**APLIKASI MONITORING GEMPA BUMI   
BERBASIS SMARTPHONE SISTEM OPERASI ANDROID  
(MEMANFAATKAN DATA DARI USGS)**

# HALAMAN JUDUL

**TUGAS AKHIR**

Diajukan Untuk Memenuhi Salah Satu Syarat Kelulusan Sarjana

Pada Jurusan Teknik Informatika STMIK BANDUNG BALI

Program Pendidikan Strata-1 (S1)

Disusun Oleh :

**I NENGAH ADI SAYOGA**

**1208004**



**JURUSAN TEKNIK INFORMATIKA**

**SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER**

**BANDUNG BALI**

**DENPASAR**

**2012**

**APLIKASI MONITORING GEMPA BUMI**

**BERBASIS SMARTPHONE SISTEM OPERASI ANDROID  
(MEMANFAATKAN DATA DARI USGS)**

# HALAMAN PENGESAHAN

**TUGAS AKHIR**

Diajukan Untuk Memenuhi Salah Satu Syarat Kelulusan Sarjana   
Pada Jurusan Teknik Informatika STMIK BANDUNG BALI   
Program Pendidikan Srtrata-1 (S1)

Disusun Oleh:

**I Nengah Adi Sayoga 1208004**

Telah diperiksa dan disetujui sebagai Tugas Akhir   
Tanggal 25 Pebruari 2012

(Ir. I Gede Made Karma, MT) (I Putu Gd Sukanada Andisana, S.Kom)

Dosen Pembimbing Ketua Jurusan Teknik Informatika

# PERNYATAAN KEASLIAN

Saya sebagai penulis Tugas Akhir menyatakan dengan sesungguhnya bahwa karya tulis ini merupakan hasil pemikiran dan pemaparan asli. Apabila terdapat referensi terhadap karya orang lain/pihak lain, penulis lakukan dengan menyebut sumber secara jelas.

Demikian pernyataan ini penulis buat secara sadar dan sebenar-benarnya.

Penulis

(I Nengah Adi Sayoga)

NIM 1208004

# ABSTRAK

Gempa bumi merupakan kejadian alam yang tidak dapat diprediksi oleh setiap orang termasuk para ilmuan, sehingga pada saat terjadi gempa banyak masyarakat yang tidak siap untuk menghadapi gempa tersebut dan menimbulkan banyak korban jiwa serta kerusakan infrastruktur. Diperlukan suatu sistem yang dapat memonitoring terjadinya gempa, sehingga masyarakat memperoleh informasi yang lebih cepat dan akurat. Di sisi lain, perkembangan perangkat bergerak begitu pesatnya, salah satu pimpinan atas adalah jajaran perangkat Android.

Pada tugas akhir ini dibuat aplikasi monitoring gempa bumi untuk perangkat bergerak yang berbasis sistem operasi Android. Sistem dapat mengecek gempa baru secara real-time dan memberitahukan ke user bahwa terjadi gempa di suatu wilayah baik melalui notifikasi *status bar*, mengirimkan SMS ke orang tertentu, mengirim email, posting ke Facebook maupun posting ke Twitter. Pada aplikasi juga berisi fasilitas untuk melihat peta lokasi gempa di seluruh dunia ditambah lokasi pengguna saat ini.

Pengembangan aplikasi meliputi tahapan analisis, perancangan, dan implementasi dengan menerapkan rekayasa perangkat lunak berorientasi objek. Pemodelan sistem menggunakan *Unitied Modeling Language* (UML). Implementasi menggunakan *tools* Android SDK (Software Development Kit) dan editor Eclipse IDE (*Integrated Development Environment*) dengan bahasa pemrograman Java. Database yang digunakan adalah SQLite.

Kata kunci: Gempa Bumi, Berorientasi Objek, UML, Java, Android, Android SDK, SQLite, Eclipse

# *ABSTRACT*

*Earthquakes are a natural occurrence that can not be predicted by any person including the scientists, so that at the time of the earthquakes, many people are not prepared to deal with and caused many casualties and infrastructure damage. Thus needed a system that can monitor the occurrence of earthquakes, so the public can obtain information more quickly and accurately. On the other hand, development of mobile devices grows so fast, one of the top leaders is a line of Android devices.*

*In this thesis author makes an earthquake monitoring application for mobile devices based on Android operating system. The system can check for new earthquakes in real-time and notify the user that there was an earthquake in a region either through status bar notifications, send SMS to a specific person, send an Email, posting to Facebook, or posting to Twitter. The application also contains a facility to view an earthquakes map around the world plus the user's current location.*

*Stages of application development includes analysis, design, and implementation by applying object-oriented software engineering. Modeling system used Unitied Modeling Language (UML). Implementation used Android SDK (Software Development Kit) tools and an Eclipse IDE (Integrated Development Environment) editor, with Java programming language. The database used is SQLite.*

*Keywords: Earthquake, Object Oriented, UML, Java, Android, Android SDK, SQLite, Eclipse*

# KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Tuhan Yang Maha Esa (Ida Sang Hyang Widhi Wasa) sehingga Penulis dapat menyelesaikan tugas akhir dengan judul **“Aplikasi Monitoring Gempa Bumi Berbasis Smartphone Sistem Operasi Android (Memanfaatkan data dari USGS)”** sebagai salah satu syarat dalam menyelesaikan program sarjana pada jurusan Teknik Informatika STMIK BANDUNG BALI.

Dalam penyusunan laporan ini, penulis telah banyak menerima dukungan, bantuan dan bimbingan dari berbagai pihak. Pada kesempatan ini Penulis ingin berterima kasih yang sebesar-besarnya kepada:

1. Bapak Ir. I Gede Made Karma, MT, selaku dosen pembimbing dan penguji pada seminar I, yang telah memberikan bimbingan, sumbangan pemikiran dan wawasannya kepada Penulis.
2. Bapak I Putu Gede Sukanada Andisana, S.Kom, selaku penguji pada seminar I, yang telah memberikan masukan dan saran-saran yang sangat bermanfaat.
3. Bapak Komang Mardika, S.Kom, selaku penguji pada seminar I dan seminar II, yang telah memberikan perhatian dan saran-saran yang sangat bermanfaat.
4. Bapak I Wayan Candra Winetra M.Kom, selaku penguji pada seminar II, yang telah banyak memberikan masukan dan ide yang sangat berarti bagi kesempurnaan tugas akhir.
5. Bapak Moch Rizki Romdoni S.Kom, selaku penguji pada seminar II dan sidang tugas akhir telah memberikan koreksi dan perbaikan yang sangat berguna bagi Penulis.
6. Ibu Luh Putu Ayu Prapitasari, MSc, selaku penguji pada sidang tugas akhir yang telah memberikan koreksi perbaikan dan nasihat-nasihat yang sangat berguna bagi Penulis.
7. Seluruh staf dosen di STMIK BANDUNG BALI yang telah mengajari penulisan selama ini.
8. Teman-teman dan semua pihak yang tidak dapat Penulis sebutkan satu-persatu.

Penulis menyadari bahwa dalam penyusunan laporan ini masih terdapat banyak kekurangan dan jauh dari sempurna. Mengingat kemampuan dan pengalaman Penulis yang masih terbatas dalam penelitian di lapangan, teknik pengolahan data dan teknik penyajian, sehingga dengan rendah hati Penulis mengharapkan kritik dan saran yang bersifat membangun dari semua pihak.

Akhir kata, semoga laporan ini dapat bermanfaat bagi penulis khususnya dan bagi semua pihak yang berkepentingan pada umumnya.

Denpasar, Pebruari 2012

Penulis

# DAFTAR ISI

[HALAMAN JUDUL i](#_Toc318743219)

[HALAMAN PENGESAHAN ii](#_Toc318743220)

[PERNYATAAN KEASLIAN iii](#_Toc318743221)

[ABSTRAK iv](#_Toc318743222)

[*ABSTRACT* v](#_Toc318743223)

[KATA PENGANTAR vi](#_Toc318743224)

[DAFTAR ISI viii](#_Toc318743225)

[DAFTAR GAMBAR xii](#_Toc318743226)

[DAFTAR TABEL xv](#_Toc318743227)

[DAFTAR PROGRAM xvi](#_Toc318743228)

[BAB I PENDAHULUAN 1](#_Toc318743229)

[1.1 Latar Belakang 1](#_Toc318743230)

[1.2 Rumusan Masalah 3](#_Toc318743231)

[1.3 Batasan Masalah 3](#_Toc318743232)

[1.4 Tujuan 4](#_Toc318743233)

[1.5 Metode Penelitian 5](#_Toc318743234)

[1.5.1 Metode Pengumpulan Data 5](#_Toc318743235)

[1.5.2 Metode Perancangan Sistem 5](#_Toc318743236)

[BAB II LANDASAN TEORI 9](#_Toc318743237)

[2.1 Gempa Bumi 9](#_Toc318743238)

[2.2 Geolokasi 12](#_Toc318743239)

[2.3 Java 13](#_Toc318743240)

[2.3.1 Apa itu teknologi JAVA? 14](#_Toc318743241)

[2.3.2 Sebagian Fitur dari Java 15](#_Toc318743242)

[2.4 Android 16](#_Toc318743243)

[2.4.1 Versi Android 17](#_Toc318743244)

[2.4.2 Perbedaan Antara Android *Original* dan OHD 18](#_Toc318743245)

[2.4.3 Arsitektur Android 18](#_Toc318743246)

[2.4.4 Dasar-Dasar Aplikasi Android 20](#_Toc318743247)

[2.4.5 Komponen Aplikasi Android 21](#_Toc318743248)

[2.4.5.1 Activity 21](#_Toc318743249)

[2.4.5.2 *Service* 23](#_Toc318743250)

[2.4.5.3 *Content Provider* 24](#_Toc318743251)

[2.4.5.4 *Broadcast Receiver* 24](#_Toc318743252)

[2.5 Basis Data 25](#_Toc318743253)

[2.6 SQLite 27](#_Toc318743254)

[2.7 *Unified Modeling Language* (UML) 28](#_Toc318743255)

[2.7.1 *Use Case Diagram* 30](#_Toc318743256)

[2.7.2 *Activity Diagram* 31](#_Toc318743257)

[2.7.3 *Sequence Diagram* 33](#_Toc318743258)

[2.7.4 *Class Diagram* 36](#_Toc318743259)

[2.7.5 Package Diagram 38](#_Toc318743260)

[BAB III ANALISA DAN PERANCANGAN SISTEM 39](#_Toc318743261)

[3.1 Deskripsi Skenario 39](#_Toc318743262)

[3.2 *Use Case Diagram* 41](#_Toc318743263)

[3.3 *Use Case Description* 42](#_Toc318743264)

[3.3.1 *Refresh* Data 42](#_Toc318743265)

[3.3.2 Pemberitahuan 43](#_Toc318743266)

[3.3.3 Menampilkan Notifikasi 44](#_Toc318743267)

[3.3.4 Kirim SMS 45](#_Toc318743268)

[3.3.5 Kirim Email 46](#_Toc318743269)

[3.3.6 *Share* ke Facebook 46](#_Toc318743270)

[3.3.7 *Share* ke Twitter 47](#_Toc318743271)

[*3.3.8* Menampilkan pada *List* 48](#_Toc318743272)

[3.3.9 Menampilkan Detail Gempa 48](#_Toc318743273)

[3.3.10 Menampilkan pada Peta 49](#_Toc318743274)

[3.4 *Activity Diagram* 50](#_Toc318743275)

[3.4.1 *Refresh* Data 50](#_Toc318743276)

[3.4.2 Pemberitahuan 51](#_Toc318743277)

[3.4.3 Menampilkan Notifikasi 52](#_Toc318743278)

[3.4.4 Kirim SMS 52](#_Toc318743279)

[3.4.5 Kirim Email 52](#_Toc318743280)

[3.4.6 *Share* ke Facebook 53](#_Toc318743281)

[3.4.7 *Share* ke Twitter 53](#_Toc318743282)

[3.4.8 Menampilkan pada *List* 53](#_Toc318743283)

[3.4.9 Menampilkan Detail Gempa 54](#_Toc318743284)

[3.4.10 Menampilkan pada Peta 54](#_Toc318743285)

[3.5 *Sequence Diagram* 55](#_Toc318743286)

[3.5.1 *Refresh* Data 55](#_Toc318743287)

[3.5.2 Pemberitahuan 56](#_Toc318743288)

[3.5.3 Menampilkan Notifikasi 57](#_Toc318743289)

[3.5.4 Kirim SMS 58](#_Toc318743290)

[3.5.5 Kirim Email 58](#_Toc318743291)

[3.5.6 *Share* ke Facebook 59](#_Toc318743292)

[3.5.7 Share ke Twitter 59](#_Toc318743293)

[3.5.8 Menampilkan pada *List* 60](#_Toc318743294)

[3.5.9 Menampilkan Detail Gempa 60](#_Toc318743295)

[3.5.10 Menampilkan pada Peta 61](#_Toc318743296)

[3.6 *Package Diagram* 62](#_Toc318743297)

[3.7 Class Diagram 63](#_Toc318743298)

[3.8 Struktur *Database* 65](#_Toc318743299)

[3.8.1 Tabel Earthquake 65](#_Toc318743300)

[3.8.2 Tabel Contact 65](#_Toc318743301)

[3.9 Perancangan Sistem 66](#_Toc318743302)

[3.9.1 Daftar Gempa 66](#_Toc318743303)

[3.9.2 Detail Gempa 70](#_Toc318743304)

[3.9.3 Peta Gempa 71](#_Toc318743305)

[3.9.4 Pengaturan 73](#_Toc318743306)

[BAB IV IMPLEMENTASI SISTEM 84](#_Toc318743307)

[4.1 Kebutuhan Implementasi 84](#_Toc318743308)

[4.1.1 Google Maps 86](#_Toc318743309)

[4.1.2 Facebook 88](#_Toc318743310)

[4.1.3 Twitter 89](#_Toc318743311)

[4.2 Kebutuhan Perangkat Keras 90](#_Toc318743312)

[4.3 Komponen Sistem 92](#_Toc318743313)

[4.3.1 Service 92](#_Toc318743314)

[4.3.2 Daftar Gempa 93](#_Toc318743315)

[4.3.3 Detail Gempa 96](#_Toc318743316)

[4.3.4 Peta Gempa 100](#_Toc318743317)

[4.3.5 Pengaturan 102](#_Toc318743318)

[4.4 Pengujian Sistem 104](#_Toc318743319)

[BAB V PENUTUP 109](#_Toc318743320)

[5.1 Kesimpulan 109](#_Toc318743321)

[5.2 Saran 110](#_Toc318743322)

[DAFTAR PUSTAKA 111](#_Toc318743323)

[LAMPIRAN KODE PROGRAM 113](#_Toc318743324)

# DAFTAR GAMBAR

[Gambar 1.1 Model *Waterfall* (*Classic Life Cycle*) 6](#_Toc318659381)

[Gambar 2.1 Arsitektur Android 19](#_Toc318659382)

[Gambar 2.2 Siklus Hidup *Activity* 22](#_Toc318659383)

[Gambar 2.3 Kruchten 4+1 *Model View* 28](#_Toc318659384)

[Gambar 2.4 Elemen didalam *Sequence Diagram* 33](#_Toc318659385)

[Gambar 3.1 *Use Case Diagram* 41](#_Toc318659386)

[Gambar 3.2 *Activity Diagram* – *Refresh* Data 50](#_Toc318659387)

[Gambar 3.3 *Activity Diagram* – Pemberitahuan 51](#_Toc318659388)

[Gambar 3.4 *Activity Diagram* – Menampilkan Notifikasi 52](#_Toc318659389)

[Gambar 3.5 *Activity Diagram* – Kirim SMS 52](#_Toc318659390)

[Gambar 3.6 *Activity Diagram* – Kirim Email 52](#_Toc318659391)

[Gambar 3.7 *Activity Diagram* – *Share* ke Facebook 53](#_Toc318659392)

[Gambar 3.8 *Activity Diagram* – *Share* ke Twitter 53](#_Toc318659393)

[Gambar 3.9 *Activity Diagram* – Menampilkan pada *List* 53](#_Toc318659394)

[Gambar 3.10 *Activity Diagram* – Menampilkan Detail Gempa 54](#_Toc318659395)

[Gambar 3.11 *Activity Diagram* – Menampilkan pada Peta 54](#_Toc318659396)

[Gambar 3.12 *Sequence Diagram* – *Refresh* Data 55](#_Toc318659397)

[Gambar 3.13 *Sequence Diagram* – Pemberitahaun 56](#_Toc318659398)

[Gambar 3.14 *Sequence Diagram* – Menampilkan Notifikasi 57](#_Toc318659399)

[Gambar 3.15 *Sequence Diagram* – Notifikasi Diklik 57](#_Toc318659400)

[Gambar 3.16 *Sequence Diagram* – Kirim SMS 58](#_Toc318659401)

[Gambar 3.17 *Sequence Diagram* – Kirim Email 58](#_Toc318659402)

[Gambar 3.18 *Sequence Diagram* – *Share* ke Facebook 59](#_Toc318659403)

[Gambar 3.19 *Sequence Diagram* – *Share* ke Facebook 59](#_Toc318659404)

[Gambar 3.20 *Sequence Diagram* – Menampilkan pada *List* 60](#_Toc318659405)

[Gambar 3.21 *Sequence Diagram* – Menampilkan Detail Gempa 60](#_Toc318659406)

[Gambar 3.22 *Sequence Diagram* – Menampilkan pada Peta 61](#_Toc318659407)

[Gambar 3.23 *Package Diagram* 62](#_Toc318659408)

[Gambar 3.24 *Class Diagram* 63](#_Toc318659409)

[Gambar 3.25 *Class Diagram* (Lanjutan) 64](#_Toc318659410)

[Gambar 3.26 Rancangan Tampilan Daftar Gempa 66](#_Toc318659411)

[Gambar 3.27 Rancangan Tampilan *Context Menu* 67](#_Toc318659412)

[Gambar 3.28 Rancangan Tampilan *QuickAction* 68](#_Toc318659413)

[Gambar 3.29 Rancangan Tampilan Kirim Email 69](#_Toc318659414)

[Gambar 3.30 Rancangan Tampilan Kirim Pesan SMS 69](#_Toc318659415)

[Gambar 3.31 Rancangan Tampilan Detail Gempa 70](#_Toc318659416)

[Gambar 3.32 Rancangan Tampilan Peta Gempa 71](#_Toc318659417)

[Gambar 3.33 Rancangan Tampilan *Balloon View* 72](#_Toc318659418)

[Gambar 3.34 Rancangan Tampilan Layer Peta 73](#_Toc318659419)

[Gambar 3.35 Rancangan Tampilan Pengaturan Lokasi Manual 75](#_Toc318659420)

[Gambar 3.36 Rancangan Tampilan *Template* Twitter 78](#_Toc318659421)

[Gambar 3.37 Rancangan Tampilan Pengaturan Akun Email 79](#_Toc318659422)

[Gambar 3.38 Rancangan Tampilan Pengaturan *Template* Email 80](#_Toc318659423)

[Gambar 3.39 Rancangan Tampilan *Social* *Connect* 81](#_Toc318659424)

[Gambar 3.40 Rancangan Tampilan Daftar Kontak 82](#_Toc318659425)

[Gambar 4.1 Implementasi Tampilan Daftar Gempa 94](#_Toc318659426)

[Gambar 4.2 Implementasi Tampilan *Context Menu* dan *Quick Action* 95](#_Toc318659427)

[Gambar 4.3 Implementasi Tampilan Detail Gempa 96](#_Toc318659428)

[Gambar 4.4 Implementasi Tampilan Share Menggunakan Intent 98](#_Toc318659429)

[Gambar 4.5 Implementasi Tampilan Kirim SMS dan Kirim Email 98](#_Toc318659430)

[Gambar 4.6 Implementasi Tampilan *Share* ke Facebook dan Twitter 99](#_Toc318659431)

[Gambar 4.7 Implementasi Tampilan pada Halaman Facebook 99](#_Toc318659432)

[Gambar 4.8 Implementasi Tampilan pada Halaman Twitter 99](#_Toc318659433)

[Gambar 4.9 Implementasi Tampilan Peta Gempa 100](#_Toc318659434)

[Gambar 4.10 Implementasi Tampilan *Layers* Peta 101](#_Toc318659435)

[Gambar 4.11 Implementasi Tampilan Pengaturan 102](#_Toc318659436)

[Gambar 4.12 Implementasi Tampilan Input Lokasi Manual 102](#_Toc318659437)

[Gambar 4.13 Implementasi Tampilan Template Pesan 103](#_Toc318659438)

[Gambar 4.14 Implementasi Tampilan Daftar Kontak 103](#_Toc318659439)

[Gambar 4.15 Implementasi Tampilan *Social* *Connect* 104](#_Toc318659440)

# DAFTAR TABEL

[Tabel 2.1 Skala Kekuatan Gempa Bumi 11](#_Toc318659441)

[Tabel 2.2 Fragmen pada *Sequence Diagram* 36](#_Toc318659442)

[Tabel 3.1 *Use Case Description* – *Refresh* Data 43](#_Toc318659443)

[Tabel 3.2 *Use Case Description* – Pemberitahuan 44](#_Toc318659444)

[Tabel 3.3 *Use Case Description* – Kirim Notifikasi 45](#_Toc318659445)

[Tabel 3.4 *Use Case Description* – Kirim SMS 45](#_Toc318659446)

[Tabel 3.5 *Use Case Description* – Kirim Email 46](#_Toc318659447)

[Tabel 3.6 *Use Case Description* – *Share* ke Facebook 47](#_Toc318659448)

[Tabel 3.7 *Use Case Description* – *Share* ke Twitter 47](#_Toc318659449)

[Tabel 3.8 *Use Case Description* – Menampilkan pada *List* 48](#_Toc318659450)

[Tabel 3.9 *Use Case Description* – Menampilkan Detail Gempa 49](#_Toc318659451)

[Tabel 3.10 *Use Case Description* – Menampilkan pada Peta 49](#_Toc318659452)

[Tabel 3.11 Tabel Earthquake 65](#_Toc318659453)

[Tabel 3.12 Tabel Contact 65](#_Toc318659454)

[Tabel 4.1 Keperluan Ruang Penyimpanan Android SDK 90](#_Toc318659455)

# DAFTAR PROGRAM

[Listing 4.1 Menghasilkan Sertifikat Sidik Jari 86](#_Toc317359136)

[Listing 4.2 *Reference* Maps API Key pada *Layout* 87](#_Toc317359137)

[Listing 4.3 *Reference* Maps API Key pada Kode Program 87](#_Toc317359138)

[Listing 4.4 Elemen <uses-linrary> pada AndroidManifest.xml 88](#_Toc317359139)

[Listing 4.5 Menghasilkan Tanda Tangan Aplikasi untuk Facebook 89](#_Toc317359140)

[Listing 4.6 Inisiasi Objek Facebook 89](#_Toc317359141)

[Listing 4.7 Inisiasi Objek Twitter 90](#_Toc317359142)

[Listing 4.8 Ijin Aplikasi pada AndroidManifest.xml 92](#_Toc317359143)

[Listing 4.9 Potongan Kode Program pada *Service* 93](#_Toc317359144)

[Listing 4.10 *Receiver* dari *Broadcast* Daftar Gempa 94](#_Toc317359145)

[Listing 4.11 *Handler* Daftar Gempa untuk Dilewatkan ke *Receiver* 95](#_Toc317359146)

[Listing 4.12 Memulai *Activity* Detail Gempa 97](#_Toc317359147)

[Listing 4.13 Mendapatkan Data Gempa dari *Intent* 97](#_Toc317359148)

[Listing 4.14 *Share* konten menggunakan Intent 97](#_Toc317359149)

[Listing 4.15 Potongan Program Menggambar Lokasi Gempa 101](#_Toc317359150)

[Listing 4.16 Data Gempa pada Server 106](#_Toc317359151)

[Listing 4.17 Output LogCat *Start* Awal 107](#_Toc317359152)

[Listing 4.18 Output LogCat *Refresh Data* 108](#_Toc317359153)

# PENDAHULUAN

## Latar Belakang

Secara histografi, Indonesia merupakan wilayah langganan gempa bumi dan tsunami. Berbagai daerah di Indonesia merupakan titik rawan bencana, terutama bencana gempa bumi, tsunami, banjir, dan letusan gunung berapi. Wilayah Indonesia dikepung oleh lempeng Eurasia, lempeng Indo-Australia, dan lempeng Pasifik. Sewaktu-waktu lempeng ini akan bergeser/patah dan menimbulkan gempa bumi. Selanjutnya jika terjadi tumbukan antar-lempeng tektonik, dapat menghasilkan tsunami, seperti yang terjadi di Aceh dan Sumatera Utara.

