurangi latency.



Gambar IV-2 Gambar pendaftaran instance pada CloudAMQP dengan memilih server yang akan digunakan

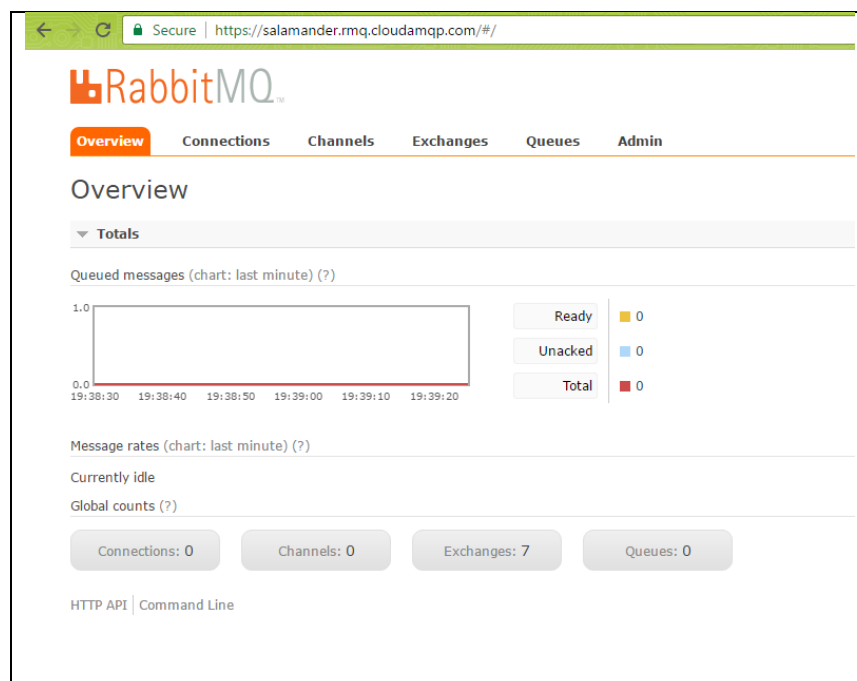
Akan muncul sebuah list instance ketika membuat instance lebih dari sekali. Artinya, ada beberapa server yang bisa digunakan sebagai messaging server. Namun, server tidak berbayar ini memiliki batasan yaitu jumlah message dan koneksi yang terbatas. Apabila hendak mengatur konfigurasi server, maka dapat dilakukan dengan klik tombol “RabbitMQ Manager di sebelah kanan layar browser.



Name	Plan	Datacenter	Actions
Earthquake Detector	Lemur	Amazon Web Services US-East-1 (Northern Virginia)	Edit RabbitMQ Manager
PythonTest2	Lemur	Amazon Web Services US-East-1 (Northern Virginia)	Edit RabbitMQ Manager
PythonTest	Lemur	Google Compute Engine Europe West	Edit RabbitMQ Manager

Gambar IV-3 Gambar daftar instance yang dapat dibuat di CloudAMQP

Setelah itu, pada browser akan terbuka halaman baru yang merujuk pada alamat server. Ketika pertama kali membuka halaman web server ini, maka tampilan yang akan dilihat adalah sebagai berikut:



Gambar IV-4 Gambar manajemen CloudAMQP pada sebuah instance

Hal yang dilakukan pertama kali adalah dengan mengklik tab Queues pada bagian atas layar.

Gambar IV-5 Gambar konfigurasi queues pada CloudAMQP

Pada tab Queues akan terdapat kolom name, durability, auto delete, dsb. Kolom name diisi dengan nama Queue (dalam proyek ini, menggunakan nama “ecn”), kolom durability diisi dengan pilihan “Durable”, sementara itu bagian lain tidak perlu diubah. Setelah itu, tekan tombol Add queue di bagian bawah halaman. Apabila sudah menekan tombol tersebut, maka akan muncul sebuah list queue dengan judul “ecn”.

Queues

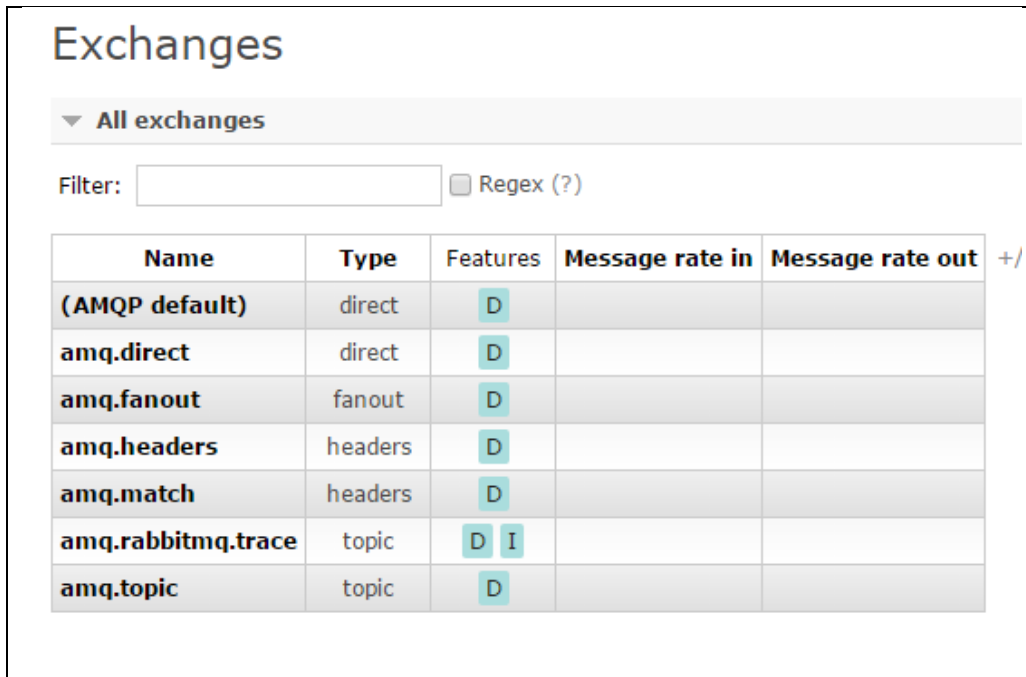
▼ All queues

Filter: ☐ Regex (?)

