TUGAS AKHIR - KI091391

DETEKSI DAN VALIDASI INFORMASI GEMPA SECARA *REAL-TIME* BERBASIS *SOCIAL SENSOR* DENGAN TWITTER

Ryan Dwi Cahyo Nugroho
NRP 5110100046

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

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2014

FINAL PROJECT – KI091391

EARTHQUAKE DETECTION AND VALIDATION OF INFORMATION IN REAL-TIME BASED ON SOCIAL SENSOR WITH TWITTER

Ryan Dwi Cahyo Nugroho
NRP 5110100046

Advisor
Waskitho Wibisono, S.Kom., M.Eng., Ph.D.
Hudan Studiawan, S.Kom., M.Kom.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2014

Deteksi dan Validasi Informasi Gempa Secara *Real-Time* Berbasis *Social Sensor* dengan Twitter

Nama Mahasiswa	**: Ryan Dwi Cahyo Nugroho**
NRP	**: 5110100046**
Jurusan	**: Teknik Informatika FTIf-ITS**
Dosen Pembimbing 1	**: Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**
Dosen Pembimbing 2	**: Hudan Studiawan, S.Kom., M.Kom.**

ABSTRAK

Twitter, merupakan sebuah layanan *microblogging* yang telah mendapatkan banyak perhatian beberapa waktu belakangan ini. Salah satu karakteristik dari Twitter adalah sifatnya yang *real-time*. Sebagai contoh, bila terjadi gempa, maka orang-orang akan segera membuat posting Twitter (*tweet*) mengenai hal-hal yang berkaitan dengan gempa tersebut. Hal ini membuat Twitter dapat digunakan sebagai *social sensor* untuk mendeteksi terjadinya gempa secara *real-time*. Namun tidak semua informasi yang didapat dari *social sensor* memiliki nilai yang valid. Maka dibutuhkan metode untuk melakukan validasi terhadap kebenaran berita tersebut.

Guna melakukan validasi informasi gempa yang didapat dari *social sensor*, diperlukan integrasi dengan situs *web* yang menangani informasi gempa. Situs *web* Earthquake Hazards Program merupakan situs yang menangani informasi gempa di seluruh dunia secara *real-time*. Perangkat lunak melakukan pengambilan data informasi gempa yang didapat dari situs *web* menggunakan JSON *parser*. Data tersebut diolah untuk kemudian digunakan untuk melakukan validasi terhadap data informasi gempa yang didapat dari *social sensor*.

Berdasarkan data informasi gempa *social sensor* yang telah diolah, dapat dihitung *level* konfidensi informasi gempa untuk menunjukkan tingkat konfidensi informasi gempa berdasarkan situs web Earthquake Hazards Program. Perangkat lunak telah diuji coba fungsionalitasnya. Pengujian dilakukan melalui beberapa skenario yang telah ditentukan sebelumnya. Hasil pengujian menunjukkan perangkat lunak berhasil menampilkan informasi gempa beserta *level* konfidensi dari informasi gempa.

Kata kunci: **Twitter, Earthquake Hazards Program, Real-Time, Social Sensor.**

Earthquake Detection And Validation Of Information In Real-Time Based On Social *Sensor* With Twitter

Student Name	**: Ryan Dwi Cahyo Nugroho**
Student ID	**: 5110100046**
Major	**: Teknik Informatika FTIf-ITS**
Advisor 1	**: Waskitho Wibisono, S.Kom., M.Eng., Ph.D.**
Advisor 2	**: Hudan Studiawan, S.Kom., M.Kom.**

ABSTRACT

Twitter, is a microblogging service that has gained a lot of attention recently. One of characteristics of Twitter is its real-time nature. For example, if there were an earthquake, people would tend to make Twitter posts about things related to that earthquake. This thing makes Twitter can be used as seocial sensor to detect an earthquake in real-time. However not all of the informations obtained from the social sensor has a valid value. So it needs a method to validate the truth of those informations.

In order to perform validation of the earthquake information obtained from social sensor, it requires an integration with web site related to earthquake information. Earthquake Hazards Program is a web site that handles earthquake information from all over the world in real-time. This software retrieve earthquake information data from the website using JSON Parser. The data is processed and used to peform validation againsts earthquake information data obtained from the social sensor.

Based on earthquake information obtained from social sensor that has been processed, level of confidentiality of earthquake information can be calculated. This software has been tested functionally. The experiment is done through some predefined scenarios. The result could successfully show earthquake information and its level of confidentiality.

Keyword: **Twitter, Earthquake Hazards Program, Real-Time, Social Sensor.**

LEMBAR PENGESAHAN

Deteksi dan Validasi Informasi Gempa Secara *Real-Time* Berbasis *Social Sensor* dengan Twitter

TUGAS AKHIR

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

Oleh:
RYAN DWI CAHYO NUGROHO
NRP: 5110100046

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Waskitho Wibisono, S.Kom., M.Eng., Ph.D.
NIP: 19741022200003100 (Pembimbing 1) .....

Hudan Studiawan, S.Kom., M.Kom
NIP: 198705112012121003 (Pembimbing 2) .....

SURABAYA
JULI 2014

KATA PENGANTAR

Alhamdulillahirabil'alamin, segala puji bagi Allah Subhanahu Wata'alla, yang telah melimpahkan segala karunia dan rahmat-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul : "**Deteksi dan Validasi Informasi Gempa Secara *Real-Time* Berbasis *Social Sensor* Dengan Twitter***"***

Dalam pelaksanaan dan pembuatan Tugas Akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Allah SWT atas limpahan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir ini dengan baik.
2. Almarhum Ibu Riyantini, yang tak pernah lelah memberikan dukungan kepada penulis semasa hidupnya.
3. Bapak Setiawan, Mas Ian, Dek Dita, dan keluarga yang selalu memberikan dukungan penuh untuk menyelesaikan Tugas Akhir ini.
4. Bapak Waskitho Wibisono dan Bapak Hudan Studiawan selaku dosen pembimbing pertama dan kedua yang telah bersedia meluangkan waktu serta memberikan kepercayaan, dukungan, bimbingan kepada penulis.
5. Bapak, Ibu dosen Jurusan Teknik Informatika ITS yang telah banyak memberikan ilmu dan bimbingan yang tak ternilai harganya bagi penulis.
6. Seluruh staf dan karyawan FTIf ITS yang banyak memberikan kelancaran administrasi akademik kepada penulis.
7. Helena Nadya Pratiwi, yang selalu memberikan semangat dan dukungan serta masukan kepada penulis.
8. Keluarga Griya Asri, Dicky, Aji, Dimas, Wido, Haryo, Kessya, Lala, Wildhan, dan Fitri yang selalu memberikan semangat dan masukan kepada penulis.

9. Akbar dan Zuhriyan yang selalu memberikan dukungan kepada penulis.
10. Seluruh teman Teknik Informatika 2010 yang telah bersama selama empat tahun atas ilmu, saran, dan dukungan terhadap pengerjaan Tugas Akhir ini.
11. Serta pihak-pihak lain yang tidak dapat disebutkan disini yang telah banyak membantu penulis dalam penyusunan Tugas Akhir ini.

Penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan dalam penulisan Tugas Akhir ini, maka penulis mengharapkan saran dan kritik yang membangun dari pembaca.

Surabaya, Juli 2014

Ryan Dwi Cahyo Nugroho

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR KODE SUMBER

DAFTAR TABEL

BAB I
PENDAHULUAN

Bab ini berisi penjelasan mengenai garis besar penulisan Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1. Latar Belakang

Kemajuan teknologi, khususnya di bidang media sosial, terus berkembang pesat belakangan ini. Salah satu media sosial yang mendapat banyak perhatian adalah Twitter[1]. Twitter adalah sebuah layanan jaringan sosial *online* yang digunakan oleh jutaan orang di seluruh dunia. Sebagai sebuah layanan jaringan sosial, Twitter memungkinkan pengguna untuk berinteraksi dengan pengguna lain. Sebuah *posting* Twitter, disebut *tweet*, merupakan sebuah pesan yang ditulis ke sebuah lini masa. Sebuah *tweet* berisi informasi mengenai *username*, isi *tweet*, lokasi, dan waktu *posting tweet*. Seorang pengguna dapat mengikuti lini masa pengguna lain, dan semua pengikut akun itu dapat melihat lini masa dari pengguna tersebut. Twitter juga dikategorikan sebagai layanan *microblogging*. *Microblogging* merupakan sebuah cara untuk melakukan *blogging* namun dengan batasan tertentu, yaitu hanya memungkinkan 140 karakter dalam sebuah *tweet*.

Salah satu karakteristik dari Twitter adalah sifatnya yang *real-time*. *Real-time* yang dimaksud adalah keadaan dengan waktu yang bisa menyamai dengan proses sebenarnya (di dunia nyata) yang sedang terjadi. Salah satu alasan Twitter memiliki sifat *real-time* adalah Twitter dapat diakses dari mana saja dan kapan saja selama perangkat terhubung ke jaringan internet, baik menggunakan PC (*Personal Computer*) melalui antarmuka situs

[1] Twitter. https://www.twitter.com

web, atau perangkat bergerak melalui SMS (*Short Message Service*) dan aplikasi perangkat bergerak. Pengguna Twitter biasanya melakukan *posting tweet* beberapa kali dalam sehari, sehingga pengikut mereka dapat mengetahui apa yang terjadi dan apa yang mereka lakukan secara *real-time*. Tidak jarang pengguna Twitter menggunakan Twitter untuk mengabarkan situasi terkini dari keadaan di sekitar mereka. Sebagai contoh, ketika terjadi gempa, pengguna Twitter akan membuat *tweet* mengenai gempa tersebut. Karena sifatnya yang *real-time*, waktu diantara kejadian dan waktu *tweet* tidak memiliki jarak yang signifikan. Alasan ini yang membuat Twitter dapat digunakan sebagai sensor untuk mengetahui keadaan di sekitar dan mengubahnya menjadi informasi. Sensor ini disebut *social sensor*. Meskipun *social sensor* memiliki sifat yang *real-time*, tidak semua informasi yang dihasilkan oleh *social sensor* memiliki nilai yang mutlak benar. Sebagai contoh, bila seorang pengguna Twitter melakukan pembaruan status mengenai gempa di Surabaya, namun yang sebenarnya terjadi adalah gempa terjadi di Jakarta, maka informasi ini termasuk informasi yang tidak benar/tidak valid. Untuk itulah, diperlukan suatu metode untuk melakukan validasi terhadap informasi gempa ini.

Earthquake Hazards Program[2] merupakan sebuah situs *web* yang menyediakan data *feed* mengenai informasi gempa yang terjadi di seluruh dunia. Data *feed* informasi gempa dari Earthquake Hazards Program selalu terbaharui setiap detik sehingga dapat dikatakan *real-time*. Alasan tersebut membuat data *feed* informasi gempa ini dapat digunakan sebagai acuan untuk melakukan validasi terhadap informasi gempa yang terdapat pada Twitter.

Dalam Tugas Akhir ini penulis mengintegrasikan informasi data *feed* gempa yang berasal dari situs *web* Earthquake Hazards Program dengan informasi gempa yang berasal dari *social sensor* yaitu Twitter. Data *tweet* gempa yang didapat dari *social*

[2] Earthquake Hazards Program. http://usgs.gov.us/

sensor diolah dan divalidasi dengan informasi gempa dari situs *web* Earthquake Hazards program untuk menghitung *level* konfidensi informasi gempa. Apabila *level* konfidensi informasi gempa dari Twitter semakin tinggi, maka tingkat konfidensi informasi terhadap situs *web* tersebut akan semakin tinggi, sehingga informasi gempa semakin valid. Begitu pula dengan sebaliknya, apabila *level* konfidensi informasi gempa dari Twitter semakin rendah, maka tingkat konfidensi informasi terhadap situs *web* tersebut semakin rendah.

1.2. Rumusan Permasalahan

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

1. Bagaimana cara mendapatkan informasi gempa dari situs *web* Earthquake Hazards Program?
2. Bagaimana cara mendapatkan informasi gempa dari *social sensor* yaitu Twitter?
3. Bagaimana cara mengintegrasi informasi gempa yang didapat dari Earthquake Hazards Program dengan informasi gempa dari *social sensor* yaitu Twitter?
4. Bagaimana cara melakukan validasi terhadap kebenaran informasi gempa dari Twitter?
5. Bagaimana cara menentukan *level* konfidensi informasi gempa?

1.3. Batasan Permasalahan

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, di antaranya sebagai berikut:

1. Sampel data yang diambil adalah gempa yang terjadi di Indonesia.
2. Bahasa yang digunakan sebagai kata kunci adalah bahasa Indonesia.

3. Data gempa yang digunakan adalah data gempa yang terakhir terjadi dalam satu hari.
4. Data yang digunakan berupa koordinat *geolocation* dalam *latitude* dan *longitude* dari *tweet* dan data koordinat gempa dari situs *web* Earthquake Hazards Program.
5. Data *tweet* yang digunakan hanya data *tweet* yang memiliki koordinat *geolocation* dalam *latitude* dan *longitude*.
6. Perangkat lunak yang dibuat adalah berupa situs *web* dan perangkat lunak *smartphone* berbasis Android.

1.4. Tujuan dan Manfaat

Tujuan dari pembuatan Tugas Akhir ini adalah membuat perangkat lunak yang dapat mengintegrasi *social sensor* dalam hal ini Twitter dengan situs *web* Earthquake Hazards Program dan melakukan validasi informasi gempa dari *social sensor* dengan menghitung *level* konfidensi informasi gempa tersebut.

Manfaat yang bisa didapat dari pembuatan Tugas Akhir ini adalah sebagai berikut:

1. Mengintegrasikan informasi gempa dari *social sensor* dengan data dari situs *web* Earthquake Hazards Program.
2. Mendapatkan informasi gempa berdasarkan *social sensor* yang valid dan memiliki tingkat kebenaran tinggi.

1.5. Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini yaitu:

1. Penyusunan proposal Tugas Akhir

   Proposal Tugas Akhir ini berisi tentang penjabaran topik dan penjelasan mengenai Deteksi dan Validasi Informasi Gempa Secara *Real-Time* Berbasis *Social Sensor* dengan Twitter, bagaimana cara mengerjakan dan apa saja hal-hal

yang dibutuhkan untuk membangun aplikasi ini.

2. Studi literatur

   Studi literatur adalah tahap pengumpulan informasi dan pembelajaran beberapa hal-hal yang diperlukan untuk menunjang pengembangan aplikasi ini, diantaranya adalah sebagai berikut:

   1. Mengintegrasikan informasi gempa dari Twitter dengan situs *web* Earthquake Hazards Program.
   2. Menghitung *level* konfidensi informasi gempa dari *social sensor*.

   Referensi yang menunjang pengembangan perangkat lunak ini, seperti GSON, Twitter API, *geolocation*, rumus Haversine dan Google Maps API.

3. Analisis dan desain perangkat lunak

   Analisis dan desain perangkat lunak merupakan pengkajian lebih lanjut terhadap literatur agar informasi yang dihasilkan akurat dan tepat. Tahapan ini merupakan tahap yang penting dimana bentuk awal perangkat lunak didefinisikan. Selain itu pada tahapan ini dibuat *prototype system*, yang merupakan rancangan dasar dari perangkat lunak yang dibuat.

4. Implementasi perangkat lunak

   Implementasi merupakan tahap pengembangan perangkat lunak berdasarkan rancangan sistem yang telah dibuat sebelumnya. Tahap ini merealisasikan hasil dari tahapan sebelumnya, sehingga menjadi sebuah perangkat lunak yang sesuai dengan yang telah direncanakan.

5. Pengujian dan evaluasi

   Pada tahap ini penulis melakukan uji coba terhadap perangkat lunak yang telah dibuat. Tujuan uji coba perangkat lunak ini adalah untuk menemukan kesalahan yang mungkin terjadi pada saat pengembangan perangkat lunak.

6. Penyusunan buku Tugas Akhir

   Pada tahap ini disusun buku sebagai dokumentasi dari pelaksanaan Tugas Akhir. Buku ini mencakup seluruh konsep, teori, analisis, desain, implementasi, dan hasil Tugas Akhir yang telah dikerjakan.

1.6. Sistematika Penulisan

Secara garis besar, buku Tugas Akhir ini terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

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

Bab II Dasar Teori

Bab ini berisi penjelasan secara umum mengenai beberapa teori penunjang yang mendukung pengembangan Tugas Akhir ini.

Bab III Analisis dan Perancangan Sistem

Bab ini berisi penjelasan mengenai diagram aplikasi, arsitektur perangkat lunak, diagram alir perangkat lunak, dan desain antarmuka perangkat lunak yang dibuat.

Bab IV Implementasi Perangkat Lunak

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

Bab V Pengujian dan Evaluasi

Bab ini berisi penjelasan kemampuan perangkat lunak dengan melakukan beberapa pengujian yaitu pengujian fungsionalitas dan pengujian performa dalam beberapa skenario.

Bab VI Kesimpulan

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan Tugas Akhir.

Lampiran

Merupakan bab tambahan yang berisi daftar istilah yang penting pada aplikasi ini.

BAB II
DASAR TEORI

Bab ini berisi penjelasan mengenai teori-teori yang menjadi dasar mengimplementasikan perangkat lunak. Penjelasan ini bertujuan untuk memberi gambaran secara umum mengenai perangkat lunak yang dibuat. Adapun teori-teori tersebut meliputi Earthquake Hazards Program, JSON, GSON, *geocoding*, Google Maps API, Twitter4J, Geonames, Twitter API, rumus Haversine, dan ADT.

2.1. Earthquake Hazards Program

Earthquake Hazards Program [1] merupakan situs *web* yang disediakan oleh USGS (*United States Geological Survey*), lembaga survei geologi di Amerika, sebagai salah satu usaha untuk mengurangi dampak dari gempa bumi. Situs *web* ini menyediakan informasi gempa yang terjadi di seluruh dunia secara *real-time*. Informasi data gempa yang disajikan berupa lokasi terjadinya gempa, *magnitude*, waktu terjadinya gempa, dan koordinat *geolocation* gempa dalam *latitude* dan *longitude*. Informasi ini tersedia dalam bentuk *feed* yang memiliki format XML (*Extensible Markup Language*) dan JSON. Adapun kelebihan dari situs *web* Earthquake Hazards Program adalah sebagai berikut:

1. Data *feed* gempa yang terjadi selalu diperbaharui setiap detik (*real-time*).
2. Informasi yang disediakan oleh data *feed* gempa lengkap.
3. Format hasil data *feed* gempa adalah XML dan JSON sehingga mudah diolah.

Contoh informasi gempa yang diambil dari situs *web* Earthquake Hazards Program ini dapat dilihat pada Gambar 2.1. Pada Gambar 2.1, gempa terjadi di selatan Pulau Jawa. Tanda bintang menunjukkan lokasi terjadinya gempa. Selain menunjukkan lokasi terjadinya gempa, halaman situs *web* ini juga menunjukkan *magnitude* gempa, waktu terjadinya gempa, dan *geolocation* gempa

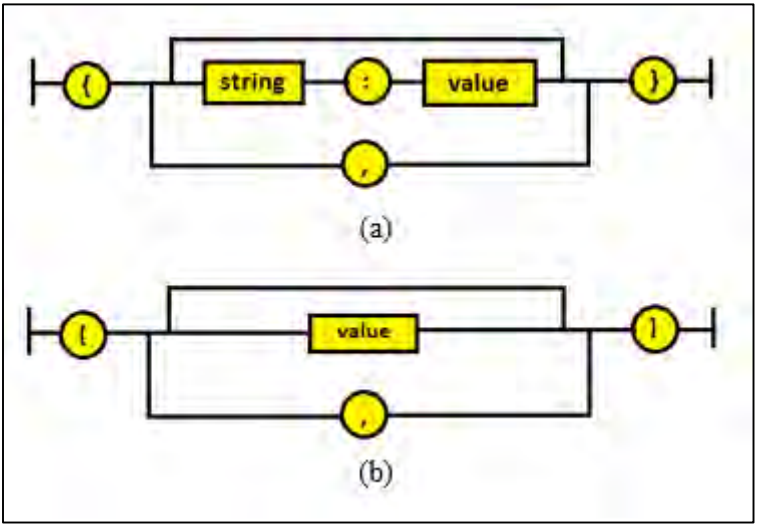
dalam *latitude* dan *longitude*. Pada Tugas Akhir ini, situs *web* Earthquake Hazards Program digunakan sebagai acuan dalam melakukan validasi informasi gempa bumi menggunakan *social sensor*.

Gambar 2.1 Informasi Gempa pada Situs *Web* Earthquake Hazards Program [1]

2.2. JSON

JSON [2] (*JavaScript Object Notation*) merupakan sebuah format data ringan yang mudah untuk diolah. JSON memudahkan manusia untuk membaca dan menulis dan memudahkan mesin untuk mengurai dan menghasilkan data. JSON merupakan sebuah format teks yang independen namun memiliki pustaka yang dapat dikenali oleh berbagai macam bahasa pemrograman seperti C, C++, Java, JavaScript, Perl, Python dan lain-lain. JSON dibangun dalam dua bentuk antara lain:

1. Sebuah objek, merupakan sekumpulan pasangan nama dan nilai. Sebuah objek dimulai dengan { (kurung kurawal kiri) dan diakhiri dengan } (kurung kurawal kanan). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama dan nilai dipisahkan dengan , (koma). Contoh sebuah objek pada JSON dapat dilihat pada Gambar 2.2.a.
2. Sebuah nilai yang berurutan, dalam bahasa pemrograman dikenal dengan sebutan *array*, *vector*, atau *list*. Sebuah *array* dimulai dengan [(kurung siku kiri) dan berakhir dengan] (kurung siku kanan). Setiap nilai yang berbeda dipisahkan dengan tanda koma. Sebuah *array* tersebut dapat berisi sebuah *array* lain. *Array* ini disebut *array* bersarang. Contoh sebuah *array* pada JSON dapat dilihat pada Gambar 2.2.b.



Gambar 2.2 (a) Struktur Objek JSON (b) Struktur *Array* dalam JSON [2]

2.2.1. GSON

GSON [3] adalah pustaka bahasa pemrograman Java yang digunakan untuk mengubah objek Java menjadi representasi JSON. GSON juga dapat digunakan untuk mengubah sebuah string JSON menjadi objek Java. GSON bersifat *open source* sehingga dapat digunakan secara gratis. Adapun tujuan dari GSON adalah sebagai berikut:

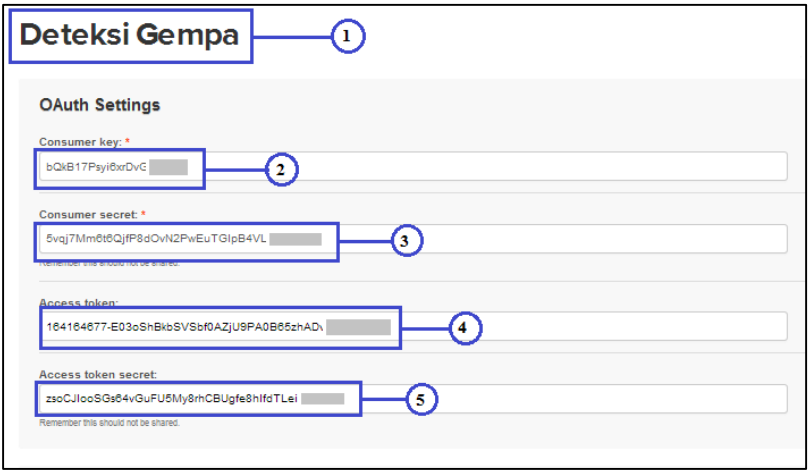
1. Menyediakan fungsi toJSON() dan fromJson() yang simpel untuk mengubah objek Java menjadi JSON atau mengubah JSON menjadi objek Java.
2. Memungkinkan objek yang tidak dapat dimodifikasi untuk diubah ke dan dari JSON.
3. Memungkinkan representasi *custom* dari objek.
4. Dukungan yang baik terhadap bahasa pemrograman Java.
5. Memungkinkan representasi objek yang kompleks.