Selain dikepung tiga lempeng tektonik dunia, Indonesia juga merupakan jalur *The Pasific Ring of Fire* (Cincin Api Pasifik), yang merupakan jalur rangkaian gunung api aktif di dunia. Cincin api Pasifik membentang diantara subduksi maupun pemisahan lempeng Pasifik dengan lempeng Indo-Australia, lempeng Eurasia, lempeng Amerika Utara dan lempeng Nazca yang bertabrakan dengan lempeng Amerika Selatan.

Gempa bumi merupakan kejadian alam yang tidak dapat diprediksi oleh setiap orang termasuk para ilmuan, sehingga pada saat terjadi gempa banyak masyarakat yang tidak siap untuk menghadapi gempa tersebut dan menimbulkan banyak korban jiwa serta kerusakan infrastruktur. Melihat permasalahan yang ada, maka diperlukan suatu sistem yang dapat memonitoring terjadinya gempa, sehingga masyarakat memperoleh informasi yang lebih cepat dan akurat. Untuk memperoleh informasi yang cepat dan akurat diperlukan suatu perangkat dan aplikasi yang dapat menunjang pengaksesan data yang maksimal. Dibutuhkan sebuah aplikasi monitoring yang dapat memperlihatkan posisi pusat gempa terjadi pada peta (*Map*) dan dibutuhkan pula aplikasi untuk mengetahui keberadaan seseorang.

Kemajuan dan perkembangan dunia terus melaju dan berkembang dengan pesat. Saat ini adalah waktu yang menarik bagi pengembang *mobile*. Ponsel yang *stylish* dan serbaguna dikemas dengan fitur hardware seperti GPS, akselerometer, dan layar sentuh, dikombinasikan dengan harga terjangkau menyediakan platform yang menarik untuk membuat aplikasi *mobile* yang inovatif.

Sejumlah *handset* Android sekarang tersedia untuk menggoda konsumen. Kemenangan sesungguhnya, adalah untuk pengembang. Dengan banyaknya pengembangan *mobile* yang dibangun pada sistem operasi berpemilik yang membatasi pengembangan dan penyebaran aplikasi pihak ketiga, Android menawarkan alternatif terbuka. Tanpa hambatan, pengembang Android bebas untuk menulis aplikasi dengan mengambil keuntungan penuh dari *hardware* ponsel yang semakin tangguh dan mendistribusikannya dalam pasar terbuka.

Oleh karena itu, pada tugas akhir ini Penulis mengusulkan sebuah aplikasi “MONITORING GEMPA BUMI BERBASIS SMARTPHONE SISTEM OPERASI ANDROID”, sekaligus ingin mengeksplorasi teknologi baru Android yang saat ini berkembang semakin pesat.

## Rumusan Masalah

Masalah yang akan dibahas dalam penelitian ini adalah:

1. Bagaimana mendapatkan data gempa bumi?
2. Dimana lokasi pusat gempa pada peta dan berapa kekuatannya?
3. Daerah mana saja yang berimbas pada gempa bumi yang terjadi?
4. Bagaimana cara memberitahu orang terdekat tentang gempa yang terjadi?

## Batasan Masalah

Agar pembahasan penelitian ini tidak menyimpang dengan apa yang menjadi pokok permasalahan maka diperlukan batasan-batasan. Batasan masalah dalam penelitian ini adalah:

* 1. Data gempa bumi didapat dari situs web USGS (<http://earthquake.usgs.gov/earthquakes/catalogs/>). Pada situs web ini, disediakan tiga jenis feed yaitu RSS (*Really Simple Syndication*), Atom, dan CSV (*Comma-Separated Values*). Untuk meminimalkan *traffic* data, aplikasi akan menggunakan CSV.
  2. Aplikasi berbasis sistem operasi Android dengan bahasa pemrograman Java dan database yang digunakan adalah SQLite.
  3. Sistem juga akan berjalan pada *background service* sehingga dapat memonitor gempa secara *real-time*.
  4. Lokasi pusat gempa dapat dilihat pada peta (Google *Maps*), dan sistem juga menggunakan GPS untuk menentukan lokasi *user* atau dapat juga menginputkan lokasinya secara manual.
  5. Sistem akan memberikan notifikasi atau pemberitahuan jika terjadi gempa bumi. Pemberitahuan ini dibedakan menjadi dua, pemberitahuan untuk lokasi yang dekat dengan wilayah *user* (regional) dan lokasi seluruh dunia (global).
  6. *User* dapat mengatur berapa magnitudo minimal notifikasi akan ditampilkan, baik untuk lokasi wilayahnya maupun global.
  7. Sistem juga dapat mengirim pesan SMS dan Email ke tujuan tertentu yang dapat ditentukan oleh *user*, dan juga dapat mempostingnya ke Facebook dan Twitter.
  8. *User* dapat mengatur apakah akan otomatis mengirim SMS, mengirim Email, posting ke Facebook dan/atau posting ke Twitter jika terjadi gempa pada magnitudo tertentu atau tidak. Isi pesan juga dapat diatur.

## Tujuan

Adapun tujuan penulisan tugas akhir ini yaitu:

1. Memberikan informasi tentang gempa bumi yang terjadi secepat mungkin sehingga dapat meminimalkan kepanikan yang berlebihan.
2. Untuk menambah wawasan bahwa di belahan bumi tertentu telah terjadi bencana gempa bumi.

## Metode Penelitian

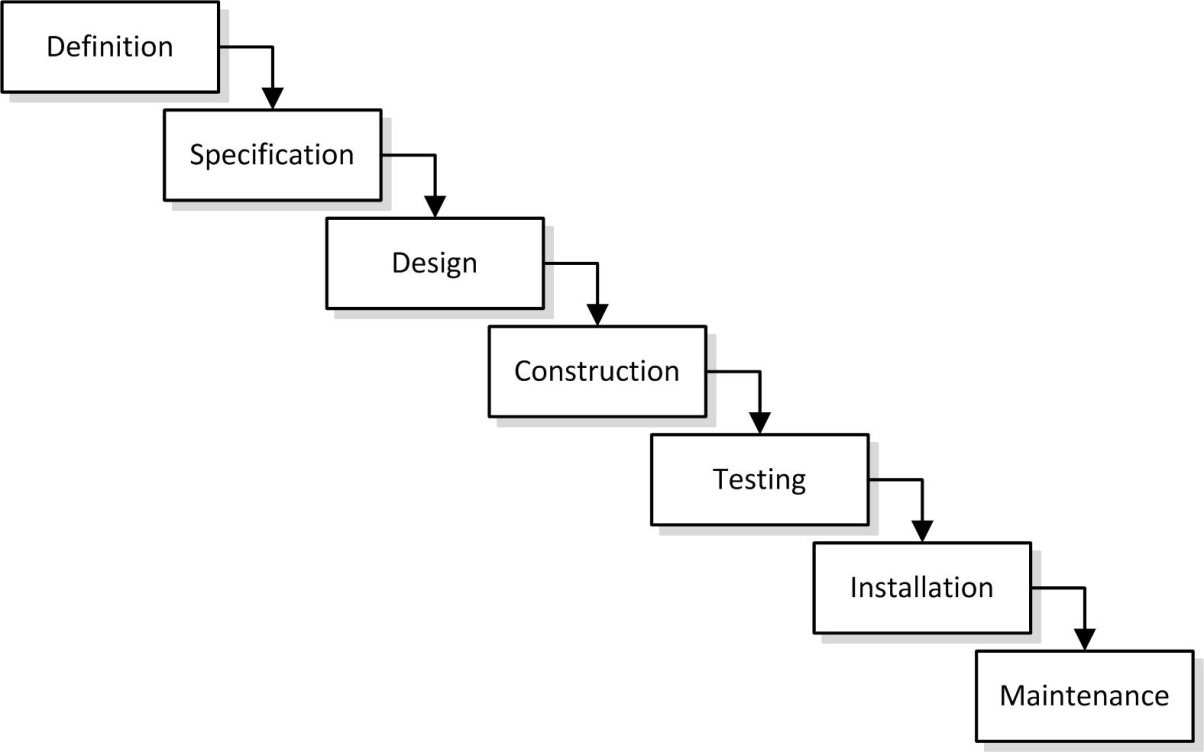
Metode penelitian sangat menentukan suatu penelitian, karena menyangkut cara yang benar dalam pengumpulan data, analisa data dan pengambilan kesimpulan hasil penelitian. Adapun metode penelitian yang digunakan dalam tugas akhir ini adalah:

### Metode Pengumpulan Data

Metode pengumpulan data ini dilakukan untuk menunjang pengembangan sistem yang dibuat. Adapun teknik yang dilakukan adalah Studi Literatur yaitu melakukan pengumpulan berbagai data dan informasi baik dari buku maupun website yang berkaitan dengan gempa bumi maupun sistem operasi Android untuk mendukung aplikasi yang akan dirancang.

### Metode Perancangan Sistem

Paradigma yang digunakan dalam penyusunan tugas akhir ini adalah metode *Waterfall Model* (*Classic Life Cycle*) yang dilakukan dari tahap definisi sampai pemeliharaan. Model *Waterfall*, membutuhkan pendekatan yang sistematis dalam pengembangan sebuah perangkat lunak.



Gambar 1. Model *Waterfall* (*Classic Life Cycle*)

Sumber: Dan Brandon

Keterangan gambar:

1. *Definition* (definisi).

Langkah definisi melibatkan pembuatan tujuan yang jelas, mengidentifikasi mengapa dan bagaimana sistem yang diusulkan akan lebih baik, lebih murah, lebih cepat daripada yang digantikan, dan biasanya secara keseluruhan/kasar analisis biaya-manfaat.

2. *Specification* (spesifikasi/kebutuhan).

Langkah dari spesifikasi melibatkan lingkup pernyataan lengkap (kebutuhan), skenario *use case*, penyusunana *user manual* (spesifikasi desain eksternal), rincian rencana proyek, spesifikasi sumber daya yang dibutuhkan, menyaring perkiraan waktu dan biaya, menyaring analisis biaya-manfaat, dan menandatangani persetujuan persyaratan dan *user manual* oleh para pemangku kepentingan.

3. *Design* (desain).

Langkah desain melibatkan resolusi dari masalah teknis yang kritis, pemilihan arsitektur dan *platform*, adopsi standar, penempatan staf, penyelesaian desain eksternal (desain *user interface*), desain struktur data penting dan *database*, desain internal dari algoritma dan proses, persyaratan matrix ketertelusuran, naskah uji pendahuluan, waktu final dan perkiraan biaya, dan analisis biaya-manfaat akhir.

4. *Construction* (konstruksi/pemrograman dan *unit testing*)

Langkah konstruksi melibatkan pelaksanaan desain (misalnya, melalui *coding*), *unit* testing, integrasi sistem, draf dokumentasi internal, dan penyelesaian uji skrip.

5. *Testing* (pengujian/sistem dan integrasi).

Langkah pengujian melibatkan integrasi skala penuh dan pengujian sistem, penyelesaian dokumentasi pengguna, penyelesaian material pelatihan, penerapan prosedur pengendalian perubahan formal, penyelesaian dokumentasi internal.

6. *Installation* (Instalasi).

Langkah instalasi melibatkan produk jadi, pelatihan pengguna akhir, memproduksi dokumentasi pelajaran, dan mendefinisikan prosedur untuk operasi penanganan, dukungan pengguna, dan manajemen konfigurasi.

*7. Maintenance* (operasi dan pemeliharaan).

Langkah pemeliharaan melibatkan mengikuti dan merevisi prosedur untuk menyelesaikan masalah dan eskalasi masalah, operasi, *backup* keamanan, kontrol konfigurasi, dan kualitas/pemantauan kerja.

# LANDASAN TEORI

## Gempa Bumi

Gempa bumi adalah getaran atau guncangan yang terjadi di permukaan bumi. Gempa bumi biasa disebabkan oleh pergerakan kerak bumi (lempeng bumi). Kata gempa bumi juga digunakan untuk menunjukkan daerah asal terjadinya kejadian gempa bumi tersebut. Bumi kita walaupun padat, selalu bergerak, dan gempa bumi terjadi apabila tekanan yang terjadi karena pergerakan itu sudah terlalu besar untuk dapat ditahan.

Tipe gempa bumi diantaranya:

1. Gempa bumi vulkanik (Gunung Api). Gempa bumi ini terjadi akibat adanya aktivitas magma, yang biasa terjadi sebelum gunung api meletus.

2. Gempa bumi tektonik. Gempa bumi ini disebabkan oleh adanya aktivitas tektonik, yaitu pergeseran lempeng-lempeng tektonik secara mendadak yang mempunyai kekuatan dari yang sangat kecil hingga yang sangat besar. Gempa bumi ini banyak menimbulkan kerusakan atau bencana alam di bumi, getaran gempa bumi yang kuat mampu menjalar keseluruh bagian bumi.

Kebanyakan gempa bumi disebabkan dari pelepasan energi yang dihasilkan oleh tekanan yang dilakukan oleh lempengan yang bergerak. Semakin lama tekanan itu kian membesar dan akhirnya mencapai pada keadaan dimana tidak dapat ditahan lagi oleh pinggiran lempengan. Pada saat itulah gempa bumi akan terjadi. Gempa bumi dapat direkam oleh seismometer hingga jarak yang jauh, karena gelombang seismik melakukan perjalanan melalui interior seluruh bumi ini.

Beberapaistilah yang sering digunakan dalam gempa bumi:

1. *Origin Time* yaitu waktu terjadinya gempa bumi yang dinyatakan dalam hari, tanggal, bulan, tahun, jam, menit dan detik.

2. *Episenter* (pusat gempa bumi) yaitu titik di permukaan bumi yang refleksi tegak lurus dari pusat gempa bumi, Lokasi *episenter* dinyatakan dalam derajat lintang dan derajat bujur.

3. *Hiposenter* (Kedalaman sumber gempa bumi) yaitu jarak pusat gempa bumi dihitung tegak lurus dari permukaan bumi dan dinyatakan oleh besaran jarak dalam satuan kilometer.

4. *Magnitudo* (Kekuatan gempa bumi) yaitu ukuran skala kekuatan gempa bumi yang dinyatakan dalam satuan Skala Richter (SR).

Charles Francis Richter — ahli seismologi dari Amerika Serikat, lahir di Hamilton, Ohio26 April 1900 — mengembangkan skala untuk mengukur kekuatan gempa bumi pada tahun 1935 yang dikenal sebagai Skala Richter.

|  |  |  |
| --- | --- | --- |
| **Magnitudo (SR)** | **Keterangan** | **Efek Gempa** |
| < 2.0 | *Micro* | Gempa kecil, tidak terasa |
| 2.0-2.9 | *Minor* | Tidak terasa, namun terekam oleh alat |
| 3.0-3.9 | *Minor* | Seringkali terasa, namun jarang menimbulkan kerusakan |
| 4.0-4.9 | *Light* | Dapat diketahui dari bergetarnya perabot dalam ruangan, suara gaduh bergetar. Kerusakan tidak terlalu signifikan |
| 5.0-5.9 | *Moderate* | Dapat menyebabkan kerusakan besar pada bangunan pada area yang kecil. Umumya kerusakan kecil pada bangunan yang didesain dengan baik |
| 6.0-6.9 | *Strong* | Dapat merusak area hingga jarak sekitar 160 km |
| 7.0-7.9 | *Major* | Dapat menyebabkan kerusakan serius dalam area lebih luas |
| 8.0-8.9 | *Great* | Dapat menyebabkan kerusakan serius hingga dalam area ratusan mil |
| 9.0-9.9 | *Great* | Menghancurkan area ribuan mil |
| 10.0+ | *Massive* | Belum pernah terekam |

Tabel 2. Skala Kekuatan Gempa Bumi

## Geolokasi

Ratuasan tahun yang lalu, orang mengandalkan bentuk visual dari geolokasi untuk membantu menentukan dimana meraka berada pada suatu area. Satu dari bentuk visual paling awal yang dicatat dalam sejarah adalah sinyal asap.

Awal dari abad ke 20, radio pada pesawat digunakan untuk mengecek keakuratan koordinat pesawat untuk menentukan arah (yang juga digunakan untuk komunikasi, tentunya). Hal ini dicapai dengan mengkalkulasi jalur berdasarkan pada arah dari sinyal yang diterima dari suatu sumber transmisi, dikenal dengan *Direction Finding* (DF). Ketika informasi arah dari dua atau lebih penerima digabungkan, lokasi dari transmisi dapat ditentukan melauli perhitungan yang dikenal dengan *triangulation*.

Sejak 1978, 59 satelit GPS (*Global Positioning System*) telah berhasil ditempatkan di orbit bumi, meskipun tahun 2010, hanya 30 dari satelit itu yang masih berfungsi.(Anthony T. Holdener,2011)

Saat ini, ada banyak cara perangkat komputasi modern untuk mendapatkan informasi lokasi, dan tidak semua dari mereka mengandalkan satelit GPS. Berikut ini daftar dari cara lokasi didapatkan:

1. *Global Positioning System* (GPS). GPS dapat digunakan pada banyak perangkat modern yang mendukung GPS.

2. Alamat IP. Penggunaan alamat IP juga tersedia untuk banyak perangkat yang terhubung pada jaringan atau internet – desktop, printer, router, dan sebagainya.

3. GSM/CDMA *Cell IDs*. Digunakan pada *carrier* handphone di seluruh dunia.

4. WiFi dan Bluetooth MAC *Address*. WiFi dan penggunaan Bloetooth MAC *Address* tersedia pada perangkat yang menggunakan teknologi *wireless*.

5. Diinput oleh *user*. Tersedia pada setiap perangkat dan perangkat lunak yang meminta lokasi, seperti kode pos, dari *user* melalui metode input tertentu, biasanya textbox.

## Java

Pada 1991, sekelompok insinyur Sun dipimpin oleh Patrick Naughton dan James Gosling ingin merancang bahasa komputer untuk perangkat konsumer seperti *cable TV Box*. Karena perangkat tersebut tidak memiliki banyak memori, bahasa harus berukuran kecil dan mengandung kode yang liat. Juga karena manufaktur–manufaktur berbeda memilih *processor* yang berbeda pula, maka bahasa harus bebas dari manufaktur manapun. Proyek diberi nama kode ”Green”.

Mulanya bahasa yang diciptakan diberi nama ”Oak” oleh James Gosling, namun dikarenakan nama Oak sendiri merupakan nama bahasa pemrograman yang telah ada sebelumnya, kemudian SUN menggantinya dengan Java. Nama Java sendiri terinspirasi pada saat mereka sedang menikmati secangkir kopi yang kemudian dengan tidak sengaja salah satu dari mereka menyebutkan kata Java yang mengandung arti asal bijih kopi.

### Apa itu teknologi JAVA?

1. Sebuah Bahasa Pemrograman

Java adalah bahasa pemrograman yang berorientasi objek (OOP) dan dapat dijalankan pada berbagai *platform* sistem operasi.

2. Sebuah *Development Environment*

Sebagai sebuah peralatan pembangun, teknologi Java menyediakan banyak *tools*: *compiler, interpreter,* penyusun dokumentasi, paket kelas dan sebagainya.

3. Sebuah Aplikasi

Aplikasi dengan teknologi Java secara umum adalah aplikasi serba guna yang dapat dijalankan pada seluruh mesin yang memiliki *Java Runtime Environment* (JRE).

4. Sebuah *Deployment Environment*

Terdapat dua komponen utama, yang pertama adalah JRE, yang terdapat pada paket J2SDK, mengandung kelas-kelas untuk semua paket teknologi Java yang meliputi kelas dasar dari Java, komponen GUI dan sebagainya. Komponen yang lain terdapat pada *web browser*. Hampir seluruh *web browser* komersial menyediakan *interpreter* dan *runtime environment* dari teknologi Java.

### Sebagian Fitur dari Java

1. Java *Virtual Machine* (JVM)

JVM adalah sebuah mesin imajiner (maya) yang bekerja menyerupai aplikasi pada sebuah mesin nyata. JVM menyediakan spesifikasi *hardware* dan *platform* dimana kompilasi kode Java terjadi. Spesifikasi inilah yang membuat aplikasi berbasis Java menjadi bebas dari *platform* manapun karena proses kompilasi diselesaikan oleh JVM.

2. *Garbage Collection*

Program Java melakulan *garbage collection* yang berarti program tidak perlu menghapus sendiri objek-objek yang tidak digunakan lagi. Fasilitas ini mengurangi beban pengelolaan memori oleh pemrogram dan mengurangi atau mengeliminasi sumber kesalahan terbesar yang terdapat pada bahasa yang memungkinkan alokasi dinamis.

3. *Code Security*

*Code security* terimplementasi pada Java melalui penggunaan Java *Runtime Environment* (JRE). Java menggunakan model pengamanan tiga lapis untuk melindungi sistem dari *untrusted* Java *code*. Pertama, *class-loader* menamgani pemuatan kelas Java ke *runtime interpreter*. Kedua, *bytecode verifier* membaca *bytecode* sebelum dijalankan dan menjamin *bytecode* memenuhi aturan-aturan dasar bahasa Java. Terakhir, Manajemen keamanan menangani keamanan tingkat aplikasi dengan mengendalikan apakah program berhak mengakses sumber daya seperti sistem *file*, *port* jaringan, proses eksternal dan sistem *windowing*.

## Android

Android adalah sistem operasi yang berbasis Linux. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak.

Kendati ponsel Android pertama kali diluncurkan pada tahun 2008, namun perjalanan panjang Android sendiri jauh sebelum itu. Pada Oktober 2003, Android, Inc. didirikan di Palo Alto, California, Amerika Serikat oleh Andy Rubin, Rich Miner, Nick Sears dan Chris White. Kemudian pada Agustus 2005, Google mengakuisi Android. Google memberikan kekuasaan ekstra untuk Android. Dibentuklah *Open Heandset Alliance* dengan lebih dari 34 rekanan untuk mengembangkan perangkat bagi Android. Sistem *open source* dipilih agar dapat memberi kebebasan pada operator dan vendor (OEM – *Original Equipment Manufacturer*) agar tidak perlu mengunci sistem operasi.

Dari keseluruhan mekanisme yang terjadi, Android membuka suatu filosofi baru sistem telekomunikasi seluler masa kini yang sesungguhnya bermuara pada empat elemen. Elemen tersebut adalah pengguna sebagai target sasaran, operator sebagai penyedia fasilitas koneksi jaringan, vendor sebagai penyedia perangkat ponsel dan *developer* yang menjadi motor perkembangan konten. Tidak ada pengecualan, tak ada eksklusifitas. Semuanya serba terbuka, itulah hakekat Android sesungguhnya. (Endah Tri Utami, 2011)

### Versi Android

Berikut ini adalah versi-versi Android dari awal sampai terakhir penulisan tugas akhir ini. Uniknya, nama kode yang digunakan berurutan dari A-Z dan merupakan makanan penutup, kecuali untuk versi 1.0 dan 1.1.

1. Android 1.0 (23 September 2008)

2. Android 1.1 (9 Februari 2009)

3. Android 1.5 – Cupcake (30 April 2009)

4. Android 1.6 – Donut (15 September 2009)

5. Android 2.0/2.1 – Eclair (26 Oktober 2009 dan 12 Januari 2010)

6. Android 2.2 – Froyo (20 Mei 2010)

7. Android 2.3 – Gingerbread (6 Desember 2010)

8. Android 3.0/3.1/3.2 – Honeycomb (22 Februari 2011, 10 Mei 2011 dan 18 Juli 2011)

9. Android 4.0 – Ice Cream Sandwitch (diperkenalkan pada acara Google I/O Mei 2011 dan diluncurkan secara resmi pada saat rilis Galaxy Nexus dan Ice Cream Sandwitch tanggal 19 Oktober 2011).

### Perbedaan Antara Android *Original* dan OHD

Jika dilihat dari dukungan google terhadap sistem operasi Android yang diadopsi oleh ponsel atau urusan distribusi, Android terbagi menjadi dua macam.

1. Google *Mobile Services* (GMS)

Android yang dilengkapi dengan GMS didukung penuh oleh Google. Pada ponsel tersebut telah dilengkapi fasilitas-fasilitas yang terhubung dengan Google *Account*.

2. *Open Handset Development* (OHD)

Sistem operasi Android OHD atau biasa dikenal dengan Android “Indie” ini memiliki fungsi yang mendasar. Sistem operasi ini dikembangkan sendiri secara kreatif tanpa dukungan Google, dan tidak ada Android Market, distribusinya pun benar-benar bebas.

### Arsitektur Android

Google mengibaratkan Android sebagai sebuah tumpukan software. Setiap lapisan dari tumpukan ini menghimpun beberapa program yang mendukung fungsi-fungsi spesifik dari sistem operasi. Berikut ini susunan dari arsitektur atau lapisan dari komponen utama platform Android.