Overview			Messages			Message rates			+/-
Name	Features	State	Ready	Unacked	Total	incoming	deliver / get	ack	
ecn	D HA	idle	0	0	0				

Gambar IV-6 daftar queues pada CloudAMQP

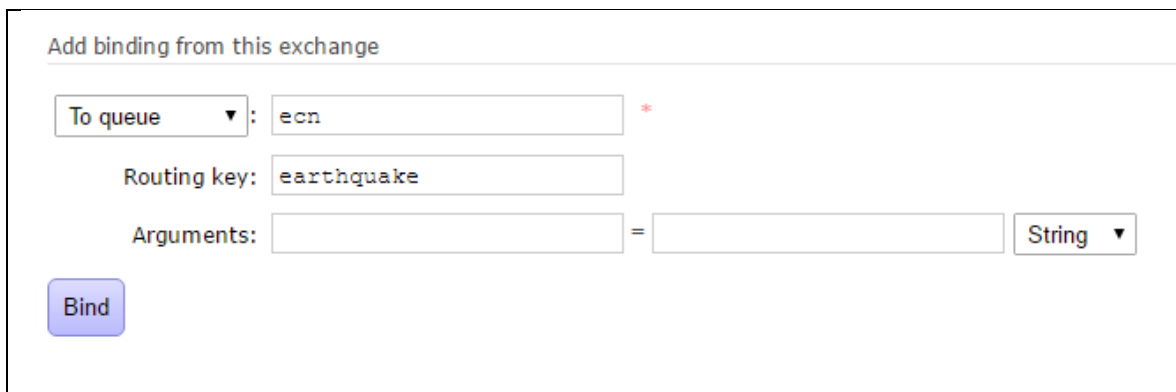
Kemudian hal selanjutnya yang harus dilakukan adalah klik tab Exchanges di bagian atas halaman.



▼ All exchanges				
Filter:	<input type="text"/>	<input type="checkbox"/> Regex (?)		
Name	Type	Features	Message rate in	Message rate out +/
(AMQP default)	direct	D		
amq.direct	direct	D		
amq.fanout	fanout	D		
amq.headers	headers	D		
amq.match	headers	D		
amq.rabbitmq.trace	topic	D I		
amq.topic	topic	D		

Gambar IV-7 daftar exchanges pada CloudAMQP

Akan ditampilkan layar exchanges pada halaman web. Klik kalimat amq.topic pada bagian bawah list exchange.



Add binding from this exchange

To queue ▼: *

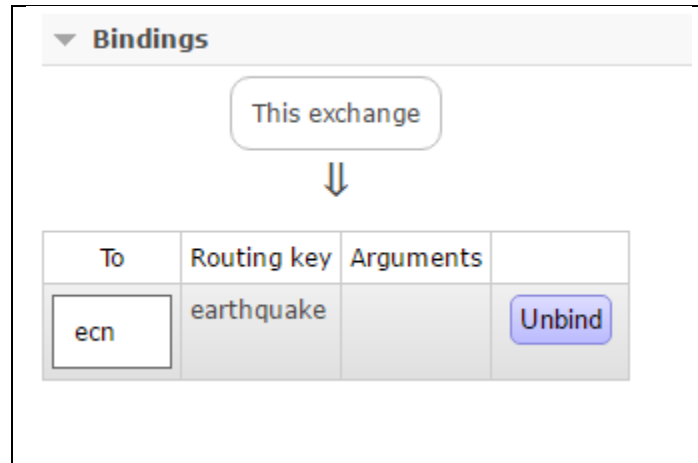
Routing key:

Arguments: = String ▼

Gambar IV-8 binding queue dengan exchange menggunakan routing key

Kemudian dilakukan binding exchange dengan queue. Isi kolom seperti gambar diatas. Hal ini dilakukan supaya queue dihubungkan dengan exchange melalui jalur routing key “earthquake”.

Apabila sudah melakukan bindings, maka akan terdapat tampilan seperti dibawah ini



Gambar IV-9 proses binding exchanges dengan queues

Konfigurasi CloudAMQP sebagai messaging server online bisa dikatakan cukup hingga tahap ini. Penggunaan server ini pada beberapa devais harus mengacu pada syntax yang telah disediakan oleh dokumentasi RabbitMQ sehingga akan diperoleh konfigurasi berbeda pada platform yang berbeda.

IV.1.4.2 Implementasi RabbitMQ dengan menggunakan server RabbitMQ PPTIK

Pada tugas akhir ini, implementasi RabbitMQ dilakukan pada server Rabbit MQ di PPTIK ITB. RabbitMQ diimplementasikan dengan menggunakan protokol AMQP seperti dibawah ini:

```
amqp://sensor_gempa:12345@167.205.7.226/%2fdisaster
```

URL diatas merupakan merupakan parameter yang harus ditambahkan pada proses koneksi pengiriman dan pengambilan data.

Keterangannya adalah sebagai berikut:

- amqp → Protokol komunikasi
- sensor_gempa → username server
- 12345 → password
- 167.205.7.226 → host server
- /%2fdisaster → /disaster → virtual host

Penggunaan %2f disini dimaksudkan untuk mengkoreksi kesalahan pembacaan /disaster karena karakter / berpengaruh terhadap protokol komunikasi dalam bentuk URL. Kemudian selain mengatur alamat server seperti yang dijelaskan diatas, pengaturan berikut-berikutnya akan dijelaskan sebagai berikut:

- a. queue → ecn + nomer sensor

penamaan queue dibebaskan namun harus unik dan khusus dialokasikan untuk sebuah sensor.

- b. exchange → amq.topic

amq.topic merupakan sebuah exchange default dengan type = topic yang dapat dipakai dalam proses implementasi

- c. routing key → exchange + queue

routing key dimaksudkan untuk menghubungkan exchange menuju queue tertentu. Hal ini bertujuan untuk memilah pesan sesuai dengan jenis dan nama sensornya.