Pada Tugas Akhir ini, pustaka GSON digunakan untuk melakukan *parsing* atau mengubah data JSON yang telah didapat dari situs *web* Earthquake Hazards Program menjadi objek Java untuk dapat diolah lebih lanjut.

2.3. Twitter API

Twitter API [4] (*Application Programming Interface*) merupakan sejumlah fungsi yang dapat digunakan pengembang perangkat lunak untuk mengolah data saat membangun perangkat lunak. Twitter API menyediakan beberapa fungsi untuk melakukan suatu tugas tertentu, sehingga pengembang perangkat lunak hanya memanggil fungsi tersebut di dalam perangkat lunak yang dibangun. Twitter API menggunakan arsitektur REST (*Representational State Transfer*) sehingga Twitter API dapat digunakan pada format data yang beragam seperti XML maupun JSON.

Twitter API terdiri atas Twitter Search API dan Twitter Streaming API. Perbedaan keduanya yaitu, Twitter Search API menitikberatkan fungsi pencarian ke masa lampau sedangkan Twitter Streaming API menitikberatkan fungsi pencarian ke masa yang akan datang. Namun, untuk menggunakan fitur dari Twitter API pengembang perangkat lunak harus mendaftarkan perangkat lunak tersebut kepada Twitter API terlebih dahulu. Setelah mendaftar, pengembang perangkat lunak akan mendapatkan *token* berupa *Consumer Key* dan *Consumer Key Secret*, serta *Access Token* dan *Access Token Secret*. Contoh halaman situs *web* Twitter API dari perangkat yang telah terdaftar dapat dilihat pada Gambar 2.3.



Gambar 2.3 Halaman Situs *Web* Twitter API [4]

Dalam Tugas Akhir ini, jenis Twitter API yang digunakan adalah Twitter Search API. Twitter Search API digunakan sebagai aplikasi *third-party* untuk melakukan pencarian *tweet* informasi gempa di masa lampau.

2.3.1. Twitter4J

Twitter4J [5] merupakan sebuah pustaka tidak resmi dari Twitter API berbasis Java. Twitter4J dapat dengan mudah mengintegrasi perangkat lunak berbasis Java yang dikembangkan dengan servis yang ada pada Twitter API. Twitter4J bersifat *open source* dan dapat digunakan secara gratis. Pengguna cukup menambahkan pustaka Twitter4J ke dalam *classpath* perangkat lunak pada IDE. Adapun kelebihan dari Twitter4J adalah sebagai berikut:

1. Siap digunakan pada Java *platform* 5 atau lebih dan Android.
2. Tidak membutuhkan pustaka tambahan lain.
3. Kompatibel terhadap Twitter API 1.1.

Pada Tugas Akhir ini, Twitter4J digunakan untuk mengembalikan *tweet* beserta informasi yang dibutuhkan. Adapun informasi dari *tweet* yang dibutuhkan adalah isi *tweet*, *geolocation tweet*, dan waktu *posting tweet*. Informasi yang telah didapat akan diolah oleh *server* untuk kemudian dilakukan validasi dan ditentukan *level* konfidensi.

2.4. *Geocoding*

Geocoding [6] merupakan proses penyimpanan identifikasi *geolocation*. *Geolocation* merupakan sebuah lokasi geografis dari sebuah objek. Secara garis besar *geocoding* melakukan dua hal: melaporkan lokasi geografis pengguna kepada pengguna lain, dan mengasosiasi suatu lokasi *real-world* (seperti restoran, taman rekreasi, dan lain-lain) dengan lokasi pengguna. *Smartphone* yang ada saat ini memiliki GPS (*Global Positioning System*) *chip* di dalamnya, yang akan digunakan untuk menentukan posisi secara *real-time*. Namun bila sinyal GPS tidak tersedia, *geocoding* dapat menggunakan informasi dari menara selular untuk memperkirakan posisi, namun tidak seakurat GPS.

Pada Tugas Akhir ini *geocoding* digunakan untuk melakukan operasi seperti mengembalikan nilai koordinat

geolocation dalam *latitude* dan *longitude* dengan masukan berupa nama objek/nama kota.

2.4.1. Google Maps API

Google Maps API [7] adalah sebuah API peta globe virtual yang disediakan oleh Google. Pada Google Maps API terdapat empat jenis pilihan model peta yang disediakan oleh Google, di antaranya adalah sebagai berikut:

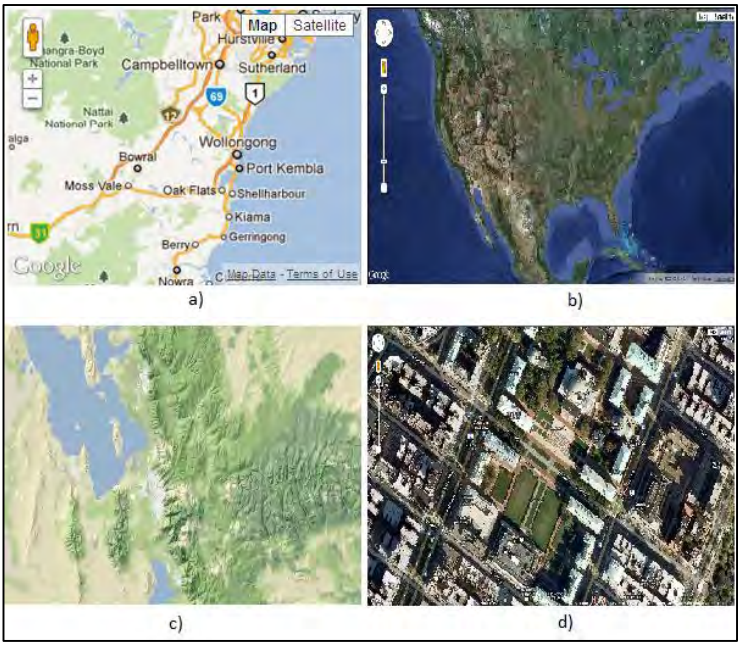
1. Roadmap, untuk menampilkan peta dua dimensi biasa.
2. Satellite, untuk menampilkan foto satelit.
3. Terrain, untuk menampilkan relief fisik permukaan bumi dan menunjukkan tinggi suatu lokasi, contohnya menunjukkan gunung dan sungai.
4. Hybrid, untuk menampilkan foto satelit yang diatasnya tergambar pula apa yang tampil pada Roadmap (jalan dan nama kota).

Contoh representasi Google Maps API dalam bentuk roadmap, satelitte, terrain, dan hybrid secara berurutan dapat dilihat pada Gambar 2.4.a, Gambar 2.4.b, Gambar 2.4.c, dan Gambar 2.4.d. Adapun kelebihan dari Google Maps API adalah sebagai berikut:

1. *Database* kota yang ada pada Google Maps API sangat lengkap.
2. Memiliki dukungan baik terhadap Java dan JavaScript.
3. Dapat dikolaborasikan dengan teknologi bahasa pemrograman lain seperti PHP, MySQL, Jquery, dan AJAX.

Namun, Google Maps API memiliki beberapa kekurangan, diantaranya adalah *request* Google Maps API yang dibatasi yaitu 2500 *request* setiap hari. Jika ingin melakukan *request* melebihi batasan, pengembang perangkat lunak harus mendapatkan lisensi Google Maps API for Business.

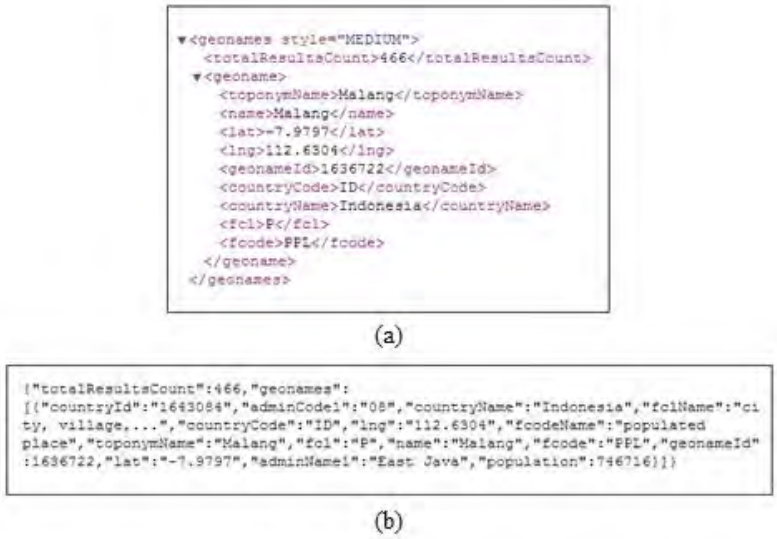
Pada Tugas Akhir ini Google Maps API digunakan untuk menampilkan lokasi gempa dalam bentuk peta pada *smartphone* Android dan situs *web*.



Gambar 2.4 Representasi Google Maps API (a) RoadMap (b) Satelitte (c) Terrain (d) Hybrid [7]

2.4.2. GeoNames

GeoNames [8] merupakan salah satu penyedia layanan *geocoding*. *Database* GeoNames berisi lebih dari 10 juta lokasi di seluruh dunia beserta dengan informasi yang diperlukan, seperti *latitude*, *longitude*, negara, dan ID yang unik. Data GeoNames dapat diperoleh secara gratis melalui *web service*. *Web service* tersebut meliputi *geocoding* secara langsung atau *reverse geocoding*, menemukan tempat melalui kode pos, dan menemukan tempat di sekitar tempat tertentu. Setiap fitur dari GeoNames direpresentasikan sebagai *web resource* dengan URI (*Uniform Resource Identifier*) yang stabil. Representasi hasil GeoNames berupa XML.



Gambar 2.5 Representasi GeoNames (a) XML (b) JSON [8]

Contoh representasi XML GeoNames dapat dilihat pada Gambar 2.5.a Selain itu terdapat juga representasi JSON dari GeoNames. Representasi JSON dari GeoNames dapat dilihat pada Gambar 2.5.b. Pada Tugas Akhir ini, GeoNames digunakan sebagai aplikasis *third-party geocoding* untuk mengembalikan nilai koordinat *geolocation* dalam *latitude* dan *longitude* dengan masukan berupa nama kota.

2.5. Rumus Haversine

Rumus Haversine [9] merupakan salah satu rumus persamaan matematika yang digunakan dalam perhitungan navigasi, yaitu untuk menghitung jarak diantara dua buah koordinat *geolocation* yang terdiri atas *latitude* dan *longitude* pada sebuah *sphere*. Adapun persamaan rumus Haversine dapat dilihat pada Persamaan 2.1.

$$d = 2r \sin^{-1}\left(\sqrt{sin^2\left(\frac{\emptyset_2 - \emptyset_1}{2}\right) + \cos(\emptyset_1)\cos(\emptyset_2)\, sin^2(\frac{\psi_2 - \psi_1}{2})}\right) \quad (2.1)$$

Dimana d adalah jarak diantara kedua titik berdasarkan koordinat dengan *latitude* dan *longitude* ($\emptyset$, ψ) dan r merupakan radius dari bumi.

Pada Tugas Akhir ini rumus Haversine digunakan untuk menghitung jarak diantara lokasi terjadinya gempa dengan koordinat *geolocation* dalam *latitude* dan *longitude.* Jarak ini didapat berdasarkan informasi dari situs *web* Earthquake Hazards Program dan koordinat *geolocation* informasi gempa berdasarkan *social sensor* yakni Twitter.

2.6. *PHP*

PHP [10] (*Personal Home Page*) merupakan sebuah bahasa pemrograman yang berjalan pada sebuah *webserver* dan digunakan untuk pengembangan situs *web*. Selain itu, PHP dapat digunakan dengan tujuan umum seperti membuat sebuah berkas XML, dan melakukan pengolahan *database* seperti menyeleksi, menambah, menghapus, atau memperbaharui data yang ada di dalam *database*. PHP dapat dikombinasikan dengan kode HTML (*Hypertext Markup Language*) untuk menghasilkan konten situs *web* yang dinamis.

Pada Tugas Akhir ini, bahasa pemrograman PHP digunakan untuk membuat berkas XML yang menampilkan *database* berisi informasi gempa dan *tweet* informasi gempa. Berkas XML ini digunakan untuk menjembatani informasi yang ada di *database* dengan situs *web* dan perangkat lunak *smartphone* berbasis Android.

2.7. *Web Service*

Web service [11] merupakan antarmuka yang menggambarkan operasi-operasi yang dapat diakses di dalam jaringan melalui pesan XML terstandarisasi. *Web service*

dideskripsikan menggunakan XML standar yang disebut deskripsi layanan. Antarmuka *web service* menyembunyikan detail implementasi dari servis, memungkinkan *web service* dapat digunakan secara independen dari *platform hardware* atau *software* dimana *web service* tersebut diimplementasikan, serta independen dari bahasa pemrograman yang digunakan. *Web service* dapat melakukan sebuah tugas yang spesifik atau serangkaian tugas. *Web service* juga dapat digunakan sendiri atau dikombinasikan dengan *web service* lain untuk menyelesaikan tugas yang lebih kompleks.

Pada Tugas Akhir ini *web service* digunakan untuk menjembatani komunikasi antara *server* dan *smartphone* berbasis Android. *Web service* ini digunakan untuk mengakses data di dalam *database* dan mengirimkan hasilnya ke *smartphone* berbasis Android.

2.8. Eclipse

Eclipse [12] merupakan sebuah IDE (*Integrated Development Environment*) yang digunakan sebagai *environment* untuk mengembangkan perangkat lunak. Bahasa pemrograman standar yang digunakan oleh Eclipse adalah Java. Eclipse memiliki banyak *plug-in* yang dapat digunakan untuk mengembangkan perangkat lunak yang memiliki bahasa pemrograman lain seperti C, C++, JavaScript, dan lain-lain. Selain itu Eclipse juga memiliki *plug-in* untuk mengembangkan aplikasi berbasis Android, yaitu ADT.

2.8.1. ADT

ADT [13] (Android *Development Tools*) merupakan sebuah *plug-in* yang disediakan oleh Eclipse yang didesain untuk memberikan *environment* yang kuat dan terintegerasi bagi pengembang perangkat lunak *smartphone* berbasis Android. ADT memudahkan pengembang perangkat lunak *smartphone* berbasis Android untuk membuat sebuah perangkat lunak baru lengkap dengan desain UI (*User Interface*) dari perangkat lunak dan

menambahkan *package* berdasarkan Android Framewok API, melakukan *debug* aplikasi menggunakan Android SDK (*Software Development Kit*) Tools, serta mengekspor berkas bertipe .apk untuk mendistribusikan perangkat lunak tersebut. Adapun kelebihan dari ADT adalah sebagai berikut:

1. Perbaikan cepat, navigasi terintegrasi antara Java dan XML karena lebih spesifik terhadap Android.
2. Desain UI yang mudah karena memiliki fitur *drag and drop*.
3. Memudahkan visualisasi UI tanpa harus melakukan *deploy* perangkat lunak ke *smartphone* terlebih dahulu.
4. Memungkin proses *debugging* melalui USB (*Universal Serial Bus*)
5. Melakukan *deploy* perangkat lunak langsung dari IDE ke perangkat yang terhubung dengan PC.

Pada Tugas Akhir ini, ADT digunakan sebagai *environment* untuk mengembangkan perangkat lunak *smartphone* berbasis Android.

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini berisi penjelasan mengenai analisis dan perancangan perangkat lunak yang dikembangkan. Perancangan merupakan bagian penting dari pengembangan perangkat lunak karena merupakan perencanaan perangkat lunak secara teknis. Adapun hal-hal yang dibahas dalam bab ini adalah deskripsi umum perangkat lunak, arsitektur perangkat lunak, diagram kasus penggunaan, perancangan *database*, diagram alur, dan desain antarmuka perangkat lunak.

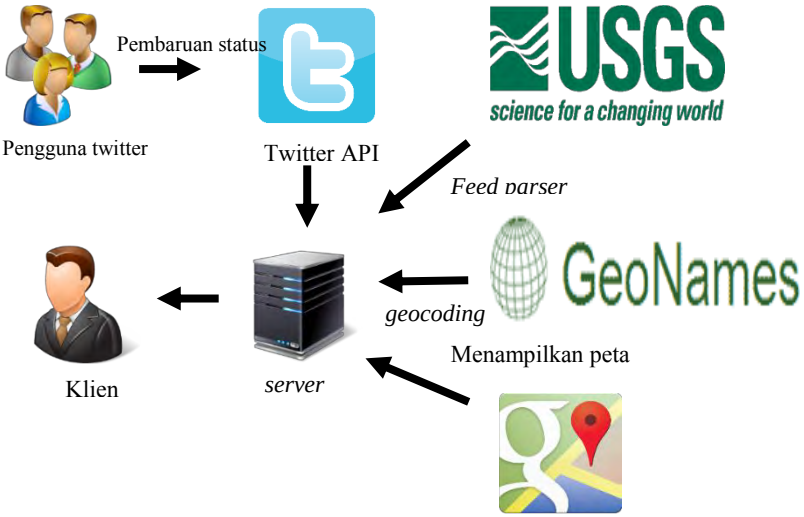
3.1. Deskripsi Umum

Pada Tugas Akhir ini dikembangkan sebuah perangkat lunak yang berfungsi untuk melakukan deteksi dan validasi informasi gempa secara *real-time* pada *social sensor* dengan Twitter. Perangkat lunak menampilkan informasi gempa beserta dengan *level* konfidensi *tweet* informasi gempa, serta menampilkan peta lokasi dimana gempa itu terjadi.

Gambaran umum dari perangkat lunak ini yaitu, perangkat lunak mengambil data gempa yang terjadi di Indonesia dari stitus *web* Earthquake Hazards Program. Data yang diambil berupa *magnitude* gempa, lokasi gempa, waktu terjadinya gempa, serta koordinat *geolocation* gempa dalam *latitude* dan *longitude*. Selanjutnya, perangkat lunak membaca pembaruan status pengguna Twitter dari Twitter Search API. Kemudian, perangkat lunak membaca koordinat *geolocation tweet* dalam *latitude* dan *longitude* menggunakan *geocoding*. Dari kedua koordinat *geolocation latitude* dan *longitude* yang ada pada situs *web* dan *tweet*, dihitung jarak antara keduanya menggunakan rumus Haversine. Berdasarkan data tersebut, dapat dilakukan validasi berdasarkan *level* konfidensi dari masing-masing *tweet*. Hasil pengolahan validasi gempa ini diletakkan ke dalam *web service* untuk kemudian dapat dikonsumsi oleh antarmuka situs *web* dan *smartphone* Android. Adapun

diagram aplikasi dari perangkat lunak ini dapat dilihat pada Gambar 3.1.

Pada diagram aplikasi ini dijelaskan sebuah *server* yang mengolah data *feed* informasi gempa yang didapat dari situs *web* Earthquake Hazards Program, *geocoding* dari GeoNames, dan informasi gempa dari Twitter Search API. Kemudian *server* mengembalikan hasil pengolahan dan menampilkan peta lokasi terjadinya gempa kepada klien menggunakan Google Maps API.



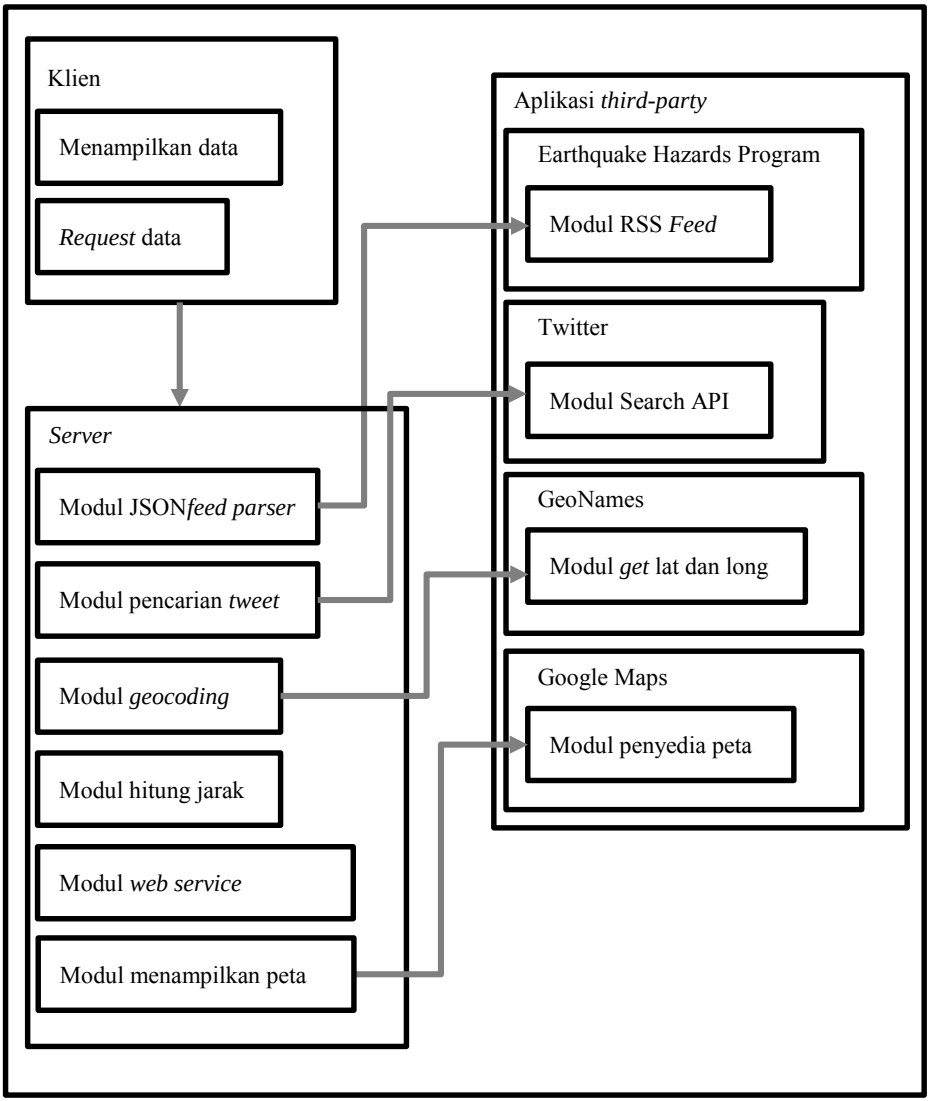
Gambar 3.1 Diagram Aplikasi

3.2. Arsitektur Umum Perangkat Lunak

Arsitektur umum perangkat lunak dalam Tugas Akhir ini dapat dilihat pada Gambar 3.2. Arsitektur perangkat lunak terdiri atas klien, *server*, dan aplikasi *third-party*. Klien merupakan pengguna yang menggunakan dan mengoperasikan perangkat lunak. *Server* digunakan sebagai wadah untuk menampung semua fungsi dan logika dari perangkat lunak. Selain klien dan *server*, terdapat

aplikasi *third-party* yang digunakan sebagai penunjang kebutuhan dari perangkat lunak yang dibangun. Adapun penjelasan untuk modul pada *server* adalah sebagai berikut.