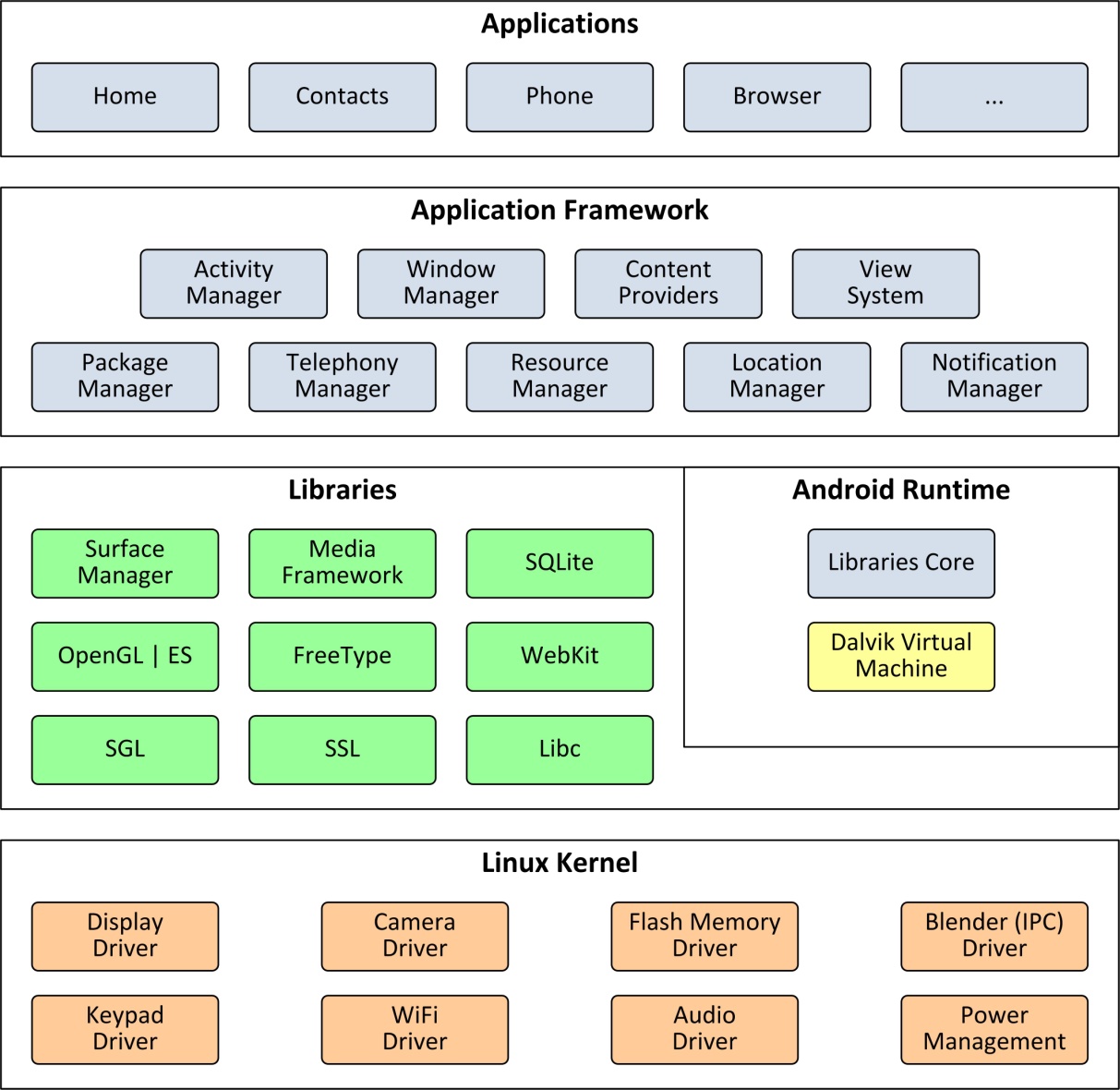
1. Linux Kernel, berperan sebagai *abstraction*/pemisah layer antara hardware dan keseluruhan software.

2. Android *Runtime*, berisi *Core Libraries* dan Dalvik *Virtual Machine*.

3. *Libraries*, menyertakan satu set *library-library* dalam bahasa C/C++ yang dapat digunakan oleh berbagai komponen yang ada pada sistem Android.

4. *Application Framework*, berperan dalam mencakup program dan mengatur fungsi-fungsi dasar *perangkat*.

5. *Applications*, pada lapisan ini terdapat aplikasi itu sendiri juga dapat menemukan fungsi-fungsi dasar seperti menelpon, mengirim pesan dan lainnya.



Gambar 2. Arsitektur Android

### Dasar-Dasar Aplikasi Android

Aplikasi Android ditulis dalam bahasa pemrograman Java. Android SDK (*Software Development Kit*) meng-*compile* kode program – bersama dengan data dan *file resource* – ke dalam Android *package*, sebuah file *archive* dengan akhiran .apk.

Sekali di-*install* pada perangkat, setiap aplikasi Android hidup dalam *security sandbox* tersendiri:

1. Sistem operasi Android adalah sistem Linux *multi-user* yang setiap aplikasi adalah *user* berbeda.

2. Secara *default*, sistem memberikan setiap aplikasi sebuah user ID Linux. Sistem memberikan ijin untuk semua file didalam aplikasi sehingga hanya user ID yang disertakan kepada aplikasi itu yang dapat mengaksesnya.

3. Setiap proses mempunyai mesin virtual (*virtual machine* – VM) tersendiri, sehingga kode aplikasi berjalan terisolasi dari aplikasi lainnya.

4. Secara *default*, setiap aplikasi berjalan di dalam proses Linux tersendiri. Android memulai proses ketika salah satu dari komponen aplikasi perlu dieksekusi, kemudian mematikan proses ketika tidak diperlukan lagi atau ketika sistem memerlukan memori untuk aplikasi lain.

Hal ini menciptakan lingkungan yang sangat aman yaitu aplikasi tidak dapat mengakses bagian dari sistem jika tidak diberikan ijin. Tetapi, terdapat jalan untuk dapat membagi data kepada aplikasi lain dan untuk sebuah aplikasi mengakses sistem service:

1. Dimungkinkan untuk mengatur dua aplikasi dapat berbagi user ID Linux yang sama, dalam kasus ini mereka dapat mengakses file satu sama lain. Mereka juga berjalan pada proses Linuk dan VM yang sama.

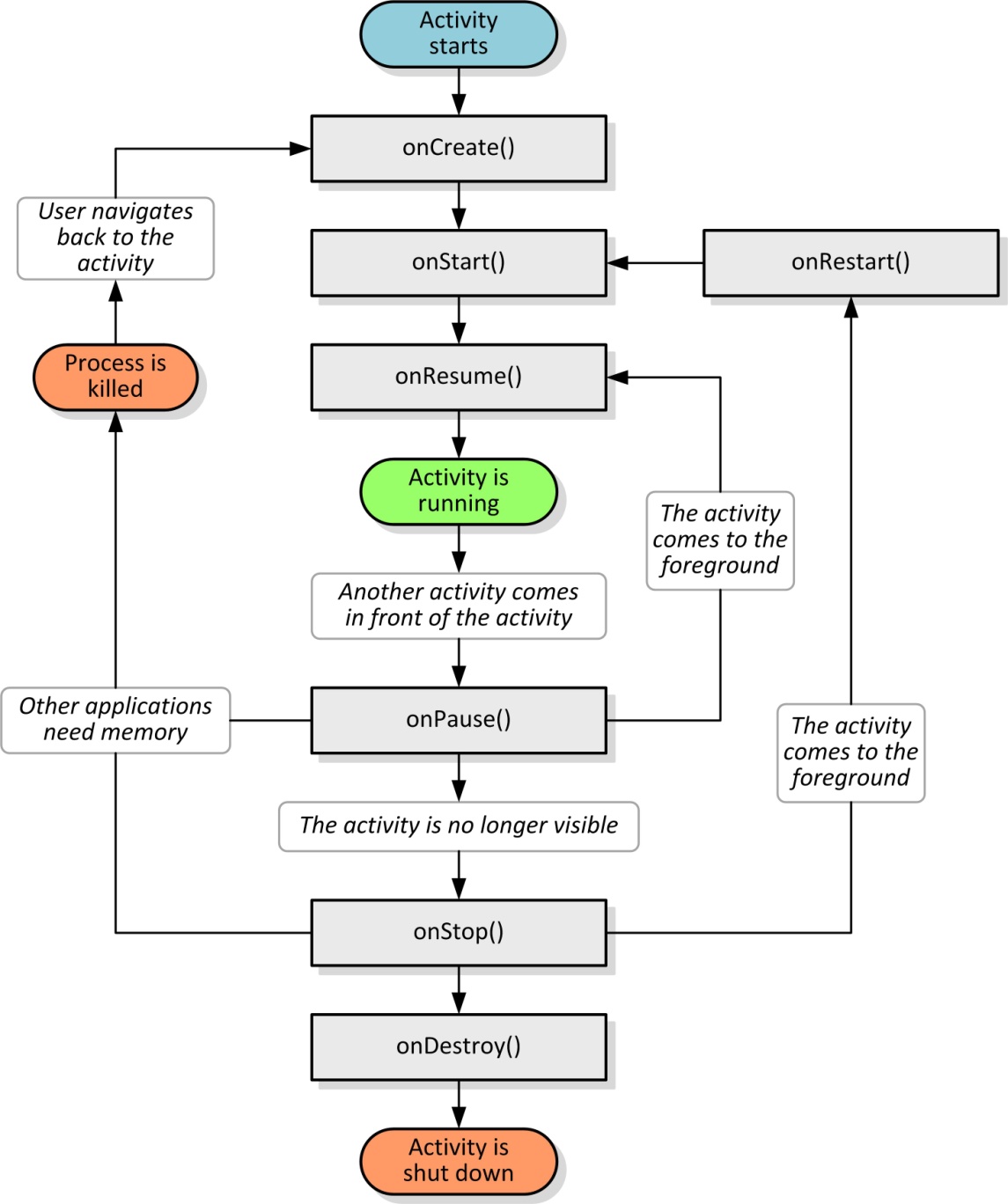
2. Aplikasi dapat meminta ijin untuk mengakses data perangkat seperti kontak, pesan SMS, SD *card*, kamera, Bluetooth, dan lainnya. Semua ijin ini harus diberikan oleh *user* pada waktu meng-*install* aplikasi.

### Komponen Aplikasi Android

Terdapat empat tipe komponen aplikasi. Setiap tipe melayani tujuan berbeda dan siklus hidup berbeda yang mendefinisikan bagaiman komponen diciptakan dan dimusnahkan.

#### Activity

*Activity* adalah komponen aplikasi yang menyediakan sebuah layar (*user interface*) sehingga user dapat berinteraksi untuk melakukan sesuatu, seperti memutar nomor telepon, mengambil foto, mengirim email, atau melihat peta. Biasanya, satu *activity* dalam aplikasi ditetapkan sebagai *activity* “utama”, yang ditunjukkan kepada *user* ketika menjalankan aplikasi pertama kalinya. Setiap *activity* kemudian dapat memulai *activity* lain. Setiap kali *activity* baru dimulai, *activity* sebelumnya dihentikan, tetapi sistem mengamankan *activity* ini dalam sebuah tumpukan (*backstack*). Ketika *activity* baru dimulai, *activity* ini didorong ke dalam *back stack* dan mengambil fokus. *Back stack* berpegang pada mekanisme antrian dasar “*last in, first out*”. Jadi, ketika *user* sudah selesai dengan *activity* saat ini dan menekan tombol kembali, *activity* ini kemudian dikeluarkan dari stack (dan dimusnahkan) dan *activity* sebelumnyapun dimulai kembali.



Gambar 2. Siklus Hidup *Activity*

Ketika *activity* dihentikan karena *activity* baru dimulai, akan terdapat pemberitahuan perubahan keadaan melalui *method callback* dari siklus hidup *activity*. Terdapat beberapa *method callback* yang dapat diterima, dan setiap *callback* menyediakan kesempatan untuk melakukan pekerjaan tertentu.

#### *Service*

*Service* adalah komponen aplikasi yang dapat melakukan operasi jangka panjang pada latar belakang dan tidak mempunyai *user interface*. Komponen aplikasi lain dapat memulai sebuah *service* dan akan tetap berjalan pada latar belakang bahkan saat *user* berpindah ke aplikasi lain. Contohnya, *service* mungkin dapat menangani transaksi jaringan, memainkan musik, melakukan operasi file I/O, berinteraksi dengan *content provider*, semuanya dari latar belakang. Sistem Android akan menghentikan *service* hanya ketika kekurangan sumber daya, itupun akan dimulai ulang segera setelah sumber daya tersedia kembali.

Perlu diperhatikan bahwa *service* berjalan pada *thread* utama, *service* tidak membuat *thread* sendiri dan tidak berjalan pada proses berbeda. Ini berarti, jika *service* akan melakukan pekerjaan intensif atau operasi memblok (seperti MP3 *playback* atau *networking*) harus dibuat *thread* baru pada *service* untuk melakukan pekerjaan itu, sehingga dapat meminimalkan resiko *error* aplikasi tidak merespon (*error* ANR – *Application Not Responding*).

#### *Content Provider*

*Content provider* mengelola data aplikasi yang dapat dibagi. Melalui *content provider*, aplikasi lain dapat mengambil atau bahkan mengubah data itu (jika diijinkan). Hanya dengan cara ini saja aplikasi dapat membagi data dengan aplikasi lain; tidak terdapat area penyimpanan biasa yang semua aplikasi dapat mengaksesnya. Contohnya, sistem Android menyediakan *content provider* yang mengelola informasi kontak *user*. Aplikasi apapun yang mempunyai ijin dapat mengambil dan menulis informasi tersebut.

#### *Broadcast Receiver*

*Broadcast receiver* adalah komponen yang merespon pada sistem siaran pemberitahuan. Banyak *broadcast* berasal dari sistem – contohnya, sebuah *broadcast* memberitahukan bahwa layar telah dimatikan, baterai hampir habis, gambar telah diambil. Aplikasi juga dapat menginisiasasikan *broadcast* – contohnya, membiarkan aplikasi lain tahu bahwa beberapa data telah diunduh dan tersedia untuk digunakan.

Meskipun *broadcast receiver* tidak menampilkan *user interface*, mereka dapat membuat notifikasi *status bar* untuk memberitahukan *user*. Lebih umum, *broadcast receiver* hanya sebuah “*gateway*” untuk komponen lain dan dimaksudkan untuk melakukan sejumlah pekerjaan yang sangat minimal. Misalnya untuk menginisiasikan sebuah *service* agar melakukan pekerjaan berdasarkan pada *event*.

## Basis Data

Basis data merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan di simpanan luar komputer dan digunakan perangkat lunak tertentu untuk mengolah/ memanipulasinya. Prinsip utama dari sistem basis data ini adalah pengaturan data atau *file* dengan tujuan untuk memudahkan dan mempercepat dalam pengambilan kembali data atau arsip. Yang diunggulkan dalam basis data adalah pengaturan, pemeliharaan, pengelompokkan atau pengorganisasian data yang akan disimpan sesuai fungsi atau jenisnya. Pembagian ini dapat berbentuk sejumlah *file* atau tabel terpisah atau dalam bentuk pendefinisian kolom-kolom data setiap *file* atau tabel.

Beberapa definisi basis data adalah sebagai berikut:

1. Sekumpulan data *store* (bisa dalam jumlah besar) yang tersimpan dalam *magnetic disc*, *optical disk*, dan media penyimpanan sekunder lainnya.
2. Sekumpulan program-program aplikasi umum yang sifatnya “*batch*” yang mengeksekusi dan memproses data secara umum (hapus, cari, *update*, dan lainnya).
3. Basis data terdiri dari data yang di-*share* bagi banyak *user* dan memungkinkan penggunaan data yang sama pada waktu bersamaan oleh banyak *user*.
4. Koleksi terpadu dari data-data yang saling berkaitan dari suatu *enterprise*. Misalkan basis data inventori akan terdiri dari data-data seperti data barang, jumlah stok, vendor dan data barang masuk dan barang keluar.

Konsep dasar basis data:

1. *Field*. *Field* merupakan implementasi dari suatu atribut data. *Field* merupakan unit terkecil dari data yang berarti (*meaningful data*) yang disimpan dalam suatu *file* atau basis data.
2. *Record*. *Field-field* tersebut diorganisasikan dalam *record-record*. *Record* merupakan koleksi dari *field-field* yang disusun dalam format yang telah ditentukan. Selama desain sistem, *record* akan diklasifikasikan sebagai *fixed-length record* atau *variable-length record*.
   1. *Fixed-length record*: tiap *instance record* mempunyai *field,* jumlah *field*, dan ukuran logik yang sama.
   2. *Variable-length record*: mengijinkan *record-record* yang berbeda-beda dalam *file* yang sama memiliki panjang yang berbeda-beda.
3. *File* dan Tabel*. File* merupakan kumpulan semua kejadian dari struktur *record-record* yang diberikan. Tabel merupakan ekivalen basis data relasional dari sebuah *file*.

## SQLite

SQLite merupakan sebuah sistem manajemen basis data relasional yang bersifat ACID-*compliant* dan memiliki ukuran pustaka kode yang relatif kecil, ditulis dalam bahasa C. SQLite merupakan proyek yang bersifat *public domain* yang dikerjakan oleh D. Richard Hipp.

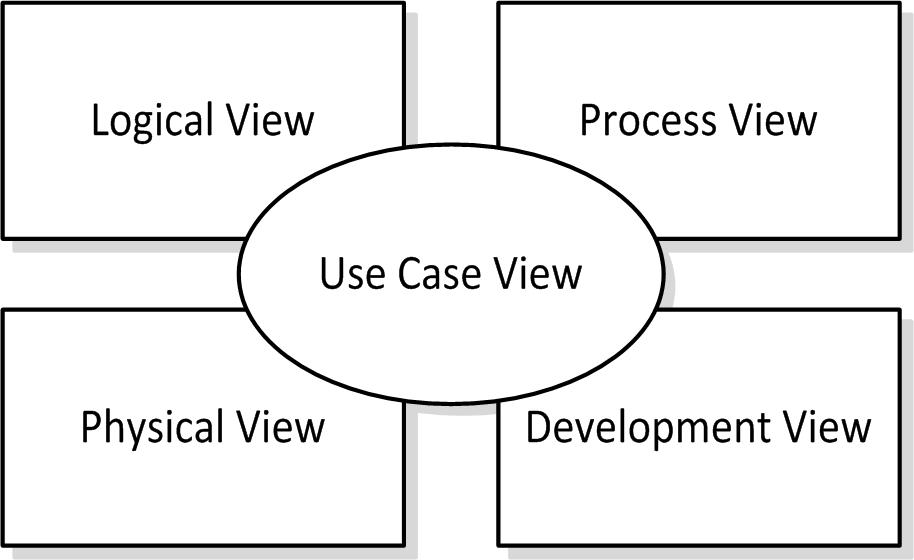
Tidak seperti pada paradigma *client-server* umumnya, Inti SQLite bukanlah sebuah sistem yang mandiri yang berkomunikasi dengan sebuah program, melainkan sebagai bagian integral dari sebuah program secara keseluruhan. Sehingga protokol komunikasi utama yang digunakan adalah melalui pemanggilan API secara langsung melalui bahasa pemrograman. Mekanisme seperti ini tentunya membawa keuntungan karena dapat mereduksi *overhead*, *latency times*, dan secara keseluruhan lebih sederhana. Seluruh elemen basis data (definisi data, tabel, indeks, dan data) disimpan sebagai sebuah *file*. Kesederhanaan dari sisi disain tersebut bisa diraih dengan cara mengunci keseluruhan file basis data pada saat sebuah transaksi dimulai.

Pustaka SQLite mengimplementasikan hampir seluruh elemen-elemen standar yang berlaku pada SQL-92, termasuk transaksi yang bersifat *atomic*, konsistensi basis data, isolasi, dan durabilitas (dalam bahasa inggris lebih sering disebut ACID), *trigger*, dan kueri-kueri yang kompleks. Tidak ada pengecekan tipe sehingga data bisa dientrikan dalam bentuk *string* untuk sebuah kolom bertipe *integer*.

Beberapa proses ataupun *thread* dapat berjalan secara bersamaan dan mengakses basis data yang sama tanpa mengalami masalah. Hal ini disebabkan karena akes baca data dilakukan secara paralel. Sementara itu akses tulis data hanya bisa dilakukan jika tidak ada proses tulis lain yang sedang dilakukan; jika tidak, proses tulis tersebut akan gagal dan mengembalikan kode kesalahan (atau bisa juga secara otomatis akan mencobanya kembali sampai sejumlah nilai waktu yang ditentukan habis).

## *Unified Modeling Language* (UML)

UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi.



Gambar 2. Kruchten 4+1 *Model View*

Ada beberapa cara untuk memecah model diagram UML, salah satunya adalah Kruchten 4+1 *model view* (Kim Hamilton dan Russell Miles, 2006):

1. *Logical view*. Menjelaskan deskripsi abstrak dari bagian suatu sistem. Jenis-jenis diagram UML ini termasuk *class*, *object*, *state machine*, dan *interaction diagrams*.
2. *Process view*. Menjelaskan proses dalam sistem. Hal ini sangat berguna ketika memvisualisasikan apa yang harus terjadi dalam sistem. *View* ini biasanya berisi *activity diagram*.
3. *Development view*. Menjelaskan bagaimana bagian-bagian sistem diatur dalam modul dan komponen. Hal ini sangat berguna untuk mengelola lapisan dalam arsitektur sistem. *View* ini biasanya berisi *package* dan *component diagrams*.
4. *Physical view*. Menjelaskan bagaimana desain sistem, seperti yang dijelaskan dalam tiga *view* sebelumnya, kemudian dibawa hidup sebagai satu set entitas dunia nyata. *View* ini biasanya berisi *deployment diagram*.
5. *Use case view*. Menjelaskan fungsi sistem yang dimodelkan dari perspektif dunia luar, diperlukan untuk menjelaskan sistem apa yang seharusnya dilakukan. *View* ini biasanya berisi *use case diagram*, *description*, dan *overview diagram*.

### *Use Case Diagram*

*Use case* merupakan titik mulai terbaik untuk setiap segi dari sistem berorientasi objek. Ia menggambarkan keperluan sistem dilihat perspektif *user*. Pada *use case*, yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”.

***Actor****.* Menggambarkan orang, sistem atau external entitas/stakeholder yang menyediakan atau menerima informasi dari sistem. *Actor* biasanya menggunakan kata benda.

***Use Case***. Diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan *actor*. Use case biasanya menggunakan kata kerja.

***Communication Line***. Mengkoneksikan *actor* dan *use case* untuk memperlihatkan bagaimana *actor* terlibat di dalam *use case*.

***System Boundaries***. Untuk menggambarkan jangkauan sistem. Digambarkan dengan kotak disekitar use case, dan *actor* dibiarkan diluar kotak.

***«include» Relationship***. Pemanggilan *use case* oleh *use case* lain, misalnya pemanggilan sebuah fungsi program.

«include»

***Generalization***. Disebut juga *inheritance*/pewarisan di-gunakan ketika ingin memperlihatkan satu *use case* adalah tipe spesial dari *use case* lain.

***«extend» Relationship***. Programmer Java mungkin akan mengira *«extend»* mirip dengan *inheritance* antar-kelas*.* Desain UML mengambil pandangan berbeda, *«extend»* digunakan untuk memperlihatkan bahwa *use case* dapat menggunakan kembali perilaku *use case* lain secara lengkap. Mirip seperti *«include»*, tetapi hal ini secara *optional*.

«extend»

### *Activity Diagram*

*Activity diagram* memperlihatkan apa yang harus dilakukan sistem. *Activity diagram m*enggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses. Struktur diagram ini mirip *flowchart* atau *Data Flow Diagram* (DFD) pada perancangan terstruktur. Perbedaan utamanya adalah *flowchart* dibuat untuk menggambarkan alur kerja dari sebuah sistem, sedangkan *activity diagram* dibuat untuk menggambarkan aktivitas dari *actor*.

***Initial node*** dan ***final node***. Merupakan awal dan akhir dari *activity*.

***Action***. Sebuah langkah tunggal didalam *activity* dimana terjadi pemrosesan atau manipulasi data. Dapat berupa kalkulasi atau sebuah tugas.

***Edge/path***. Aliran dari *activity*.

***Decision***. Percabangan, dianalogikan sebagai statemen if-else pada kode. Setiap cabang terdapat kondisi ditulis dalam kurung kotak.

***Fork*** dan ***Join***. *Fork* mengambarkan aksi parallel didalam *activity diagram*. *Fork* membagi aliran saat ini menjadi lebih dari satu aliran secara bersamaan. Sedangkan *join* merupakan kebalikan dari *fork*, *join* mensinkronkan kembali aliran *activity* itu menjadi aliran tunggal.