- d. durable → true

durable bertujuan untuk menahan pesan supaya tidak hilang ketika server mati.

Sehingga implementasi untuk parameter-parameter diatas adalah sebagai berikut:

```
factory.Uri = "amqp://sensor_gempa:12345@167.205.7.226/%2fdisaster";
channel.QueueDeclare(queue: "ecn",
                    durable: true,
                    exclusive: false,
                    autoDelete: false,
                    arguments: null);
channel.QueueBind(queue: "ecn",
                exchange: "amq.topic",
                routingKey: "amq.topic.ecn"
                );
```

Penggunaan kode diatas adalah implementasi dalam Bahasa C#.

IV.1.5 Implementasi Basis data MySQL

Supaya dapat memasukkan pesan kedalam MySQL, maka terlebih dahulu data di consume dengan menggunakan sebuah program. Program yang akan dibuat adalah program C# yang bertujuan untuk melakukan consume pada RabbitMQ dan mendapatkan pesan. Lalu pesan ini dipilah-pilah

sesuai dengan kolom yang ada pada MySQL. Algoritma dari program tersebut adalah sebagai berikut:

```
public static void Main(string[] args)
{
    connect_database();
    Timer t = new Timer(TimerCallback, null, 0, 1000);
    Console.ReadLine();
}
```

Program diatas adalah program utama yang menjalankan fungsi koneksi basis data MySQL dan kemudian membuat Thread baru untuk menjalankan Timer dimana fungsi tersebut akan dipanggil setiap 1 detik.

```
public static void connect_msgserver()
{
    ConnectionFactory factory;
    using (StreamReader r = new StreamReader("config1.json"))
    {
        string json = r.ReadToEnd();
        Config config = JsonConvert.DeserializeObject<Config>(json);

        factory = new ConnectionFactory();
        factory.Uri = "amqp://sensor_gempa:12345@167.205.7.226/%2fdisaster";
    }
}
```

Program diatas adalah sebuah fungsi untuk melakukan consume pada RabbitMQ.

```
public static void connect_database()
{
    string server_host = "localhost";
    string server_password = "MySQLRoot";
    string server_name = "root";

    try
    {
        con.ConnectionString = "server=" + server_host + ";user id=" + server_name +
        ";password=" + server_password + ";database=earthquake";
        con.Open();
        //MessageBox.Show("Connected to " + server);
    }
    catch (Exception e1)
    {
        Console.WriteLine("Connection failed due to " + e1.ToString());
    }
}

private static void write_database(string[] data)
{
    command_add = con.CreateCommand();
    try
    {
        command_add.CommandText = "INSERT INTO store_id (point_time, time_zone_id,
        interval_id, client_id, lattitude_id, longitude_id) VALUES('" + data[0] + "'", "' + data[1] + "'",
        "'" + data[2] + "'", "' + data[3] + "'", "' + data[4] + "'", "' + data[5] + "')";
        command_add.ExecuteNonQuery();
    }
    catch (Exception e1)
    {
    }
}
```

Program diatas adalah proses untuk melakukan proses koneksi pada basis data MySQL dan kemudian menuliskan data pada kolom-kolom di MySQL.

Consumer yang dimaksud disini adalah GUI yang dirancang dengan menggunakan Microsoft Visual Studio 2016. Supaya GUI dapat mengambil pesan dari messaging server, maka diberikan algoritma pada seperti berikut:

```

        {
            var body = ea.Body;
            var message = Encoding.UTF8.GetString(body);
            Console.WriteLine("[x] Received {0}", message);
            Console.WriteLine("
////////////////////////////////////
////////////////////////////////////");

            data = message;
            try
            {
                data accelReport = JsonConvert.DeserializeObject<data>(message);
                Console.WriteLine("{0} {1} {2} {3} {4}",
                    accelReport.geojson.geometry.coordinates[0], accelReport.geojson.geometry.coordinates[1],
                    accelReport.accelerations[0].x, accelReport.accelerations[0].y,
                    accelReport.accelerations[0].z);
                for (int i = 0 ; i < 20; i++)
                {
                    perfChart.AddValue((decimal)int.Parse(accelReport.accelerations[i].x));
                    perfChart1.AddValue((decimal)int.Parse(accelReport.accelerations[i].y));
                    perfChart2.AddValue((decimal)int.Parse(accelReport.accelerations[i].z));
                }
                gMapControl1.Position = new
                    GMap.NET.PointLatLng(double.Parse(accelReport.geojson.geometry.coordinates[0]),
                    double.Parse(accelReport.geojson.geometry.coordinates[1]));

            }
            catch (Exception ex)
            {
                Console.WriteLine("ERROR: {0}", ex);
            }

        };
        channel.BasicConsume(queue: "ecn", //"emergency_gui",
            noAck: true,
            consumer: consumer);

        Console.WriteLine("Already BasicConsume");
    }
}

```

Dalam algoritma tersebut, terdapat syntax yang bertugas membaca file json dalam suatu directory yang serupa dengan directory GUI. Setelah membaca file json tersebut, akan diambil isi dari file yang bertugas untuk mengakses sebuah messaging server.

```

using (StreamReader r = new StreamReader("config1.json"))
{
    string json = r.ReadToEnd();
    Config config = JsonConvert.DeserializeObject<Config>(json);

    factory = new ConnectionFactory{Uri = config.url};
}

```

Kemudian dilakukan akses pada queue yang terdapat pada messaging server.

```

using (var connection = factory.CreateConnection())
    using (var channel = connection.CreateModel())
    {
        channel.QueueDeclare(queue: "ecn",

```

```

        durable: true,
        exclusive: false,
        autoDelete: false,
        arguments: null);
channel.QueueBind(queue: "ecn",
        exchange: "amq.topic",
        routingKey: "amq.topic.ecn"
        );

```

Setelah itu, dilakukan proses consume data pada queue yang berkaitan.

```

var consumer = new EventingBasicConsumer(channel);
consumer.Received += (model, ea) =>
{
    var body = ea.Body;
    var message = Encoding.UTF8.GetString(body);
    ...
}