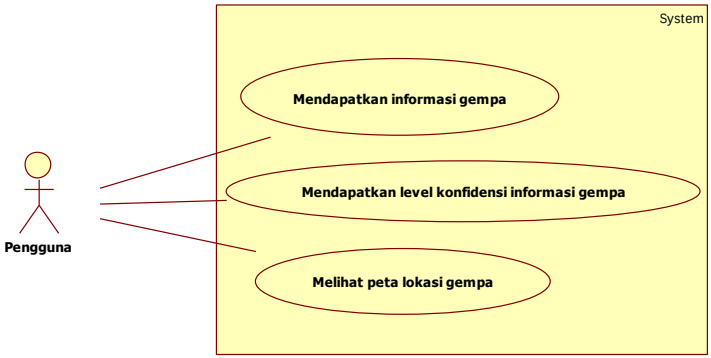
1. Modul *feed parser*
   Modul *feed parser* digunakan untuk melakukan *parsing* terhadap *feed* yang ada pada aplikasi *third-party* yaitu situs *web* Earthquake Hazards Program. Format data yang digunakan oleh situs *web* tersebut adalah JSON. JSON yang didapat dari situs ini diubah menjadi sebuah objek di dalam Java. Karena perangkat lunak berjalan dalam sebuah *thread*, maka modul *feed parser* selalu mengecek situs *web* tersebut setiap jangka waktu yang telah ditentukan.
2. Modul pencarian *tweet*
   Modul pencarian *tweet* merupakan modul yang digunakan untuk mencari *tweet* dengan menggunakan aplikasi *third-party* Twitter Search API. Data yang dihasikan dari modul ini berupa *username,* isi *tweet*, *geolocation tweet*, dan waktu *posting tweet*.
3. Modul *geocoding*
   Modul *geocoding* digunakan untuk mengetahui koordinat *geolocation* (*latitude* dan *longitude*) suatu daerah di dalam *string* isi *tweet*. Modul *geocoding* menggunakan GeoNames sebagai aplikasi *third-party*.
4. Modul menghitung jarak
   Modul menghitung jarak digunakan untuk menghitung jarak diantara dua koordinat *geolocation* (*latitude* dan *longitude*) menggunakan rumus Haversine.
5. Modul *posting web service*
   Modul ini digunakan untuk melakukan *posting* hasil validasi *tweet* informasi gempa ke dalam sebuah *web service.*
6. Modul menampilkan peta
   Modul menampilkan peta digunakan untuk menampilkan peta ke dalam *smartphone* berbasis Android dan situs *web*. Modul ini menggunakan aplikasi *third-party* Google Maps untuk menampilkan lokasi terjadinya gempa dan lokasi *tweet*.



Gambar 3.2 Arsitektur Perangkat Lunak

3.3. Perancangan Diagram Kasus Penggunaan

Diagram kasus penggunaan menggambarkan peran aktor yang terlibat dalam fungsionalitas perangkat lunak yang dibangun. Adapun perancangan diagram kasus penggunaan perangkat lunak dalam Tugas Akhir ini dapat dilihat pada Gambar 3.3.



Gambar 3.3 Diagram Kasus Penggunaan

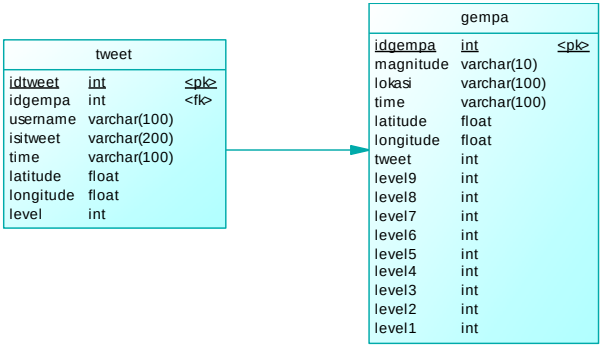
Pada Tugas Akhir ini, terdapat satu aktor yang berperan untuk melakukan *request* data. Aktor dapat menjalankan fitur yang ada pada perangkat lunak ini, yaitu sebagai berikut:

1. Mendapatkan informasi gempa
   Aktor mendapatkan informasi gempa yang berasal dari *feed* informasi gempa melalui situs *web* Eathquake Hazards Program. Informasi yang didapat berupa besar *magnitude* gempa, lokasi gempa beserta koordinat *latitude* dan *longitude*, dan waktu terjadinya gempa.
2. Mendapatkan *level* konfidensi *tweet* informasi gempa
   Aktor mendapatkan informasi *level* konfidensi *tweet* informasi gempa dari *social sensor* berbasis Twitter yang telah diolah untuk melakukan validasi gempa yang telah terjadi.

3. Melihat peta lokasi gempa
   Aktor dapat melihat peta lokasi terjadinya gempa berdasarkan informasi yang didapat dari situs *web* Earthquake Hazards Program.

3.4. Perancangan *Database*

Perancangan *database* merupakan tahapan untuk merancang *database* yang diperlukan oleh perangkat lunak. *Database* ini digunakan untuk menyimpan informasi dari entitas *tweet* dan gempa. Pada Tugas Akhir ini, terdapat dua tabel yaitu tabel *tweet* dan tabel informasi gempa yang saling berhubungan dan memiliki relasi *one to many*. Gambar perancangan *database* beserta dengan relasinya dapat dilihat pada Gambar 3.4. Penjelasan dari setiap tabel dapat dilihat pada subbab berikut ini.

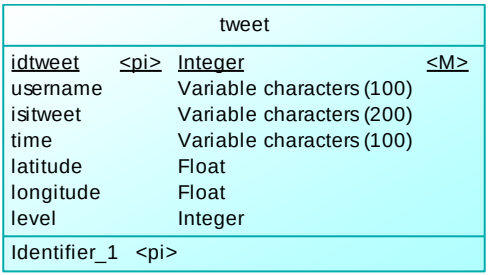


Gambar 3.4 Tabel *Tweet* dan Tabel Gempa serta Relasinya.

3.4.1. Tabel *Tweet*

Tabel *tweet* merupakan tabel yang menyimpan informasi yang terdapat pada sebuah *tweet*. Informasi yang disimpan adalah id *tweet*, *username tweet*, isi dari *tweet*, waktu *posting tweet*, koordinat *geolocation tweet* dalam *latitude* dan *longitude*, serta *level*

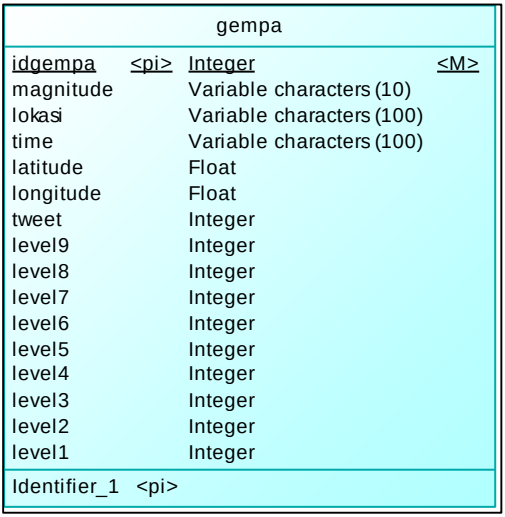
konfidensi dari *tweet* tersebut. Tabel *tweet* ditunjukkan oleh Gambar 3.5.



Gambar 3.5 Tabel *Tweet*

3.4.2. Tabel Gempa

Gambar 3.6 merupakan tabel gempa yang menyimpan informasi gempa yang didapat dari situs *web* Earthquake Hazards Program.

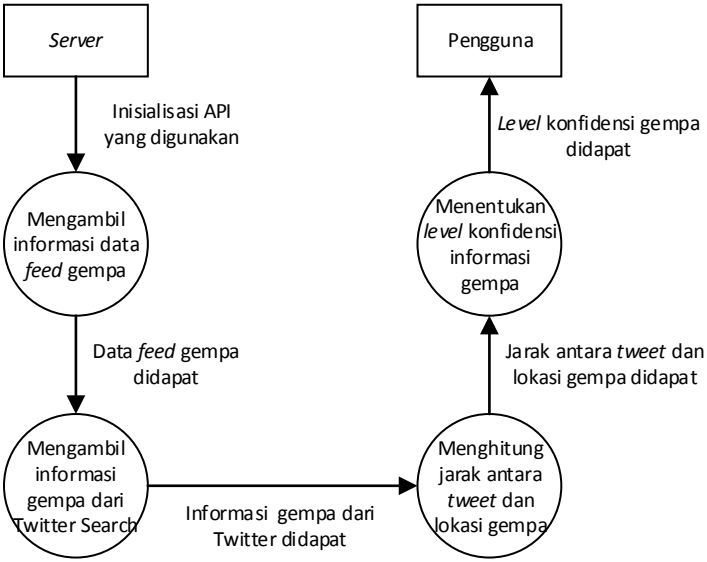


Gambar 3.6 Tabel Gempa

Informasi ini terdiri atas id gempa, *magnitude* gempa, lokasi gempa, waktu terjadinya gempa, dan koordinat *geolocation* gempa dalam *latitude* dan *longitude*. Selain itu tabel gempa juga berisi jumlah *tweet* dengan *level* konfidensi yang telah diolah. Untuk penjelasan *level* konfidensi informasi gempa ini akan dijelaskan pada subbab perancangan diagram alir data perangkat lunak.

3.5. Perancangan Diagram Alir Data *Level* 0

Diagram alir data *level* 0 adalah gambaran fungsionalitas perangkat lunak dan faktor eksternal yang terlibat. Diagram alir data *level* 0 dapat dilihat pada Gambar 3.7.



Gambar 3.7 Diagram Alir Data *Level* 0

Berdasarkan Gambar 3.7, hal pertama yang dilakukan adalah *server* mengambil informasi data *feed* gempa dari situs *web* Earthquake Hazards Program. Setelah informasi dari situs ini

didapat maka *server* melakukan pencarian informasi gempa dari *social sensor* dalam hal ini adalah Twitter dengan menggunakan Twitter Search API. Setelah itu dihitung jarak antara *tweet* dan lokasi terjadinya gempa. Dengan menggunakan jarak tersebut, dapat ditentukan *level* konfidensi *tweet* informasi gempa untuk kemudian dikembalikan kepada pengguna.

3.6. Perancangan Alir Aplikasi

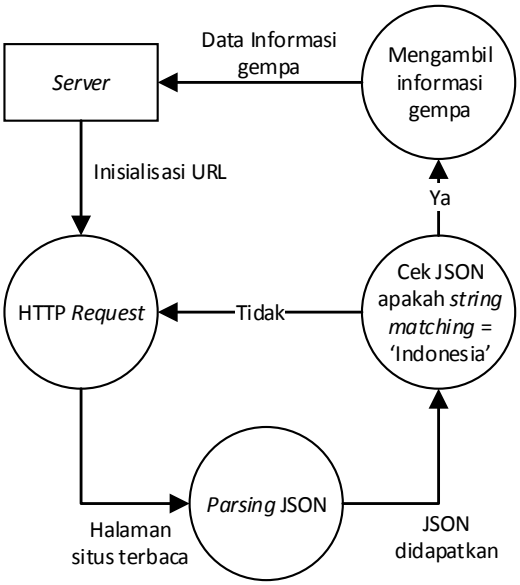
Pada Tugas Akhir ini, alur setiap proses di dalam perangkat lunak digambarkan dengan diagram alir, untuk memudahkan pemahaman secara garis besar proses yang ada pada perangkat lunak ini. Diagram alir yang terdapat pada perangkat lunak ini terdiri atas diagram alir parseJSON, pencarian *tweet*, penentuan *level* konfidensi *tweet*, dan *web service*.

3.6.1. Diagram Alir Data JSON *Parser*

Pada saat perangkat lunak dijalankan pada *server*, perangkat lunak membaca URL JSON dari halaman situs *web* Earthquake Hazards Program. Setelah URL dibaca, *server* melakukan HTTP (*Hypertext Transfer Protocol*) *request* ke halaman situs *web* untuk mendapatkan JSON yang berisi informasi gempa. Selanjutnya dilakukan inisialisasi fungsi dari GSON untuk melakukan *parsing* terhadap JSON. Fungsi dari GSON yang digunakan untuk melakukan *parsing* adalah fromJSON. Adapun informasi data gempa yang di-*parsing* adalah lokasi gempa, waktu terjadinya gempa, *magnitude* gempa, serta koordinat gempa dalam satuan *latitude* dan *longitude*.

Setelah dilakukan *parsing*, perangkat lunak mengecek JSON tersebut apakah lokasi gempa mengandung kata ‘Indonesia’ dengan menggunakan *string matching*. Tujuannya agar hanya data gempa di Indonesia saja yang diambil. Jika JSON mengandung kata ‘Indonesia’ maka JSON tersebut diubah menjadi objek di dalam Java. Selanjutnya informasi gempa yang telah berbentuk objek

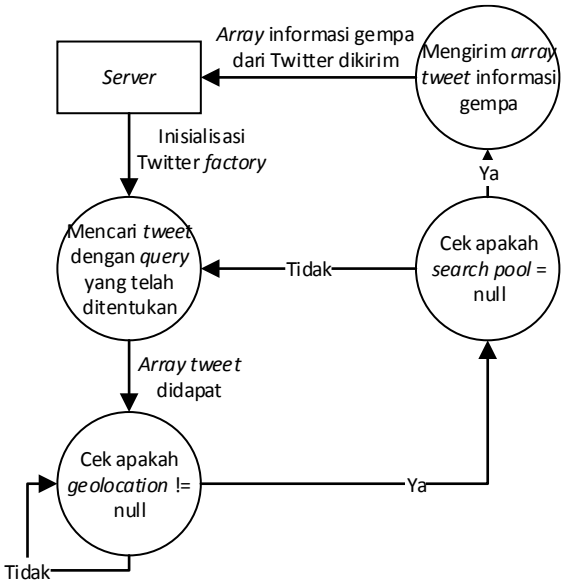
dalam Java dikirim kembali ke *server* untuk pengolahan selanjutnya. Gambar diagram alir dari JSON *parse* dapat dilihat pada Gambar 3.8.



Gambar 3.8 Diagram Alir Data JSON *Parse*

3.6.2. Diagram Alir Data Pencarian *Tweet*

Setelah mendapatkan informasi gempa dari situs *web* Earthquake Hazards Program, hal yang selanjutnya dilakukan adalah melakukan pencarian informasi gempa dari *social sensor* yaitu Twitter. Adapun diagram alir data pencarian *tweet* dapat dilihat pada Gambar 3.9.



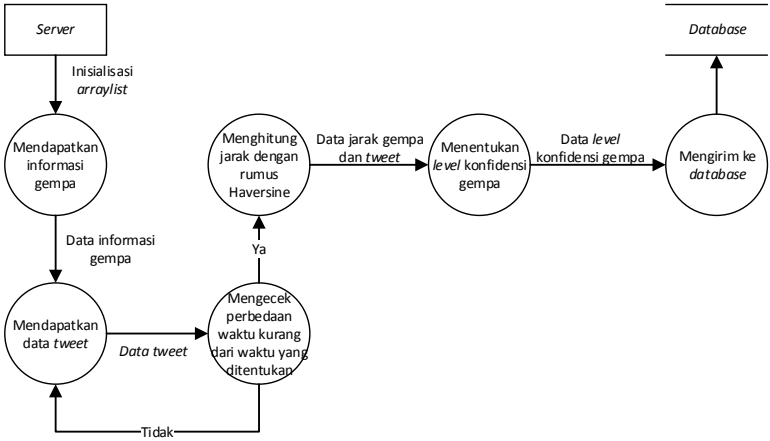
Gambar 3.9 Diagram Alir Data Pencarian *Tweet*

Hal pertama yang dilakukan adalah *server* melakukan inisialisasi Twitter *factory*. Hal-hal yang dilakukan pada saat inisialisasi Twitter *factory* adalah autentikasi *consumer key* dan *access token*. Autentikasi ini diperlukan agar perangkat lunak dapat menggunakan fitur dari Twitter Search API. Jika autentikasi berhasil, maka perangkat lunak memulai pencarian *tweet* berikut data yang dibutuhkan seperti *username*, isi *tweet*, waktu posting *tweet*, dan koordinat *geolocation tweet* dalam *latitude* dan *longitude* pada Twitter Search *pool* berdasarkan *query*. *Query* yang digunakan untuk pencarian sudah ditentukan sebelumnya. Setelah *tweet* yang memenuhi *query* didapat, *tweet* disimpan ke dalam sebuah *array*. Selanjutnya, perangkat lunak menyeleksi *array* tersebut agar semua *tweet* memiliki *geolocation*. Kemudian, *server* mengecek apakah masih terdapat *tweet* di dalam Twitter Search *pool*. Jika tidak ada,

perangkat lunak mengirim data informasi gempa dari Twitter ke *server*.

3.6.3. Diagram Alir Data Menentukan *Level* Konfidensi Informasi Gempa

Tahapan selanjutnya yang dilakukan oleh perangkat lunak adalah menghitung *level* konfidensi gempa. Diagram alir data menghitung *level* konfidensi informasi gempa dapat dilihat pada Gambar 3.10. Pertama, *server* melakukan inisialisasi *arraylist* yang memuat data gempa dan data *tweet*. Setelah mendapatkan data gempa dan *tweet*, perangkat lunak mengecek perbedaan waktu antara waktu posting *tweet* dan waku terjadinya gempa. Hal ini dilakukan agar informasi gempa yang dihimpun tidak kadaluarsa.



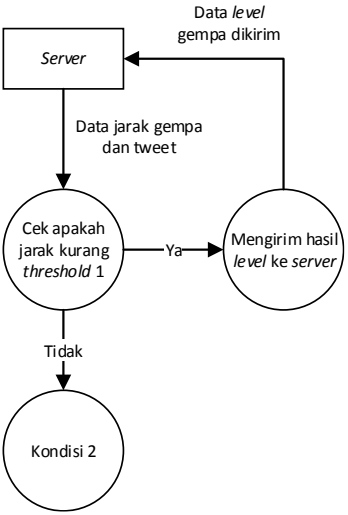
Gambar 3.10 Diagram Alir Data Menentukan *Level* Konfidensi Gempa

Setelah mendapatkan data *tweet* yang dimaksud, langkah selanjutnya yaitu menghitung jarak diantara lokasi dimana gempa terjadi dan lokasi *posting tweet*. Berdasarkan jarak tersebut, dapat

ditentukan *level* konfidensi informasi gempa dari *social sensor*. Langkah berikutnya yaitu *level* konfidensi informasi gempa yang telah diolah diletakkan ke dalam sebuah tabel *database* beserta dengan informasi gempa dari situs *web* Earthquake Hazards Program. Selain informasi gempa, semua *tweet* informasi gempa juga diletakkan ke dalam *database* untuk selanjutnya diletakkan ke dalam *web service*. Terdapat tiga kondisi yang menjadi tolak ukur penentuan *level* konfidensi informasi gempa. Ketiga kondisi tersebut akan dijelaskan pada subbab berikut.

3.6.3.1. Diagram Alir Data Menentukan *Level* Konfidensi Gempa (Kondisi 1)

Gambar 3.11 merupakan diagram alir data menentukan *level* konfidensi gempa dengan kondisi 1. Pada Gambar 3.11, pertama-tama *server* mengecek apakah jarak antara lokasi terjadinya gempa dan *tweet* kurang dari *threshold* 1.



Gambar 3.11 Diagram Alir Data Menentukan *Level* Konfidensi Gempa (Kondisi 1)

Berdasarkan data dari BMKG (Badan Meteorologi dan Geofisika), rata-rata gempa yang sering terjadi di Indonesia berkisar antara 3,0 sampai 5,0 Skala Ritcher. *Threshold* 1 merupakan jarak efek gempa jika *magnitude* gempa adalah 3,0. Adapun jarak efek gempa jika *magnitude* gempa 3,0 Skala Ritcher adalah berkisar antara 50 km [14]. Jika jarak kurang dari *threshold* 1, maka *level* konfidensi gempa adalah 9. Informasi ini kemudian dikirim ke *server*. Jika jarak melebihi *threshold* 1, maka jarak tersebut masuk ke dalam kondisi 2.

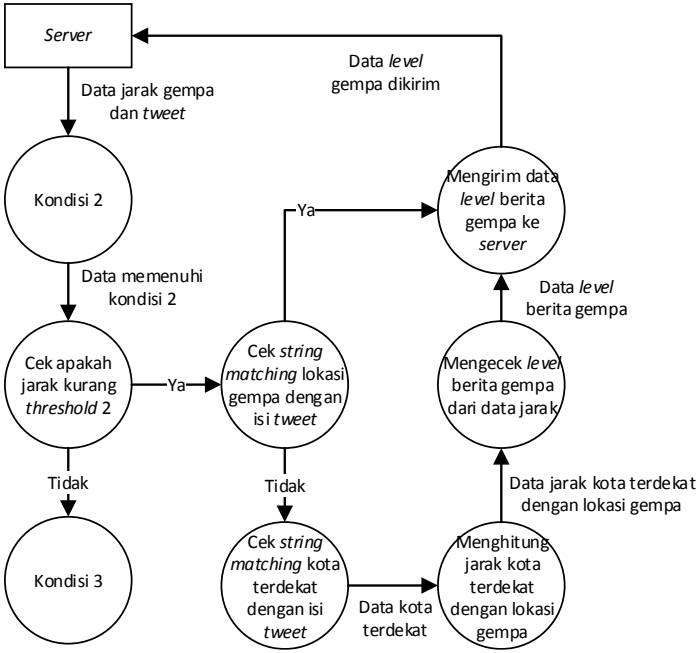
3.6.3.2. Diagram Alir Data Menentukan *Level* Konfidensi Gempa (Kondisi 2)

Gambar 3.12 menunjukkan diagram alir data untuk menentukan *level* konfidensi gempa dengan kondisi 2. Pertama-tama *server* mengecek apakah jarak diantara lokasi terjadinya gempa dan *tweet* tidak memenuhi kondisi 1. Selanjutnya perangkat lunak mengecek apakah jarak melebihi *threshold* 2. *Threshold* 2 merupakan jarak efek gempa jika *magnitude* gempa adalah 5,0. Adapun jarak efek gempa jika *magnitude* gempa 5,0 Skala Ritcher adalah berkisar antara 300 km [14]. Jika jarak melebihi *threshold* 2, maka jarak tersebut masuk ke dalam kondisi 3.

Jika jarak kurang dari *threshold* 2, perangkat lunak melakukan pengecekan *string matching* lokasi gempa dengan isi *tweet*. Jika *string matching* lokasi gempa sama dengan isi *tweet*, *level* konfidensi informasi gempa adalah 8. Namun bila tidak, perangkat lunak melakukan pengecekan *string matching* untuk menemukan kota yang terkandung di dalam isi *tweet*.

Selanjutnya, dilakukan pencarian jarak antara kota yang terkandung di dalam isi *tweet* dengan lokasi gempa sebenarnya. Hal ini bertujuan untuk mengecek apakah jarak kota itu masih berada di sekitar kota lokasi dimana gempa terjadi. Dari jarak itu dicek apakah melebihi *threshold* yang ditentukan atau tidak. Bila jarak kurang dari *threshold*, maka *level* konfidensi informasi gempa adalah 7. Namun

bila melebihi *threshold*, maka *level* konfidensi informasi gempa adalah 6. Informasi ini lalu dikirim ke *server*.