***Time Event***. Untuk memodelkan masa menunggu, seperti menunggu tiga hari setelah pengapalan order untuk mengirim tagihan. *Time event* tanpa aliran datang, diaktifkan secara periodik.

***Sub-Activity Indicator***. Mengindikasikan *activity* diurai-kan oleh *activity diagram* lebih detail. Ini mirip dengan pemanggilan prosedur program.

***Objects***. Menggambarkan sebuah objek yang terdapat pada titik khusus pada *activity*.

***Send Signal*** dan ***Receive Signal***. Pada *activity diagram*, sinyal menggambarkan interaksi dengan partisipan luar. *Send signal* yaitu sinyal yang dikirim ke partisipan luar. *Receive signal* memiliki efek untuk membangunkan aksi pada *activity diagram*. Penerima sinyal tahu bagaimana menangani sinyal tetapi tidak tahu pasti kapan.

***Partition (swimline)***. Mengelompokkan *activity* dalam sebuah urutan yang sama.

Partition1

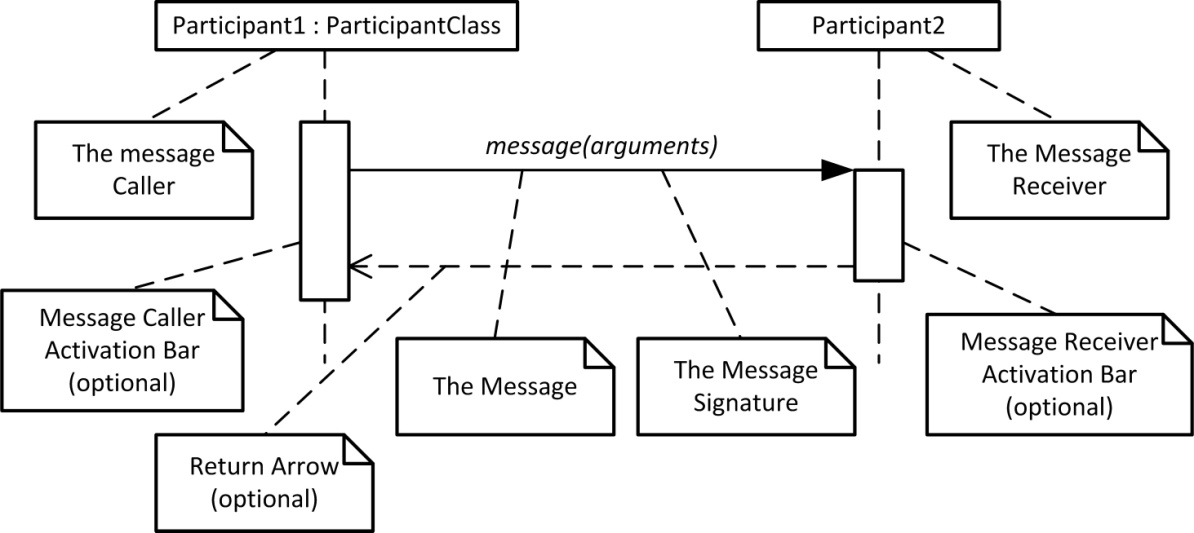
Partition2

### *Sequence Diagram*

Merupakan suatu diagram interaksi yang memodelkan suatu skenario tunggal yang dijalankan pada sistem. *Sequence diagram* digunakan untuk memperlihatkan interaksi antar-objek dalam perintah yang berurut. Tujuan utamanya adalah untuk mendefinisikan urutan kejadian yang dapat menghasilkan output yang diinginkan.

Elemen-elemen dalam *sequence diagram*:

1. Partisipan, merupakan obyek atau entitas yang bertindak dalam sequence diagram.
2. *Message*, merupakan komunikasi antar obyek partisipan.
3. Garis, terdapat 2 tipe garis yaitu vertikal dan horisontal: Vertikal merupakan waktu (maju berdasarkan waktu), Horisontal merupakan obyek mana yang beraksi.



Gambar 2. Elemen didalam *Sequence Diagram*

Jenis dari tanda panah pada sebuah pesan juga penting untuk menentukan tipe pesan apa yang akan dilewatkan. Contohnya, pemanggil pesan mungkin ingin untuk mengunggu pesan sebelum membawanya bekerja pada pesan sinkron. Atau mungkin hanya ingin mengirim pesan tanpa menunggu kembalian sebagai bentuk “*fire and forget*” pesan tidak sinkron.

***Synchronous Message***. Pesan yang dikirim oleh 1 obyek ke obyek lain dan obyek pertama menunggu sampai hasil aksi selesai.

***Asynchronous Message***. Pesan yang dikirim oleh 1 obyek ke obyek lain dan obyek pertama tidak menunggu sampai hasil aksi selesai.

***Return Message***. Menunjukkan nilai kembali dari obyek ke obyek yang megirim pesan.

***Participant Creation Message***. Menunjukkan pesan yang menyebabkan terjadinya pembentukan instansi obyek.

«create»

P1:Class

***Participant Destruction Message***. Menunjukkan penghapusan instansi objek. Pada implementasi bahasa pemrograman, seperti Java, tidak dijumpai metoda *destroy* secara ekplisit, jadi hal ini tidak memiliki arti untuk diperlihatkan dalam *sequence diagram*.

«destroy»

UML 2.0 berisi sejumlah besar tipe fragmen yang dapat diterapkan pada *sequence diagram* untuk membuatnya lebih kaya:

| **Tipe** | **Parameter** | **Keterangan** |
| --- | --- | --- |
| ref | tidak ada | Membantu untuk mengelola diagram dengan memecah, dan berpotensi penggunaan kembali, koleksi dari interaksi. Mirip seperti «include» pada *use case*. |
| assert | tidak ada | Mengindikasikan harus tepat terjadi sesuai dengan yang diindikasikan. Bekerja mirip dengan statemen assert pada java. |
| loop | waktu min,  waktu max,  [guard] | Perulangan dengan jumlah tertentu sampai kondisi *guard* bernilai *false*. Mirip seperti perulangan for(..) pada Java dan C# |
| break | tidak ada | Interaksi yang dilingkupi (umumnya fragmen *loop*) harus dikeluarkan. Mirip seperti statemen break pada Java dan C# |
| alt | [guard1] ...  [guard2] ...  [else] | Tergantung pada *guard* yang mana yang bernilai *true*. Mirip seperti statemen if(..) else pada kode. |
| opt | [guard] | Akan dieksekusi hanya jika bernilai *true*. Mirip seperti statemen if(..) tanpa else pada kode. Merupakan hubungan dari «extend» pada *use case*. |
| neg | tidak ada | Mengindikasikan fragmen selalu tidak dieksekusi. Berguna jika ingin mencoba menandai tidak dieksekusinya interaksi sampai yakin dapat dihapus. |
| par | tidak ada | Dapat dieksekusi secara paralel. |
| region | tidak ada | Merupakan bagian dari wilayah kritis. Wilayah kritis adalah area dimana *shared* partisipan di-*update*. Memiliki kemiripan dengan blok *synchronized* dan *object lock* pada Java. |

Tabel 2. Fragmen pada *Sequence Diagram*

### *Class Diagram*

*Class* adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok yaitu: nama (dan stereotype), atribut, dan metoda.

Baik atribut maupun metode memiliki empat katakteristik *visibility*:

1. *Public* (+), dapat dipanggil oleh siapa saja.
2. *Protected* (#), hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
3. *Package* (~), hanya dapat dipanggil oleh *class* yang memiliki *package* yang sama.
4. *Private* (-), tidak dapat dipanggil dari luar *class* yang bersangkutan.

UML menyediakan beberapa cara untuk menggambarkan hubungan antar-*class*. Setiap hubungan digambarkan dalam tipe koneksi yang berbeda diantara *class*:

***Dependency***. Relasi paling lemah antar-*class*. *Dependency* berarti satu *class* menggunakan atau mempunyai pengetahuan dari kelas lain. *Dependency* biasanya dibaca “menggunakan”.

***Association***. Lebih kuat dari *dependenciy* dan mengindikasikan satu *class* memegang teguh relasi dengan *class* lain sampai waktu tertentu. *Association* biasanya dibaca “mempunyai (*has*)”.

***Aggregation***. *Aggregation* adalah versi lebih kuat dari *association*. Tidak seperti *association*, *aggregation* cirinya dinyatakan secara tidak langsung sebagai kepemilikan. *Aggregation* biasanya dibaca “memiliki (*own*)”.

***Composition***. Menggambarkan relasi yang sangat kuat diantara *class*. *Composition* digunakan untuk menangkap semua bagian relasi. Relasi ini biasanya dibaca “bagian dari”.

***Generalization***. *Generalization* memberitahukan bahwa relasi adalah versi umum, atau kurang spesifik dari sumber *class* atau *interface*. Biasanya dibaca “adalah”. Contohnya “kucing adalah seekor binatang”

Terkadang atribut akan mewakili lebih dari satu objek. *Multiplicity* memungkinkan untuk merincikan bahwa atribut sebenarnya mewakili koleksi dari objek. Berikut ini beberapa contoh *multiplicity*:

1 Persis satu

0..\* Kosong atau lebih (banyak, tidak terbatas)

2..4 *range*, dari dua sampai empat

### Package Diagram

Sebagaimana program perangkat lunak berkembang dalam kompleksitas, akan dengan mudah program ini mengandung ratusan kelas. Salah satu cara untuk menentukan struktur adalah dengan mengorganisir kelas ke dalam kelompok logis terkait.

Pada UML, kelompok dari kelas dimodelkan menggunakan *package*. *Package diagram* seringkali digunakan untuk melihat dependensi antar *package*. *Package* dapat mengorganisir hampir semua elemen UML tidak hanya kelas. Contohnya *package* juga umumnya digunakan untuk mengelompokkan *use case*.

***Package***. Mekanisme tujuan umum untuk mengorganisir elemen model dan diagram ke group.

***Dependensi***. Tanda panah untuk menggambarkan ketergantungan antar *package*.

# ANALISA DAN PERANCANGAN SISTEM

## Deskripsi Skenario

Aplikasi mendapatkan data gempa bumi terbaru dari situs *web* USGS (United States *Geological Survey*) dan menyimpannya ke *database*. Untuk mendapatkan data tersebut, akan dilakukan pengecekan secara periodik. Periode pengecekan dapat diatur setiap menit, setiap 5 menit, setiap 15 menit, setiap setengah jam, setiap jam, setiap 3 jam, setiap 6 jam, setiap setengah hari atau setiap hari. Atau *user* juga bisa mengklik menu *refresh* untuk me­-*refresh* secara manual.

Situs USGS menyediakan tiga jenis *feed* yaitu RSS (*Really Simple Syndication*), Atom, dan CSV (*Comma-Separated Values*). Selain itu *feed* juga dibedakan berdasarkan magnitudo dan periode *update*-nya. Untuk meminimalkan *traffic* data, digunakan CSV dan juga dapat memilih *feed* yang sesuai dengan pengaturan periode dan minimal magnitudo yang ditentukan *user*. *Feed* CSV dapat dibedakan menjadi: *past hour* (eqs1hour-M1.txt, eqs1hour-M0.txt), *past day* (eqs1day-M2.5.txt, eqs1day-M1.txt, eqs1day-M0.txt), dan *past 7 days* (eqs7day-M7.txt, eqs7day-M5.txt, eqs7day-M2.5.txt).

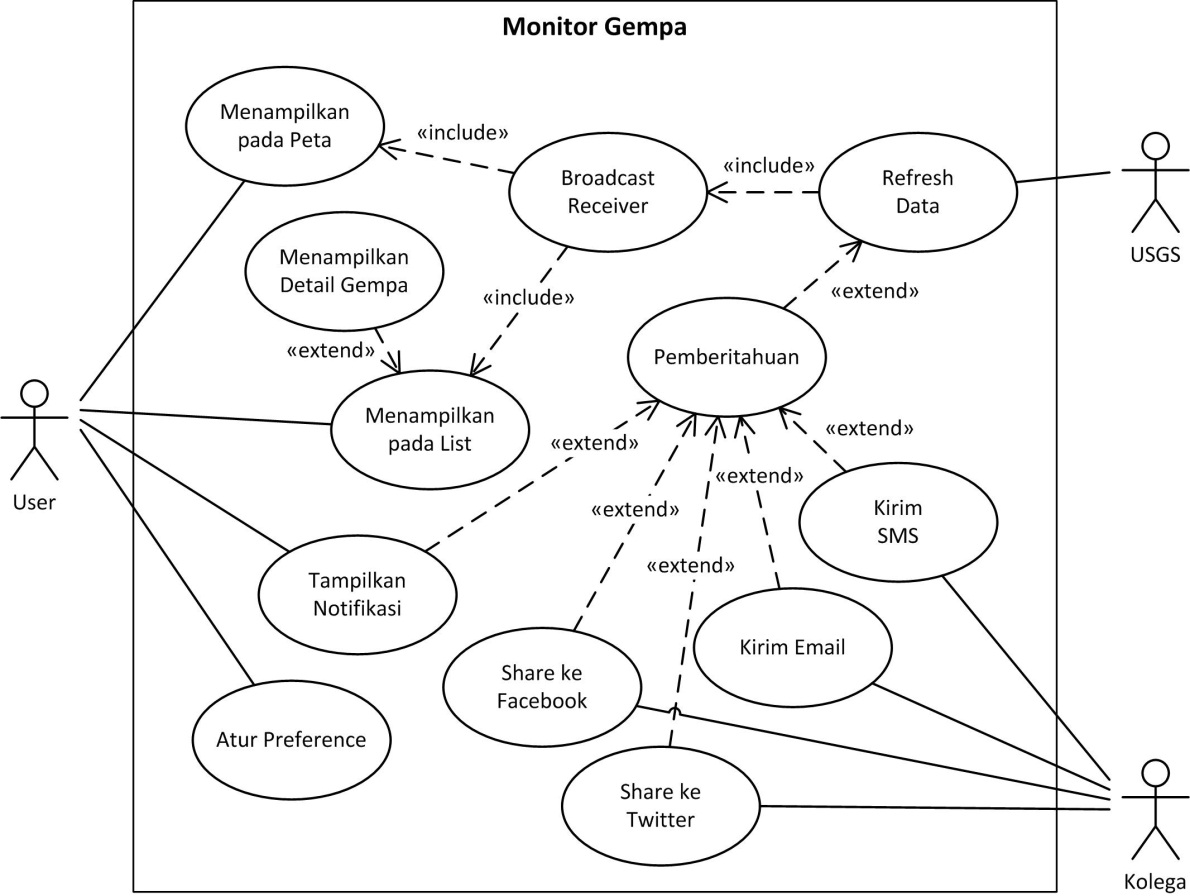
Agar data tidak terlalu banyak, harus terdapat *filter* minimal magnitudo dan data yang dikategorikan lama akan dihapus secara otomatis sehingga tidak memenuhi memori perangkat. Lamanya data disimpan dapat dipilih 1 hari, 2 hari, 3 hari, 5 hari, 1 minggu, 2 minggu, 1 bulan atau 6 bulan.

Menurut lokasinya, gempa dibedakan menjadi dua, gempa yang dekat dengan wilayah *user* dan gempa lokasi global. Lokasi bisa ditentukan melalui GPS atau menginput alamat secara manual. Data yang didapat akan ditampilkan pada *list* dan juga pada peta (Google Map). Disini juga ditampilkan berapa jarak lokasi *user* dengan pusat gempa.

Jika terdapat gempa yang cukup besar (magnitudonya dapat ditentukan dari magnitudo 0 sampai magnitudo 9), akan ada notifikasi kepada user. Notifikasi akan ditampilkan pada status bar notifikasi android. Notifikasi ini juga akan mengeluarkan suara, *flash*, dan bergetar sesuai dengan kekuatan gempa. Saat *user* mengklik notifikasi, akan ditampilkan detail gempa itu.

Aplikasi juga dapat mengirim pesan secara otomatis jika terdapat gempa yang cukup kuat ke nomor tertentu dan mengirim Email ke alamat email tertentu. Nomor telepon dan alamat email bisa didapat dari daftar kontak *user*. Isi pesan dan besarnya magnitudo gempa bisa ditentukan oleh *user*. Dan agar lebih sosial, aplikasi dapat memposting informasi gempa ke jejaring sosial Facebook dan Twitter. Selain informasi gempa, *user* juga dapat memposting pesan tambahan apa yang dipikirkannya.

## *Use Case Diagram*



Gambar 3. *Use Case Diagram*

Pada *use case* *refresh data*, data akan dicek secara periodik untuk mendapatkan gempa terbaru dan menyimpannya ke *database*. *Use case* pemberitahuan akan melakukan kalkulasi apakah data yang didapat sesuai dengan pengaturan untuk notifikasi *status bar* Android, mengirim SMS, mengirim Email, *posting* ke Facebook ataupun *posting* ke Twitter.

Jika notifikasi sesuai dengan pengaturan, maka pada *use case* tampilkan notifikasi, data akan diproses sehingga dapat ditampilkan di status bar notifikasi Android. Jika data sesuai dengan pengaturan mengirim SMS, pada *use case* mengirim SMS data ini akan diproses menjadi teks polos dan dikirim ke kolega yang sudah ditentukan. Jika sesuai dengan pengaturan mengirim email, pada *use case* mengirim email data ini akan dikirim ke alamat email kolega yang sudah ditentukan. Begitu juga pada *use case share* ke Facebook dan Twitter, jika sesuai pengaturan, setiap data akan *diposting* ke Facebook dan Twitter.

*Use case refresh data* juga mengirim *broadcast* setiap kali pengecekan gempa baru. *Use case* menampilkan pada *list* akan mendengarkan/menerima *broadcast* ini, jika merupakan data baru maka list akan diambil dari *database* dan ditampilkan ke daftar gempa. Saat daftar gempa diklik, maka akan ditampilkan informasi gempa yang lebih lengkap pada *use case* menampilkan detail gempa. *Use case* menampilkan pada peta juga mendengarkan *broadcast*, jika terdapat data baru maka titik-titik lokasi gempa akan digambar pada peta.

## *Use Case Description*

### *Refresh* Data

|  |  |
| --- | --- |
| **Nama *use case*** | *Refresh* Data |
| **Deskripsi** | Mendapatkan data gempa terbaru dari situs USGS |
| **Kondisi awal** | - |
| **Kondisi akhir** | Terdapat data baru dan disimpan ke *database* |
| **Aktor** | USGS |
| **Aliran utama** | 1. Hapus data lama 2. Mengambil data dari USGS berdasarkan magnitudo dan waktu terakhir mengambil data sebelumnya 3. Simpan ke *database* 4. extend::Pemberitahuan. Beritahukan *user*/kolega telah terjadi gempa (jika ada) 5. Include::Broadcast Receiver. Kirim *broadcast* terdapat data baru |
| **Aliran alternatif** | 2.1. Include::Broadcast Receiver. Kirim *broadcast* jaringan *error*  3.1. Include::Broadcast Receiver. Kirim *broadcast* tidak terdapat data baru |
| ***Exception*** | Tidak dapat menghubungi *web* USGS |

Tabel 3. *Use Case Description* – *Refresh* Data

### Pemberitahuan

|  |  |
| --- | --- |
| **Nama *use case*** | Pemberitahuan |
| **Deskripsi** | Memberitahukan *user* bahwa terdapat gempa baru |
| **Kondisi awal** | Terdapat gempa baru |
| **Kondisi akhir** | Notifikasi ditampilkan pada *status bar* sistem notifikasi Android  Pesan SMS dikirim  Pesan diposting ke Facebook |
| **Aktor** | *User*, Kolega |
| **Aliran utama** | 1. Menentukan jarak dari setiap gempa 2. Menentukan apakah jarak ini dekat dengan *user* atau tidak 3. Memilih magnitudo paling besar untuk notifikasi *user* (sistem notifikasi Android memiliki keterbatasan tempat, jadi hanya dipilih yang paling besar) 4. Menambahkan ke daftar gempa untuk dikirimkan SMS 5. Menambahkan ke daftar gempa untuk diposting ke Facebook 6. extend::Kirim Notifikasi. Kirimkan notifikasi ke sistem notifikasi Android 7. extend::Kirim Pesan. Kirimkan pesan SMS telah terjadi gempa 8. extend::Posting ke Facebook. Posting setiap gempa ke Facebook |
| **Aliran alternatif** | 7.1. Tidak ada notifikasi  8.1. Tidak ada pesan yang dikirim  9.1. Tidak ada yang diposting |
| ***Exception*** | - |

Tabel 3. *Use Case Description* – Pemberitahuan

### Menampilkan Notifikasi

|  |  |
| --- | --- |
| **Nama *use case*** | Menampilkan Notifikasi |
| **Deskripsi** | Menampilkan notifikasi di sistem notifikasi Android |
| **Kondisi awal** | Terdapat gempa baru yang melebihi minimal magnitudo notifikasi |
| **Kondisi akhir** | Notifikasi ditampilkan pada *status bar* sistem notifikasi Android |
| **Aktor** | *User* |
| **Aliran utama** | 1. Atur suara (jika pengaturan suara aktif) 2. Atur *flash* (jika pengaturan *flash* aktif) 3. Atur getaran (jika pengaturan getar aktif) 4. Kirim notifikasi 5. *User* mengklik *status bar* notifikasi Android 6. Tampilkan detail gempa bumi |
| **Aliran alternatif** | 6.1. Tidak terdapat data gempa pada extra, detail gempa tidak ditampilkan |
| ***Exception*** | - |

Tabel 3. *Use Case Description* – Kirim Notifikasi

### Kirim SMS

|  |  |
| --- | --- |
| **Nama *use case*** | Kirim SMS |
| **Deskripsi** | Mengirim pesan SMS ke kolega |
| **Kondisi awal** | Terdapat gempa baru yang melebihi minimal magnitudo mengirim pesan |
| **Kondisi akhir** | Pesan dikirim |
| **Aktor** | Kolega |
| **Aliran utama** | 1. Menentukan penerima pesan 2. Menghasilkan teks polos berdasarkan daftar gempa 3. Mengirim pesan |
| **Aliran alternatif** | - |
| ***Exception*** | Gagal mengirim pesan |

Tabel 3. *Use Case Description* – Kirim SMS

### Kirim Email

|  |  |
| --- | --- |
| **Nama *use case*** | Kirim Email |
| **Deskripsi** | Mengirim Email ke kolega |
| **Kondisi awal** | Terdapat gempa baru yang melebihi minimal magnitudo mengirim email |
| **Kondisi akhir** | Email dikirim |
| **Aktor** | Kolega |
| **Aliran utama** | 1. Memgatur *host*, *port*, *sport*, *username* dan *password* email. 2. Menentukan siapa yang mengirim email 3. Menentukan kemana email dikirim 4. Menentukan subjek email 5. Menentukan *body* email 6. Mengirim email |
| **Aliran alternatif** | - |
| ***Exception*** | Gagal mengirim email |

Tabel 3. *Use Case Description* – Kirim Email

### *Share* ke Facebook

|  |  |
| --- | --- |
| **Nama *use case*** | *Share* ke Facebook |
| **Deskripsi** | Memposting informasi gempa ke *wall* Facebook |
| **Kondisi awal** | Terdapat gempa baru yang melebihi minimal magnitudo share ke Facebook |
| **Kondisi akhir** | Informasi gempa diposting ke Facebook |
| **Aktor** | Kolega |
| **Aliran utama** | 1. Menghasilkan parameter posting untuk setiap gempa 2. Posting pesan |
| **Aliran alternatif** | 2.1. Belum login  2.2. Login |
| ***Exception*** | - |

Tabel 3. *Use Case Description* – *Share* ke Facebook

### *Share* ke Twitter

|  |  |
| --- | --- |
| **Nama *use case*** | *Share* ke Twitter |
| **Deskripsi** | Memposting informasi gempa ke Twitter |
| **Kondisi awal** | Terdapat gempa baru yang melebihi minimal magnitudo share ke Twitter |
| **Kondisi akhir** | Informasi gempa diposting ke Twitter |
| **Aktor** | Kolega |
| **Aliran utama** | 1. Login ke Twitter 2. Menentukan pesan yang akan diposting untuk setiap data gempa 3. Posting pesan |
| **Aliran alternatif** | - |
| ***Exception*** | - |

Tabel 3. *Use Case Description* – *Share* ke Twitter

### Menampilkan pada *List*

|  |  |
| --- | --- |
| **Nama *use case*** | Menampilkan pada *List* |
| **Deskripsi** | Menampilkan data gempa bumi pada *list* |
| **Kondisi awal** | - |
| **Kondisi akhir** | Data ditampilkan di *list* |
| **Aktor** | *User* |
| **Aliran utama** | 1. Menentukan lokasi *user*saat ini 2. Mengambil data dari *database* 3. Menampilkan data di *list* |
| **Aliran alternatif** | 3.1. Database kosong, sembunyikan *list* dan tampilkan pesan tidak ada data |
| ***Exception*** | - |