```

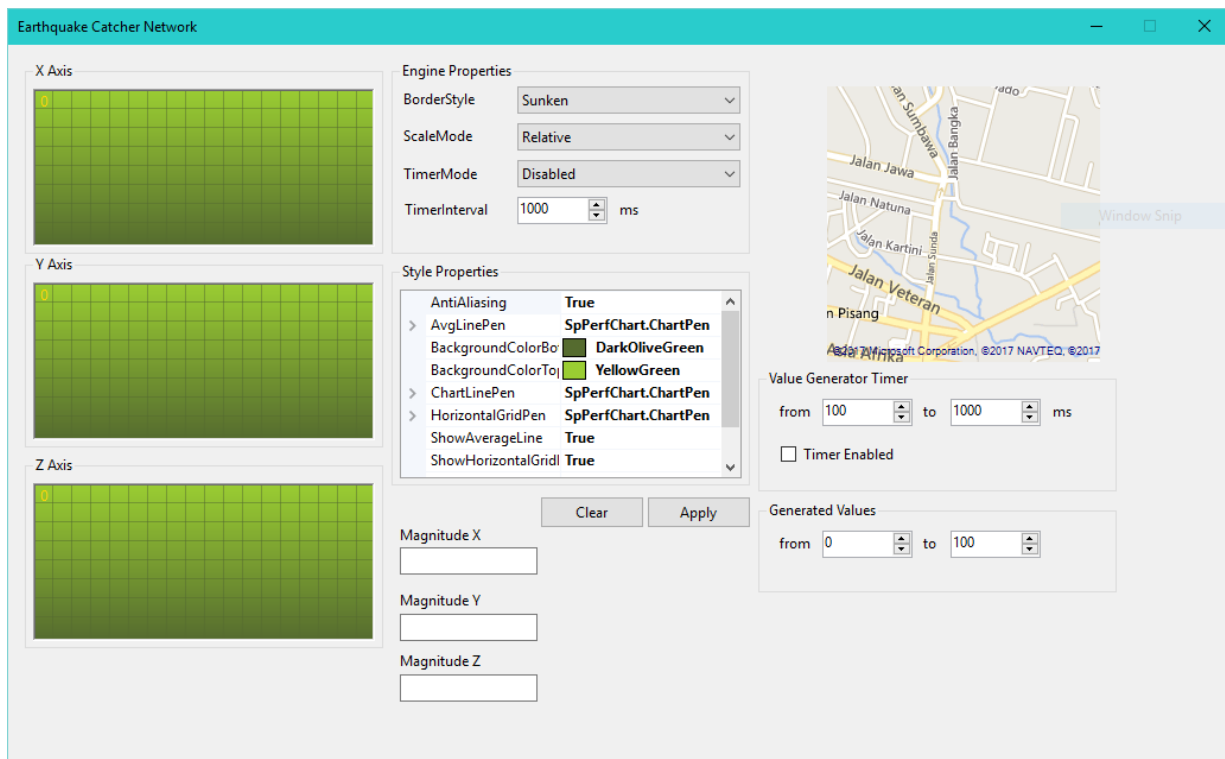
Kemudian dilakukan parsing dengan menambahkan syntax berikut ini:

```

data accelReport = JsonConvert.DeserializeObject<data>(message);

```

IV.1.6 Implementasi GUI untuk pengguna umum



Gambar IV-10 tampilan GUI untuk pengguna umum yang dapat mengamati akuisisi data pada sensor gempa

Tampilan ini adalah tampilan awal dari antarmuka pengguna grafis sistem detektor gempa ini. Hanya data dari satu sensor yang dimasukkan ke dalam proses yang ada dibalik tampilan ini. Data tersebut berupa percepatan ke arah x,y,dan z, serta data posisi yang ditampilkan melalui peta di pojok kanan atas. Sensor akan ditandai dengan pin berwarna biru di peta tersebut. Bila ada lebih dari satu sensor yang terhubung, maka akan ada banyak pin biru di peta. Tombol dan bar lainnya digunakan untuk melakukan pengaturan terhadap grafik data percepatan yang ditampilkan pada kotak hijau di bagian kiri. Selain itu, magnituda dari getaran pada masing-masing sumbu juga ditampilkan secara kuantitatif di bagian bawah.

Peta tersebut disediakan oleh Bing secara gratis, melalui proyek gMap.Net, yaitu paket peta dan gps yang khusus dibuat untuk platform .NET, termasuk didalamnya C#. Program akan menerima data lokasi dari masing-masing sensor berupa lintang dan bujur, dan program lalu akan membuat sebuah pin biru di peta yang lokasinya sesuai dengan lintang dan bujur yang diterima. Pin dapat dibuat pada peta dengan jumlah yang tidak terbatas. Pengguna juga dapat menggeser peta sehingga bagian lain dari bumi yang terlihat pada layar, dan juga melakukan pembesaran atau pengecilan.

Tampilan grafik yang merepresentasikan data percepatan yang dikirim sensor bisa diatur sebagai berikut. Tiga pilihan yang terletak pada bagian atas berfungsi untuk mengatur *engine* dari grafik. Sebagai contoh, ada yang mengatur kecepatan grafik bergeser ke kiri, ada yang mengatur pembesaran/pengecilan skala grafik secara otomatis, dan juga mengatur jenis dari perbatasan grafik, apakah absolut atau bergerak mengikuti nilai grafik yang sekarang.

IV.2 Pengujian Messaging Server dan Basis data

Aspek yang ditinjau pada pengujian Messaging Server adalah pesan dapat diterima oleh Messaging Server dengan baik kemudian dapat meneruskan pesan tersebut ke consumer yang sesuai. Harus dilakukan sebuah pengaturan pada pengirim sehingga sebuah pesan tidak akan bisa dikonsumsi oleh consumer yang tidak sesuai. Setelah pesan diterima oleh consumer, pesan tersebut akan dikirimkan menuju sebuah basis data. Data yang telah disimpan dapat digunakan untuk diolah datanya.