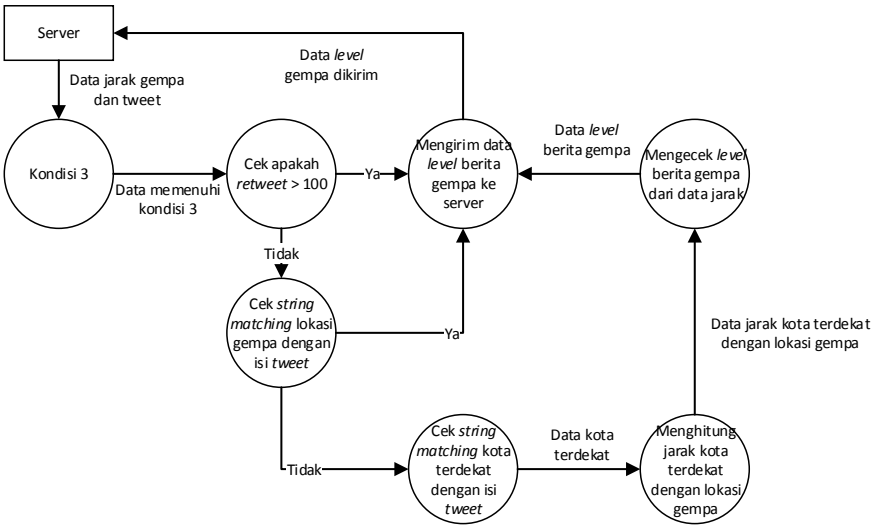


Gambar 3.12 Diagram Alir Data Menentukan *Level* Konfidensi Gempa (Kondisi 2)

3.6.3.3. Diagram Alir Data Menentukan *Level* Konfidensi Gempa (Kondisi 3)

Gambar 3.13 merupakan diagram alir data untuk menentukan *level* konfidensi gempa dengan kondisi 3. Hal pertama yang dilakukan adalah *server* mengecek apakah jarak tidak memenuhi kondisi 1 dan kondisi 2. Langkah berikutnya adalah mengecek apakah jumlah *retweet* dari *tweet* melebihi *threshold.* Mengecek jumlah *retweet* bertujuan apabila seseorang memiliki

banyak *follower* di Twitter namun tidak berada di dalam kawasan yang dekat dengan gempa. Jika jumlah *retweet* melebihi *threshold* maka *level* konfidensi informasi gempa adalah 5.



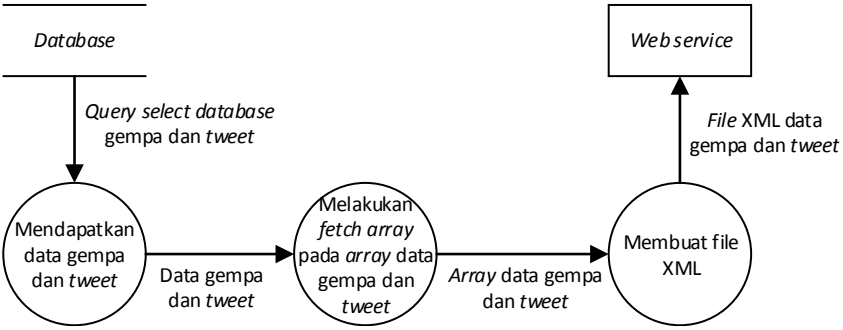
Gambar 3.13 Diagram Alir Data Menentukan *Level* konfidensi Gempa (Kondisi 3)

Untuk pengecekan selanjutnya sama dengan kondisi 2 dimana perangkat lunak melakukan pengecekan *string matching* lokasi gempa dengan isi *tweet*, dan pengecekan *string matching* untuk menemukan kota yang terkandung di dalam isi *tweet*. *Level* konfidensi untuk masing-masing informasi gempa berturut-turut dari hasil pengecekan *string matching* lokasi gempa dengan isi *tweet*, pengecekan *string matching* untuk menemukan kota yang terkandung di dalam isi *tweet* dengan nilai kurang dari *threshold*, dan pengecekan *string matching* untuk menemukan kota yang terkandung di dalam isi *tweet* dengan nilai melebihi *threshold* adalah 4, 3, dan 2. Adapun bila informasi gempa tidak memenuhi semua kondisi di atas maka *level* konfidensi informasi gempa adalah 1.

3.6.4. Diagram Alir Data *Web Service*

Langkah selanjutnya setelah perangkat lunak menentukan *level* konfidensi informasi gempa adalah melakukan *posting* informasi gempa dan *tweet* ke dalam *web service*. Gambar diagram alir data *web service* ditunjukkan oleh Gambar 3.14. Pertama-tama, *database* yang menyimpan informasi gempa dan *tweet* perangkat lunak melakukan *query select* dari *database* gempa dan *database tweet*.

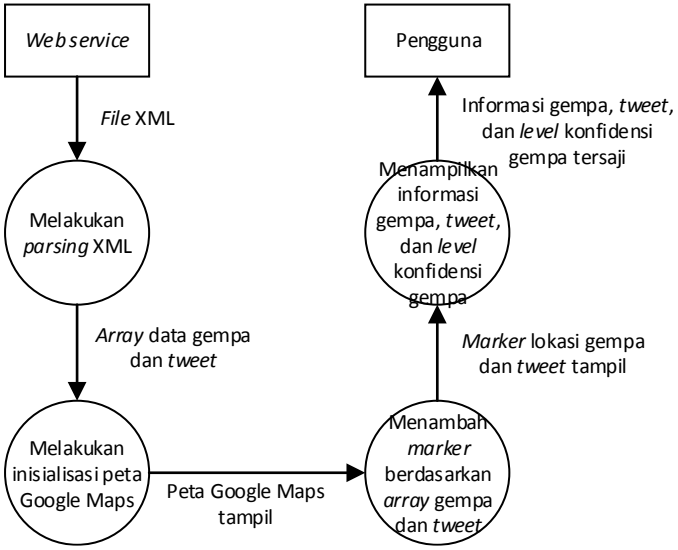
Selanjutnya, dari data gempa dan *tweet* yang telah di-*query* dilakukan *fetch array* untuk memasukkan setiap baris hasil *query* tersebut ke dalam sebuah *array*. Setelah melakukan proses *fetch array*, data gempa dan *tweet* dimasukkan masing-masing ke dalam sebuah *array*. Dari kedua *array* tersebut dibuat dua berkas berekstensi XML yang memuat informasi gempa dan *tweet*. Berkas tersebut diletakkan ke dalam *web service* untuk dapat dikonsumsi oleh situs *web* dan *smartphone* berbasis Android.



Gambar 3.14 Diagram Alir Data *Web service*

3.6.5. Diagram Alir Data Menampilkan Peta dan Informasi Gempa

Gambar 3.15 merupakan diagram alir data untuk menampilkan peta dan informasi kepada pengguna baik menggunakan situs *web* atau menggunakan *smartphone* berbasis Android. Pertama berkas XML *web service* di-*parsing* dengan menggunakan XML *parser* bawaan yang ada pada PHP dan *smartphone* berbasis Android. Setelah dilakukan *parsing*, data dari berkas XML berubah menjadi *array* informasi gempa dan *tweet*. Setelah data diletakkan ke dalam *array*, situs *web* dan *smartphone* Android melakukan inisialisasi Google Maps API untuk menampilkan peta dunia Google Maps.



Gambar 3.15 Diagram Alir Data Menampilkan Peta dan Informasi Gempa

Setelah peta Google Maps berhasil ditampilkan, langkah selanjutnya adalah mengasosiasikan *marker* yang ada pada Google

Maps dengan *array* yang berisi data informasi gempa dan *tweet*. Setelah *marker* yang menunjukkan lokasi gempa dan *tweet* berhasil ditampilkan, hal selanjutnya adalah menyajikan informasi gempa, *tweet* informasi gempa, dan *level* konfidensi informasi gempa.

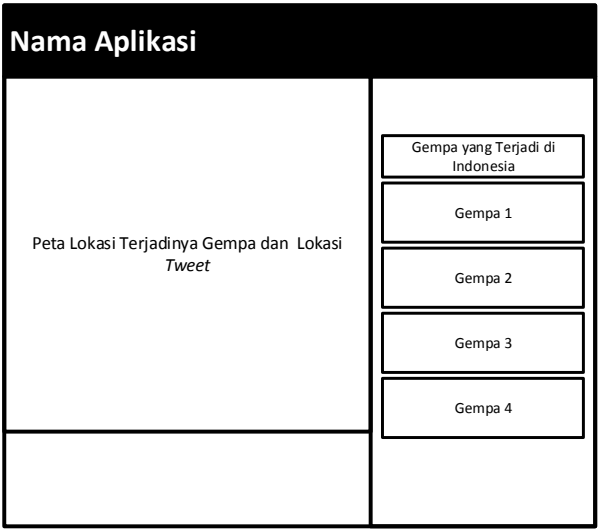
3.7. Rancangan Antarmuka

Pada Tugas Akhir ini, antarmuka perangkat lunak terdiri atas halaman situs *web* dan *activity* pada *smartphone* berbasis Android. Tujuan keduanya sama, yaitu untuk menampilkan peta lokasi terjadinya gempa, dan menyajikan informasi gempa dan *level* konfidensi gempa. Penjelasan rancangan antarmuka akan dijelaskan lebih lanjut pada subbab berikut ini.

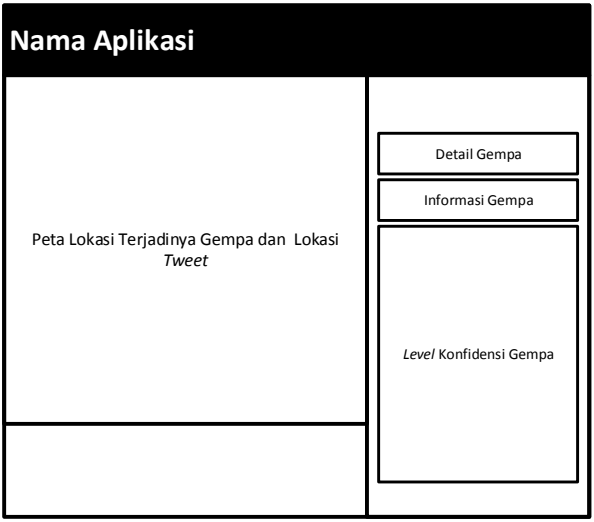
3.7.1. Rancangan Antarmuka Situs *Web*

Rancangan antarmuka situs *web* ini dapat dilihat pada Gambar 3.16 dan Gambar 3.17. Gambar 3.16 merupakan halaman antarmuka utama situs *web* ini. Ketika pengguna mengakses situs *web* ini, terdapat sebuah panel dan sebuah *header* bertuliskan nama perangkat lunak. Halaman antarmuka utama situs *web* ini terdiri atas dua bagian. Bagian pertama di sebelah kiri menunjukkan peta lokasi semua gempa yang terjadi dalam satu hari. Di bagian sebelah kanan terdapat panel yang menunjukkan detail dari masing-masing gempa.

Gambar 3.17 merupakan halaman detail informasi dari masing-masing gempa. Pada Gambar 3.17, halaman situs *web* juga terbagi ke dalam dua bagian. Bagian pertama terletak di sebelah kiri untuk menampilkan peta lokasi terjadinya gempa dan lokasi *tweet*, dan satu bagian lagi terletak di sebelah kanan untuk menyajikan informasi gempa serta *level* konfidensi gempa.



Gambar 3.16. Rancangan Antarmuka Utama Situs *Web*



Gambar 3.17 Rancangan Antarmuka Situs *Web* Detail Gempa

3.7.2. Rancangan Antarmuka Android

Pada Gambar 3.18, terdapat sebuah diagram *activity* yang menampilkan peta lokasi terjadinya gempa. Selain itu terdapat sebuah tombol '*Refresh*' untuk memuat data lokasi terjadinya gempa dan *tweet*. Selain tombol '*Refresh*' juga terdapat tombol informasi untuk menampilkan informasi gempa dan tombol konfidensi *level* konfidensi gempa.

Gambar 3.18 Rancangan Antarmuka di *Smartphone* Android

BAB IV
IMPLEMENTASI

Bab ini berisi penjelasan mengenai implementasi perangkat lunak yang dikembangkan. Implementasi perangkat lunak merupakan bentuk realisasi dari perancangan perangkat lunak yang telah dijelaskan pada bab sebelumnya. Adapun hal-hal yang dibahas dalam bab ini adalah lingkungan implementasi perangkat lunak, implementasi perangkat lunak disertai dengan *pseudocode* perangkat lunak, dan *screenshot* hasil implementasi perangkat lunak.

4.1. Lingkungan Implementasi

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

4.1.1. Lingkungan Implementasi Perangkat Keras

Lingkungan implementasi perangkat keras yang digunakan untuk mengembangkan perangkat lunak ini terdiri atas sebuah *notebook* dan sebuah *smartphone* berbasis Android. Adapun spesifikasi dari perangkat lunak tersebut adalah sebagai berikut:

- Notebook ASUS X450JF
  - Windows 8.1 Pro 64bit
  - Prosesor Intel® Quad Core™ i7-4700HQ CPU @2.40 GHz, dan
  - Memori 4GB DDR3
- SONY Xperia™ SL LT26ii
  - Prosesor Qualcomm MSM8260 Snapdragon Dual Core CPU @1.7 GHz, dan
  - Memori 1GB

4.1.2. Lingkungan Implementasi Perangkat Lunak

Lingkungan implementasi perangkat lunak yang digunakan untuk menunjang pengembangkan perangkat lunak ini terdiri atas sebuah sistem operasi, dua buah IDE dan beberapa pustaka. Adapun lingkungan implementasi perangkat lunak tersebut adalah sebagai berikut:

- Microsoft Windows 8.1 Pro 64 bit sebagai sistem operasi.
- Microsoft Visio 2013 untuk melakukan perancangan diagram alir data dan perancangan antarmuka perangkat lunak situs *web* dan Android.
- StarUML untuk membuat diagram kasus penggunaan perangkat lunak.
- Power Designer v15.0 untuk membuat rancangan *database* informasi gempa dan *database tweet* informasi gempa.
- Eclipse Juno v4.2.2 untuk membuat logika dan fungsionalitas perangkat lunak dalam bahasa pemrograman Java.
- Android Development Tools v22.6.2 untuk membuat implementasi perangkat lunak pada *smartphone* berbasis Android.
- MySQL v5.6.16 sebagai RDBMS (*Relational Database Management System*) untuk menyimpan data informasi gempa dan *tweet*.
- Apache v2.4.7 sebagai *platform web server* untuk menjalankan situs *web*.
- Notepad++ untuk membuat situs *web* dan logika dalam bahasa pemrograman PHP, JavaScript, dan Jquery.
- Pustaka GSON digunakan untuk melakukan *parsing* JSON dari situs *web* Earthquake Hazards Program.
- Pustaka Twitter4J digunakan untuk mengambil informasi yang dibutuhkan perangkat lunak dari Twitter API.
- Pustaka Google Maps API digunakan untuk menampilkan peta lokasi terjadinya gempa pada situs *web* dan *smartphone* berbasis Android.

4.2. Implementasi Perangkat Lunak

Implementasi pada perangkat lunak terbagi menjadi implementasi *server* sebagai wadah untuk menampung fungsi dan logika dari perangkat lunak, implementasi situs *web*, dan implementasi pada *smartphone* berbasis Android. Setiap bagian implementasi akan dijelaskan pada subbab berikut ini.

4.2.1. Implementasi JSON *Parser*

Gambar 4.1 menunjukkan *pseudocode* implementasi JSON *parser* berdasarkan alir diagram dari bab sebelumnya.

Prosedur 1. JSON Parser

Input: URL situs *web* Earthquake Hazards Program

```
1   Inisialisasi URL
2   response ⇦ decode JSON dari URL
3   FOR setiap baris dalam response
4     s ⇦ response.getPlace()
5     IF s is 'Indonesia'
6         magnitude ⇦ response.getMagnitude()
7         place ⇦ response.getPlace()
8         time ⇦ response.getTime()
9         latitude ⇦ response.getLatitude()
10        longitude ⇦ response.getLongitude()
11    ENDIF
12  Mengirim output ke server
```

Output: Informasi gempa terkirim ke *server*

Gambar 4.1 *Pseudocode* Implementasi JSON *Parser*

Proses ini membutuhkan *input* berupa URL situs *web* Earthquake Hazards Program dan mengeluarkan *output* informasi data gempa yang dikirim kembali ke *server*.

Penjelasan dari *pseudocode* JSON *parser* adalah sebagai berikut:

1. Melakukan inisialisasi URL dengan URL situs *web* Earthquake Hazards Program.

2. Memanggil fungsi fromJSON dari pustaka GSON untuk melakukan *decode* JSON dari URL yang telah dibaca menjadi objek di dalam Java.
3. Untuk setiap lokasi gempa di dalam JSON, jika mengandung kata 'Indonesia', maka perangkat lunak mengambil informasi gempa yang terdiri atas *magnitude* gempa, lokasi terjadinya gempa, waktu terjadinya gempa, dan koordinat *geolocation* dalam *latitude* dan *longitude*.
4. Mengirim informasi gempa ke *server*.

4.2.2. Implementasi Pencarian *Tweet*

Implementasi pencarian *tweet* merupakan bentuk implementasi berdasarkan diagram alir data pencarian *tweet* yang dijelaskan pada bab sebelumnya. Implementasi dalam bentuk *pseudocode* pencarian *tweet* dapat dilihat pada Gambar 4.2. *Pseudocode* ini tidak memiliki *input*, namun memiliki *output* yaitu *array* berisi informasi *tweet* seperti *username*, isi *tweet,* waktu posting *tweet*, dan koordinat *geolocation tweet* dalam *latitude* dan *longitude*.

Prosedur 2. Pencarian *tweet*

Input: -

```
1   Inisialisasi Twitter factory
2   Query ⇦ query
3   Result ⇦ TwitterSearch(Query)
4   FOR setiap tweet pada result
5      IF tweet.getGeolocation tidak null
6         username ⇦ tweet.getUsername()
7         isiTweet ⇦ tweet.getText()
8         time ⇦ tweet.getTime()
9         geolocation ⇦ tweet.getGeolocation()
10     ENDIF
11  Mengirim output to server
12
```

Output: Informasi gempa terkirim ke *server*

Gambar 4.2 *Pseudocode* Implementasi Pencarian *Tweet*

Penjelasan dari *pseudocode* pencarian *tweet* adalah sebagai berikut:

1. Inisialisasi Twitter *factory* dengan sebelumnya melakukan autentikasi *access token* untuk menggunakan fitur dari Twitter Search API.
2. Menentukan kata kunci pencarian *query*.
3. Memasukkan hasil pencarian ke dalam *result*.
4. Untuk setiap *tweet* pada *result*, jika *geolocation tweet* tidak *null*, maka perangkat lunak mengambil informasi pada *tweet* seperti *username*, isi *tweet*, waktu *posting tweet*, dan koordinat *geolocation* dalam *latitude* dan *longitude*.
5. Mengirim informasi gempa ke *server.*

4.2.3. Implementasi Menghitung *Level* Konfidensi Informasi Gempa

Gambar 4.3 merupakan *pseudocode* bentuk implementasi dari diagram alir data menghitung *level* konfidensi informasi gempa. *Pseudocode* ini tidak memiliki *input* namun memiliki *output* yaitu informasi gempa dan data *tweet* yang dikirim ke dalam *database*.

Penjelasan dari *pseudocode* menghitung *level* konfidensi *tweet* informasi gempa adalah sebagai berikut:

1. Mengambil data informasi gempa dari proses JSON *parser*.
2. Mengambil data *tweet* informasi gempa dari proses pencarian *tweet*.
3. Jika selisih waktu posting dan waktu terjadinya gempa kurang dari satu jam maka menghitung jarak antara lokasi terjadinya gempa dan lokasi *tweet* dengan menggunakan rumus Haversine.
4. Menghitung *level* konfidensi informasi gempa dengan prosedur menentukan *level* konfidensi informasi gempa.
5. Menyimpan *level* konfidensi informasi gempa ke dalam array.
6. Menyimpan informasi gempa ke *database*.
7. Menyimpan data *tweet* informasi gempa ke *database*.

Prosedur 3. Menghitung *level* konfidensi gempa	
Input: -	

1
2
3
4
5
6
7
8
9
10
11
12
13
	Load data informasi gempa
	Load data *tweet* informasi gempa
	IF (selisih waktu informasi gempa dan *tweet* kurang dari satu jam)
	jarak ⇦ Haversine(lokasi gempa, lokasi *tweet*)
	level ⇦ call check*Level*(jarak, isi*Tweet*, lokasi gempa, re*tweet*)
	array*Level*[*level*-1]++
	ENDIF
	Menyimpan informasi gempa ke *database*
	Menyimpan data *tweet* informasi gempa ke *database*

Output: Informasi gempa dan data *tweet* ke *database*	

Gambar 4.3 *Pseudocode* Implementasi Menghitung *Level* Konfidensi Informasi Gempa

4.2.4. Implementasi Rumus Haversine

Rumus Haversine digunakan untuk menghitung jarak diantara lokasi gempa dan lokasi *tweet*. Adapun gambar *pseudocode* dari implementasi rumus Haversine dapat dilihat pada Gambar 4.4.

Prosedur 4. Rumus Haversine	
Input: lokasi gempa, lokasi *tweet*	
1 2 3 4 5 6	dLat ⇦ selisih latitude lokasi gempa dan lokasi *tweet* dLon ⇦ selisih longitude lokasi gempa dan lokasi *tweet* a ⇦ sin(dLat / 2) * sin(dLat / 2) + sin(dLon / 2) * sin(dLon / 2) * cos(lat1) * cos(lat2) c ⇦ 2 * asin(sqrt(a))
Output: Informasi gempa dan data *tweet* ke *database*	

Gambar 4.4 *Pseudocode* Implementasi Rumus Haversine

Penjelasan dari *pseudocode* rumus Haversine adalah sebagai berikut:

1. Menghitung selisih *latitude* lokasi gempa dan lokasi *tweet*.
2. Menghitung selisih *longitude* gempa dan lokasi *tweet*.
3. Menghitung jarak berdasarkan selisih *latitude* dan *longitude*.

4.2.5. Implementasi Menentukan *Level* Konfidensi Informasi Gempa (Kondisi 1)

Implementasi menentukan *level* konfidensi informasi gempa merupakan bentuk implementasi dari diagram alir data menentukan *level* konfidensi informasi gempa. *Input* yang dibutuhkan adalah jarak diantara lokasi gempa dan *tweet*, lokasi gempa, isi *tweet*, dan jumlah *retweet* dari *tweet*. *Output* yang dikeluarkan adalah informasi *level* konfidensi informasi gempa. Adapun untuk *pseudocode* dari implementasi ini dapat dilihat pada Gambar 4.5.

Prosedur 5. Menentukan *level* konfidensi gempa (kondisi 1)	
Input: jarak antara lokasi gempa dan *tweet*, lokasi gempa, isi *tweet*, *retweet*	
1	IF jarak kurang dari threshold1
2	*Level* ⇦ 9
3	ELSE kondisi 2
4	ENDIF
5	Mengirim output ke *server*
6	
Output: Informasi *level* konfidensi *tweet* informasi gempa	

Gambar 4.5 *Pseudocode* Implementasi Menentukan *Level* Konfidensi Gempa (Kondisi 1)

Penjelasan dari implementasi menentukan *level* konfidensi informasi gempa dengan kondisi 1 adalah sebagai berikut:

1. Jika jarak antara lokasi gempa dan *tweet* kurang dari *threshold*, maka *level* konfidensi informasi gempa adalah 9 (paling tinggi).

2. Jika tidak memenuhi kondisi 1, maka menuju kondisi 2.
3. Mengirim informasi *level* konfidensi *tweet* informasi gempa.