Tabel 3. *Use Case Description* – Menampilkan pada *List*

### Menampilkan Detail Gempa

|  |  |
| --- | --- |
| **Nama *use case*** | Menampilkan Detail Gempa |
| **Deskripsi** | Menampilkan detail gempa bumi |
| **Kondisi awal** | - |
| **Kondisi akhir** | Detail gempa bumi dan gambar *globe* ditampilkan |
| **Aktor** | *User*, USGS |
| **Aliran utama** | 1. Menentukan lokasi *user* saat ini 2. Mendapatkan data dari extra 3. Tampilkan detail gempa 4. *Load* gambar globe dari USGS |
| **Aliran alternatif** | 2.1. Tidak terdapat extra  3.1. tidak ada yang ditampilkan  4.1. Gagal mengambil gambar |
| ***Exception*** | - |

Tabel 3. *Use Case Description* – Menampilkan Detail Gempa

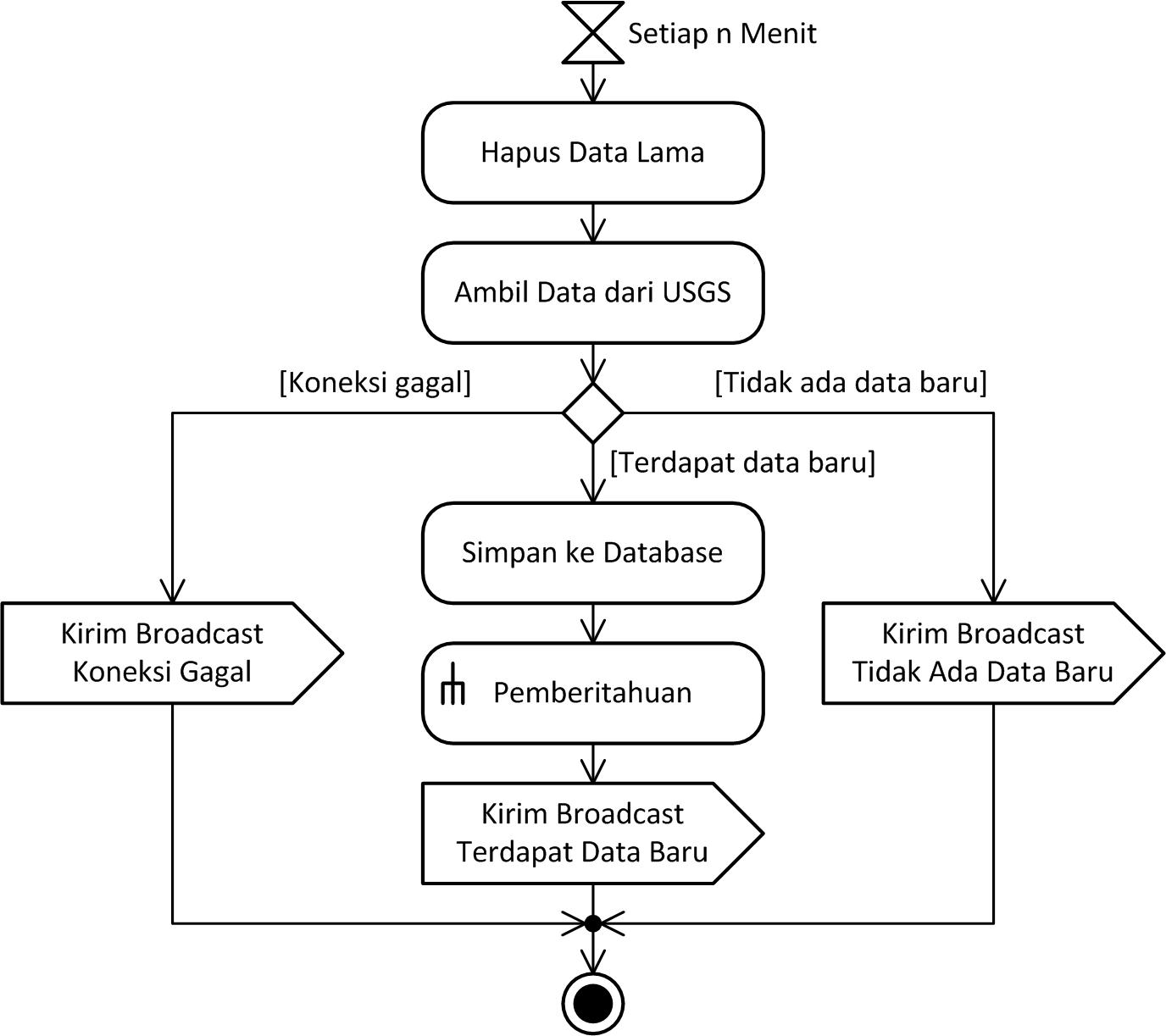
### Menampilkan pada Peta

|  |  |
| --- | --- |
| **Nama *use case*** | Menampilkan pada Peta |
| **Deskripsi** | Menampilkan data gempa bumi pada Google Map |
| **Kondisi awal** | - |
| **Kondisi akhir** | Data ditampilkan di peta berupa titik-titik lokasi gempa |
| **Aktor** | *User* |
| **Aliran utama** | 1. Menentukan lokasi *user* saat ini 2. Mengambil data dari *database* 3. Menggambar titik-titik lokasi gempa 4. Menggambar lokasi *user* saat ini |
| **Aliran alternatif** | 3.1. Tidak ada data, tidak ada yang digambar  4.1. Lokasi tidak diketahui, lokasi tidak digambar |
| ***Exception*** | - |

Tabel 3. *Use Case Description* – Menampilkan pada Peta

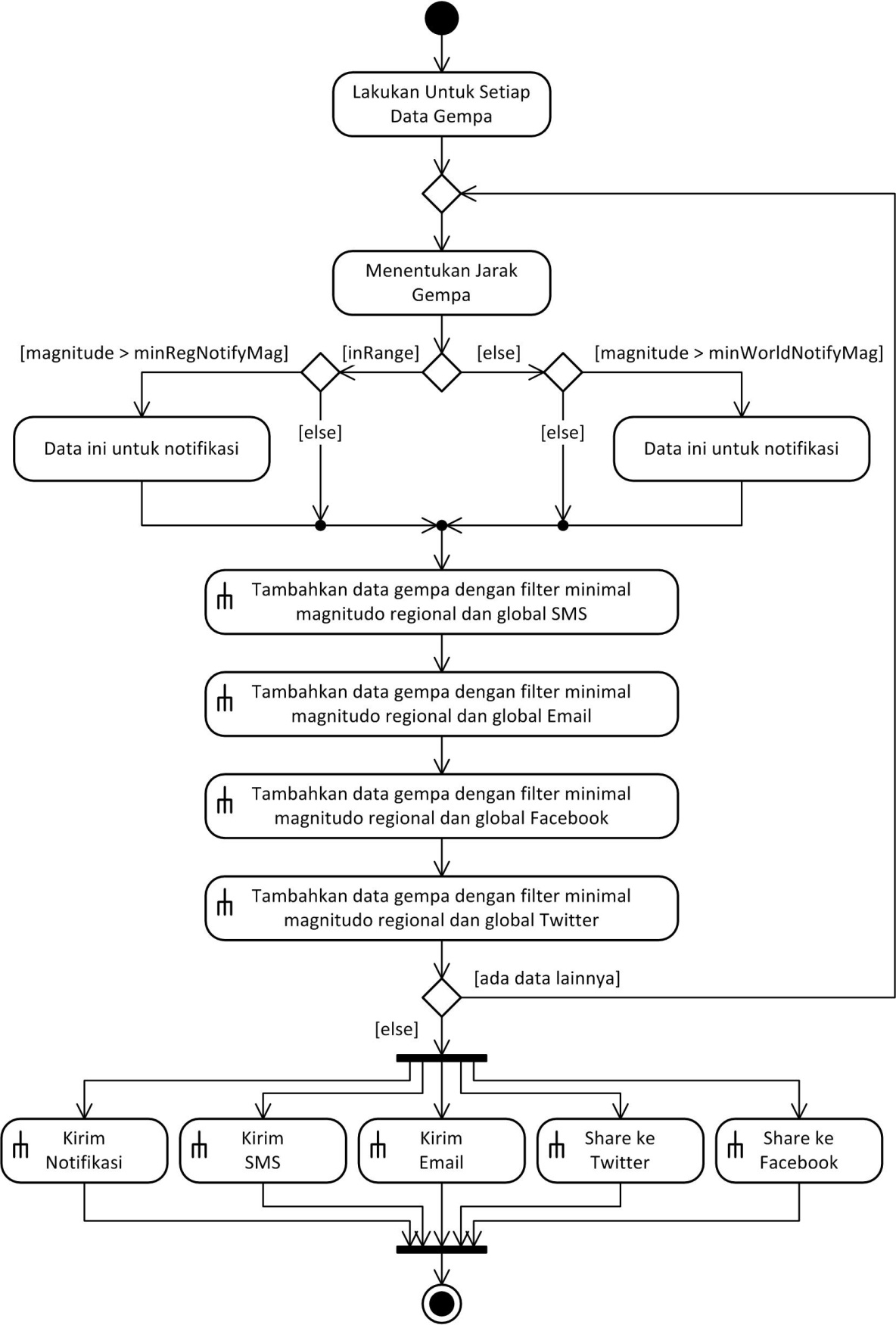
## *Activity Diagram*

### *Refresh* Data



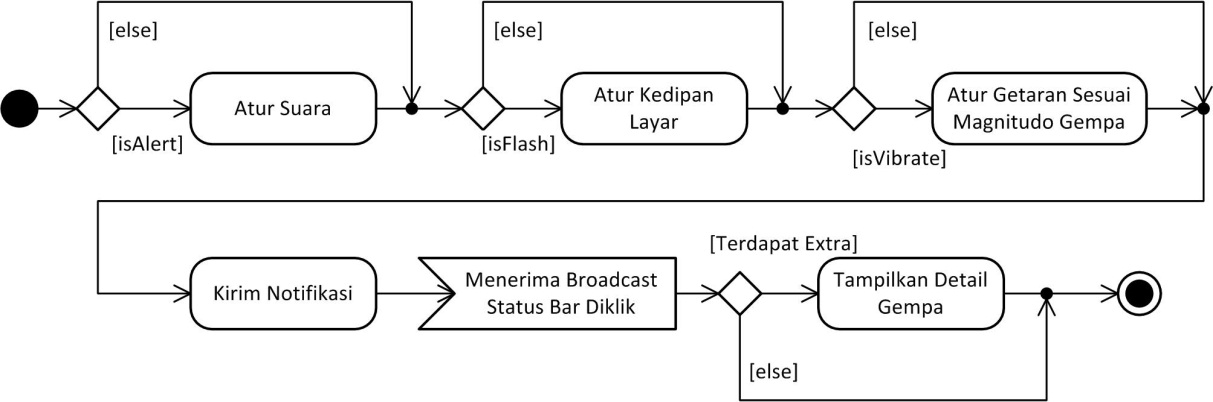
Gambar 3. *Activity Diagram* – *Refresh* Data

### Pemberitahuan



Gambar 3. *Activity Diagram* – Pemberitahuan

### Menampilkan Notifikasi



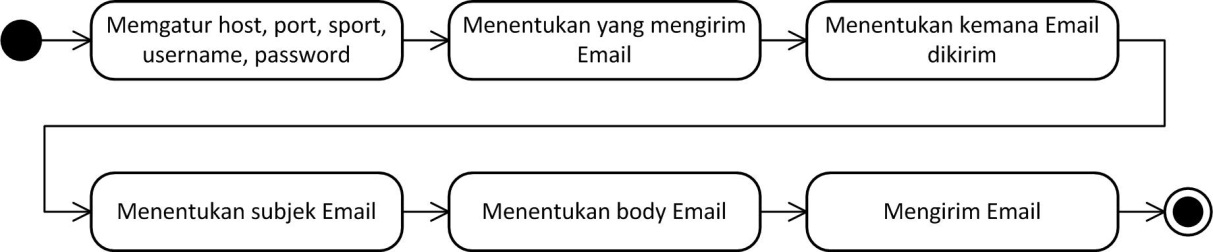
Gambar 3. *Activity Diagram* – Menampilkan Notifikasi

### Kirim SMS



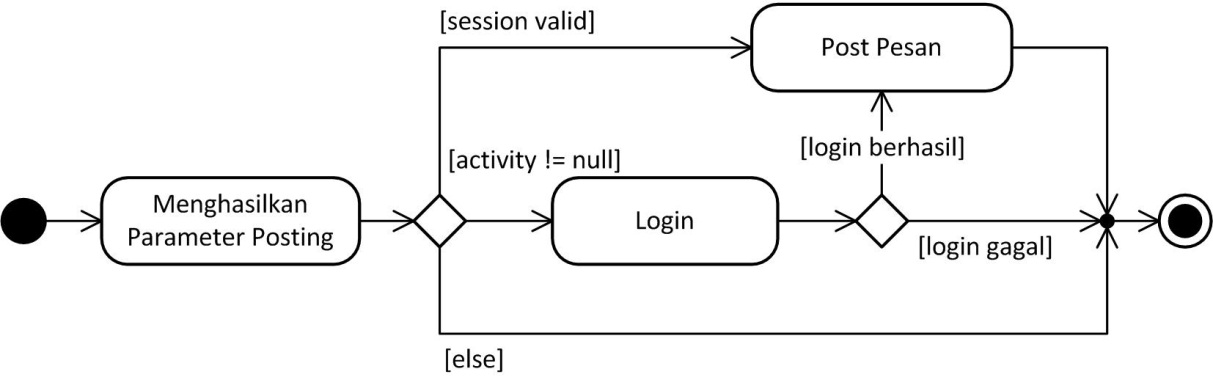
Gambar 3. *Activity Diagram* – Kirim SMS

### Kirim Email



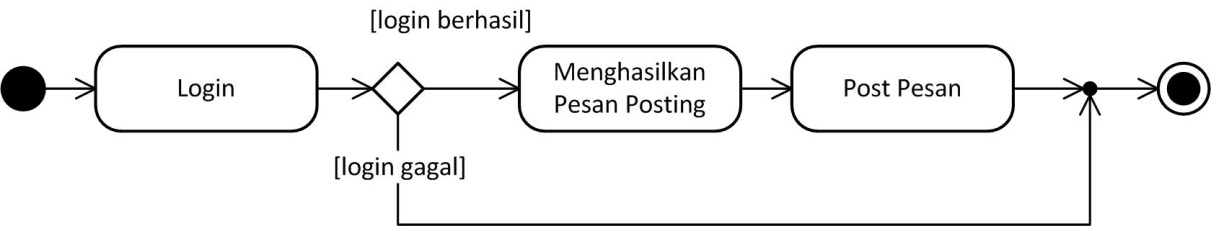
Gambar 3. *Activity Diagram* – Kirim Email

### *Share* ke Facebook



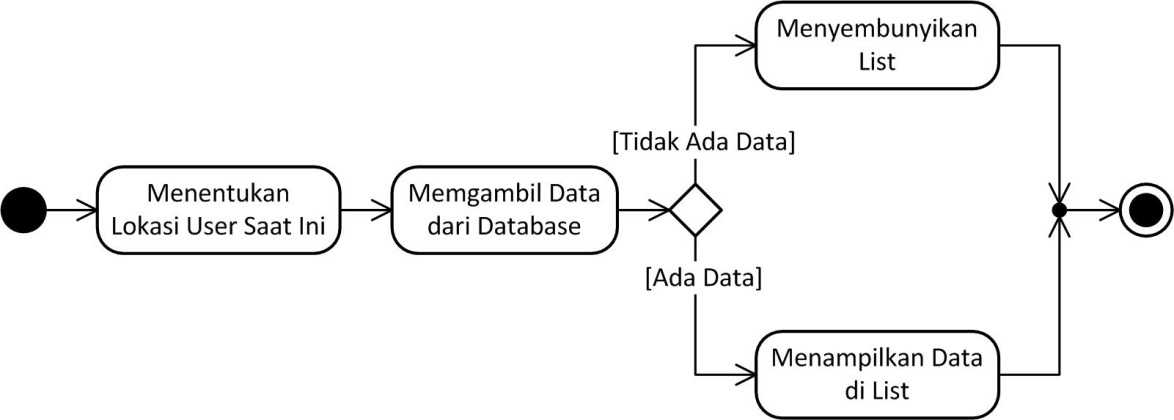
Gambar 3. *Activity Diagram* – *Share* ke Facebook

### *Share* ke Twitter



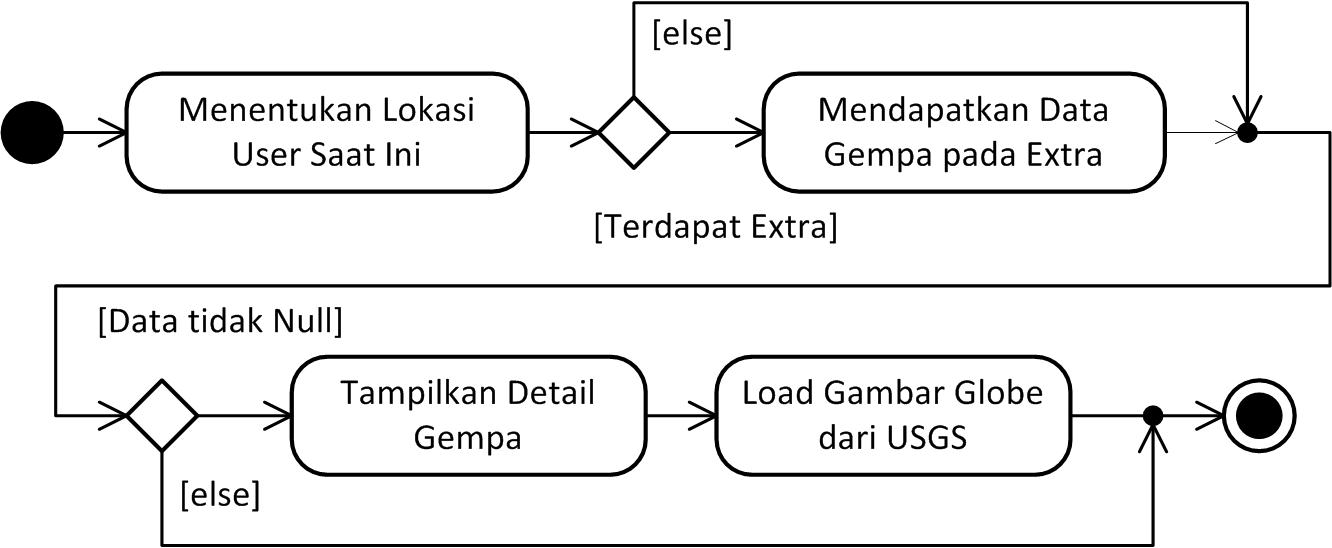
Gambar 3. *Activity Diagram* – *Share* ke Twitter

### Menampilkan pada *List*



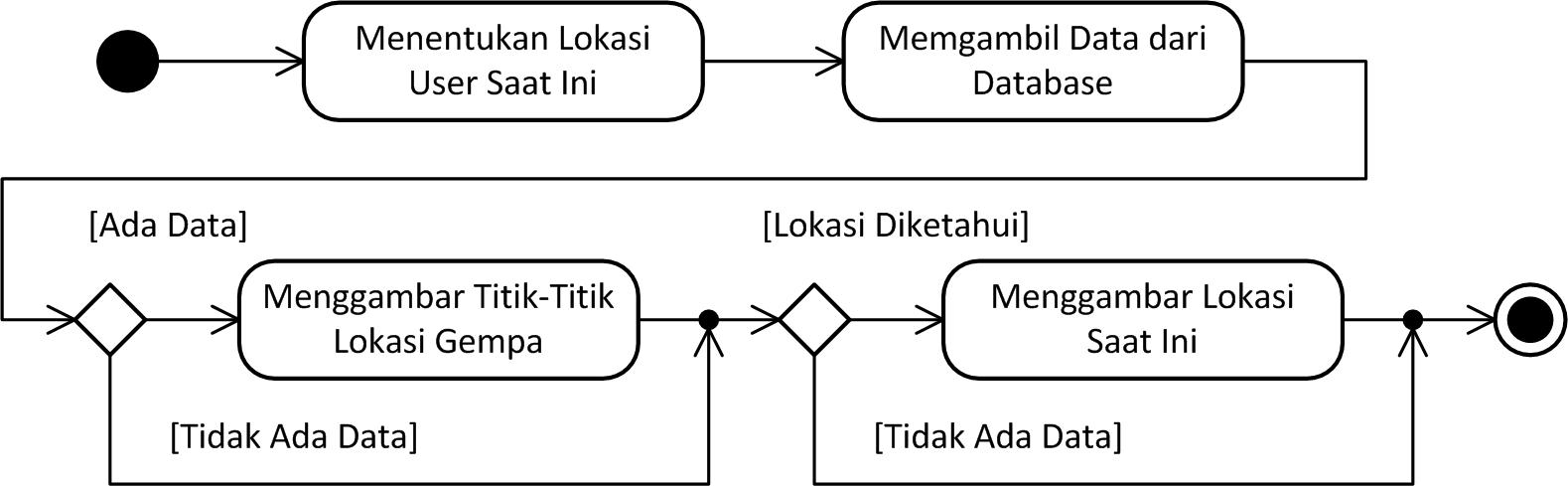
Gambar 3. *Activity Diagram* – Menampilkan pada *List*