Pengujian dilakukan dengan mengirimkan pesan dari sensor maupun dari program penguji. Pesan kemudian akan dikirimkan menuju Messaging Server. Supaya mengetahui bahwa pesan dapat terkirim, maka harus ada consumer yang menjalankan perintah consume Messaging Server. Apabila pesan dapat di consume, maka proses pengiriman dari sensor menuju Messaging Server berjalan dengan lancar.

IV.2.1 Pengujian Messaging Server RabbitMQ

Sensor mengirimkan sebuah pesan setiap 1 detik menuju Messaging Server dengan jaringan WiFi yang sudah di set. Pesan ini merupakan pesan informasi geospasial dengan format JSON. Penguji melakukan debugging pada sensor dengan membaca komunikasi serial sehingga mengetahui apa yang dikirimkan oleh sensor.

Berikut ini data yang dihasilkan oleh sensor saat berhasil mengirimkan sebuah pesan beserta tampilan debugging dari pesan yang dihasilkan:



```
publish success
{ "pointTime": "2000-0-0T0:0:0Z", "timeZone": "Asia/Jakarta", "interval": "1000", "clientID": "ECI"
publish success
{ "pointTime": "2000-0-0T0:0:0Z", "timeZone": "Asia/Jakarta", "interval": "1000", "clientID": "ECI"
publish success
{ "pointTime": "2000-0-0T0:0:0Z", "timeZone": "Asia/Jakarta", "interval": "1000", "clientID": "ECI"
publish success
{ "pointTime": "2000-0-0T0:0:0Z", "timeZone": "Asia/Jakarta", "interval": "1000", "clientID": "ECI"
publish success
{ "pointTime": "2000-0-0T0:0:0Z", "timeZone": "Asia/Jakarta", "interval": "1000", "clientID": "ECI"
publish success
{ "pointTime": "2000-0-0T0:0:0Z", "timeZone": "Asia/Jakarta", "interval": "1000", "clientID": "ECI"
Soft WDT reset
```

Gambar IV-11 hasil dari pengujian pengiriman pesan yang dilakukan oleh serial monitor IDE Arduino

Kemudian isi pesan yang dikirimkan adalah sebagai berikut:

```
{
  "pointTime": "2000-0-
0T0:0:0Z", "timeZone": "Asia/Jakarta", "interval": "1000", "clientID": "ECN-1", "geojson"
: {"geometry": {"type": "Point", "coordinates": [
6.89, 107.61]}, "properties": {"name": "ITB"}}, "accelerations": [{"x": -2.0661, "y":
4.8328, "z": 5.2528}, {"x": -2.0859, "y": 4.8904, "z": 5.3096}, {"x": -2.1068, "y":
4.9404, "z": 5.3676}, {"x": -2.1281, "y": 4.9897, "z": 5.4209}, {"x": -2.1516, "y":
5.0375, "z": 5.4726}, {"x": -2.1710, "y": 5.0862, "z": 5.5196}, {"x": -2.1899, "y":
5.1320, "z": 5.5701}, {"x": -2.2104, "y": 5.1750, "z": 5.6138}, {"x": -2.2282, "y":
5.2156, "z": 5.6570}, {"x": -2.2458, "y": 5.2601, "z": 5.7015}, {"x": -2.2648, "y":
5.3003, "z": 5.7426}, {"x": -2.2820, "y": 5.3374, "z": 5.7857}, {"x": -2.2963, "y":
5.3774, "z": 5.8231}, {"x": -2.3130, "y": 5.4122, "z": 5.8600}, {"x": -2.3297, "y":
5.4444, "z": 5.8985}, {"x": -2.3458, "y": 5.4768, "z": 5.9321}, {"x": -2.3593, "y":
5.5074, "z": 5.9641}, {"x": -2.3726, "y": 5.5376, "z": 5.9999}, {"x": -2.3879, "y":
5.5690, "z": 6.0290}, {"x": -2.3978, "y": 5.5987, "z": 6.0641}, {"x": -2.4110, "y":
5.6320, "z": 6.0913}, {"x": -2.4203, "y": 5.6601, "z": 6.1175}, {"x": -2.4339, "y":
5.6857, "z": 6.1457}, {"x": -2.4411, "y": 5.7115, "z": 6.1718}, {"x": -2.4504, "y":
5.7361, "z": 6.1995}, {"x": -2.4614, "y": 5.7576, "z": 6.2212}, {"x": -2.4707, "y":
5.7819, "z": 6.2485}, {"x": -2.4815, "y": 5.8033, "z": 6.2715}, {"x": -2.4923, "y":
5.8249, "z": 6.2971}, {"x": -2.5006, "y": 5.8451, "z": 6.3182}, {"x": -2.5088, "y":
5.8637, "z": 6.3422}, {"x": -2.5156, "y": 5.8826, "z": 6.3636}, {"x": -2.5221, "y":
5.9009, "z": 6.3845}, {"x": -2.5288, "y": 5.9186, "z": 6.4063}, {"x": -2.5368, "y":
5.9352, "z": 6.4273}, {"x": -2.5464, "y": 5.9518, "z": 6.4467}, {"x": -2.5534, "y":
5.9692, "z": 6.4634}, {"x": -2.5586, "y": 5.9802, "z": 6.4793}, {"x": -2.5656, "y":
5.9951, "z": 6.4893}, {"x": -2.5717, "y": 6.0085, "z": 6.5053}]]}
```

Pesan yang terkirim oleh sensor memiliki format yang sesuai dan data yang valid. Validitas pesan ini akan dicek kemudian oleh consumer.