4.2.6. Implementasi Menentukan *Level* Konfidensi Informasi Gempa (Kondisi 2)

Implementasi *pseudocode* dari menentukan *level* konfidensi informasi gempa dengan kondisi 2 dapat dilihat pada Gambar 4.6. *Input* yang diperlukan adalah jarak antara lokasi gempa dan *tweet*, lokasi gempa, isi *tweet*, dan jumlah *retweet* dari *tweet*. *Output* yang keluar merupakan informasi *level* konfidensi *tweet* informasi gempa.

Prosedur 6. Menentukan *level* konfidensi gempa (kondisi 2)

Input: jarak antara lokasi gempa dan *tweet*, lokasi gempa, isi *tweet*, *retweet*

```
1   IF jarak kurang dari threshold2
2     IF string matching lokasi gempa isi tweet = true
3       Level ⇦ 8
4     ELSE
5       string matching nama kota dengan isi tweet
6       menghitung jarak kota dengan lokasi gempa
7       IF jarak kota kurang dari threshold
8         Level ⇦ 7
9       ELSE
10        Level ⇦ 6
11      ENDIF
12    ENDIF
13  ELSE kondisi 3
14  Mengirim output ke server
```

Output: Informasi *level* konfidensi *tweet* informasi gempa

Gambar 4.6 *Pseudocode* Implementasi Menentukan *Level* Konfidensi Gempa (Kondisi 2)

Penjelasan dari implementasi menentukan *level* konfidensi informasi gempa dengan kondisi 1 adalah sebagai berikut:

1. Jika jarak lokasi gempa dan *tweet* kurang dari *threshold 2,*

2. Melakukan pengecekan *string matching* lokasi gempa dengan isi *tweet*.
3. Jika memuat *string* nama kota lokasi gempa, maka *level* konfidensi informasi gempa adalah 8.
4. Jika tidak memuat *string* nama kota lokasi gempa, maka melakukan pengecekan *string matching database* nama kota dengan isi *tweet*.
5. Jika pada isi *tweet* terdapat salah satu nama kota, maka menghitung jarak antara kota tersebut dengan lokasi terjadinya gempa.
6. Jika jarak kurang dari *threshold*, *level* konfidensi informasi gempa adalah 7. Jika melebihi *threshold*, *level* konfidensi informasi gempa adalah 6.
7. Selain itu jika jarak antara lokasi gempa dengan *tweet* melebihi *threshold* 2, maka menuju kondisi 3.
8. Mengirim informasi *level* konfidensi *tweet* informasi gempa.

4.2.7. Implementasi Menentukan *Level* Konfidensi Informasi Gempa (Kondisi 3)

Gambar 4.7 merupakan bentuk implementasi dari menentukan *level* konfidensi informasi gempa dengan kondisi 3. Adapun *input* untuk implementasi ini adalah jarak antara lokasi gempa dan *tweet*, lokasi gempa, isi *tweet*, dan jumlah *retweet* dari *tweet*. Sedangkan untuk *output* adalah informasi *level* konfidensi *tweet* informasi gempa. Adapun penjelasan dari implementasi menentukan *level* konfidensi *tweet* informasi gempa dengan kondisi 3 adalah sebagai berikut:

1. Jika jumlah re*tweet* melebihi *threshold*, maka *level* konfidensi *tweet* informasi gempa adalah 5.
2. Selain itu jika jumlah re*tweet* kurang dari *threshold*, maka melakukan pengecekan *string matching* lokasi gempa dengan isi *tweet*.
3. Jika memuat *string* nama kota lokasi gempa, maka *level* konfidensi informasi gempa adalah 8.

4. Jika tidak memuat *string* nama kota lokasi gempa, maka melakukan pengecekan *string matching database* nama kota dengan isi *tweet*.
5. Jika pada isi *tweet* terdapat salah satu nama kota, maka menghitung jarak antara kota tersebut dengan lokasi terjadinya gempa
6. Jika jarak kurang dari *threshold*, *level* konfidensi informasi gempa adalah 7. Jika melebihi *threshold*, *level* konfidensi informasi gempa adalah 6.
7. Selain itu jika jarak antara lokasi gempa dengan *tweet* melebihi *threshold* 2, maka menuju kondisi 3.
8. Mengirim informasi *level* konfidensi *tweet* informasi gempa.

Prosedur 7. Menentukan *level* konfidensi gempa (kondisi 3)

Input: jarak antara lokasi gempa dan *tweet*, lokasi gempa, isi *tweet*, *retweet*

```
1  IF retweet melebihi threshold
2     Levek ⇦ 5
3  ELSE
4    IF string matching lokasi gempa isi tweet = true
5      Level ⇦ 4
6    ELSE
7      string matching nama kota dengan isi tweet
8      menghitung jarak kota dengan lokasi gempa
9      IF jarak kota kurang dari threshold
10       Level ⇦ 3
11     ELSE
12       Level ⇦ 2
13     ENDIF
14   ENDIF
15 ELSE Level ⇦ 1
16 Mengirim output ke server
```

Output: Informasi *level* konfidensi *tweet* informasi gempa

Gambar 4.7 *Pseudocode* Implementasi Menentukan *level* konfidensi gempa (Kondisi 3)

4.2.8. Implementasi *Web service*

Implementasi *web service* ditunjukkan oleh Gambar 4.8. Implementasi *web service* merupakan bentuk realisasi dari diagram alir data yang telah dijelaskan pada bab sebelumnya. *Input* dari implementasi *web service* adalah *database* yang berisi informasi gempa dan *database* yang berisi *tweet* informasi gempa, sedangkan *output* dari implementasi *web service* adalah sebuah berkas XML yang berisi *database* informasi gempa dan *tweet* informasi gempa di dalam *web service*.

Prosedur 8. *Web* service	
Input: *database* informasi gempa dan *database tweet* informasi gempa	
1	Membuka koneksi ke *database*
2	Load *database* informasi gempa
3	Load *database tweet* informasi gempa
4	Query ⇦ select * from table
5	WHILE row ⇦ fetchArray(Query)
6	Result ⇦ row
7	ENDWHILE
8	Menutup koneksi *database*
9	Membuat berkas XML
10	Mengirim berkas XML ke *web service*
11	
Output: Berkas XML berisi *database* informasi gempa dan *tweet* informasi gempa	

Gambar 4.8 *Pseudocode* Implementasi *Web Service*

Penjelasan dari *pseudocode* implementasi *web service* adalah sebagai berikut:

1. Membuka koneksi ke *database*.
2. Mendapatkan *database* informasi gempa
3. Mendapatkan *database tweet* informasi gempa
4. Melakukan deklarasi query '*select*' untuk menyeleksi semua baris dalam *database*.

5. Untuk setiap baris, dilakukan fetchArray() terhadap hasil seleksi *query*.
6. Menutup koneksi *database*.
7. Membuat sebuah berkas XML yang berisi array hasil fetchArray().
8. Menutup koneksi *database*.
9. Mengirim berkas XML ke dalam *web service*.

4.2.9. Implementasi Menampilkan Peta dan Informasi Gempa

Gambar 4.9 merupakan implementasi dari diagram alir data untuk menampilkan peta dan informasi gempa. *Input* dari implementasi menampilkan peta adalah sebuah berkas XML berisi *database* informasi gempa dan *tweet* informasi gempa di dalam *web service*. *Output* dari implementasi menampilkan peta dan informasi ini adalah sebuah peta lokasi terjadinya gempa, dan menyajikan informasi gempa beserta *tweet* informasi gempa kepada *end user*. Adapun tampilan untuk menampilkan peta dan informasi gempa berupa sebuah situs *web* dan sebuah aplikasi *smartphone* berbasis Android.

Penjelasan dari implementasi untuk menampilkan peta dan informasi gempa adalah sebagai berikut:

1. Melakukan *parsing* XML dari berkas XML yang berisi *database* informasi gempa dan *database tweet* informasi gempa.
2. Melakukan inisialisasi Google Maps untuk membuka peta dari Google Maps.
3. Untuk setiap baris sepanjang *array* yang terdapat pada *database* informasi gempa dan l data *tweet*, maka
4. Menambahkan *marker* berdasarkan lokasi terjadinya gempa dan *tweet*.
5. Menampilkan informasi gempa.
6. Menampilkan *tweet* informasi gempa.
7. Menampilkan *level* konfidensi *tweet* informasi gempa.

Prosedur 9. Menampilkan peta dan informasi gempa	
Input: Berkas XML berisi *database* informasi gempa dan *database tweet* informasi gempa	
1	array ⇦ parseXML(XML)
2	Inisialisasi Google Maps
3	WHILE array ⇦ jumlah array
4	*marker*Array ⇦ array.latitude, array.longitude
5	ENDWHILE
6	Menampilkan informasi gempa
7	Menampilkan *tweet* informasi gempa
8	Menampilkan *level* konfidensi *tweet* informasi gempa
Output: Informasi gempa, tweet informasi gempa, dan peta lokasi terjadinya gempa	

Gambar 4.9 *Pseudocode* Menampilkan Peta dan Informasi Gempa

4.3. Implementasi Antarmuka Perangkat Lunak

Implementasi antarmuka perangkat lunak ini terdiri atas situs *web* dan perangkat lunak *smartphone* berbasis Android. Kedua bagian ini akan dijelaskan pada masing-masing subbab.

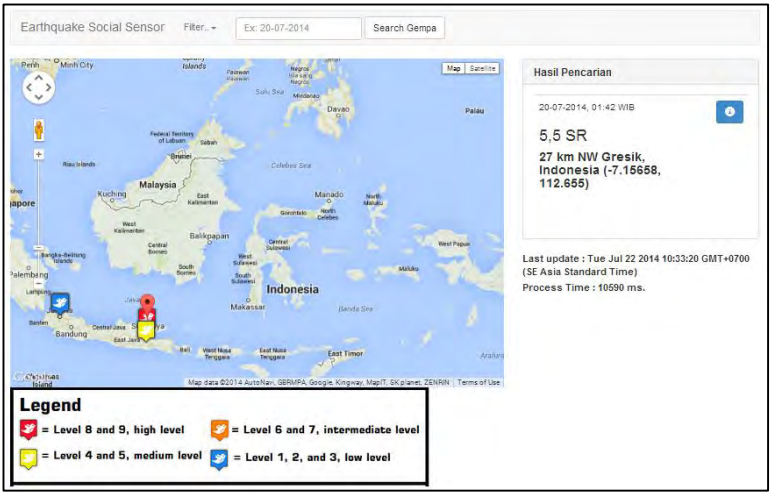
4.3.1. Implementasi Antarmuka Situs *Web*

Gambar 4.10 merupakan *screenshot* hasil implementasi halaman antarmuka utama situs *web*. Gambar tersebut merupakan halaman antarmuka utama pada saat pengguna mengakses halaman situs *web*. Pada halaman situs *web* tersebut terdapat dua bagian yaitu sebelah kiri yang berisi peta Indonesia. *Marker* penunjuk berwarna merah menunjukkan lokasi terjadinya gempa dan *marker* bergambar *tweet* menjelaskan informasi gempa. Selain itu terdapat legenda yang menjelaskan *level* konfidensi berdasarkan warna *marker tweet*. Adapun penjelasan warna *marker tweet* adalah sebagai berikut:

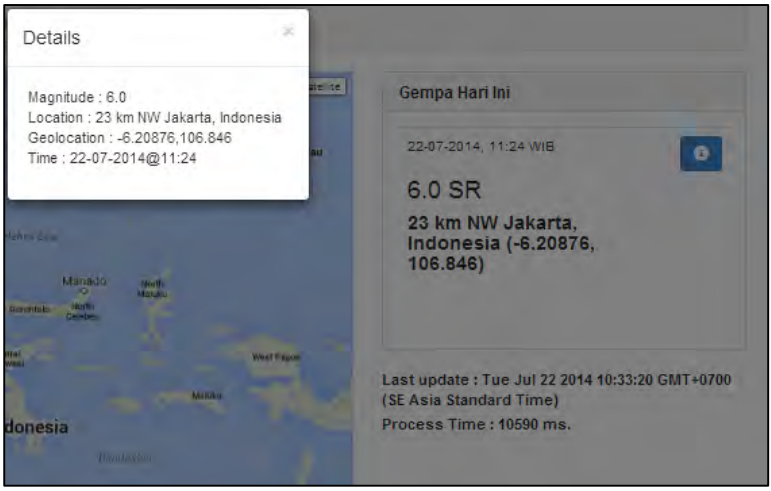
1. Marker *tweet* merah menunjukkan level konfidensi informasi gempa memiliki nilai 8 dan 9. Ini menunjukkan bahwa informasi gempa tersebut memiliki nilai konfidensi terhadap situs *web* yang tinggi.

2. Marker *tweet* oranye menunjukkan level konfidensi informasi gempa memiliki nilai 6 dan 7. Ini menunjukkan bahwa informasi gempa tersebut memiliki nilai konfidensi terhadap situs *web* yang lebih lanjut.
3. Marker *tweet* oranye menunjukkan level konfidensi informasi gempa memiliki nilai 4 dan 5. Ini menunjukkan bahwa informasi gempa tersebut memiliki nilai konfidensi terhadap situs *web* yang sedang.
4. Marker *tweet* biru menunjukkan level konfidensi informasi gempa memiliki nilai 1 sampai 3. Ini menunjukkan bahwa informasi gempa tersebut memiliki nilai konfidensi terhadap situs *web* yang rendah.

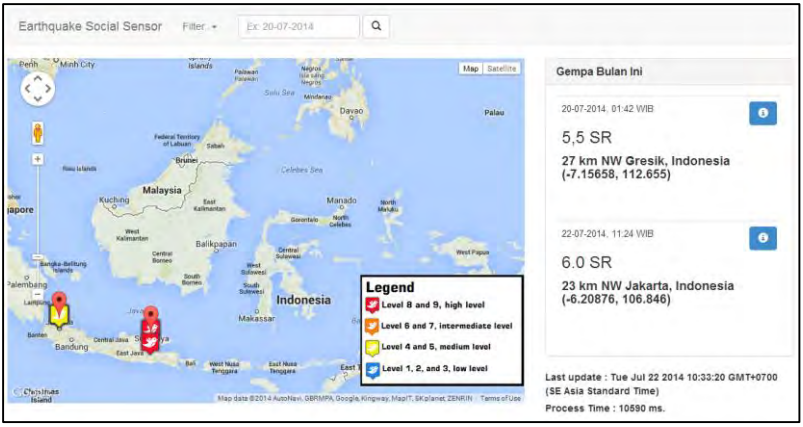
Di sebelah kanan terdapat tabel dan tombol yang bila ditekan menampilkan panel berisi informasi gempa dari masing-masing gempa. Gambar panel informasi gempa tersebut dapat dilihat pada Gambar 4.11.



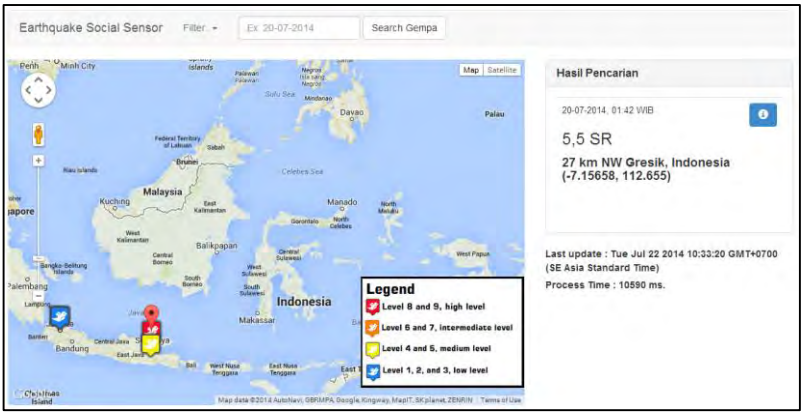
Gambar 4.10 Implementasi Halaman Antarmuka Utama Situs *Web*



Gambar 4.11 Panel Informasi Gempa pada Halaman Antarmuka Utama Situs *Web*



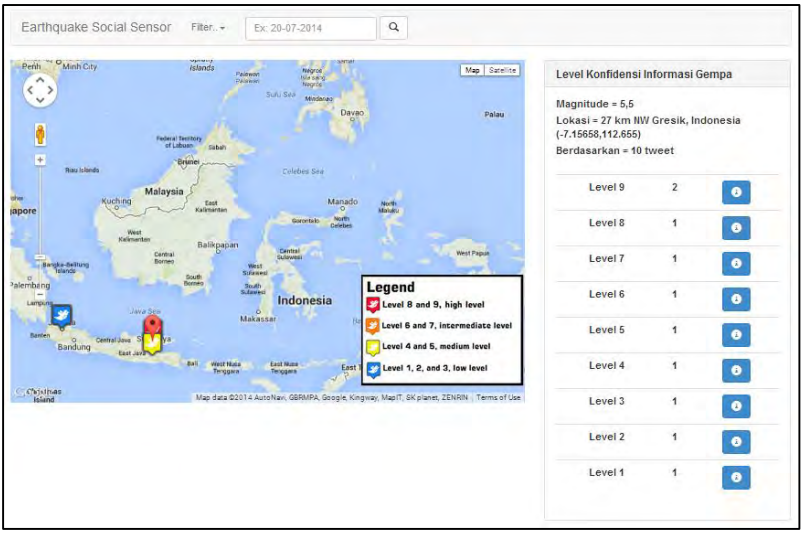
Gambar 4.12 Implementasi Halaman *Filter By* Berdasarkan Gempa yang Terjadi Bulan Ini



Gambar 4.13 Implementasi Halaman Hasil Pencarian Berdasarkan Tanggal Terjadinya Gempa

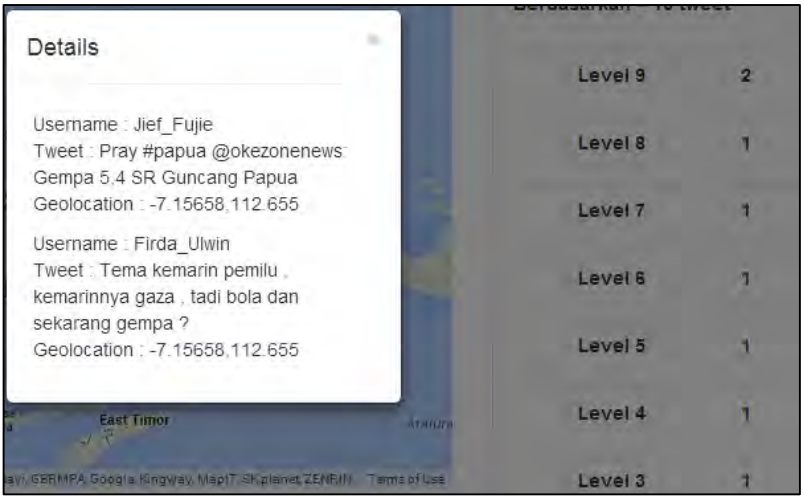
Adapun fitur yang ada di dalam halaman utama adalah sebagai berikut:

1. *Filter by*, merupakan fitur untuk melakukan pencarian gempa berdasarkan *filter*. *Filter* tersebut terdiri dari gempa yang terjadi hari ini, gempa yang terjadi bulan ini, dan gempa yang terjadi tahun ini. Contoh pencarian *filter by* dapat dilihat pada Gambar 4.12. Gambar 4.12 menunjukkan gempa yang terjadi dalam bulan ini.
2. *Search*, merupakan fitur untuk melakukan pencarian gempa berdasarkan kata kunci. Kata kunci yang digunakan yaitu tanggal terjadinya gempa. Hasil pencarian gempa tersebut dapat dilihat pada Gambar 4.13. Gambar 4.13 menunjukkan hasil pencarian gempa berdasarkan tanggal terjadinya gempa yaitu pada tanggal 20 Juli 2014.



Gambar 4.14 Implementasi Halaman Antarmuka Detail Gempa

Gambar 4.14 merupakan halaman antarmuka detail gempa untuk menunjukkan informasi gempa dan *level* konfidensi pada gempa yang bersangkutan. Halaman antarmuka ini terbagi menjadi dua, di sebelah kiri terdapat peta Indonesia, *marker* gempa, dan *marker tweet*, dan di sebelah kanan terdapat tabel *level* konfidensi gempa dan tombol yang bila ditekan menampilkan panel yang berisi *tweet* dengan *level* konfidensi bersangkutan. Panel informasi *tweet* tersebut dapat dilihat pada Gambar 4.15. Selain itu pada halaman detail gempa, terdapat tabel *log* yang berisi seluruh *tweet* yang divalidasi berdasarkan informasi gempa yang bersangkutan. Tabel *log* ini diperlukan untuk melakukan *monitoring* terhadap *tweet* yang masuk dan divalidasi. Gambar tabel *log* tersebut dapat dilihat pada Gambar 4.16.



Gambar 4.15 Panel Detail *Tweet* Halaman Antarmuka

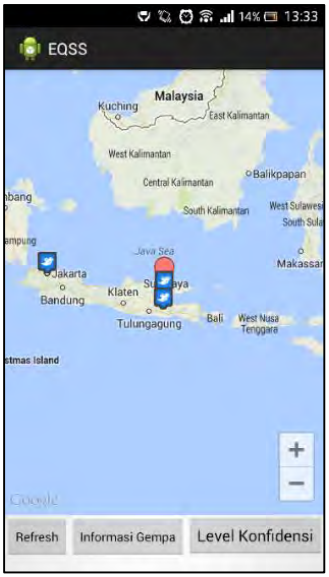
Log

ID Gempa	Username	Isi Tweet	Date	Time	Koordinat	Level	URL
0	Jief_Fujie	Pray #papua @okezonenews: Gempa 5,4 SR Guncang Papua	20-07-2014	01:42	(-7.15658, 112.655)	9	http://twitter.com
1	darinholic	Gempa di Keerom. Pantes kerasa bergelinjang tadi.	20-07-2014	01:42	(-7.98391, 112.621)	8	http://twitter.com
2	AzwanVL	Namun demikian ketika Dajjal datang ke pergunungan di luar kota Madinah, kota Madinah bergoncang seperti gempa bumi.	20-07-2014	01:42	(-7.98391, 112.621)	7	http://twitter.com
3	EarthQuake_id	#BMKG Magnitudo: 5.4 SR, 76 km BaratDaya KEEROM-PAPUA:	20-07-2014	01:42	(-7.98391, 112.621)	6	http://twitter.com
4	AlvaroMolina365	[BMKG] Magnitudo: 5.4 SR, 76 km BaratDaya KEEROM-PAPUA:	20-07-2014	01:42	(-6.20876, 106.846)	5	http://twitter.com
5	hanumayu29	Ini berasa..... "@infoBMKG: Gempa	20-07-2014	01:42	(-6.20876, 106.846)	4	http://twitter.com

Gambar 4.16 Halaman Tabel *Log* Informasi Gempa

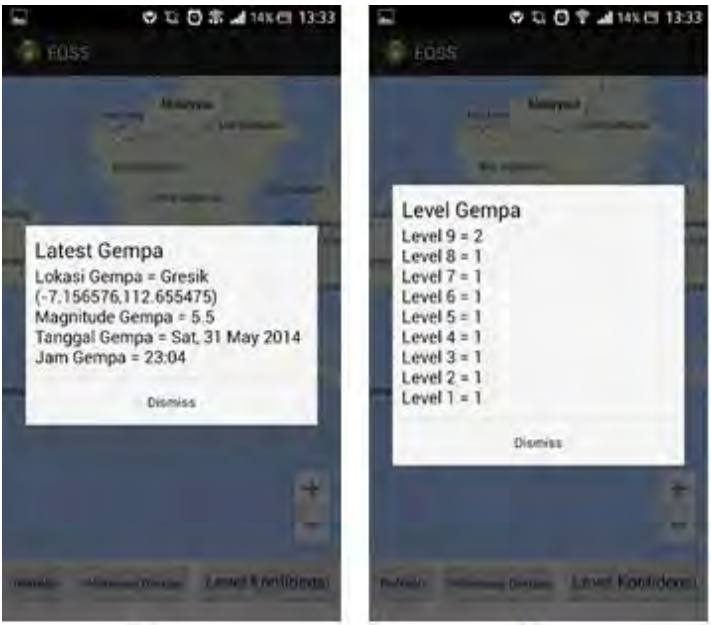
4.3.2. Implementasi Antarmuka Android

Gambar 4.17 merupakan halaman antarmuka hasil implementasi perangkat lunak pada *smartphone* berbasis Android.