### Menampilkan Detail Gempa



Gambar 3. *Activity Diagram* – Menampilkan Detail Gempa

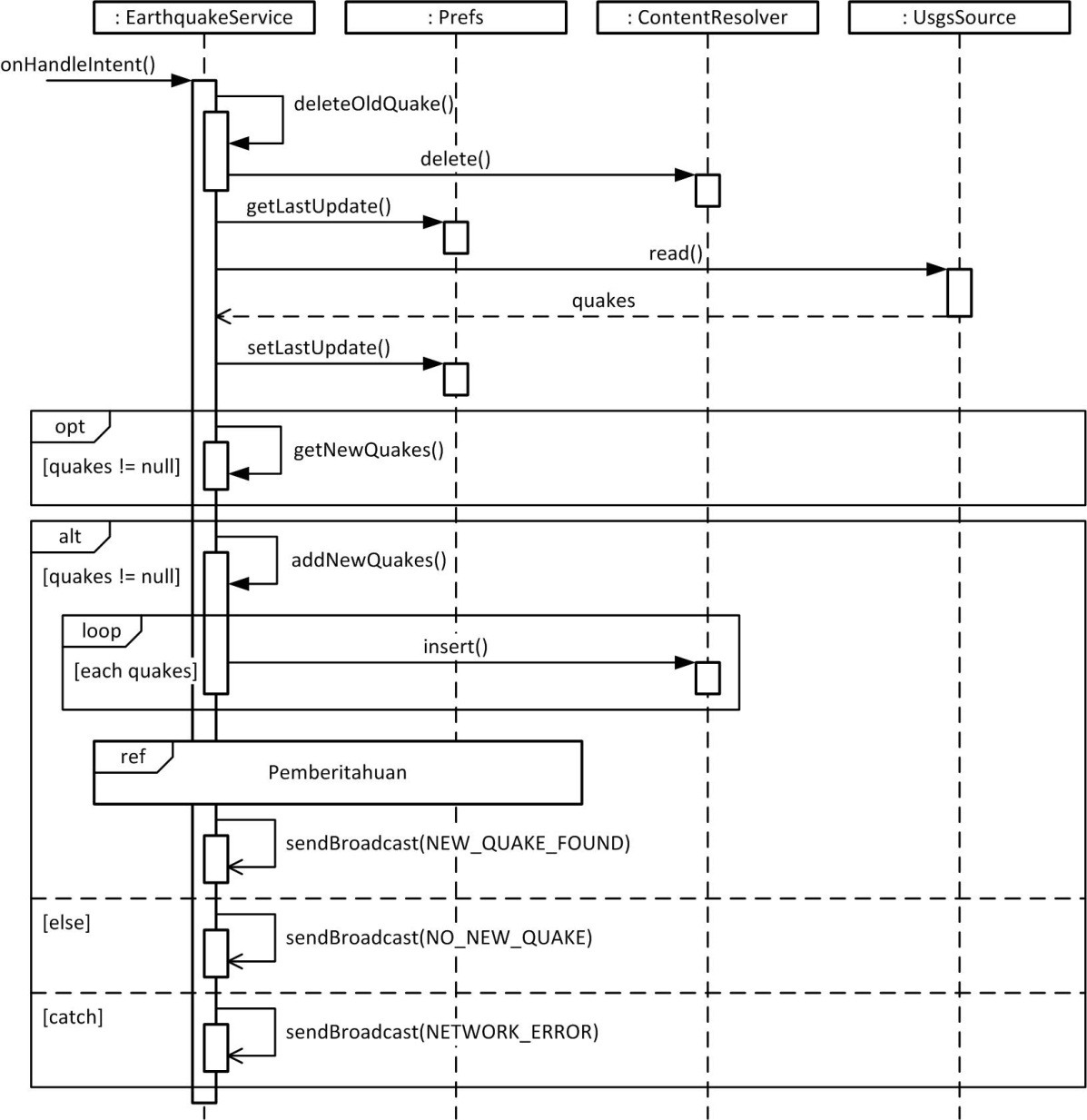
### Menampilkan pada Peta



Gambar 3. *Activity Diagram* – Menampilkan pada Peta

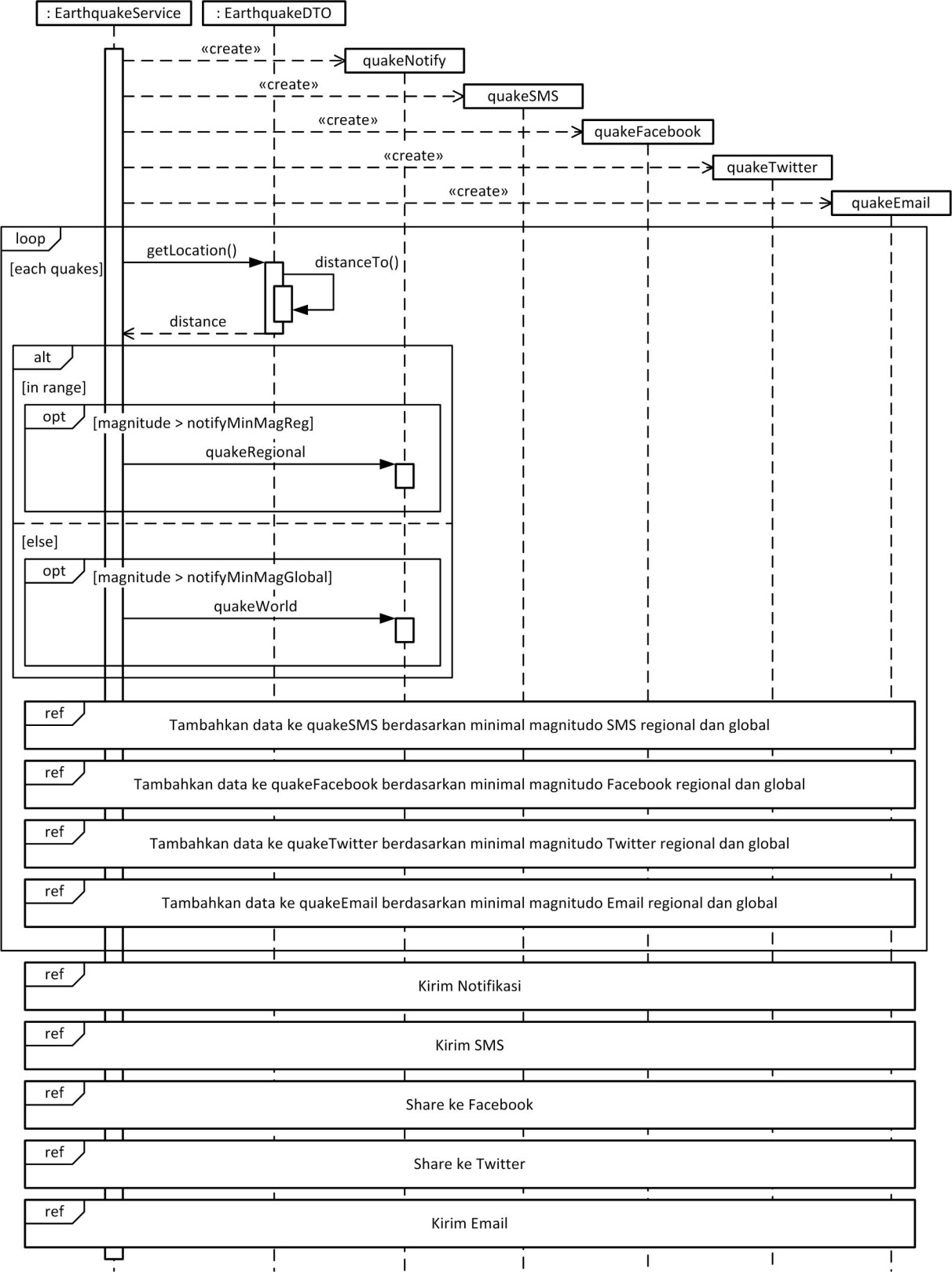
## *Sequence Diagram*

### *Refresh* Data



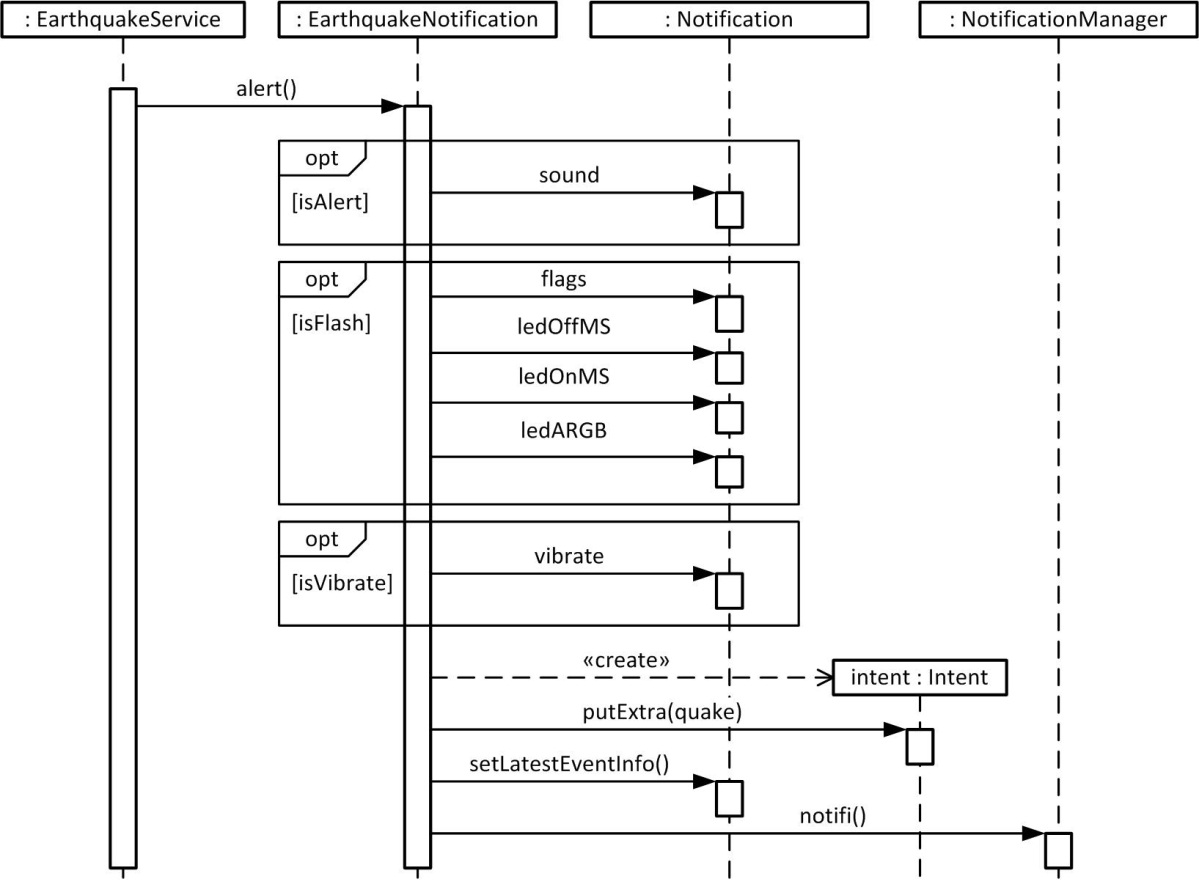
Gambar 3. *Sequence Diagram* – *Refresh* Data

### Pemberitahuan

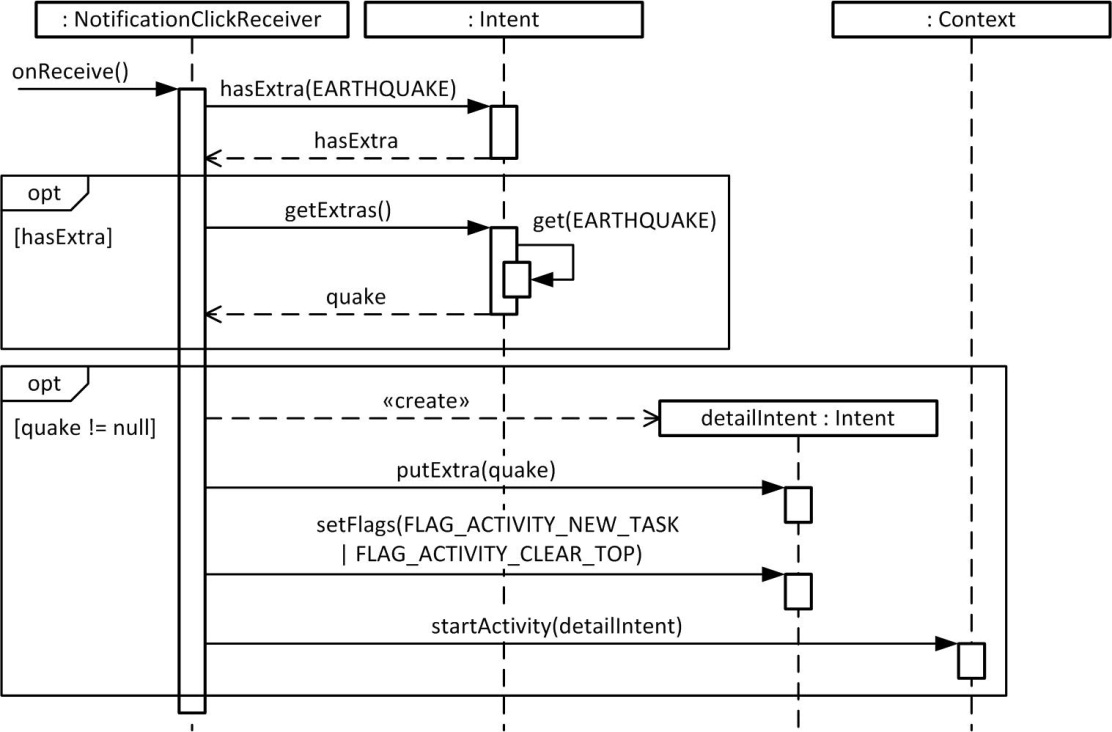


Gambar 3. *Sequence Diagram* – Pemberitahaun

### Menampilkan Notifikasi

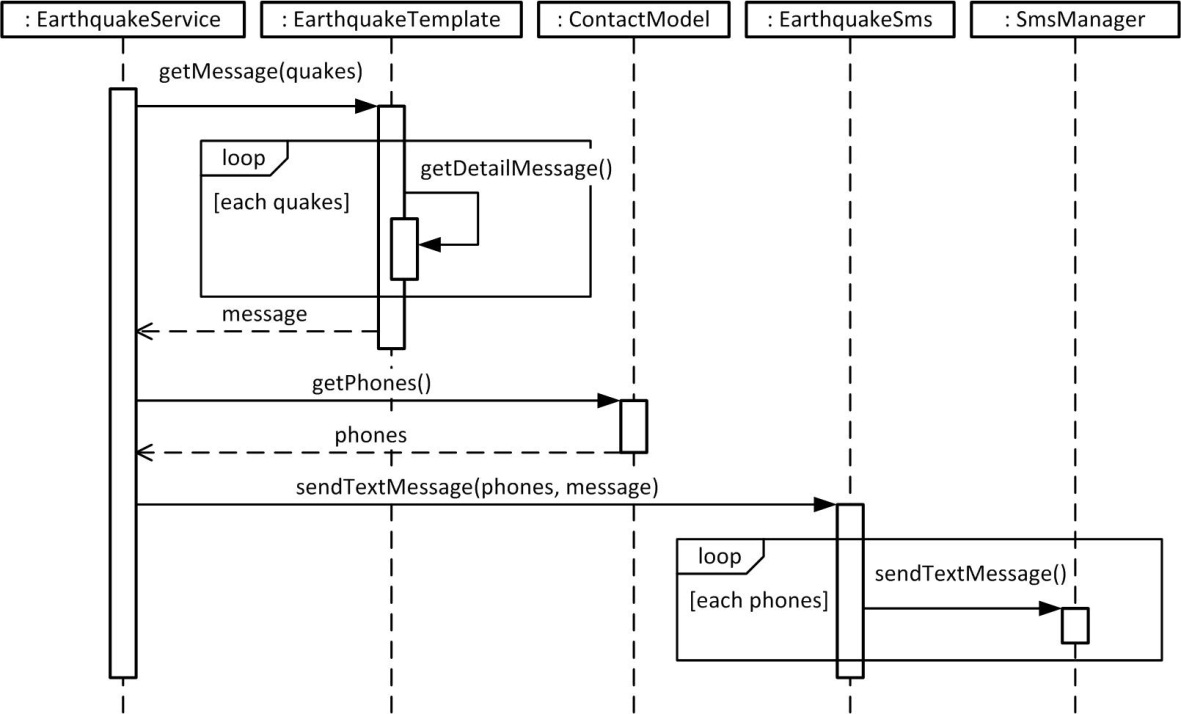


Gambar 3. *Sequence Diagram* – Menampilkan Notifikasi



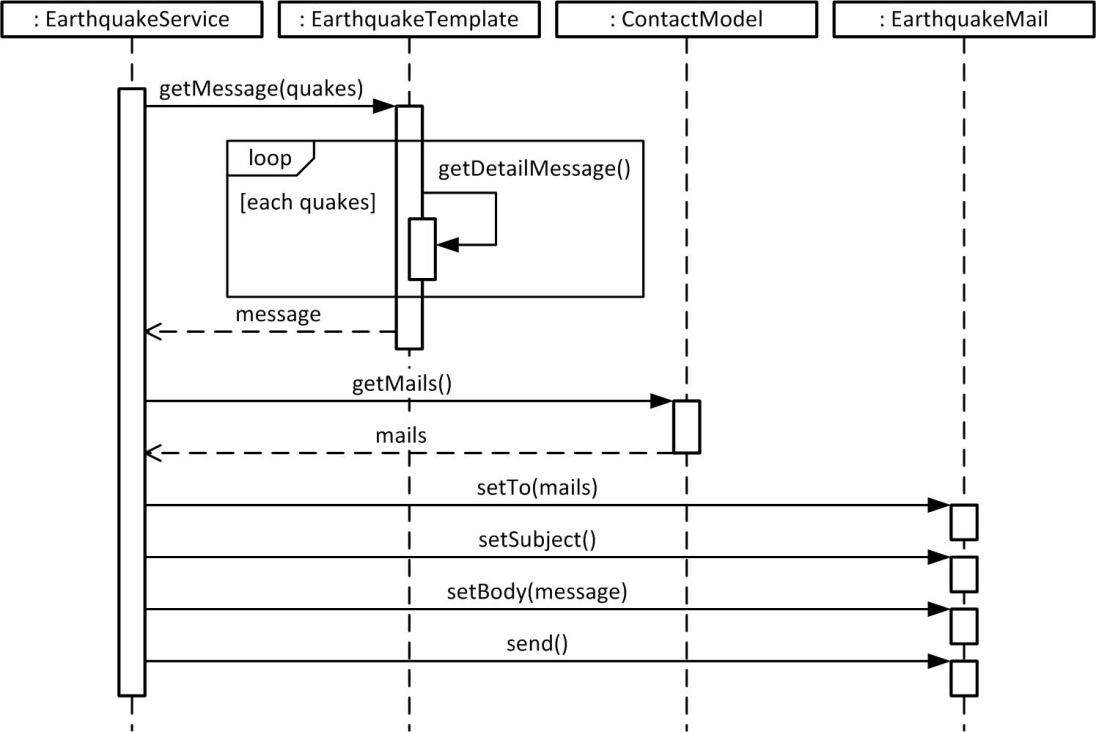
Gambar 3. *Sequence Diagram* – Notifikasi Diklik

### Kirim SMS



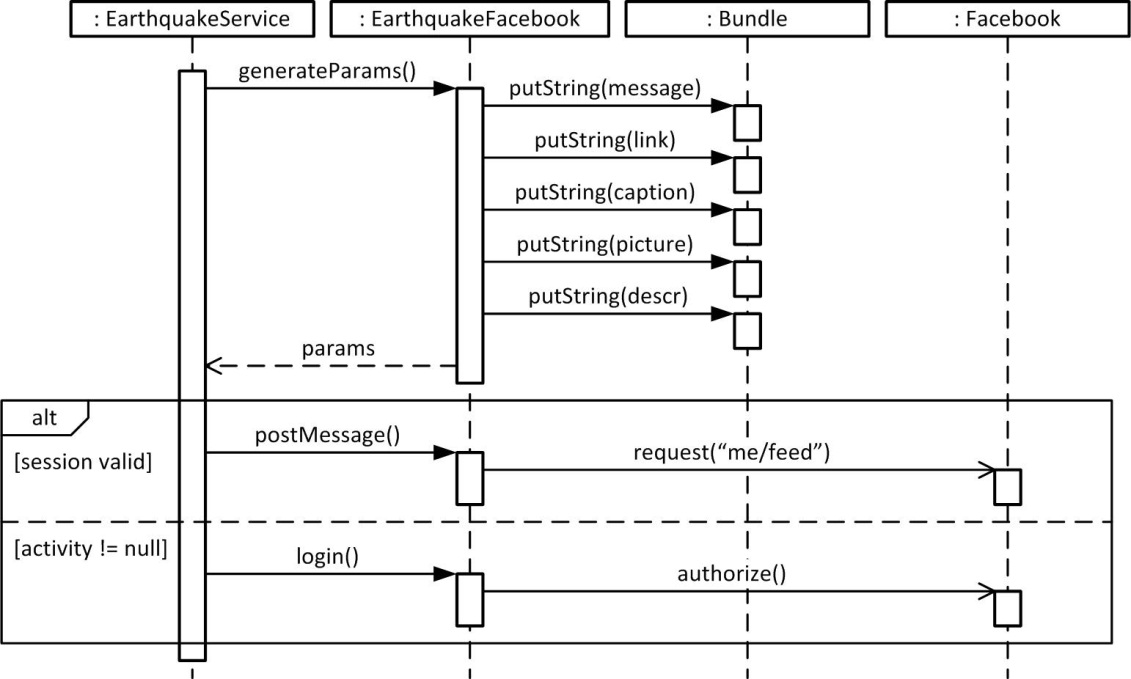
Gambar 3. *Sequence Diagram* – Kirim SMS

### Kirim Email



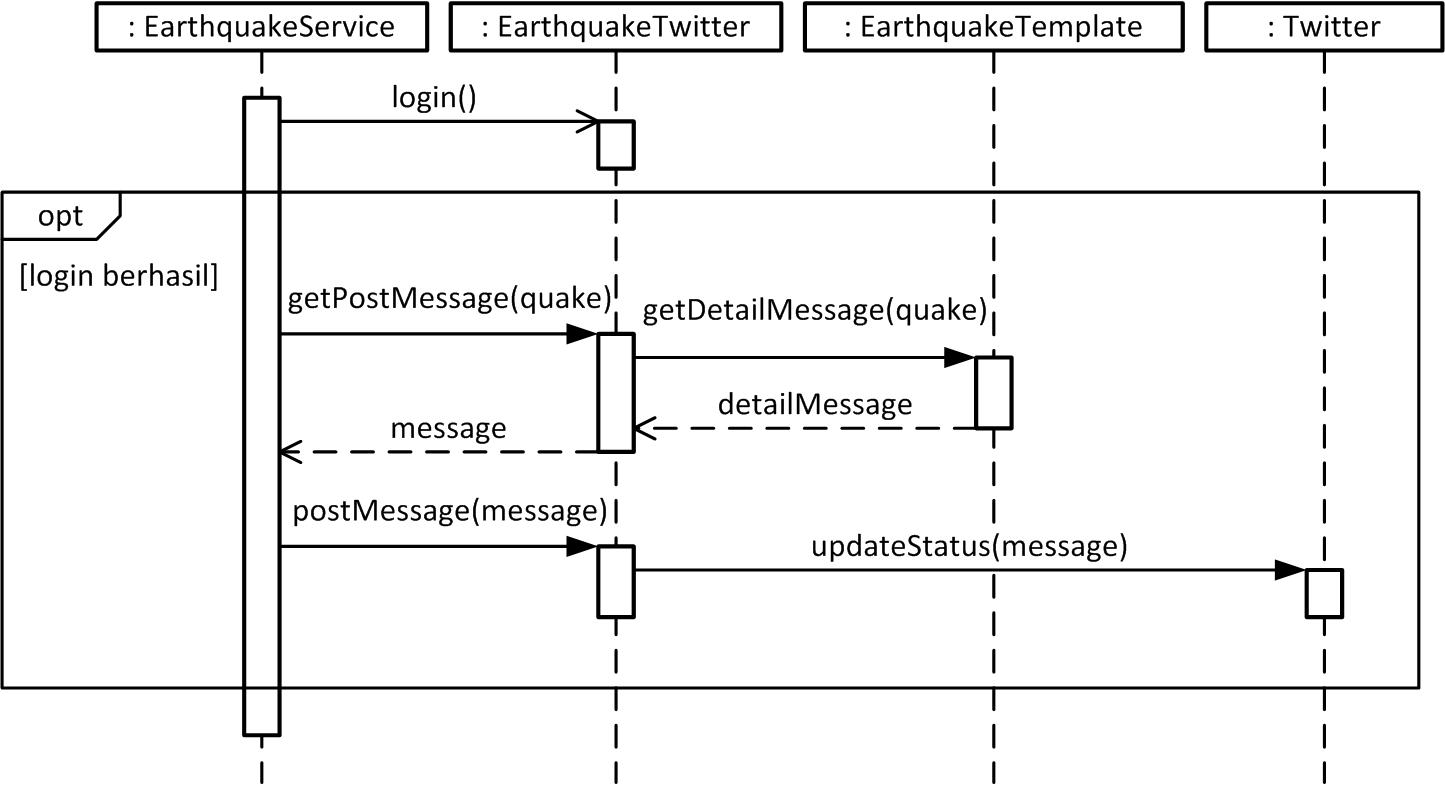
Gambar 3. *Sequence Diagram* – Kirim Email

### *Share* ke Facebook



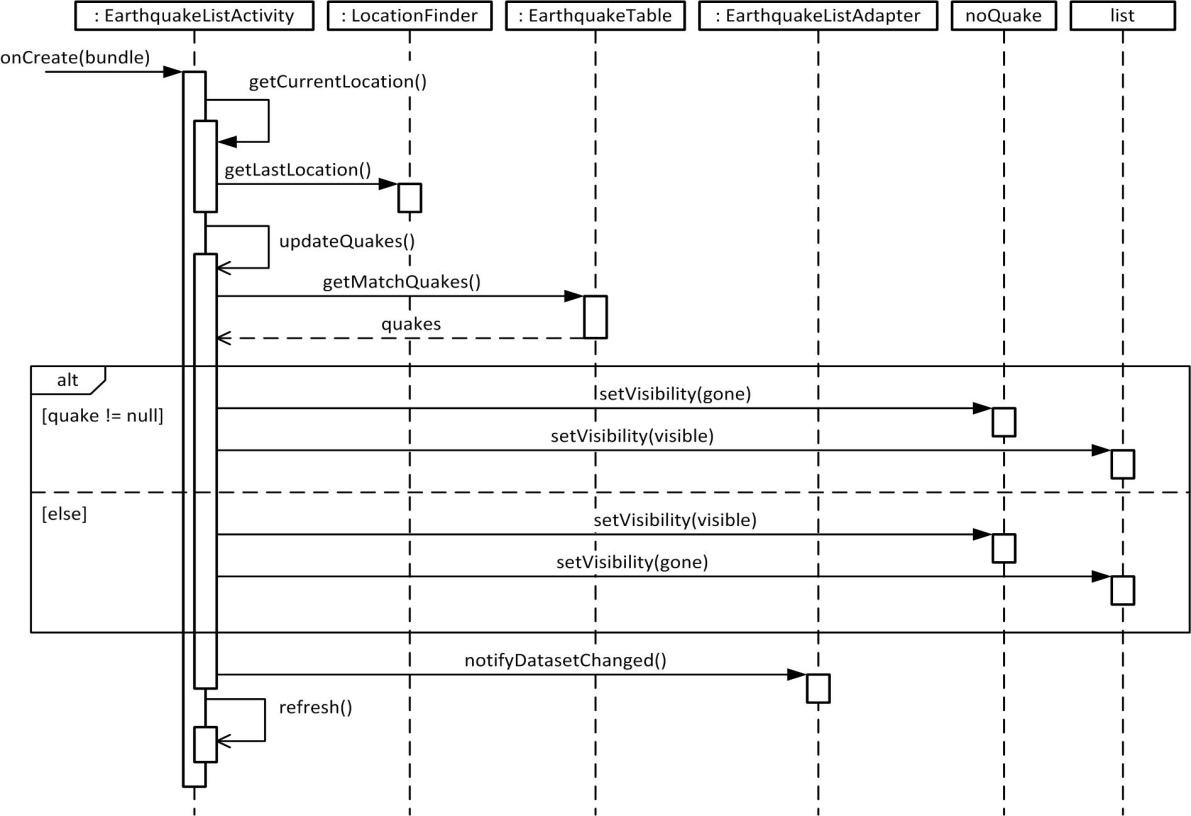
Gambar 3. *Sequence Diagram* – *Share* ke Facebook