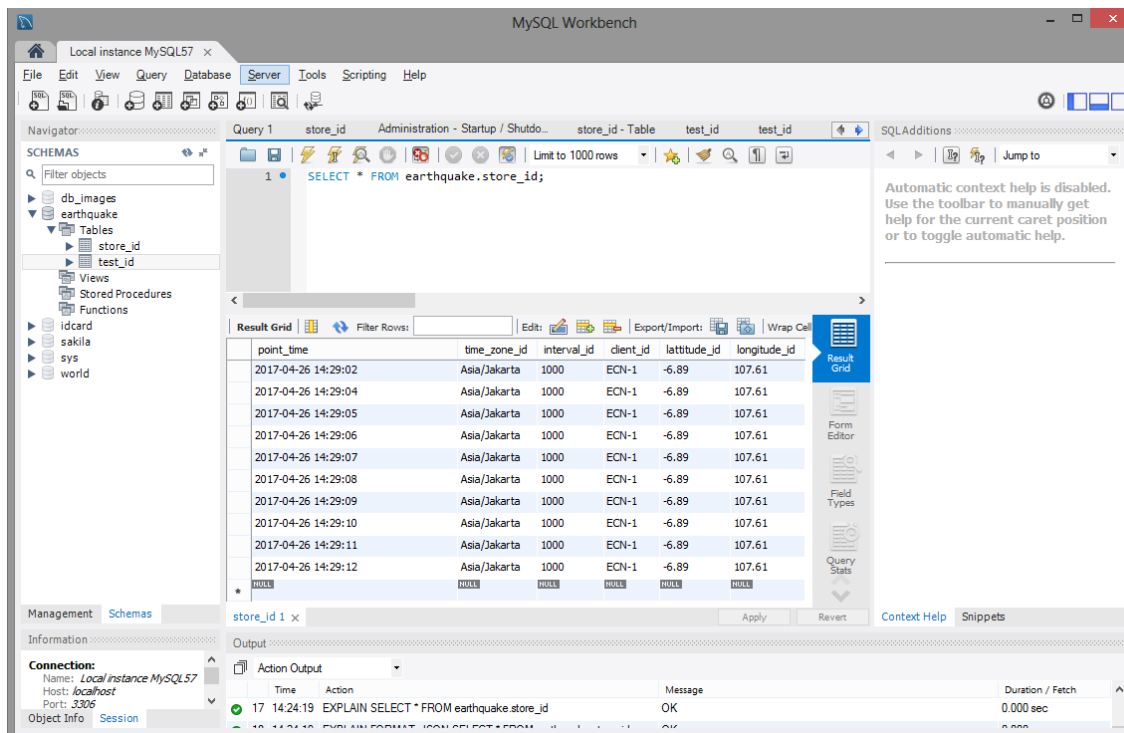
IV.2.2 Pengujian Basis data MySQL

Langkah selanjutnya, akan dicek apabila pesan tersebut berhasil diunduh dari Messaging Server. Ketika program consume oleh console C# dijalankan, maka hasilnya adalah sebagai berikut:

```
file:///C:/Users/X450JB/Desktop/Database/Database/bin/Debug/Database.EXE
Queue Declare Emergency GUI
Already BasicConsume
2017-04-26 14:27:31
Queue Declare Emergency GUI
Already BasicConsume
[1] Received { "pointTime": "2000-0-0T0:0:0Z", "timeZone": "Asia/Jakarta", "interval": "1000", "clientID": "ECN-1", "geojson": { "geometry": { "type": "Point", "coordinates": [ -6.89, 107.61 ] }, "properties": { "name": "ITB" }, "accelerations": [ { "x": -2.6833, "y": 6.3658, "z": 7.0531 }, { "x": -2.6821, "y": 6.3661, "z": 7.0531 }, { "x": -2.6801, "y": 6.3652, "z": 7.0496 }, { "x": -2.6790, "y": 6.3639, "z": 7.0497 }, { "x": -2.6773, "y": 6.3652, "z": 7.0527 }, { "x": -2.6770, "y": 6.3659, "z": 7.0548 }, { "x": -2.6787, "y": 6.3653, "z": 7.0527 }, { "x": -2.6760, "y": 6.3664, "z": 7.0510 }, { "x": -2.6755, "y": 6.3678, "z": 7.0514 }, { "x": -2.6736, "y": 6.3695, "z": 7.0526 }, { "x": -2.6727, "y": 6.3698, "z": 7.0509 }, { "x": -2.6731, "y": 6.3709, "z": 7.0499 }, { "x": -2.6748, "y": 6.3696, "z": 7.0513 }, { "x": -2.6740, "y": 6.3678, "z": 7.0528 }, { "x": -2.6735, "y": 6.3664, "z": 7.0565 }, { "x": -2.6715, "y": 6.3644, "z": 7.0574 }, { "x": -2.6716, "y": 6.3622, "z": 7.0576 }, { "x": -2.6697, "y": 6.3634, "z": 7.0555 }, { "x": -2.6713, "y": 6.3650, "z": 7.0564 }, { "x": -2.6724, "y": 6.3655, "z": 7.0586 }, { "x": -2.6728, "y": 6.3644, "z": 7.0573 }, { "x": -2.6733, "y": 6.3655, "z": 7.0554 }, { "x": -2.6743, "y": 6.3655, "z": 7.0574 }, { "x": -2.6726, "y": 6.3666, "z": 7.0578 }, { "x": -2.6730, "y": 6.3660, "z": 7.0542 }, { "x": -2.6716, "y": 6.3649, "z": 7.0543 }, { "x": -2.6727, "y": 6.3645, "z": 7.0546 }, { "x": -2.6738, "y": 6.3666, "z": 7.0531 }, { "x": -2.6736, "y": 6.3661, "z": 7.0501 }, { "x": -2.6738, "y": 6.3655, "z": 7.0512 }, { "x": -2.6740, "y": 6.3645, "z": 7.0508 }, { "x": -2.6754, "y": 6.3647, "z": 7.0511 }, { "x": -2.6747, "y": 6.3663, "z": 7.0539 }, { "x": -2.6747, "y": 6.3661, "z": 7.0555 }, { "x": -2.6715, "y": 6.3672, "z": 7.0552 }, { "x": -2.6715, "y": 6.3663, "z": 7.0539 }, { "x": -2.6737, "y": 6.3657, "z": 7.0538 }, { "x": -2.6725, "y": 6.3670, "z": 7.0523 }, { "x": -2.6735, "y": 6.3696, "z": 7.0527 }, { "x": -2.6750, "y": 6.3688, "z": 7.0542 } ] }
```