Gambar 4.17 Halaman Antarmuka Utama Android

Pada halaman ini terdapat peta Indonesia dan sebuah tombol '*Refresh*'. Tombol '*Refresh'* ini bila ditekan akan mengambil data XML yang berisi informasi gempa dan *tweet*. Selain itu terdapat tombol untuk menampilkan informasi gempa, dan tombol untuk menampilkan *level* konfidensi gempa. Kedua informasi ini ditampilkan dalam sebuah panel. Panel informasi gempa dan panel *level* konfidensi gempa ditampilkan pada dan Gambar 4.18(a) dan Gambar 4.18(b) secara berturut-turut.



Gambar 4.18 (a) Panel Informasi Gempa (b) Panel *Level* Konfidensi Informasi Gempa

BAB V
PENGUJIAN DAN EVALUASI

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

5.1. Lingkungan Uji Coba

Lingkungan uji coba menjelaskan mengenai lingkungan dimana perangkat lunak diuji. Lingkungan uji coba ini terdiri atas lingkungan uji coba perangkat keras dan alat kakas. Lingkungan uji coba perangkat keras terdiri atas sebuah *notebook* yang berfungsi sebagai *server*, dan sebuah *smartphone* yang digunakan oleh pengguna dan berfungsi untuk melakukan *request* data. Spesifikasi lingkungan ujicoba perangkat keras dan alat kakas adalah sebagai berikut:

- Notebook ASUS X450JF
  Spesifikasi perangkat keras
  - Intel® Quad Core i7-4700HQ CPU @2.40GHz, dan
  - Memori 4GB DDR3

  Spesifikasi alat kakas
  - Windows 8.1 Pro 64bit
  - Eclipse Juno v4.2.2. sebagai IDE untuk membuat *server* perangkat lunak.
  - Android Development Tools untuk mengembangkan perangkat lunak *smartphone* berbasis Android.
  - MySQL v5.6.16 sebagai RDBMS (*Relational Database Management System*) untuk menyimpan data informasi gempa dan *tweet*.
  - Apache v2.4.7 sebagai *platform web server* untuk menjalankan situs *web*.

  - Google Chrome untuk menampilkan peta dan informasi gempa.

- SONY Xperia™ SL LT26ii
  - Sistem Operasi Android v4.1.2 Jelly Bean.
  - Prosesor Qualcomm MSM8260 Snapdragon Dual Core CPU @1.7 GHz.
  - Memori 1GB.
  - Memori interal 16GB.
  - Speed: HSDPA, 7.2 Mbps.

5.2. Skenario Uji Coba

Skenario uji coba merupakan skenario untuk melakukan pengujian terhadap perangkat lunak yang dikembangkan. Skenario uji coba terbagi menjadi dua bagian yaitu uji coba fungsionalitas untuk menguji fungsi-fungsi dari perangkat lunak yang telah dijelaskan sebelumnya dan uji coba performa untuk menguji apakah performa perangkat lunak dapat diandalkan.

5.2.1. Pengujian Fungsionalitas

Pengujian fungsionalitas perangkat lunak dilakukan untuk menguji apakah fungsionalitas perangkat lunak berjalan sesuai dengan yang diharapkan. Adapun fungsionalitas perangkat lunak akan dijelaskan lebih lanjut pada subbab berikut.

5.2.1.1. Pengujian Melakukan *Parsing* JSON *Parser*

Pengujian melakukan *parsing* JSON *parser* merupakan pengujian terhadap kemampuan perangkat lunak dalam melakukan *parsing* terhadap JSON dengan menggunakan pustaka GSON. Tujuan uji coba melakukan *parsing* JSON *parser* adalah untuk menguji apakah fungsionalitas fungsi ini berjalan sesuai dengan

rancangan. Rincian prosedur uji coba melakukan *parsing* JSON *parser* dapat dilihat pada Tabel 5.1.

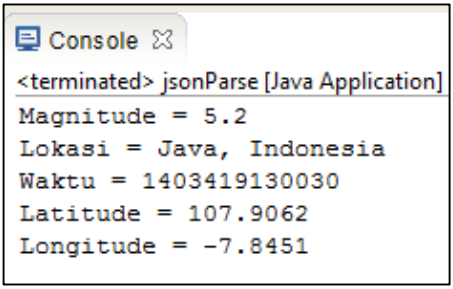
Tabel 5.1 Prosedur Uji Coba Melakukan *Parsing* JSON *Parser*

ID	**UJ-01**
Nama	**Uji Coba Melakukan *Parsing* JSON *Parser***
Tujuan Uji Coba	Menguji fungsionalitas untuk melakukan *parsing* dengan menggunakan JSON *parser*.
Kondisi Awal	Perangkat lunak dijalankan.
Skenario	**Perangkat lunak dijalankan dan *server* melakukan *request* URL ke situs *web* Earthquake Hazards Program**
Masukan	URL Earthquake Hazards Program
Keluaran	Objek gempa dalam bahasa Java
Hasil uji coba	**Berhasil**

Tabel 5.1 merupakan rincian prosedur uji coba melakukan *parsing* JSON *parser* dengan menggunakan pustaka GSON. Berdasarkan skenario uji coba, perangkat lunak melakukan *request* URL situs *web* Earthquake Hazards Program dan melakukan *parsing* JSON *parser*. Melalui fungsi fromJSON() dari pustaka GSON, JSON yang didapat dari URL tersebut diubah ke dalam objek di dalam Java.

Gambar 5.1 menunjukkan tampilan berbentuk *console* hasil keluaran uji coba fungsionalitas melakukan *parsing* JSON *parser* menggunakan pustaka GSON yang dilakukan di dalam *server*. Hasil yang didapat adalah sebuah objek pada Java yang bernama gempa. Objek ini memuat data gempa terakhir yang terjadi di Indonesia. Adapun data gempa yang diperlukan adalah besar *magnitude* gempa, lokasi dimana gempa terjadi, waktu terjadinya gempa dalam

satuan *milisecond*, dan koordinat *geolocation* gempa dalam *latitude* dan *longitude*.



Gambar 5.1 Tampilan Hasil Uji Coba Melakukan *Parsing* JSON Parser

5.2.1.2. Pengujian Melakukan Pencarian *Tweet*

Pengujian melakukan pencarian *tweet* merupakan pengujian untuk menguji fungsionalitas perangkat lunak untuk melakukan pencarian informasi pada *social sensor* yaitu Twitter (*tweet*). Tujuan pengujian ini adalah untuk memastikan fungsionalitas perangkat lunak tersebut berjalan dengan baik. Rincian prosedur pengujian melakukan pencarian *tweet* dapat dilihat pada Tabel 5.2.

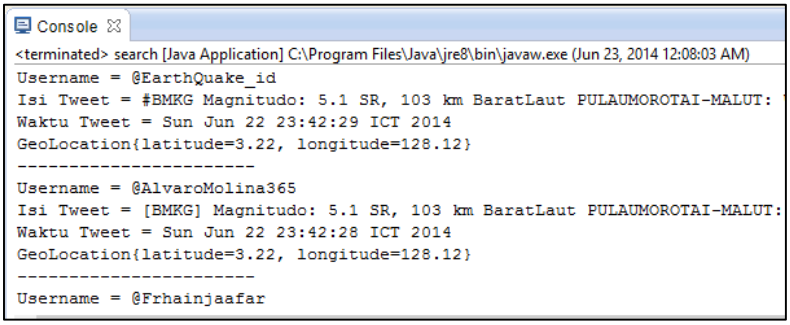
Tabel 5.2 Prosedur Uji Coba Melakukan Pencarian *Tweet*

ID	**UJ-02**
Nama	**Uji Coba Melakukan Pencarian *Tweet***
Tujuan Uji Coba	Menguji fungsionalitas untuk melakukan pencarian *tweet*
Kondisi Awal	Perangkat lunak dijalankan.
Skenario	**Perangkat lunak dijalankan dan menjalankan fungsi pencarian *tweet* menggunakan *query* 'gempa'**

ID	UJ-02
Masukan	-
Keluaran	Objek *tweet* dalam bahasa Java
Hasil uji coba	**Berhasil**

Berdasarkan skenario pada Tabel 5.2, perangkat lunak melakukan pencarian *tweet* berdasarkan *query*. Untuk uji coba ini *query* yang digunakan untuk pencarian *tweet* adalah kata 'gempa'. Adapun pustaka yang digunakan untuk melakukan pencarian *tweet* adalah Twitter4J.

Hasil keluaran uji coba fungsionalitas melakukan pencarian *tweet* yang dijelaskan pada Tabel 5.2 adalah berupa objek *tweet* dalam bahasa Java. Objek tersebut terdiri atas *username*, isi *tweet*, waktu *posting tweet*, dan koordinat *geolocation* dalam *latitude* dan *longitude*. Hasil keluaran uji coba fungsionalitas melakukan pencarian *tweet* sesuai rincian pengujian tersebut dapat dilihat pada Gambar 5.2.



Gambar 5.2 Tampilan Hasil Uji Coba Melakukan Pencarian *Tweet*

5.2.1.3. Pengujian Menghitung Jarak Koordinat *Geolocation* dengan Rumus Haversine

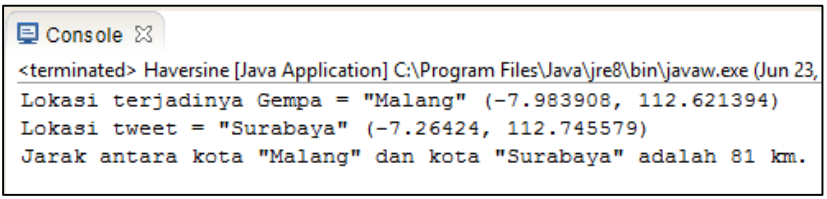
Pengujian menghitung jarak koordinat *geolocation* dengan rumus Haversine merupakan pengujian untuk menguji fungsionalitas perangkat lunak tersebut berjalan dengan baik. Adapun rincian prosedur uji coba menghitung jarak koordinat *geolocation* dengan rumus Haversine dapat dilihat pada Tabel 5.3.

Tabel 5.3 Prosedur Uji Coba Menghitung Jarak Koordinat *Geolocation* dengan Rumus Haversine

ID	**UJ-03**
Nama	**Uji Coba Menghitung Jarak Koordinat *Geolocation* dengan Rumus Haversine**
Tujuan Uji Coba	Menguji fungsionalitas untuk menghitung jarak *geolocation* dengan rumus Haversine
Kondisi Awal	Perangkat lunak memiliki informasi gempa dari situs *web* Earthquake Hazards Program dan informasi gempa dari *social sensor*
Skenario	***Server* mendapatkan jarak lokasi terjadinya gempa dan lokasi *tweet* lalu menghitung jarak diantara keduanya**
Masukan	Data informasi gempa dan data *tweet*
Keluaran	Jarak diantara lokasi terjadinya gempa dan lokasi *tweet*
Hasil uji coba	**Berhasil**

Skenario dari uji coba ini adalah berdasarkan Tabel 5.3 adalah setelah *server* mendapatkan jarak lokasi terjadinya gempa dari uji coba UJ-01 dan jarak *tweet* dari uji coba UJ-02, hal yang selanjutnya dilakukan adalah menghitung jarak koordinat *geolocation* diantara lokasi terjadinya gempa dan lokasi *tweet* menggunakan rumus Haversine.

Tampilan masukan dan hasil keluaran uji coba fungsionalitas menghitung jarak koordinat *geolocation* dengan rumus Haversine dapat dilihat pada Gambar 5.3. Pada Gambar 5.3 terlihat masukan uji coba ini adalah koordinat *geolocation* dari lokasi terjadinya gempa, yaitu kota Malang dan koordinat *geolocation* dari lokasi *tweet*, yaitu kota Surabaya. Hasil keluaran uji coba ini adalah jarak dari kota Malang dan kota Surabaya yaitu 81 dalam satuan kilometer.



Gambar 5.3 Tampilan Hasil Uji Coba Menghitung Jarak Koordinat Geolocation dengan Rumus Haversine

5.2.1.4. Pengujian Menentukan *Level* Konfidensi Informasi Gempa

Pengujian ini dilakukan untuk menentukan *level* konfidensi gempa. Pengujian ini sendiri terbagi ke dalam tiga bagian yaitu kondisi 1, kondisi 2, dan kondisi 3. Adapun penjelasan dari setiap bagian akan dijelaskan pada subbab berikut ini.

5.2.1.4.1. Pengujian Menentukan *Level* Konfidensi Informasi Gempa (Kondisi 1)

Pada fungsionalitas menentukan *level* konfidensi dengan kondisi 1, syarat jarak lokasi terjadinya gempa dan lokasi *tweet* memenuhi kondisi 1 adalah jarak tersebut kurang dari *threshold* yang telah ditentukan pada bab sebelumnya. Adapun rincian

prosedur uji coba menentukan *level* konfidensi informasi gempa dengan kondisi 1 dapat dilihat pada Tabel 5.4.

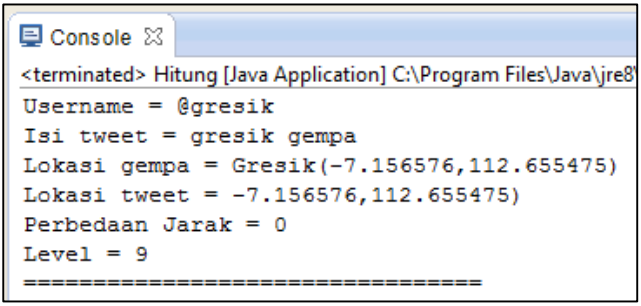
Tabel 5.4 Prosedur Uji Coba Menentukan *Level* Konfidensi Informasi Gempa dengan Kondisi 1

ID	**UJ-05**
Nama	**Uji Coba Menentukan *Level* Konfidensi Informasi Gempa dengan Kondisi 1**
Tujuan Uji Coba	Menguji fungsionalitas untuk menentukan konfidensi informasi gempa dengan kondisi 1.
Kondisi Awal	*Server* memiliki informasi jarak diantara lokasi gempa yang terjadi dan lokasi *tweet*
Skenario	**Terdapat *tweet* yang memiliki jarak lokasi terjadinya gempa dan *tweet* memenuhi kondisi 1.**
Masukan	Jarak lokasi gempa dan *tweet*
Keluaran	*Level* konfidensi gempa
Hasil uji coba	**Berhasil**

Pada Tabel 5.4, skenario uji coba ini adalah terdapat *tweet* yang memiliki jarak lokasi terjadinya gempa dan *tweet* memenuhi kondisi 1, dengan kondisi awal *server* memiliki informasi jarak diantara lokasi terjadinya gempa dan lokasi *tweet.*

Gambar 5.4. merupakan hasil keluaran uji coba menentukan *level* konfidensi informasi gempa dengan kondisi 1 berbentuk *console*. Pada Gambar 5.4 terlihat lokasi gempa adalah di Gresik dengan koordinat *geolocation* dalam *latitude* dan *longitude* secara berututan adalah `-7.156576` dan `112.655475`. Lalu lokasi *tweet* memiliki koordinat *geolocation* dengan *latitude* dan *longitude*. Dengan menggunakan rumus Haversine jarak dari keduanya adalah 0. Karena jarak keduanya adalah 0, yaitu memenuhi kondisi 1, dimana jarak tersebut kurang dari *threshold* yang telah ditentukan,

maka *level* konfidensi dari informasi gempa yang berasal dari *tweet* tersebut adalah 9.



Gambar 5.4 Tampilan Hasil Uji Coba Menentukan *Level* Konfidensi Informasi Gempa dengan Kondisi 1

5.2.1.4.2. Pengujian Menentukan *Level* Konfidensi Informasi Gempa (Kondisi 2)

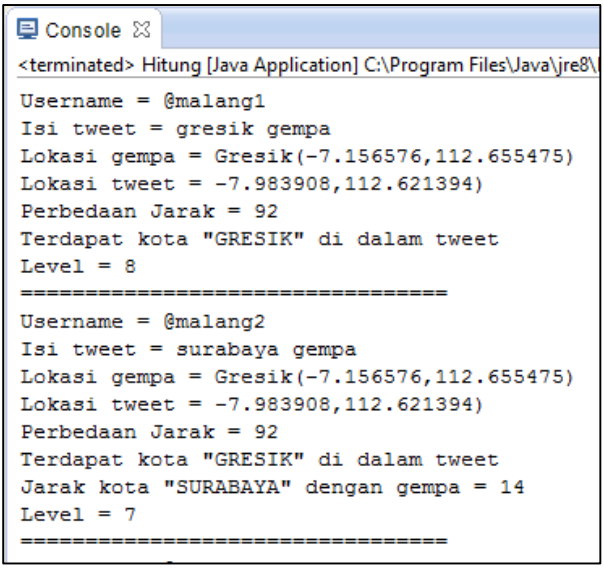
Pada fungsionalitas menentukan *level* konfidensi dengan kondisi 2, syarat jarak lokasi terjadinya gempa dan lokasi *tweet* memenuhi kondisi 2 adalah jarak tersebut tidak memenuhi kondisi 1 dan kurang dari *threshold* yang telah ditentukan pada bab sebelumnya. Adapun rincian prosedur uji coba menentukan *level* konfidensi informasi gempa dengan kondisi 2 dapat dilihat pada Tabel 5.5.

Tabel 5.5 Prosedur Uji Coba Menentukan *Level* Konfidensi Informasi Gempa dengan Kondisi 2

ID	**UJ-05**
Nama	**Uji Coba Menentukan *Level* Konfidensi Informasi Gempa dengan Kondisi 2**

ID	**UJ-05**
Tujuan Uji Coba	Menguji fungsionalitas untuk menentukan *level* konfidensi informasi gempa dengan kondisi 2
Kondisi Awal	*Server* memiliki informasi jarak diantara lokasi gempa yang terjadi dan lokasi *tweet*
Skenario 1	**Terdapat *tweet* yang memiliki jarak diantara lokasi terjadinya gempa dan *tweet* memenuhi kondisi 2 dan *tweet* mengandung kota lokasi gempa di dalam isi *tweet***
Masukan	Jarak lokasi gempa dan *tweet*
Keluaran	*Level* konfidensi gempa
Hasil uji coba	**Berhasil**
Skenario 2	**Terdapat *tweet* yang memiliki jarak diantara lokasi terjadinya gempa dan *tweet* memenuhi kondisi 2 dan jarak kota yang ada di isi *tweet* memenuhi kondisi**
Masukan	Jarak lokasi gempa dan *tweet*
Keluaran	*Level* konfidensi gempa
Hasil uji coba	**Berhasil**

Pada rincian prosedur uji coba fungsionalitas menentukan *level* konfidensi informasi gempa dengan kondisi 2 dapat dilihat kondisi awal yaitu *server* memiliki informasi jarak diantara lokasi terjadinya gempa dan lokasi *tweet*. Selain itu terdapat dua buah skenario dimana pada skenario pertama terdapat *tweet* yang memiliki jarak diantara lokasi terjadinya gempa dan *tweet* memenuhi kondisi 2 dan *tweet* mengandung kota lokasi gempa di dalam isi *tweet*. Selain itu pada skenario kedua terdapat *tweet* yang memiliki jarak diantara lokasi terjadinya gempa dan *tweet* memenuhi kondisi 2 dan jarak kota yang ada di isi *tweet* memenuhi kondisi.



Gambar 5.5 Tampilan Hasil Uji Coba Menentukan *Level* Konfidensi Informasi Gempa dengan Kondisi 2

Hasil keluaran uji coba fungsionalitas menentukan *level* konfidensi informasi gempa dengan kondisi 2 dapat dilihat pada Gambar 5.5. Pada Gambar 5.5 terlihat lokasi gempa terjadi di Gresik. Selain itu, terdapat informasi *tweet* dan informasi *level* konfidensi dari kedua *tweet* berdasarkan kedua skenario. *Tweet* pertama merupakan hasil skenario pertama dimana jarak lokasi terjadinya gempa dan lokasi *tweet* memenuhi kondisi 2, dan terdapat kata 'Gresik' di dalam *tweet*. Alasan ini membuat *level* konfidensi informasi gempa dari *tweet* ini memiliki nilai 8. Pada *tweet* kedua, tidak terdapat kata 'Gresik' melainkan kata 'Surabaya'. Maka dihitung jarak koordinat *geolocation* diantara kota terjadinya gempa yaitu 'Gresik' dan kota 'Surabaya' dengan menggunakan rumus Haversine. Kata 'Surabaya' didapat dengan menggunakan fungsi *string matching* yaitu mencocokkan kata yang ada pada isi *tweet* dengan berkas yang berisi nama kota di Indonesia. Koordinat

geolocation kota 'Surabaya' didapat dengan mencari koordinat *geolocation* kota tersebut menggunakan GeoNames. Berdasarkan hal tersebut, *level* konfidensi informasi gempa dari *tweet* pada skenario kedua bernilai 7.

5.2.1.4.3. Pengujian Menentukan *Level* Konfidensi Informasi Gempa (Kondisi 3)

Pengujian menentukan *level* konfidensi informasi gempa merupakan fungsionalitas terkakhir untuk menentukan *level* konfidensi informasi gempa. Pada pengujian ini, syarat jarak lokasi terjadinya gempa dan lokasi *tweet* memenuhi kondisi 3 adalah jarak tersebut tidak memenuhi kondisi 1 dan kondisi 2. Pada pengujian dengan kondisi 3, selain jarak diantara lokasi terjadinya gempa dan lokasi *tweet*, jumlah *retweet* pada *social sensor* juga digunakan sebagai parameter untuk menentukan *level* konfidensi informasi gempa. Adapun rincian prosedur uji coba menentukan *level* konfidensi informasi gempa dengan kondisi 3 dapat dilihat pada Tabel 5.6.