### Share ke Twitter



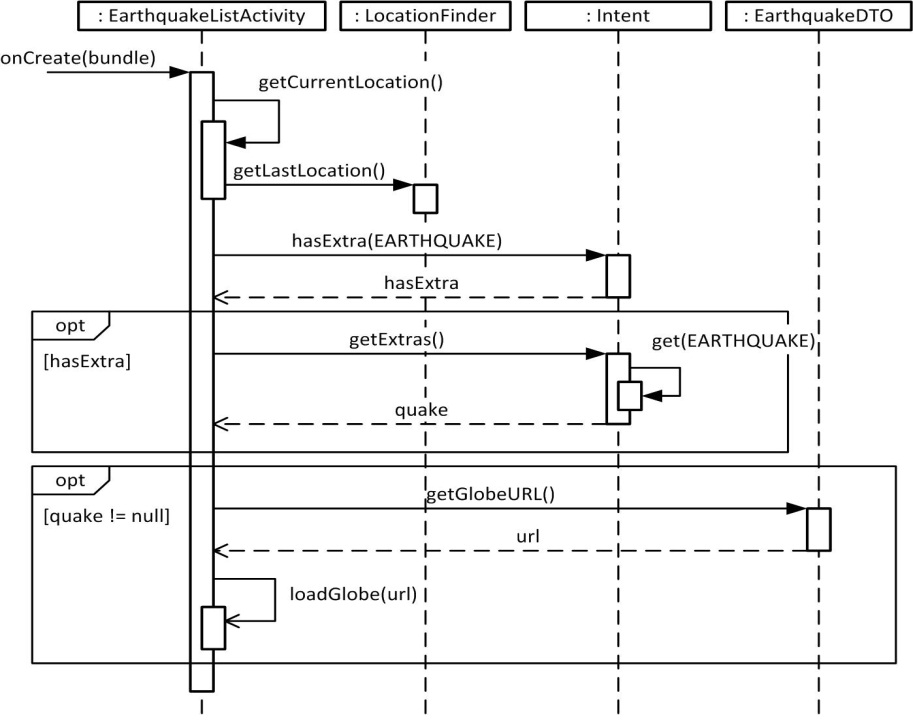
Gambar 3. *Sequence Diagram* – *Share* ke Facebook

### Menampilkan pada *List*



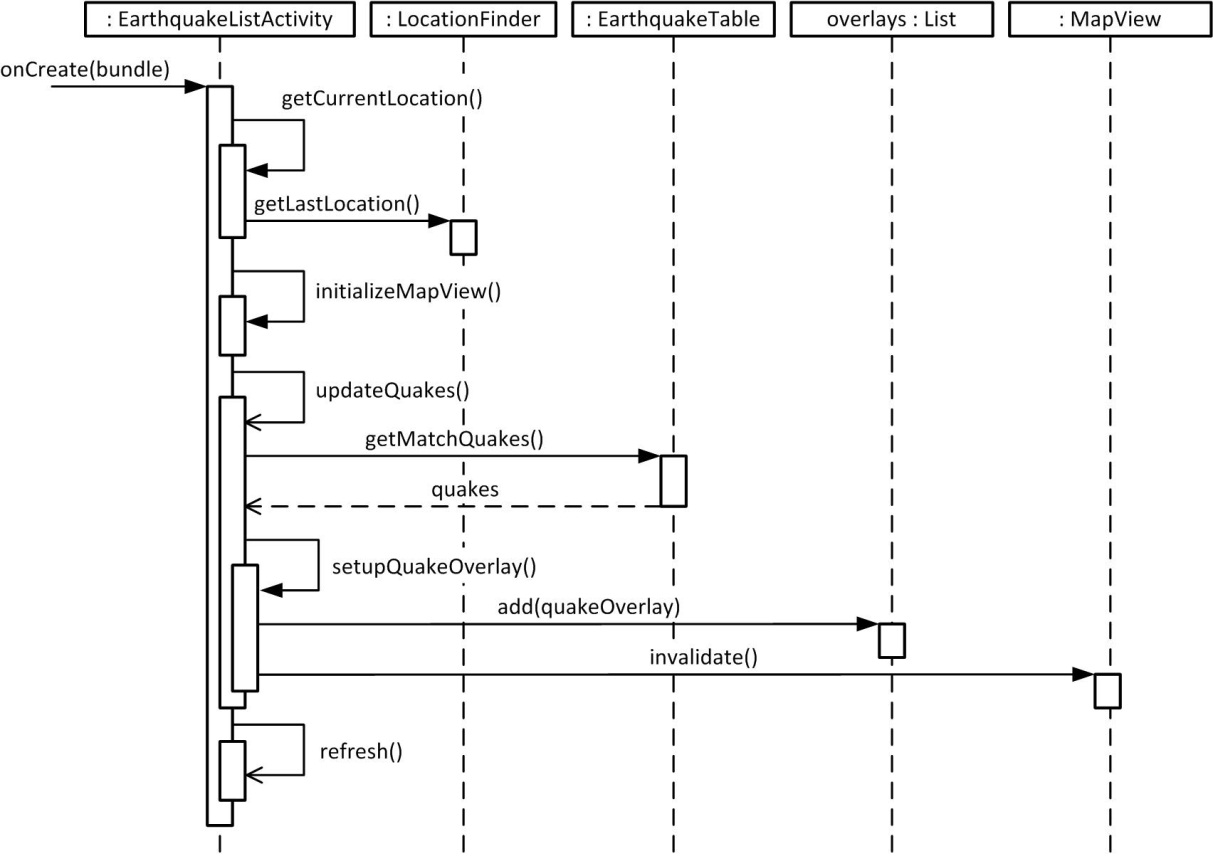
Gambar 3. *Sequence Diagram* – Menampilkan pada *List*

### Menampilkan Detail Gempa



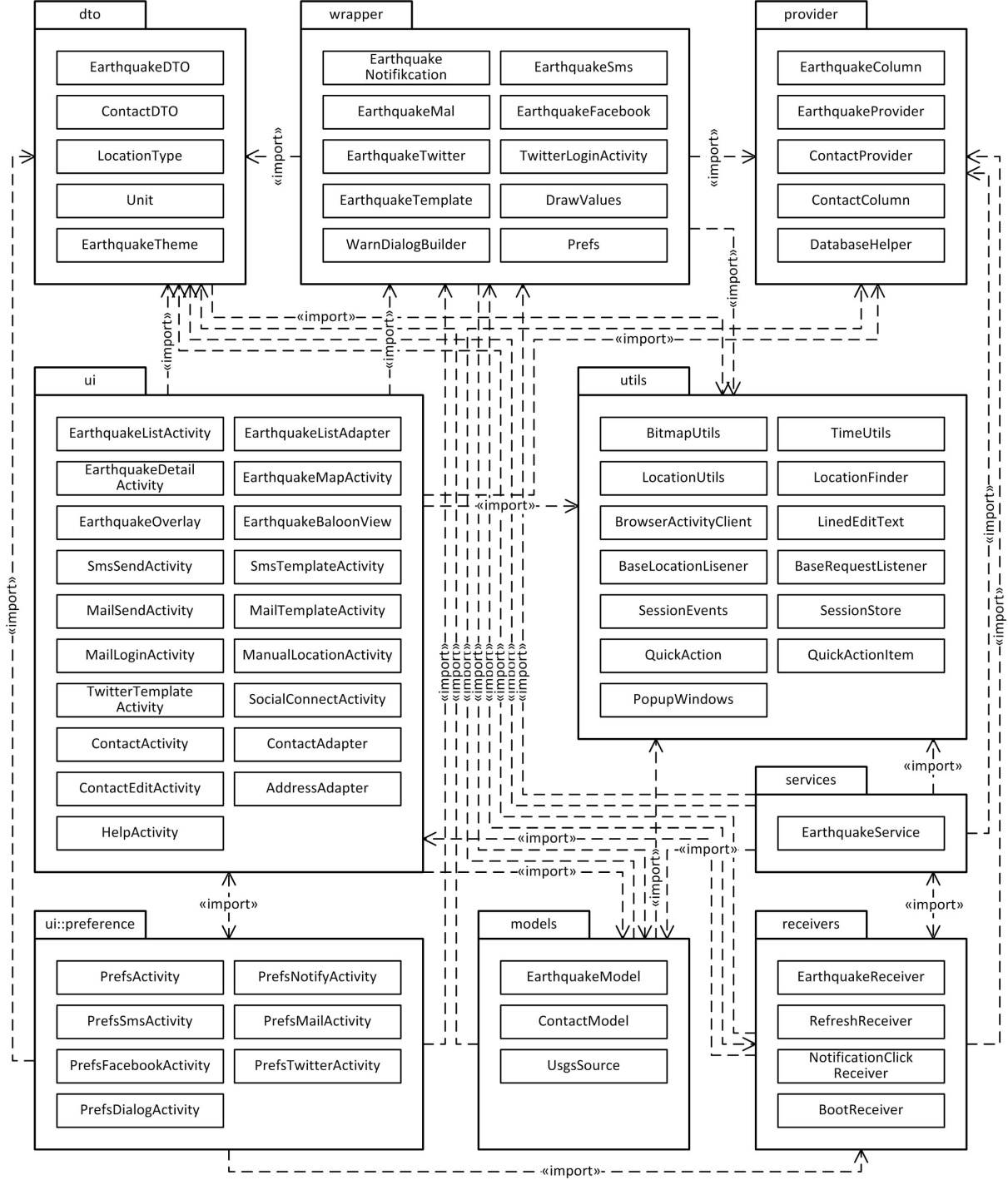
Gambar 3. *Sequence Diagram* – Menampilkan Detail Gempa

### Menampilkan pada Peta



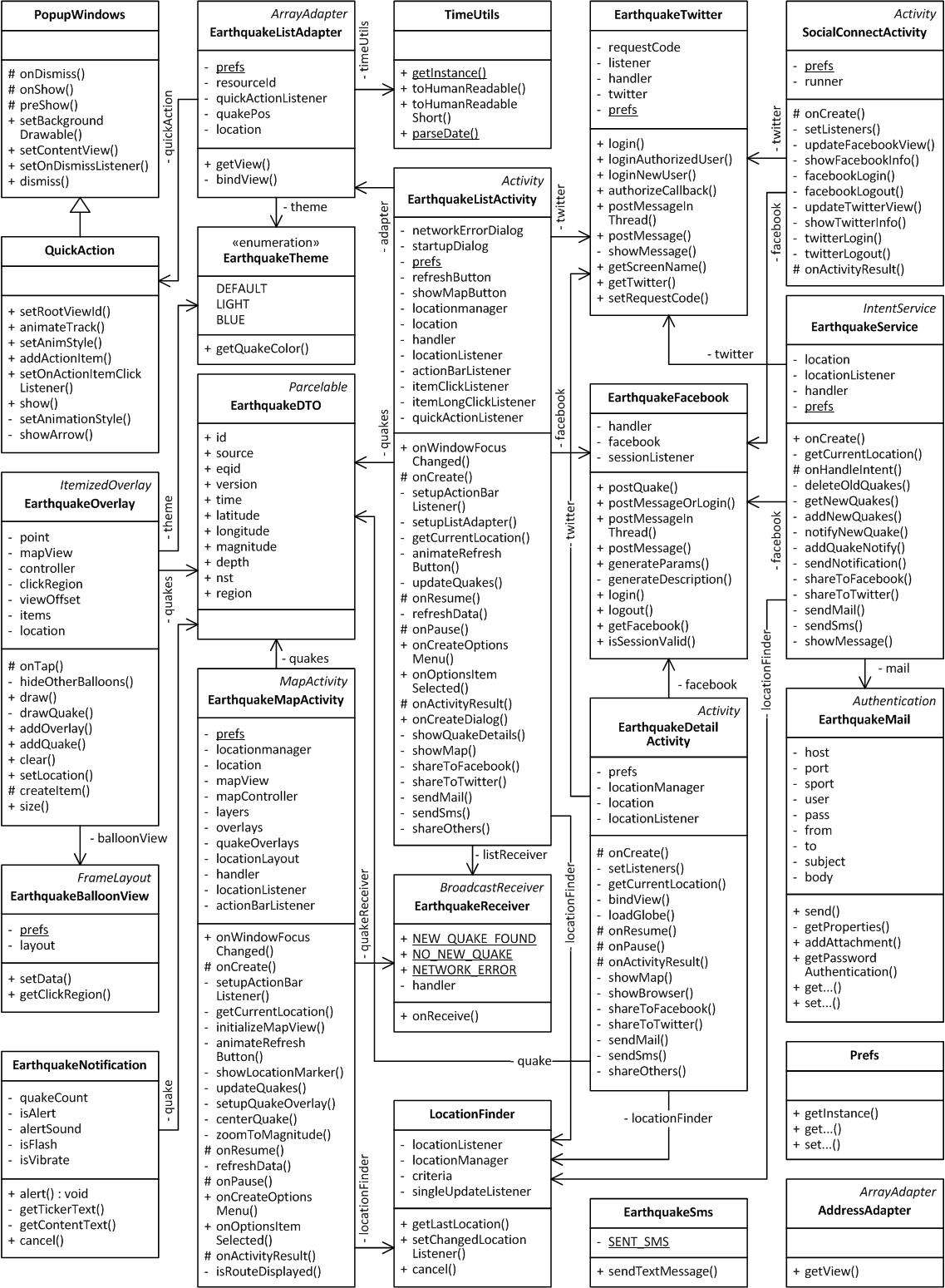
Gambar 3. *Sequence Diagram* – Menampilkan pada Peta

## *Package Diagram*

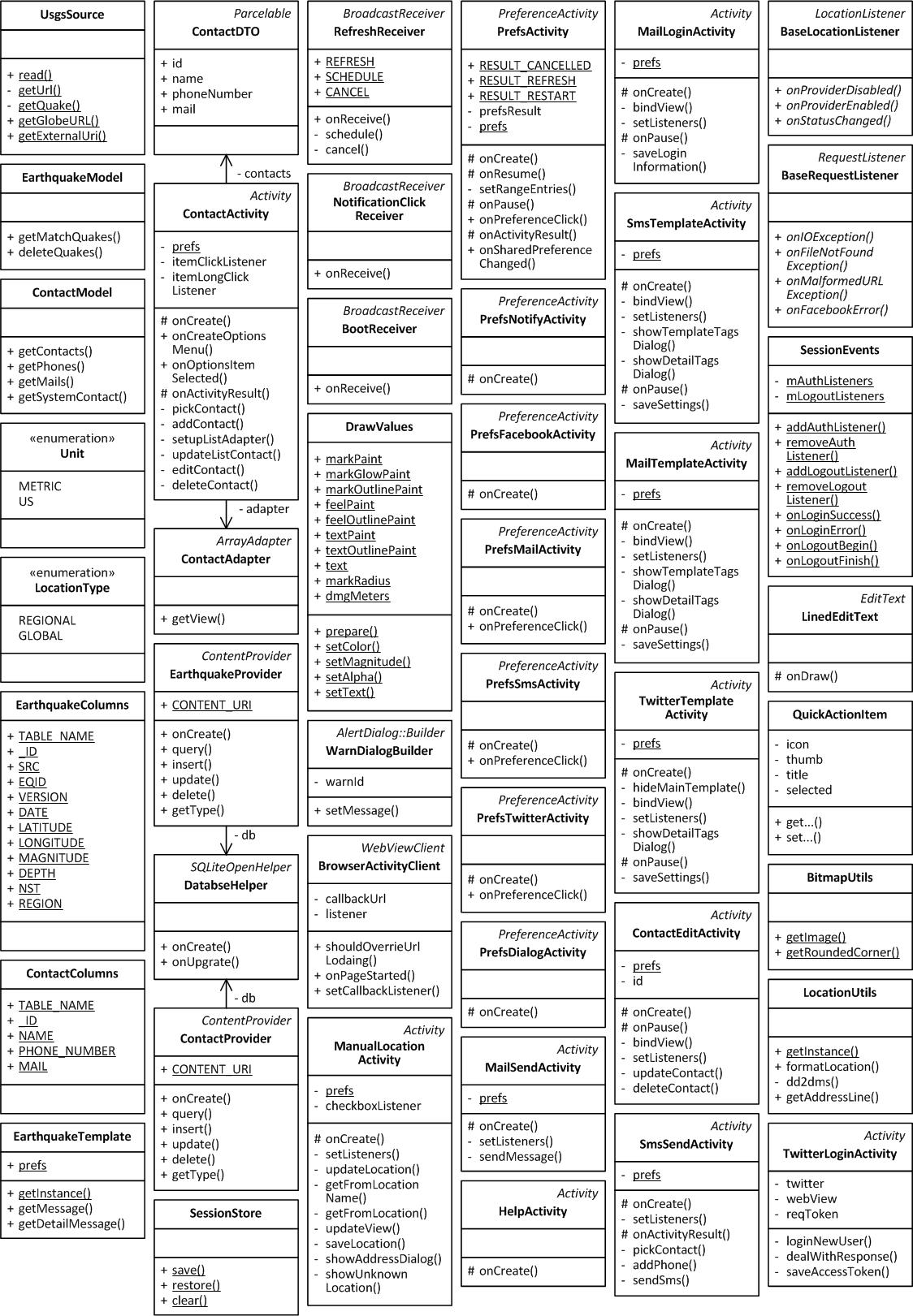


Gambar 3. *Package Diagram*

## Class Diagram



Gambar 3. *Class Diagram*



Gambar 3. *Class Diagram* (Lanjutan)

## Struktur *Database*

### Tabel Earthquake

|  |  |  |  |
| --- | --- | --- | --- |
| ***Key*** | ***Field*** | ***Type*** | **Keterangan** |
| PK | \_id | Integer | Id tabel, auto increment |
|  | src | Text | *Source network* |
|  | eqid | Text | *Earthquake* Id |
|  | version | Text | Versi |
|  | date | Integer | Tanggal/waktu dalam *timestamp* |
|  | latitude | Float | Lokasi latitude |
|  | longitude | Float | Lokasi longitude |
|  | magnitude | Float | Magnitudo gempa |
|  | depth | Float | Kedalaman gempa |
|  | nst | Integer | *Number of reporting stations* |
|  | region | Text | Wilayah terjadinya gempa |

Tabel 3. Tabel Earthquake

### Tabel Contact

|  |  |  |  |
| --- | --- | --- | --- |
| ***Key*** | ***Field*** | ***Type*** | **Keterangan** |
| PK | \_id | Integer | Id tabel, auto increment |
|  | name | Text | Nama kontak |
|  | phone\_number | Text | Nomor telepon |
|  | mail | Text | Alamat email |

Tabel 3. Tabel Contact

## Perancangan Sistem

### Daftar Gempa

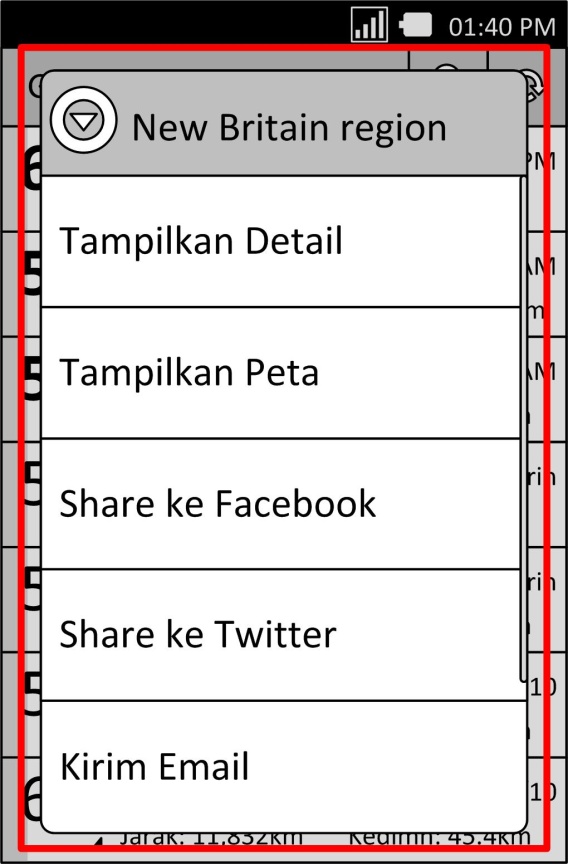
*Activity* bagian atas (disebut *action bar*) terdiri dari dua tombol masing-masing untuk langsung menampilkan peta dan untuk *refresh* data. Bagian bawahnya adalah ListView daftar gempa dan pada kotak paling kiri akan dibuat berwarna sesuai dengan magnitudo gempa, yaitu dari warna hijau untuk magnitudo paling kecil sampai merah untuk magitudo paling besar. Gempa baru yang kurang dari sehari akan ditampilkan dalam *font* yang dicetak tebal. Jika pada *list* diklik maka akan ditampilkan detail gempa.



Gambar 3. Rancangan Tampilan Daftar Gempa

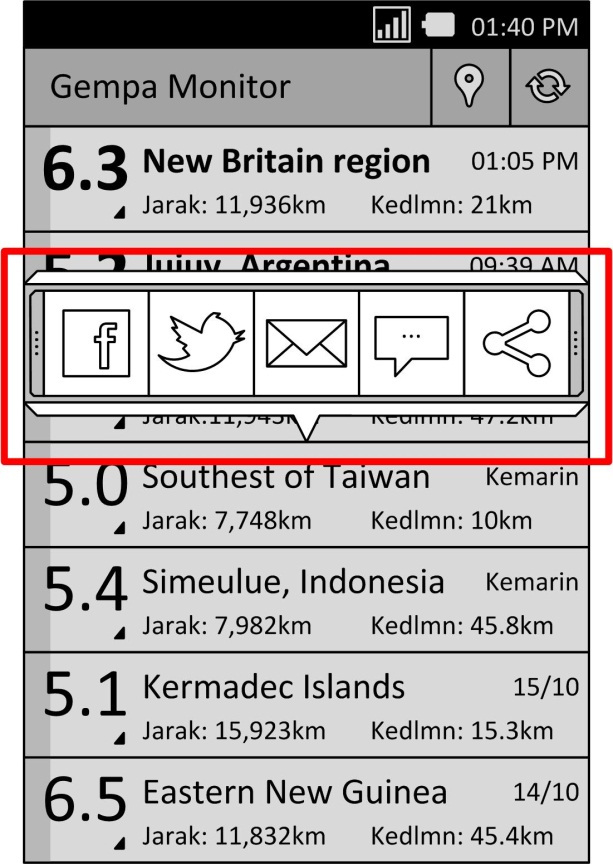
(Kiri: Terdapat Data, Kanan: Tidak Terdapat Data)

Pada daftar gempa ini terdapat menu pilihan yang terdiri dari menu daftar kontak, pengaturan, dan bantuan. Untuk saat ini, bantuan hanya akan berisi informasi tentang aplikasi ini.



Gambar 3. Rancangan Tampilan *Context Menu*

Jika list diklik cukup lama akan ditampilkan *context menu*. Pada *context menu* ini terdapat pilihan yaitu: tampilkan detail (sama fungsinya dengan mengklik pada *list*), tampilkan peta, *share* ke Facebook, *share* ke Twitter, kirim Email, kirim pesan sms, dan *share* lainnya (menampilkan pilihan *share* ke aplikasi lain).