Gambar IV-12 hasil pengujian yang dilakukan oleh program console C# pada proses pemerolehan data dari RabbitMQ

Setelah melakukan consume, maka program consumer akan mengirimkan pesan menuju MySQL. Hasilnya adalah sebagai berikut:

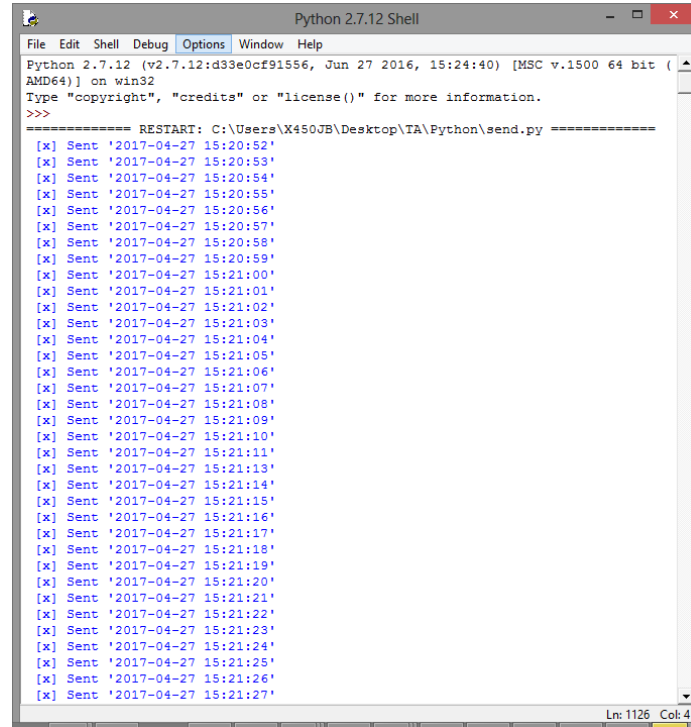


Gambar IV-13 tampilan basis data seismik pada perangkat lunak MySQL

Data yang tersimpan ini akan digunakan untuk pengolahan data gempa.

IV.2.3 Pengujian Latency Jaringan Sistem

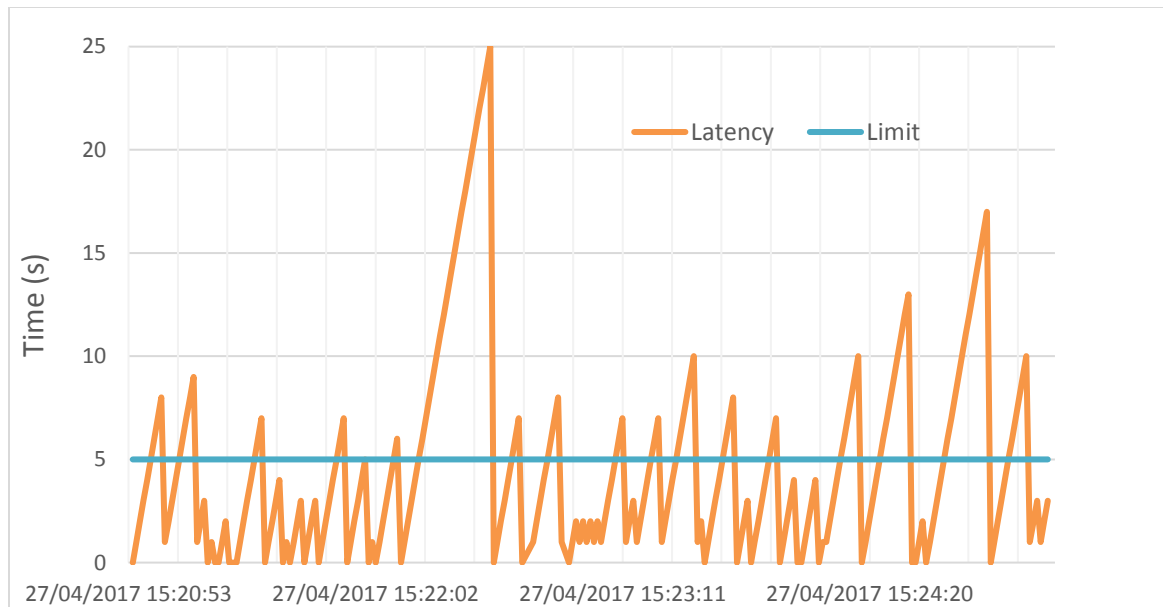
Namun selain kemampuan untuk dapat mengirimkan pesan, sistem ini juga harus handal untuk dapat berjalan selama mungkin. Oleh karena itu, keaslian data secara aktual juga harus diperhitungkan. Pengujian dilakukan dengan mengirimkan sebuah pesan Timestamp dari sebuah producer kemudian mencocokkan Timestamp tsb dengan Timestamp consumer. Hasil pengiriman Timestamp producer dengan menggunakan Python adalah sebagai berikut:



```
Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
Python 2.7.12 (v2.7.12:d39e0cf91556, Jun 27 2016, 15:24:40) [MSC v.1500 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\X450JB\Desktop\TA\Python\send.py =====
[x] Sent '2017-04-27 15:20:52'
[x] Sent '2017-04-27 15:20:53'
[x] Sent '2017-04-27 15:20:54'
[x] Sent '2017-04-27 15:20:55'
[x] Sent '2017-04-27 15:20:56'
[x] Sent '2017-04-27 15:20:57'
[x] Sent '2017-04-27 15:20:58'
[x] Sent '2017-04-27 15:20:59'
[x] Sent '2017-04-27 15:21:00'
[x] Sent '2017-04-27 15:21:01'
[x] Sent '2017-04-27 15:21:02'
[x] Sent '2017-04-27 15:21:03'
[x] Sent '2017-04-27 15:21:04'
[x] Sent '2017-04-27 15:21:05'
[x] Sent '2017-04-27 15:21:06'
[x] Sent '2017-04-27 15:21:07'
[x] Sent '2017-04-27 15:21:08'
[x] Sent '2017-04-27 15:21:09'
[x] Sent '2017-04-27 15:21:10'
[x] Sent '2017-04-27 15:21:11'
[x] Sent '2017-04-27 15:21:13'
[x] Sent '2017-04-27 15:21:14'
[x] Sent '2017-04-27 15:21:15'
[x] Sent '2017-04-27 15:21:16'
[x] Sent '2017-04-27 15:21:17'
[x] Sent '2017-04-27 15:21:18'
[x] Sent '2017-04-27 15:21:19'
[x] Sent '2017-04-27 15:21:20'
[x] Sent '2017-04-27 15:21:21'
[x] Sent '2017-04-27 15:21:22'
[x] Sent '2017-04-27 15:21:23'
[x] Sent '2017-04-27 15:21:24'
[x] Sent '2017-04-27 15:21:25'
[x] Sent '2017-04-27 15:21:26'
[x] Sent '2017-04-27 15:21:27'
```