Tabel 5.6 Prosedur Uji Coba Menentukan *Level* Konfidensi Informasi Gempa dengan Kondisi 3

ID	**UJ-06**
Nama	**Uji Coba Menentukan Konfidensi Informasi Gempa dengan Kondisi 3**
Tujuan Uji Coba	Menguji fungsionalitas untuk menentukan konfidensi informasi gempa dengan kondisi 3
Kondisi Awal	Perangkat lunak memiliki informasi jarak diantara lokasi gempa yang terjadi dan lokasi *tweet*, dan jumlah *retweet* dari *tweet*
Skenario 1	**Terdapat *tweet* yang memiliki jarak diantara lokasi gempa dan *tweet* memenuhi kondisi 3, jumlah re*tweet* memenuhi**

ID	UJ-06
	kondisi, dan *tweet* mengandung kota lokasi gempa di dalam isi *tweet*
Masukan	Jarak lokasi gempa dan *tweet,* jumlah *retweet*
Keluaran	*Level* konfidensi gempa
Hasil uji coba	**Berhasil**
Skenario 2	**Terdapat *tweet* yang memiliki jarak diantara lokasi gempa dan *tweet* memenuhi kondisi 3 jumlah *retweet* memenuhi kondisi, dan jarak kota yang ada di isi *tweet* memenuhi kondisi**
Masukan	Jarak lokasi gempa dan *tweet*
Keluaran	*Level* konfidensi gempa
Hasil uji coba	**Berhasil**

Pada rincian prosedur uji coba fungsionalitas menentukan *level* konfidensi informasi gempa dengan kondisi 3, kondisi awal yaitu *server* memiliki informasi jarak diantara lokasi terjadinya gempa dan lokasi *tweet,* serta jumlah *retweet* dari *tweet*. Skenario pertama dari pengujian ini yaitu terdapat *tweet* yang memiliki jarak diantara lokasi terjadinya gempa dan *tweet* memenuhi kondisi 3, jumlah *retweet* dari *tweet* memenuhi kondisi, dan *tweet* mengandung kota lokasi gempa di dalam isi *tweet*. Selain itu pada skenario kedua terdapat *tweet* yang memiliki jarak diantara lokasi terjadinya gempa dan *tweet* memenuhi kondisi 3, jumlah *retweet* dari *tweet* memenuhi kondisi, dan jarak kota yang ada di isi *tweet* memenuhi kondisi.

Hasil keluaran uji coba fungsionalitas menentukan *level* konfidensi informasi gempa dengan kondisi 3 dapat dilihat pada Gambar 5.6. Berdasarkan Gambar 5.6, lokasi gempa terjadi di Gresik. Selain itu, terdapat informasi *tweet* dan informasi *level* konfidensi dari kedua *tweet* berdasarkan kedua skenario. *Tweet* pertama merupakan hasil skenario pertama dimana jarak lokasi

terjadinya gempa dan lokasi *tweet* memenuhi kondisi 3, jumlah *retweet* memenuhi kondisi, dan terdapat kata 'Gresik' di dalam *tweet*. Berdasarkan hal tersebut, *level* konfidensi informasi gempa dari *tweet* ini memiliki nilai 4. Pada *tweet* kedua, tidak terdapat kata 'Gresik' melainkan kata 'Surabaya'. Maka dihitung jarak koordinat *geolocation* diantara kota terjadinya gempa yaitu 'Gresik' dan kota 'Surabaya' dengan menggunakan rumus Haversine. Hasil keluaran uji coba pada skenario kedua adalah *tweet* tersebut memiliki *level* konfidensi informasi gempa bernilai 3.

Gambar 5.6 Tampilan Uji Coba Menentukan *Level* Konfidensi Informasi Gempa dengan Kondisi 3

5.2.1.5. Pengujian Menghitung *Level* Konfidensi Informasi Gempa

Pengujian menghitung *level* konfidensi informasi gempa dilakukan untuk menguji fungsionalitas perangkat lunak untuk menghitung jumlah *tweet* dengan *level* konfidensi informasi gempa

yang telah diolah berdasarkan data dari situs *web* Earthquake Hazards Program dan data dari *social sensor*. Adapun rincian dari pengujian menghitung *level* konfidensi informasi gempa dapat dilihat pada Tabel 5.7.

Tabel 5.7 Prosedur Ujicoba Menghitung *Level* Konfidensi Informasi Gempa

ID	**UJ-07**
Nama	**Uji Coba Menghitung Konfidensi Informasi Gempa**
Tujuan Uji Coba	Menguji fungsionalitas untuk menghitung konfidensi informasi gempa
Kondisi Awal	*Server* memiliki informasi gempa dari situs *web* Earthquake Hazards Program dan informasi gempa dari *social sensor*, dan *level* konfidensi informasi gempa
Skenario	***Server* memasukkan data informasi gempa, *tweet*, dan *level* konfidensi informasi gempa ke dalam *database*.**
Masukan	Data informasi gempa dari situs *web* dan *social sensor*, dan *level* konfidensi informasi gempa
Keluaran	*Level* konfidensi, data informasi gempa, dan data *tweet* masuk ke dalam *database*.
Hasil uji coba	**Berhasil**

Berdasarkan Tabel 5.7, kondisi awal dari uji coba ini adalah *server* telah mendapatkan informasi hasil keluaran uji coba UJ-01 dan UJ-02. Selain itu *server* memiliki *level* konfidensi informasi gempa hasil keluaran uji coba UJ-04, UJ-05, dan UJ-06. Skenario dari uji coba ini adalah *server* menghitung jumlah *tweet* hasil keluaran uji coba UJ-02, berdasarkan *level* konfidensi hasil keluaran

uji coba UJ-04, UJ-05, dan UJ-06. Setelah itu, hasil uji coba dimasukkan ke dalam *database*.

Hasil keluaran uji coba menghitung *level* konfidensi informasi gempa dapat dilihat pada Gambar 5.7 dan Gambar 5.8. Gambar 5.7 merupakan tabel gempa yang berisi informasi gempa dari situs *web* Earthquake Hazards Program dan jumlah *tweet* berdasarkan *level* konfidensi informasi gempa.

MAGNITUDE	LOKASI	TIME	LATITUDE	LONGITUDE	TWEET	LEVEL9	LEVEL8	LEVEL7	LEVEL6	LEVEL5	LEVEL4	LEVEL3	LEVEL2	LEVEL1
5.5	Gresik	1401552240000	-7.15658	112.655	10	4	2	2	2	2	2	2	2	2
6.0	Malang	1401552420000	-7.98391	112.621	10	6	0	0	4	2	2	0	4	2
6.0	Malang	1401552420000	-7.98391	112.621	10	3	0	0	2	1	1	0	2	1

Gambar 5.7 Tampilan Tabel Gempa Hasil Uji Coba Menghitung *Level* Konfidensi Informasi Gempa

USERNAME	ISITWEET	TIME	LATITUDE	LONGITUDE
@gresik	gresik gempa	1401555840000	-7.15658	112.655
@malang1	gresik gempa	1401555840000	-7.98391	112.621
@malang2	surabaya gempa	1401555840000	-7.98391	112.621
@malang3	malang gempa	1401555840000	-7.98391	112.621
@jakarta1	gempa	1401555840000	-6.20876	106.846
@jakarta1	gempa gresik	1401555840000	-6.20876	106.846

Gambar 5.8 Tampilan Tabel *Tweet* Hasil Uji Coba Menghitung *Level* Konfidensi Informasi Gempa

5.2.1.6. Pengujian Membuat Berkas XML pada *Web Service*

Pengujian membuat berkas XML pada *web service* merupakan pengujian terhadap kemampuan perangkat lunak dalam membuat berkas XML pada *web service*. Tujuan uji coba ini adalah untuk menguji apakah fungsionalitas perangkat lunak tersebut

bekerja dengan baik. Adapun rincian prosedur uji coba membuat berkas XML pada *web service* dapat dilihat pada Tabel 5.8.

Tabel 5.8 Prosedur Uji Coba Membuat Berkas XML pada *Web Service*

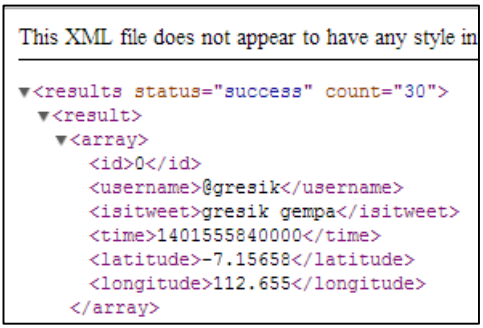
ID	**UJ-08**
Nama	**Uji Coba Membuat Berkas XML pada *web service***
Tujuan Uji Coba	Menguji fungsionalitas untuk membuat berkas XML pada *web service*
Kondisi Awal	Perangkat lunak memiliki *database* informasi gempa dan *tweet*
Skenario	***Server* mengambil informasi gempa dan *tweet* dari *database* dan membuat berkas XML**
Masukan	*Database* informasi gempa dan *tweet*
Keluaran	File XML berisi informasi gempa dan *tweet*
Hasil uji coba	**Berhasil**

Berdasarkan Tabel 5.8, kondisi awal dari uji coba ini adalah *server* memiliki *database* berisi informasi gempa dan *tweet*. Adapun skenario dari pengujian ini adalah *server* mengambil baris data tabel gempa dan *tweet* dari *database* dan membuat berkas XML berdasarkan data tersebut.

Hasil keluaran uji coba membuat berkas XML pada *web service* dapat dilihat pada Gambar 5.9 dan Gambar 5.10. Pada Gambar 5.9, berkas XML berisi informasi gempa dan jumlah *tweet* berdasarkan *level* konfidensi yang bersangkutan. Informasi gempa ini terdiri atas *magnitude* gempa, lokasi terjadinya gempa, waktu terjadinya gempa dalam satuan *milisecond*, koordinat *geolocation* gempa, dan jumlah *tweet* beserta *level* konfidensinya. Gambar 5.10 menunjukkan berkas XML yang berisi *tweet* informasi gempa.

Informasi yang ada pada berkas ini adalah *username*, isi *tweet*, waktu *posting tweet* dan koordinat *geolocation tweet*.

Gambar 5.9 Tampilan Berkas XML Gempa Hasil Uji Coba Membuat Berkas XML pada *Web Service*



Gambar 5.10 Tampilan Berkas XML *Tweet* Hasil Uji Coba Membuat Berkas XML pada *Web Service*

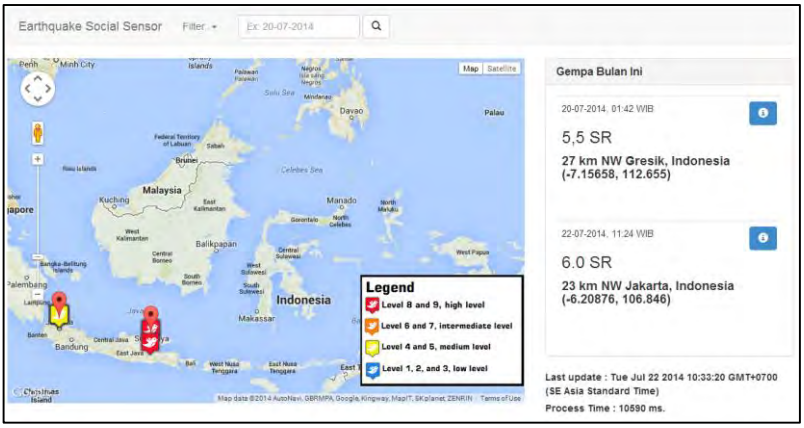
5.2.1.7. Menampilkan Peta dan Informasi

Pengujian menampilkan peta dan informasi merupakan pengujian terhadap kemampuan perangkat lunak dalam menampilkan peta dan informasi gempa. Tujuan uji coba ini adalah untuk menguji apakah fungsionalitas perangkat lunak tersebut bekerja dengan baik. Adapun rincian prosedur uji coba menampilkan peta dan informasi dapat dilihat pada Tabel 5.9.

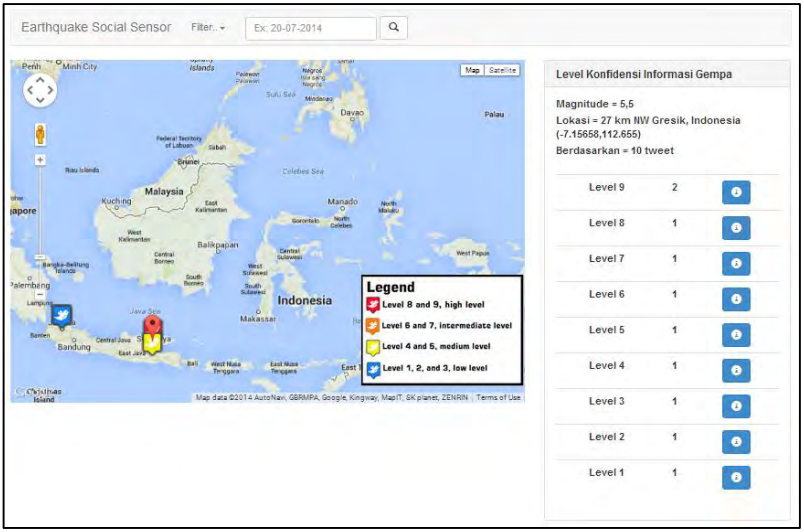
Tabel 5.9 Prosedur Uji Coba Menampilkan Peta dan Informasi

ID	**UJ-09**
Nama	**Uji Coba Menampilkan Peta dan Informasi**
Tujuan Uji Coba	Menguji fungsionalitas untuk menampilkan peta dan informasi
Kondisi Awal	*Server* memiliki berkas XML berisi informasi gempa dan *tweet*
Skenario	***Server* mengambil informasi gempa dan *tweet* dari berkas XML dan menampilkan peta lokasi terjadinya gempa dan informasi gempa**
Masukan	Berkas XML berisi informasi gempa dan *tweet*
Keluaran	Peta lokasi terjadinya gempa dan informasi gempa
Hasil uji coba	**Berhasil**

Berdasarkan Tabel 5.9, kondisi awal uji coba menampilkan peta dan informasi adalah *server* memiliki berkas XML hasil keluaran uji coba UJ-08. Skenario dari pengujian ini adalah *server* mengambil informasi dari berkas XML yang berisi informasi gempa dan *tweet*. Lalu perangkat lunak melakukan inisialisasi Google Maps. Selanjutnya adalah membuat *marker* berdasarkan informasi gempa dan *tweet* yang didapat.



Gambar 5.11 Tampilan Hasil Uji Coba Menampilkan Peta dan Informasi



Gambar 5.12 Tampilan Hasil Uji Coba Menampilkan *Level* Konfidensi Informasi Gempa

Hasil keluaran uji coba menampilkan peta dan informasi dapat dilihat pada Gambar 5.11 dan Gambar 5.12. Pada Gambar 5.11 terdapat halaman situs *web* yang menampilkan peta lokasi terjadinya gempa. Di sebelah kanan terdapat *panel* untuk menunjukkan informasi gempa yang terjadi. Gambar 5.12 menunjukkan detail gempa yang terjadi. Selain itu juga terdapat *level* konfidensi gempa beserta jumlah *tweet*nya.

5.2.2. Pengujian Performa

Pengujian performa perangkat lunak merupakan pengujian untuk menguji apakah performa perangkat lunak dapat diandalkan. Pengujian performa pada perangkat lunak ini terdiri atas pengujian waktu pengolahan data pada *server* berdasarkan data yang ada dan jumlah *bandwidth* yang digunakan saat program melakukan *request* data. Adapun penjelasan dari setiap bagian akan dijelaskan pada subbab berikut ini.

5.2.2.1. Pengujian Performa Waktu Pengolahan Data pada *Server*

Pengujian performa waktu pengolahan data pada *server* dilakukan untuk mengukur kemampuan *server* dalam mengolah data gempa dan *tweet* dan mendapatkan *level* konfidensi gempa. Pengujian performa waktu pengolahan data pada *server* terbagi menjadi dua bagian, yaitu pengujian waktu performa pengolahan data pada *server* berdasarkan jumlah gempa, dan pengujian performa waktu pengolahan data pada *server* berdasarkan jumlah *tweet.*

5.2.2.1.1. Pengujian Performa Waktu Pengolahan Data pada *Server* Berdasarkan Data Jumlah Gempa

Pengujian performa waktu pengolahan data pada *server* berdasarkan data jumlah gempa merupakan pengujian untuk menguji apakah jumlah data gempa yang terjadi yang didapat dari

situs *web* Earthquake Hazards Program mempengaruhi waktu pengolahan data pada *server*. Untuk menghitung waktu pengolahan data pada *server*, diperlukan data jumlah gempa, jumlah *tweet*, waktu mulai mengolah data, dan waktu selesai mengolah data. Adapun waktu yang dihitung menggunakan satuan detik (s).

Tabel 5.10 merupakan sebuah contoh uji coba menghitung rata-rata waktu yang dibutuhkan untuk mengolah sebuah data gempa dengan *test case* lima data gempa dan sepuluh data *tweet*. Waktu mulai pengecekan gempa adalah waktu dimulainya proses penentuan *level* konfidensi setiap *tweet* dalam satu data gempa. Selain itu terdapat waktu selesai pengecekan gempa sebagai tanda proses penentuan *level* konfidensi setiap *tweet* pada gempa tersebut selesai. Kolom ΔT gempa digunakan untuk menghitung selisih waktu proses penentuan *level* konfidensi setiap *tweet* dalam satu data gempa.

Tabel 5.10 Menghitung Rata-Rata Waktu Pengolahan Sebuah Data Gempa

No	Jumlah Gempa	Waktu Mulai Pengecekan Gempa (detik)	Waktu Selesai Pengecekan Gempa (detik)	ΔT Gempa (detik)
1	1	00:17:23.000	00:17:27:792	4.792
2	2	00:17:27:792	00:17:30.582	3.582
3	3	00:17:30.582	00:17:37.605	7.023
4	4	00:17:37.605	00:17:44.527	6.922
5	5	00:17:44.527	00:17:51.428	6.901
Rata-Rata Waktu Pengolahan Gempa				5.844

Tabel 5.11 merupakan hasil uji coba waktu pengolahan data pada *server* berdasarkan jumlah gempa yang terjadi. Pada tabel ini terlihat pengujian performa pengolahan data pada *server* berdasarkan data jumlah gempa dengan jumlah *tweet* yang sama namun memiliki jumlah informasi gempa yang berbeda.

Tabel 5.11 Hasil Uji Coba Waktu Pengolahan Data pada *Server* Berdasarkan Jumlah Gempa

No	Jumlah Gempa	Jumlah *Tweet*	Waktu Mulai (detik)	Waktu Selesai (detik)	Selisih Waktu (detik)	ΔT Gempa (detik)
1	5	10	00:10:11.034	00:10:42.232	31.198	6.036
2	10	10	00:10:50.235	00:11:52.910	62.675	6.163
3	15	10	00:12:46.450	00:14:22.492	96.042	6.340
4	20	10	00:14:59.453	00:17:4.577	125.124	6.205
5	25	10	00:18.22.252	00:20:53.348	160.148	6.364
6	30	10	00:21:07.233	00:24:11.622	184.389	6.111
7	35	10	00:24:33.347	00:28:11.522	218.175	6.225
8	40	10	00:29.02.201	00:33:15.280	254.080	6.347
9	45	10	00:34:00.236	00:38:37.073	276.837	6.146
10	50	10	00:40:57.345	00:46:8.079	310.734	6.210
Rata-rata Gempa						6.215

Adapun jumlah pengujian adalah sepuluh kali dengan jumlah *tweet* yang diujikan adalah 10, sedangkan jumlah informasi gempa yang digunakan adalah kelipatan 5, yaitu 5, 10, 15, hingga 50. Waktu mulai adalah waktu saat perangkat lunak dijalankan, dan waktu selesai adalah waktu saat perangkat lunak selesai dijalankan. Selisih waktu merupakan waktu jalannya program, didapat dengan mengurangi waktu selesai dan waktu mulai perangkat lunak. Selain itu juga dihitung ΔT gempa seperti pada Tabel 5.10. Hasil yang didapat adalah rata-rata ΔT gempa pada berbagai *test case* untuk menunjukkan waktu yang dibutuhkan dalam melakukan proses penentuan *level* konfidensi informasi gempa.

5.2.2.1.2. Pengujian Performa Pengolahan Data pada *Server* Berdasarkan Data Jumlah *Tweet*

Pengujian performa waktu pengolahan data pada *server* berdasarkan data jumlah *tweet* merupakan pengujian untuk menguji apakah jumlah *tweet* yang didapat dari *social sensor* mempengaruhi waktu pengolahan data pada *server*. Untuk menghitung waktu pengolahan data pada *server*, diperlukan data jumlah gempa, jumlah

tweet, waktu mulai mengolah data, dan waktu selesai mengolah data. Adapun waktu pengolahan data yang dihitung menggunakan satuan detik (s).

Tabel 5.12 merupakan contoh uji coba menghitung rata-rata waktu yang dibutuhkan untuk mengolah sebuah data *tweet* dengan *test case* sebuah data gempa dan sepuluh data *tweet*. Waktu mulai pengecekan *tweet* adalah waktu dimulainya proses penentuan *level* konfidensi setiap *tweet*. Selain itu terdapat waktu selesai pengecekan *tweet* sebagai tanda proses penentuan *level* konfidensi setiap *tweet* selesai. Kolom ΔT *tweet* digunakan untuk menghitung selisih waktu proses penentuan *level* konfidensi setiap *tweet*.

Tabel 5.12. Menghitung Rata-Rata Waktu Pengolahan Sebuah Data *Tweet*

No	*Tweet*	Waktu Mulai Pengecekan *Tweet* (detik)	Waktu Selesai Pengecekan *Tweet* (detik)	ΔT *Tweet* (detik)
1	1	00:41:04.045	00:41:04.354	0.309
2	2	00:41:04.676	00:41:04.708	0.032
3	3	00:41:05.002	00:41:07.008	2.006
4	4	00:41:07.224	00:41:8.866	1.642
5	5	00:41:08.900	00:41:9.001	0.101
6	6	00:41:09.029	00:41:09.061	0.032
7	7	00:41:09.077	00:41:10.709	1.632
8	8	00:41:10.800	00:41:12.417	1.617
9	9	00:41:12.500	00:41:12.556	0.056
10	10	00:41:12.678	00:41:12.711	0.033
Rata-Rata Waktu Pengolahan *Tweet*				0.746

Tabel 5.13 merupakan hasil uji coba waktu pengolahan data pada *server* berdasarkan jumlah *tweet.* Pada tabel ini terlihat pengujian performa waktu pengolahan data pada *server* berdasarkan data jumlah gempa dengan jumlah data gempa yang sama namun memiliki jumlah *tweet* yang berbeda. Adapun jumlah pengujian adalah sepuluh kali dengan jumlah informasi gempa yang diujikan

adalah lima, sedangkan jumlah *tweet* yang diujikan adalah kelipatan sepuluh, yaitu 10, 20, 30 hingga 100. Waktu mulai adalah waktu saat perangkat lunak dijalankan, dan waktu selesai adalah waktu saat perangkat lunak selesai dijalankan. Selisih waktu merupakan waktu jalannya program, didapat dengan mengurangi waktu selesai dan waktu mulai perangkat lunak. Selain itu juga dihitung ΔT *tweet* seperti pada Tabel 5.12. Hasil yang didapat adalah rata-rata ΔT *tweet* pada berbagai *test case* untuk menunjukkan waktu yang dibutuhkan dalam melakukan proses penentuan *level* konfidensi informasi gempa.

Tabel 5.13. Hasil Uji Coba Waktu Pengolahan Data pada *Server* Berdasarkan Jumlah *Tweet*

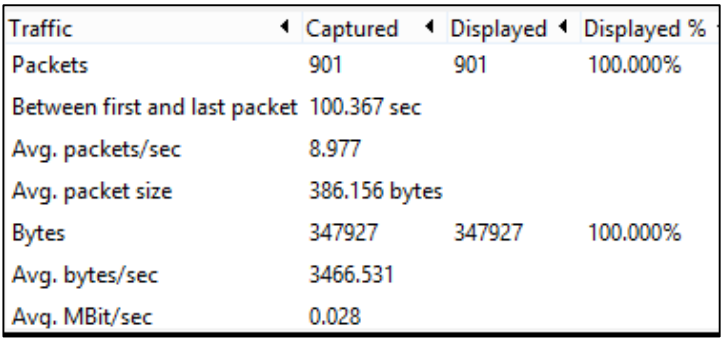
No	Gempa	*Tweet*	Waktu Mulai (detik)	Waktu Selesai (detik)	Selisih Waktu (detik)	ΔT *Tweet* (detik)
1	5	10	00:01:34.345	00:02:13.580	39.235	0.748
2	5	20	00.03.00.189	00:03:17.297	89.297	0.879
3	5	30	00:03:47.222	00:05:58.379	131.157	0.864
4	5	40	00:06:00.440	00:08:58.082	177.642	0.879
5	5	50	00:09:45.976	00:12:24.963	158.988	0.620
6	5	60	00:12:45.444	00:15:37.13	171.686	0.570
7	5	70	00:15:59.621	00:20:44.868	285.247	0.811
8	5	80	00:21:01:322	00:24:43.062	222.062	0.554
9	5	90	00:25:00.033	00:29:15.376	255.343	0.566
10	5	100	00:30:09.189	00:34:56.065	286.876	0.572
Rata-Rata *Tweet*						0.706

5.2.2.2. Pengujian Performa Penggunaan *Bandwidth*

Pengujian performa penggunaan *bandwidth* menjelaskan penggunaan *bandwidth* yang digunakan untuk mengirim dan menerima data diantara *server* dan *smartphone* Android. Untuk menguji performa tersebut, dilakukan penangkapan jumlah paket yang melewati *traffic server* dan *smartphone* Android menggunakan alat kakas Wire Shark. Gambar 5.13 menunjukkan hasil *screenshot*

menghitung penggunaan bandwith untuk mengirim dan menerima data diantara *server* dan *smartphone* Android. Pada gambar tersebut, rata-rata penggunaan *bandwidth* adalah sebesar 0.028 Mbit/detik dengan jumlah paket data yang melewati *traffic* adalah 901.

Tabel 5.14 merupakan hasil uji coba penggunaan *bandwidth* perangkat lunak berdasarkan jumlah pengguna yang mengakses *server*. Adapun jumlah pengujian adalah sepuluh kali dengan jumlah informasi gempa dan jumlah informasi *tweet* yang sama namun memiliki jumlah pengguna mengakses yang berbeda. Selain itu dihitung rata-rata jumlah penggunaan *bandwidth* setiap pengguna.



Traffic	Captured	Displayed	Displayed %
Packets	901	901	100.000%
Between first and last packet	100.367 sec		
Avg. packets/sec	8.977		
Avg. packet size	386.156 bytes		
Bytes	347927	347927	100.000%
Avg. bytes/sec	3466.531		
Avg. MBit/sec	0.028		

Gambar 5.13. Menghitung Penggunaan *Bandwidth*

Tabel 5.14 Hasil Uji Coba Penggunaan *Bandwidth*

No	Jumlah Gempa	Jumlah *Tweet*	Jumlah Pengguna	Penggunaan *Bandwidth* (Mbit/detik)	ΔBandwidth (Mbit/detik)
1	5	30	1	0.028	0.028
2	5	30	2	0.078	0.039
3	5	30	3	0.111	0.037
4	5	30	4	0.144	0.036
5	5	30	5	0.167	0.033
6	5	30	6	0.199	0.033

7	5	30	7	0.228	0.033
8	5	30	8	0.259	0.032
9	5	30	9	0.289	0.032
10	5	30	10	0.312	0.031
Rata-Rata Waktu Penggunaan *Bandwidth*					0.033

5.3. Evaluasi Hasil Uji Coba

Evaluasi hasil uji coba merupakan pemaparan mengenai hasil uji coba yang telah dilakukan pada subbab sebelumnya. Evaluasi uji coba ini membahas evaluasi hasil uji coba performa perangkat lunak.

5.3.1. Evaluasi Hasil Uji Coba Performa

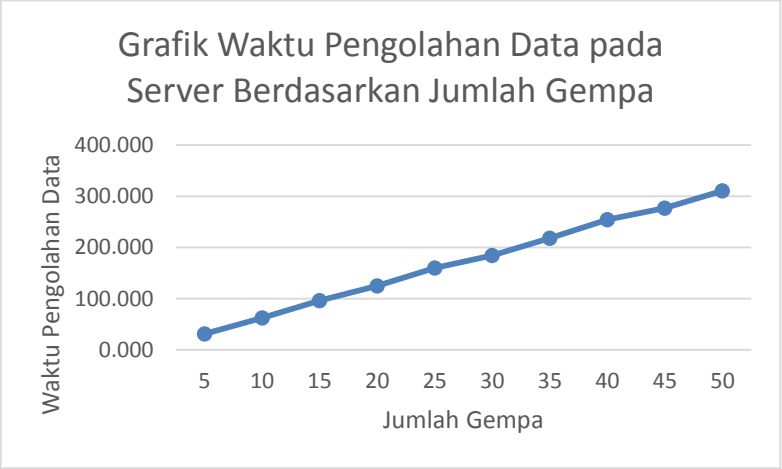
Pada evaluasi hasil uji coba performa, dilakukan pemaparan hasil uji coba performa yang telah dilakukan sebelumnya, yaitu uji coba performa waktu pengolahan data pada *server* dan uji coba peforma penggunaan *bandwidth*. Evaluasi ini akan dijelaskan pada subbab berikut.

5.3.1.1. Evaluasi Hasil Uji Coba Waktu Pengolahan Data pada *Server*

Evaluasi hasil uji coba waktu pengolahan data pada *server* terdiri atas dua bagian, yaitu evaluasi hasil uji coba waktu pengolahan data pada *server* berdasarkan jumlah gempa dan evaluasi hasil uji coba waktu pengolahan data pada *server* berdasarkan jumlah *tweet*. Kedua bagian tersebut akan dijelaskan pada subbab berikut ini.

5.3.1.1.1. Evaluasi Hasil Uji Coba Waktu Pengolahan Data pada *Server* Berdasarkan Jumlah Gempa

Sesuai dengan hasil uji coba waktu pengolahan data pada *server* berdasarkan jumlah gempa yang terdapat pada Tabel 5.10, maka didapat waktu pengolahan data yang merupakan selisih waktu saat program dijalankan dan pada saat program selesai dalam satuan *milisecond*. Grafik waktu pengolahan data pada *server* berdasarkan jumlah gempa dapat dilihat pada Gambar 5.14.



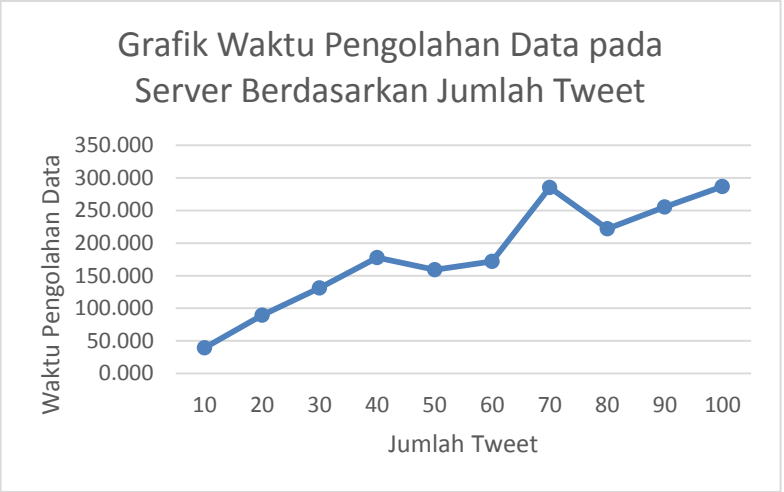
Gambar 5.14. Grafik Waktu Pengolahan Data pada *Server* Berdasarkan Jumlah Gempa

Berdasarkan Gambar 5.14, grafik waktu pengolahan data memiliki grafik yang naik seiring pertambahan data jumlah gempa. Gambar 5.14 menunjukkan jumlah gempa yang terjadi mempengaruhi waktu pengolahan data pada *server*. Hal ini disebabkan karena untuk setiap *tweet* pada setiap gempa, *server* membutuhkan waktu untuk melakukan *request* ke *web service* GeoNames pada saat mengecek *level* konfidensi gempa. Adapun rata-rata pengolahan *level* konfidensi untuk satu gempa berdasarkan hasil uji coba adalah 6.215 detik untuk sepuluh *tweet*.

5.3.1.1.2. Evaluasi Hasil Uji Coba Waktu Pengolahan Data pada *Server* Berdasarkan Jumlah *Tweet*

Sesuai hasil uji coba waktu pengolahan data pada *server* berdasarkan jumlah *tweet* pada Tabel 5.1, didapat waktu pengolahan data pada *server*.

Waktu pengolahan data untuk satu *tweet* pada satu gempa cukup bervariatif bergantung pada *level* konfidensi informasi gempa dari setiap *tweet*. Adapun grafik waktu pengolahan data pada *server* berdasarkan jumlah *tweet* dapat dilihat pada Gambar 5.15.



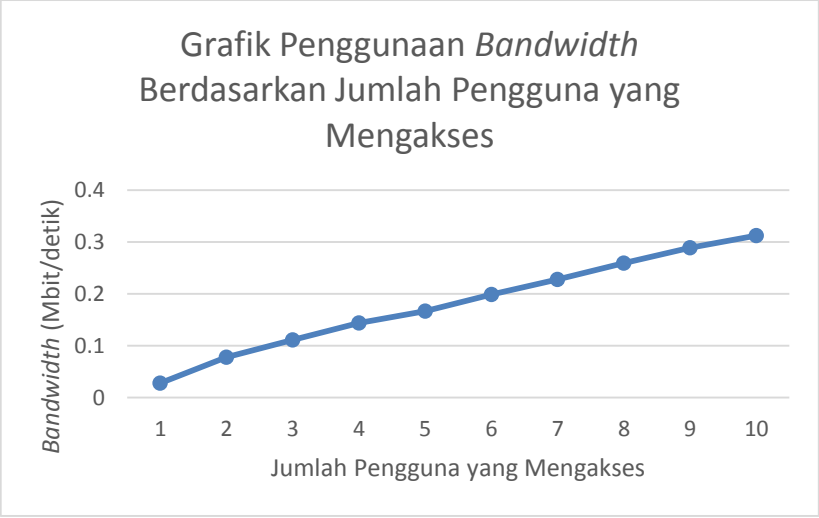
Gambar 5.15. Grafik Waktu Pengolahan Data pada *Server* Berdasarkan Jumlah *Tweet*

Berdasarkan Gambar 5.15, terjadi pertambahan waktu pengolahan data seiring bertambahnya jumlah *tweet*. Gambar 5.15 membuktikan bahwa jumlah *tweet* mempengaruhi waktu pengolahan data pada *server*. Seperti halnya evaluasi pada subbab 5.3.1.1.2, hal ini disebabkan karena *server* membutuhkan waktu untuk melakukan *request* ke *web service* GeoNames pada saat

mengecek *level* konfidensi gempa. Adapun rata-rata waktu pengolahan data untuk satu *tweet* berdasarkan hasil uji coba adalah 0.706 detik.

5.3.1.2. Evaluasi Hasil Uji Coba Penggunaan *Bandwidth*

Gambar 5.17 menunjukkan perbandingan jumlah pengguna yang mengakses dan penggunaan *bandwidth*. Pada gambar tersebut terlihat bahwa jumlah pengguna yang mengakses perangkat lunak mempengaruhi *bandwidth* perangkat lunak. Adapun rata-rata penggunaan *bandwidth* adalah sebesar 0.033 Mbit/detik.



Gambar 5.16 Grafik Penggunaan *Bandwidth*

DAFTAR PUSTAKA

[1] "Earthquake Hazars Program," United States Geological Survey, [Online]. Available: http://earthquake.usgs.gov/earthquakes/. [Diakses 26 Juni 2014].

[2] "JSON," JSON. [Online]. [Diakses 29 Juni 2014].

[3] "GSON," Google, Inc, [Online]. Available: https://code.google.com/p/google-gson/. [Diakses 29 Juni 2014].

[4] "Twitter API," Twitter, [Online]. Available: https://dev.twitter.com/. [Diakses 29 Juni 2014].

[5] "Twitter4J," Twitter4J, [Online]. Available: http://twitter4j.org/en/index.html. [Diakses 29 Juni 2014].

[6] "Google Geocoding API," Google, [Online]. Available: https://developers.google.com/maps/documentation/geocoding/. [Diakses 29 Juni 2014].

[7] "Google Maps API," Google, [Online]. Available: https://developers.google.com/maps/. [Diakses 29 Juni 2014].

[8] "GeoNames," http://www.geonames.org/, [Online]. Available: http://www.geonames.org/. [Diakses 29 Juni 2014].

[9] M. M. K. N. Dr. P. V. Ingole, "Landmark based shortest path detection by using Dijkstra Algorithm and Haversine Formula," *International Journal of Engineering Research and Applications,* vol. 3, no. 3, pp. 162-165, Juni 2013.

[10] "PHP," The PHP Group, [Online]. Available: http://www.php.net/. [Diakses 30 Juni 2014].

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

[12] “Eclipse Documentation,” Eclipse Juno, [Online]. Available: http://help.eclipse.org/juno/index.jsp. [Diakses 30 Juni 2014].

[13] “Android Developer Tools,” Google, [Online]. Available: http://developer.android.com/about/index.html. [Diakses 30 Juni 2014].

[14] C. G. Survey, “How Earthquake and Their Effects are Measured,” 2002. [Online]. Available: http://www.conservation.ca.gov/Index/Pages/Index.aspx. [Diakses 30 Juni 2014].

BIODATA PENULIS

Penulis, Ryan Dwi Cahyo Nugroho, lahir di kota Pekalongan pada tanggal 18 Nopember 1992. Penulis adalah anak kedua dari tiga bersaudara dan dibesarkan di Bandar Lampung, Lampung.

Penulis menempuh pendidikan formal di SD Negeri 2 Rawalaut (Teladan) (1998-2004), SMPN 2 Bandar Lampung (2004-2007), SMAN 2 Bandar Lampung (2007-2010). Pada tahun 2010, penulis memulai pendidikan S1 jurusan Teknik Informatika Fakultas Teknologi Informasi di Institut Teknologi Sepuluh Nopember Surabaya, Jawa Timur.

Di jurusan Teknik Informatika, penulis mengambil bidang minat Komputasi Berbasis Jaringan dan memiliki ketertarikan di bidang *web programming*, sistem terdistribusi, dan pemrograman jaringan. Penulis juga aktif dalam organisasi kemahasiswaan seperti Himpunan Mahasiswa Teknik Computer (HMTC). Dan penulis juga beberapa kali menjadi asisten dosen, diantaranya Asisten Dosen mata kuliah Pemrograman Jaringan, dan Asisten Dosen mata kuliah Sistem Terdistribusi. Penulis dapat dihubungi melalui alamat email ryan.dwi10@gmail.com

LAMPIRAN

1. Fungsi JSON *Parser*

Fungsi JSON *parser* digunakan untuk melakukan *parsing* situs *web* Earthquake Hazards Program menggunakan pustaka GSON.

```
public String gsonParsing() throws
FileNotFoundException, IOException
{
Gson gson = new Gson();
String url =
"http://earthquake.usgs.gov/earthquakes/
feed/v1.0/summary/all_hour.geojson";
Response response =
gson.fromJson(IOUtils.toString(new URL(url)),
Response.class);
for (int i = 0; i < response.features.size();
i++) {
String s = response.getFeatures().get(i).
getProperties().getPlace();
if (s.contains("Indonesia"))
{
magnitude = response.getFeatures().get(i).
getProperties().getMag();
place =  response.getFeatures().get(i).
getProperties().getPlace();
time = response.getFeatures().get(i).
getProperties().getTime();
longitude = response.getFeatures().get(i).
getGeometry().getCoordinates().get(0);
latitude = response.getFeatures().get(i).
getGeometry().getCoordinates().get(1);
Gempa objt = new Gempa(magnitude, place, time,
latitude, longitude);
gempa.add(objt);
}
}
```

```
return gempa.get(0).getMagnitude()+"`"+gempa.
   get(0).getPlace()+"`"+gempa.get(0).getTime()+
   "`"+
gempa.get(0).getLatitude()+"`"+
      gempa.get(0).getLongitude();
}
```

Kode Sumber 1 Fungsi JSON *Parser*

2. Fungsi Pencarian *Tweet*

Fungsi pencarian *tweet* digunakan untuk mengambil informasi dari *social sensor* yaitu Twitter.

```
public ArrayList<tweeps> searching() {
TwitterFactory factory = new TwitterFactory();
Distinct dis = new Distinct();
AccessToken accessToken = new
AccessToken("","");
Twitter twitter = factory.getInstance();
twitter.setOAuthConsumer("", "");
twitter.setOAuthAccessToken(accessToken);
try {
Query query = new Query("gempa");
QueryResult result;
do {
result = twitter.search(query);
List<Status> tweets = result.getTweets();
for (Status tweet : tweets) {
if (tweet.getGeoLocation() != null)
{
String test = tweet.getGeoLocation().toString();
String testlagi[] = test.split("latitude=");
String longs[] = testlagi[1].split(",
longitude=");
String lats = longs[0];
String longi[] = longs[1].split("}");
String username =
tweet.getUser().getScreenName();
String text = tweet.getText();
text = text.replaceAll("\"", "");
```

```
String times = tweet.getCreatedAt().toString();
String[] month = times.split("\\s");
Double latitu = Double.parseDouble(lats);
Double longitu = Double.parseDouble(longi[0]);
if(dis.cekDistinct(tweep,
tweet.getUser().getScreenName())==false)
{
tweeps objt = new tweeps(username, text, times,
latitu, longitu); tweep.add(objt);
System.out.println(username);
}
}
}
} while ((query = result.nextQuery()) != null);
System.out.println(tweep.size());
} catch (TwitterException te) {
te.printStackTrace();
System.out.println("Failed to search tweets: " +
te.getMessage());
System.exit(-1);
}
return tweep;
}
```

Kode Sumber 2 Pencarian Tweet

3. **Fungsi Menentukan *Level* Konfidensi Informasi Gempa**
   Fungsi menentukan *level* konfidensi informasi gempa digunakan untuk menentukan *level* konfidensi dari informasi yang didapat dari *social sensor*.

```
public int cekLevel(double jarak, String input,
String cekString, Integer retweet) throws
JsonSyntaxException, MalformedURLException,
IOException {
int level = 0;
int jar = 0;
city = readFile.loadKota();
if (jarak < THRESHOLD1) {
```

```
level = 9;
}
else if (jarak > THRESHOLD1 && jarak <=
THRESHOLD2) {
if(input.contains(cekString)) {
level = 8;
}
else {
String kota = null;
for (int i = 0; i < city.size(); i++) {
if(input.contains(city.get(i).toUpperCase())) {
kota = city.get(i).toUpperCase();
}
}
String tes[] = parse.getXml(kota).split(",");
String tes1[] =
parse.getXml(cekString).split(",");
Jar=(int)haversineFormula.haversine(Double.parse
Double(tes[0]),
Double.parseDouble(tes[1]),Double.parseDouble(te
s1[0]), Double.parseDouble(tes1[1]));
if (jar <= 20) {
level = 7;
}
else {
level = 6;
}
}
}
else {
if (retweet >= 100) {
level = 5;
}
else if (retweet >10 && retweet < 100) {
if(input.contains(cekString)) {
level = 4;
}
else {
String kota = null;
```

```
for (int i = 0; i < city.size(); i++) {
if(input.contains(city.get(i).toUpperCase())) {
kota = city.get(i).toUpperCase();
}
}
String tes[] = parse.getXml(kota).split(",");
String tes1[] =
parse.getXml(cekString).split(",");
jar=(int)haversineFormula.haversine(Double.parse
Double(tes[0]),
Double.parseDouble(tes[1]),Double.parseDouble(te
s1[0]), Double.parseDouble(tes1[1]));
if (jar <= 20) {
level = 3;
}
else {
level = 2;
}
}
}
else {
level = 1;
}
}
return level;
}
```

Kode Sumber 3 Menentukan Level Konfidensi Informasi Gempa

4. Fungsi Rumus Haversine

Rumus Haversine digunakan untuk menghitung jarak diantara lokasi terjadinya gempa dan lokasi tweet.

```
public static double haversine(double lat1,
double lon1, double lat2, double lon2) {
double dLat = Math.toRadians(lat2 - lat1);
double dLon = Math.toRadians(lon2 - lon1);
lat1 = Math.toRadians(lat1);
lat2 = Math.toRadians(lat2);
```

```
double a = Math.sin(dLat / 2) * Math.sin(dLat /
2) +
Math.sin(dLon / 2) * Math.sin(dLon / 2) *
Math.cos(lat1) * Math.cos(lat2);
double c = 2 * Math.asin(Math.sqrt(a));
return R * c;
}
```

Kode Sumber 4 Rumus Haversine

5. **Fungsi Menghitung *Level* Konfidensi Informasi Gempa**
   Fungsi ini digunakan untuk menghitung jumlah *tweet* dengan *level* konfidensi yang telah diolah sebelumnya.

```
public static void main(String[] args) throws
Exception, JsonSyntaxException, IOException,
MalformedURLException {
Thread t = new Thread(new mainProgram());
t.start();
}
public void run() {
cekLevelkonfidensi cek = new
cekLevelkonfidensi();
ArrayList<gempa> gempa = new ArrayList<gempa>();
ArrayList<tweeps> twerk = new
ArrayList<tweeps>();
List<Long> rataGempa = new ArrayList<Long>();
List<Long> rataTweet = new ArrayList<Long>();
int temps = 0;
gempa = tesGempa.loadGempa();
twerk = tesTweet.loadTweet();
while (true) {
try {
tambah.initialize();
} catch (SQLException e1) {
e1.printStackTrace();
}
System.out.println(twerk.size());
System.out.println(gempa.size());
for(int j = 0; j<gempa.size(); j++)
```

```
{
for(int i = 0; i<twerk.size(); i++)
{
long waktuTweets =
Long.parseLong(twerk.get(i).getTimes());
Date waktutweet = new Date(waktuTweets);
Date waktuGempa = new
Date(gempa.get(j).getTime());
long diff = Math.abs(waktutweet.getTime() -
waktuGempa.getTime());
if (diff <= ONE_HOUR) {
jarak=
(int)haversineFormula.haversine(twerk.get(i).get
Latitude(),
twerk.get(i).getLongitude(),gempa.get(j).getLati
tude(), gempa.get(j).getLongitude());
);
try {
level = cek.cekLevel(jarak,
twerk.get(i).getText().toUpperCase(),
gempa.get(j).getPlace().toUpperCase(),twerk.get(
i).getRetweet());
count[j][level-1]++;
try {
tambah.insertGempa(j,gempa.get(j).getMagnitude()
,
gempa.get(j).getPlace(),gempa.get(j).getTime(),
gempa.get(j).getLatitude(),gempa.get(j).getLongi
tude(),
twerk.size(),count[j][8],count[j][7],count[j][6]
,count[j][5],
count[j][4],count[j][3],count[j][2],count[j][1],
count[j][0]);
tambah.insertTweet(temp,j,twerk.get(i).getUserna
me(),
twerk.get(i).getText(),twerk.get(i).getTimes(),
twerk.get(i).getLatitude(),
twerk.get(i).getLongitude(), level);
}
```

```
catch (SQLException e) {
}
} catch (JsonSyntaxException e) {
e.printStackTrace();
} catch (MalformedURLException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
}
temp++;
}
}
try {
tambah.updateGempa(j,count[j][8],count[j][7],cou
nt[j][6],
count[j][5],count[j][4],count[j][3],count[j][2],
count[j][1], count[j][0] );
} catch (SQLException e) {
e.printStackTrace();
}
}
try {
Thread.sleep(3600000);
} catch (InterruptedException e) {
e.printStackTrace();
}
temps++;
}
}
```

Kode Sumber 5 Menghitung *Level* Konfidensi Informasi Gempa