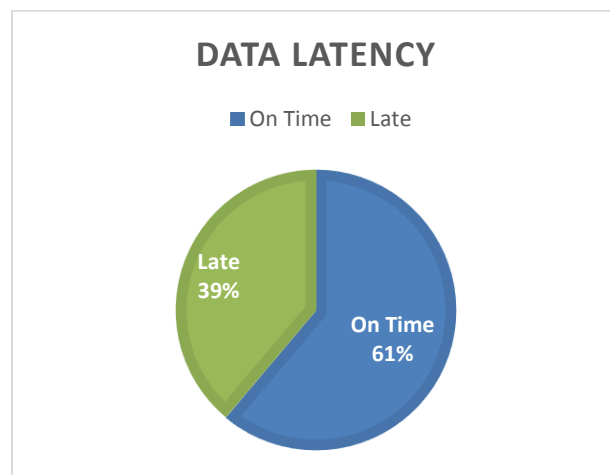
Gambar IV-14 hasil pengiriman data yang dilakukan oleh IDE Python untuk proses pengujian latency data

Kemudian pesan di dalam Messaging Server di consume oleh console C# dan digabungkan dengan Timestamp consumer. Kedua time stamp ini kemudian disimpan ke dalam Basis data MySQL dan di export datanya untuk pengolahan lebih lanjut. Hasil pengolahannya adalah sebagai berikut:



Gambar IV-15 grafik latency data, dimana garis oranye adalah ketepatan waktu data sampai ke server, dan garis biru adalah batas toleransi data untuk tidak terlambat

Artinya terdapat beberapa buah data yang tidak real time karena terlambat untuk sampai ke pengolahan server. Setelah diolah datanya, didapatkan hasil sebagai berikut:



Gambar IV-16 persentase kemungkinan data mengalami latency yang tidak dapat ditoleransi

Terdapat peluang sebesar 61% bahwa data yang dikirimkan adalah data real-time ketika melalui sistem jaringan.

BAB V

KESIMPULAN

V.1 Kesimpulan

Sistem detektor gempa membutuhkan sebuah sistem jaringan untuk menyimpan datanya ke sebuah basis data dan juga dapat menampilkan GUI yang dapat melakukan pengawasan terhadap akuisisi data seismik sehingga dapat dilihat bahwa apakah terjadi getaran seismik pada permukaan tanah yang dapat mengakibatkan gempa. Penggunaan RabbitMQ bertujuan untuk menjembatani perpindahan data dari sensor gempa dengan jumlah yang banyak hingga dapat sampai ke setiap konsumen yang membutuhkan, baik itu sebuah server basis data maupun perangkat lunak pengolahan data gempa. Pengaturan RabbitMQ disesuaikan pada perangkat lunak sesuai dengan bahasa pemrograman yang digunakan. Pengaturan basis data MySQL juga dilakukan agar perangkat lunak dapat langsung mengirim data seismik dengan responsif. Setelah melakukan perancangan dan implementasi, dilakukan uji coba dan bisa diambil beberapa kesimpulan. Sistem jaringan menggunakan RabbitMQ dapat diimplementasikan dan diuji dengan baik dan sesuai dengan yang diinginkan. Namun pada proses konsumsi data, masih terdapat keterlambatan pemerolehan data sehingga dapat mengakibatkan kerugian dalam segi ketepatan data.

V.2 Saran

Konfigurasi server pengirim data RabbitMQ harus dapat didokumentasikan lebih baik karena untuk dapat mendukung sistem multi wahana. Perancangan sensor gempa juga harus mempertimbangkan aspek dokumentasi yang lengkap sehingga kesulitan pada aktualisasi pengiriman data dapat diatasi dengan mudah. Untuk kedepannya, diharapkan konsumsi data dapat dilakukan secara aktual tanpa mengalami keterlambatan (latency).

DAFTAR PUSTAKA

- [1] "Designing A User Interface (Visual C#)". Msdn.microsoft.com. N.p., 2017. Web. 9 May 2017.
- [2] DuBois, Paul. Mysql, Fourth Edition. 1st ed. Addison-Wesley Professional, 2008. Print.
- [3] "Graphical User Interface". En.wikipedia.org. N.p., 2017. Web. 9 May 2017.
- [4] "Mysql :: Mysql 5.7 Reference Manual :: 1.3.1 What Is Mysql?". Dev.mysql.com. N.p., 2017. Web. 9 May 2017.
- [5] Perkins, Benjamin, Jacob Vibe Hammer, and Jon D Reid. Beginning C# 6 Programming with Visual Studio® 2015. 1st ed. Indianapolis, IN: Wrox, A Wiley Brand, 2016. Print.
- [6] "Rabbitmq - Messaging That Just Works". Rabbitmq.com. N.p., 2017. Web. 9 May 2017.
- [7] "Simple Performance Chart - Codeproject". Codeproject.com. N.p., 2017. Web. 9 May 2017.
- [8] Videla, Alvaro, and Jason J. W Williams. Rabbitmq In Action: Distributed Messaging For Everyone. 1st ed. Manning Publications, 2012. Print.

LAMPIRAN