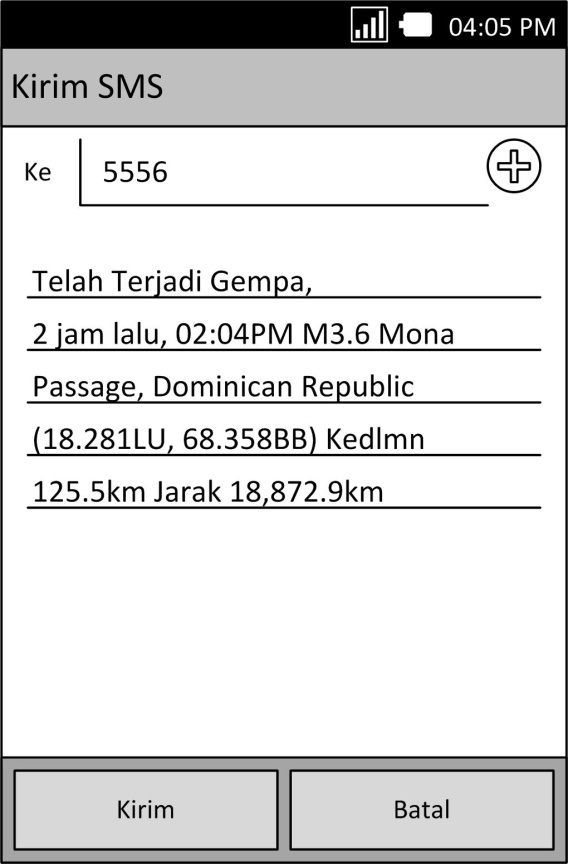
Gambar 3. Rancangan Tampilan *QuickAction*

Jika pada bagian magnitudo yang diklik maka akan ditampilkan popup (disebut quick action). Pilihannya terdiri dari *share* ke Facebook, *share* ke Twitter, kirim Email, kirim pesan sms, dan *share* lainnya.

Sebelum mengirim Email, akan ditampilkan *activity* untuk mengirim Email. Begitu juga dengan mengirim pesan SMS.



Gambar 3. Rancangan Tampilan Kirim Email

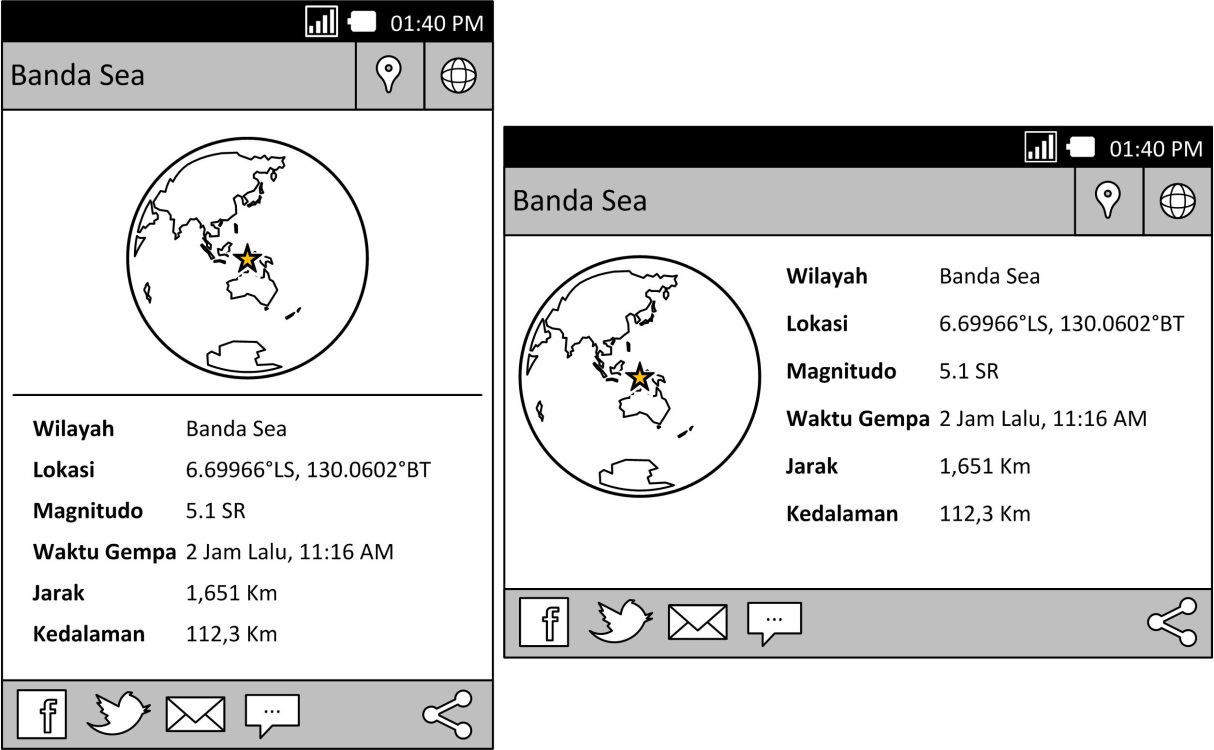


Gambar 3. Rancangan Tampilan Kirim Pesan SMS

### Detail Gempa

Untuk mengoptimalkan tampilan, detail gempa akan dibuat berbeda sesuai dengan orientasi perangkat, satu untuk orientasi *portrait* dan satunya lagi untuk orientasi *landscape*.

Pada *action bar* terdapat dua tombol yaitu untuk menampilkan lokasi gempa ini pada peta dan untuk menampilkannya pada situs resmi USGS, yaitu dengan membuka *web browser* kemudian menuju ke alamat detail gempa ini. Pada bagian bawah terdapat tombol-tombol yang fungsinya sama seperti *quick action* pada daftar gempa.

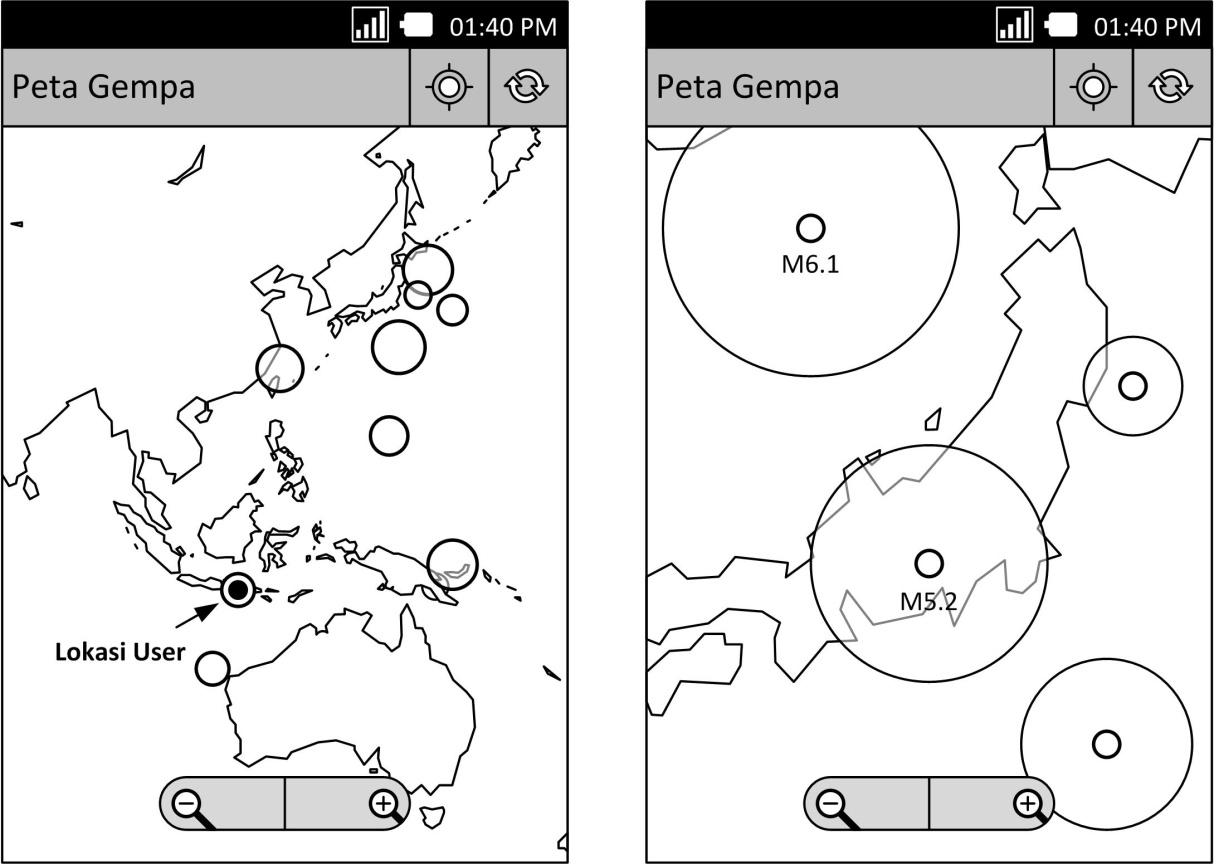


Gambar 3. Rancangan Tampilan Detail Gempa

(Kiri: *Portrait*, Kanan: *Landscape*)

### Peta Gempa

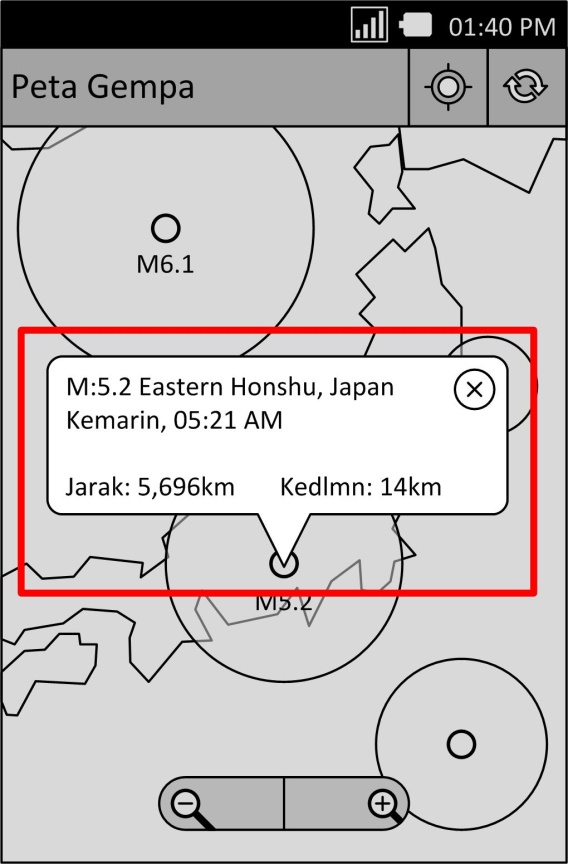
Pada *activity* peta gempa ini terdapat *action bar* yang terdiri dari dua tombol yaitu untuk memperlihatkan lokasi *user* saat ini, dan untuk refresh data. Menu pada *activity* ini terdiri dari menu *layer*, pengaturan, dan menu bantuan.



Gambar 3. Rancangan Tampilan Peta Gempa

(Kiri: Zoom Dibawah 6, Kanan: Zoom 6 Keatas)

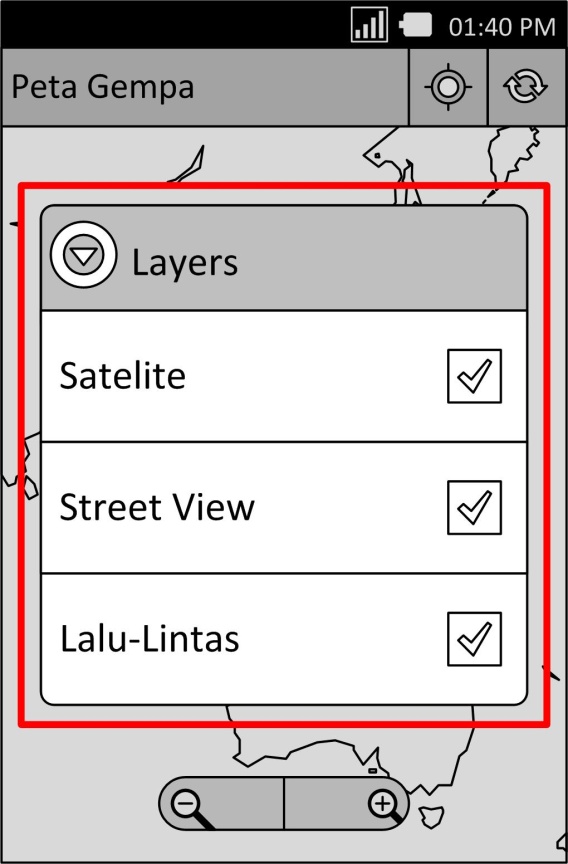
Agar lokasi gempa dapat dilihat lebih jelas, data gempa akan ditampilkan berbeda sesuai dengan tingkat zoom tertentu dan gempa yang cukup lama akan ditampilkan lebih transparan dari gempa baru. Label magnitudonya hanya ditampilkan jika melebihi atau sama dengan 4 SR dan merupakan gempa baru, serta pada tingkat zoom 6 keatas.



Gambar 3. Rancangan Tampilan *Balloon View*

Informasi gempa lebih lengkap dapat dilihat dengan mengklik pada gempa tersebut. Informasi ini ditampilkan pada sebuah *popup view* (disebut *balloon view*).

Pada peta akan terdapat tiga pilihan *layer* yaitu: satelit, menampilkan citra satelit jika diaktifkan, atau menampilkan peta jika tidak diaktifkan; *street view*, menampilkan *street view* jika diaktifkan; dan lalu-lintas, menampilkan kondisi lalu-lintas jika diaktifkan.



Gambar 3. Rancangan Tampilan Layer Peta

### Pengaturan

Pengaturan dapat dikelompokkan dalam empat group yaitu group display, group lokasi, group pemberitahuan dan group lainnya. *Activity* yang digunakan merupakan *activity* pengaturan standar Android.

1. **Group Display**
   1. Magnitudo Minimal [ListPreference]: Magnitudo minimal gempa bumi akan ditampilkan, terdiri dari: M3.0+ goncangan, M4.0+ kecil, M5.0+ menengah, M6.0+ kuat, M7.0+ destroktif, M8.0+ besar, M9.0+ bencana.
   2. Refresh Otomatis [CheckBoxPreference]: Otomatis cek gempa bumi baru secara periodik.
   3. Interval [ListPreference]: Seberapa sering pengecekan gempa bumi baru, terdiri dari: setiap menit, setiap 5 menit, setiap 15 menit\*, setiap ½ jam\*, setiap jam\*, setiap 3 jam, setiap 6 jam, setiap ½ hari\*, setiap hari\*.

Keterangan: tanda (\*) merupakan interval tidak tepat, dioptimasi untuk menghemat baterai.

* 1. Umur Maksimal [ListPreference]: Gempa bumi lebih lama dari ini akan dihapus, terdiri dari: 1 hari, 2 hari, 3 hari, 5 hari, 1 minggu, 2 minggu, 1 bulan, 6 bulan.
  2. Unit [ListPreference]: Menentukan unit yang digunakan ketika menampilkan jarak, terdiri dari: metric (kilometer), US (mil).
  3. Tema [ListPreference]: Tema yang digunakan, terdiri dari: default, cerah, biru.

1. **Group Lokasi**
   1. Deteksi Lokasi Saya [CheckBoxPreference]: Menentukan apakah lokasi dapat dideteksi atau tidak.
   2. Input Lokasi Manual [Preference]: Menginput lokasi secara manual. Akan ditampilkan *activity* input lokasi manual.

Pada saat menginput lokasi manual, akan terdapat dua pilihan yaitu menginput lokasi berdasarkan alamat atau berdasarkan koordinat lokasi.



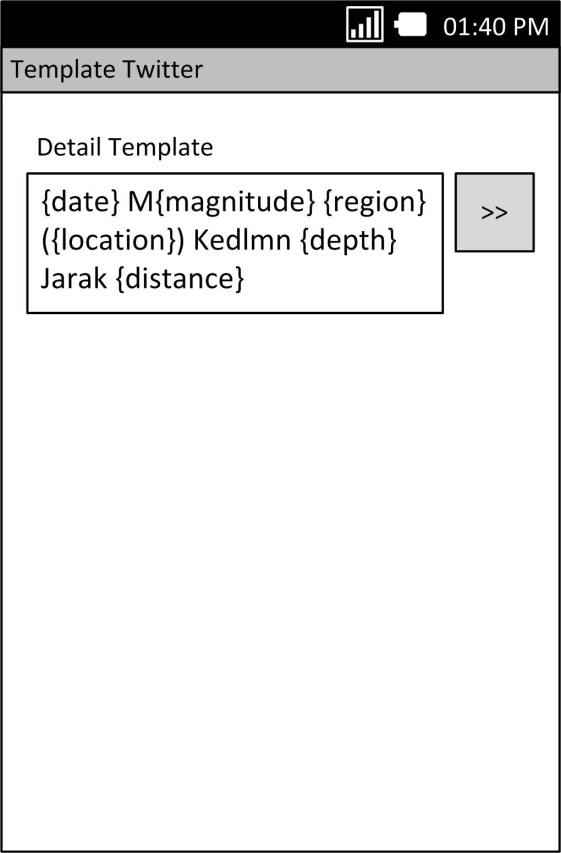
Gambar 3. Rancangan Tampilan Pengaturan Lokasi Manual

Jika berdasarkan alamat, akan digunakan *geocoder* untuk menentukan koordinat lokasi, Jika ternyata terdapat alamat yang sama lainnya maka akan ditampilkan pilihan alamat mana yang dimaksud.

Sebaliknya, jika berdasarkan lokasi, alamat ditentukan dengan menggunakan *reverse geocoder*. Jika alamat yang dihasilkan lebih dari satu, akan ditampilkan pilihan alamat, sama seperti yang berdasarkan alamat diatas.

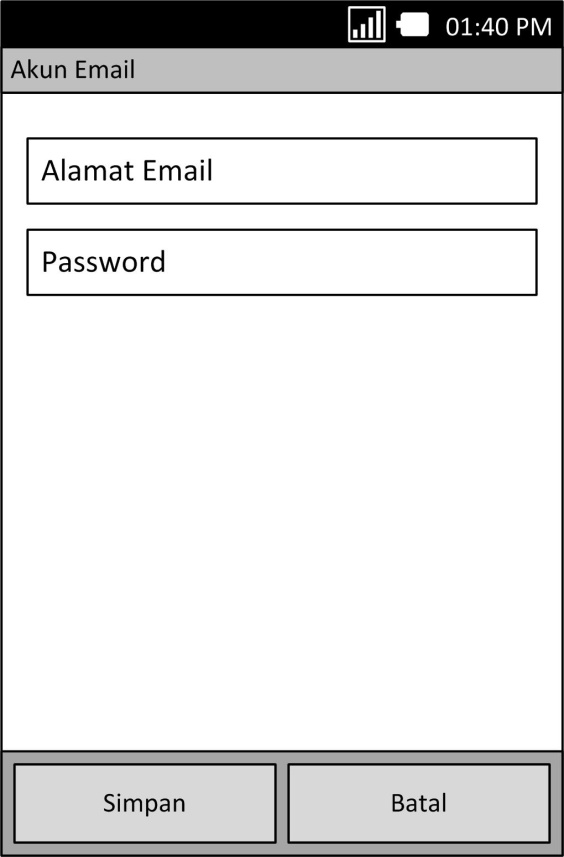
1. **Group Pemberitahuan**
   1. Jarak [ListPreference]: Jarak gempa dikategorikan sebagai wilayah regional, terdiri dari: 100km, 250km, 500km, 1000km.
   2. Kirim Pemberitahuan [CheckBoxPreference]: Kirim pemberitahu-an ketika terjadi gempa.
   3. Pengaturan Pemberitahuan [Preference]: Pengaturan pemberi-tahuan lanjutan.
      1. Magnitudo Regional [ListPreference]: Minimum magnitudo pemberitahuan untuk wilayah regional. List sama dengan magnitudo minimal diatas.
      2. Magnitudo Global [ListPreference]: Minimum magnitudo pemberitahuan untuk wilayah global. List sama dengan magnitudo regional.
      3. *Alert* [CheckBoxPreference]: Membunyikan *alert* saat pemberitahuan.
      4. Suara *Alert* [RingtonePreference]: Suara dari *alert* saat pemberitahuan.
      5. Flash [CheckBoxPreference]: Flash LED saat pemberitahuan.
      6. Getar [CheckBoxPreference]: Getar saat pemberitahuan.
   4. Posting ke Facebook [CheckBoxPreference]: Posting ke Facebook ketika terjadi gempa.
   5. Pengaturan Facebook [Preference]: Pengaturan posting ke Facebook lanjutan.
      1. Magnitudo Regional [ListPreference]: Minimum magnitudo posting pesan untuk wilayah regional. List sama dengan magnitudo minimal diatas.
      2. Magnitudo Global [ListPreference]: Minimum magnitudo posting pesan untuk wilayah global. List sama dengan magnitudo regional.
   6. Posting ke Twitter [CheckBoxPreference]: Posting ke Twitter ketika terjadi gempa.
   7. Pengaturan Twitter [Preference]: Pengaturan posting ke Twitter lanjutan.
      1. Magnitudo Regional [ListPreference]: Minimum magnitudo posting pesan untuk wilayah regional. List sama dengan magnitudo minimal diatas.
      2. Magnitudo Global [ListPreference]: Minimum magnitudo posting pesan untuk wilayah global. List sama dengan magnitudo regional.
      3. Social Connect [Preference]: Koneksi ke jejaring sosial. Akan ditampilkan *activity Social* *Connect*.
      4. Template Twitter [Preference]: Template pesan Twitter yang akan dikirim secara otomatis.

Pada template ini terdapat pilihan {date}, {magnitude}, {region}, {location}, {depth}, {distance}.



Gambar 3. Rancangan Tampilan *Template* Twitter

* 1. Kirim Email [CheckBoxPreference]: Kirim email ketika terjadi gempa bumi.
  2. Pengaturan Email [Preference]: Pengaturan email lanjutan.
     1. Magnitudo Regional [ListPreference]: Minimum magnitudo mengirim email untuk wilayah regional. List sama dengan magnitudo minimal diatas.
     2. Magnitudo Global [ListPreference]: Minimum magnitudo mengirim email untuk wilayah global. List sama dengan magnitudo regional.
     3. *Host* [EditTextPreference]: Pengaturan SMTP *server*
     4. SMTP *Port* [EditTextPreference]: Pengaturan SMTP *port*
     5. *Socket Factory Port* [EditTextPreference]: Pengaturan *port socket factory*.
     6. Akun Email [Preference]: Pengaturan akun email.

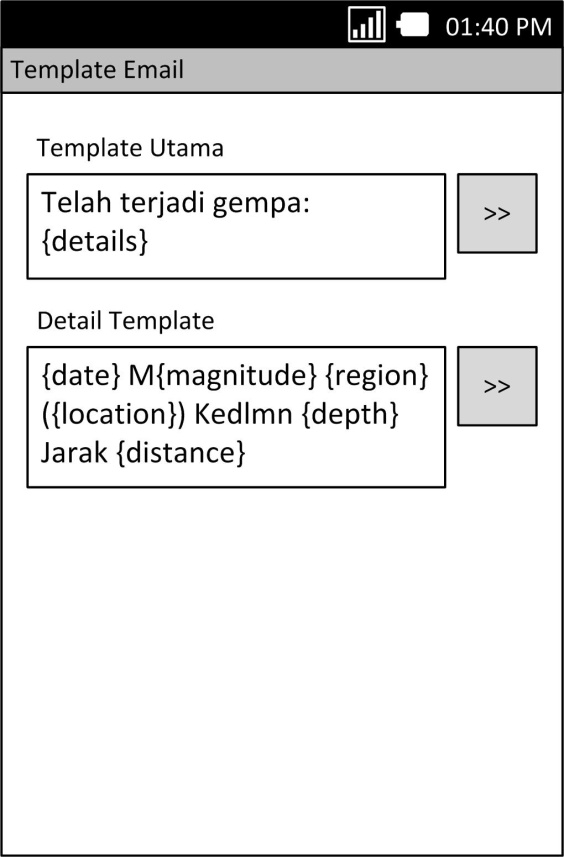


Gambar 3. Rancangan Tampilan Pengaturan Akun Email

* + 1. Template Email [Preference]: Template pesan email yang akan dikirim secara otomatis.

Template pesan dipecah menjadi dua, template utama dan detail template, yang dimaksudkan untuk dapat memungkinkan mengirim banyak gempa sekaligus. Untuk saat ini, mengirim banyak gempa sekaligus tidak tersedia jika dilakukan secara manual.

Pada template utama hanya akan ada pilihan {detail}. Sedangkan pada detail template terdapat pilihan sama seperti pada template Twitter.



Gambar 3. Rancangan Tampilan Pengaturan *Template* Email

* 1. Kirim SMS [CheckBoxPreference]: Kirim pesan ketika terjadi gempa.
  2. Pengaturan SMS [Preference]: Pengaturan sms lanjutan.
     1. Magnitudo Regional [ListPreference]: Minimum magnitudo mengirim SMS untuk wilayah regional. List sama dengan magnitudo minimal diatas.
     2. Magnitudo Global [ListPreference]: Minimum magnitudo mengirim SMS untuk wilayah global. List sama dengan magnitudo regional.
     3. Template Pesan [Preference]: Template pesan SMS. Akan ditampilkan *activity* template pesan.

Template ini mirip dengan template Email.

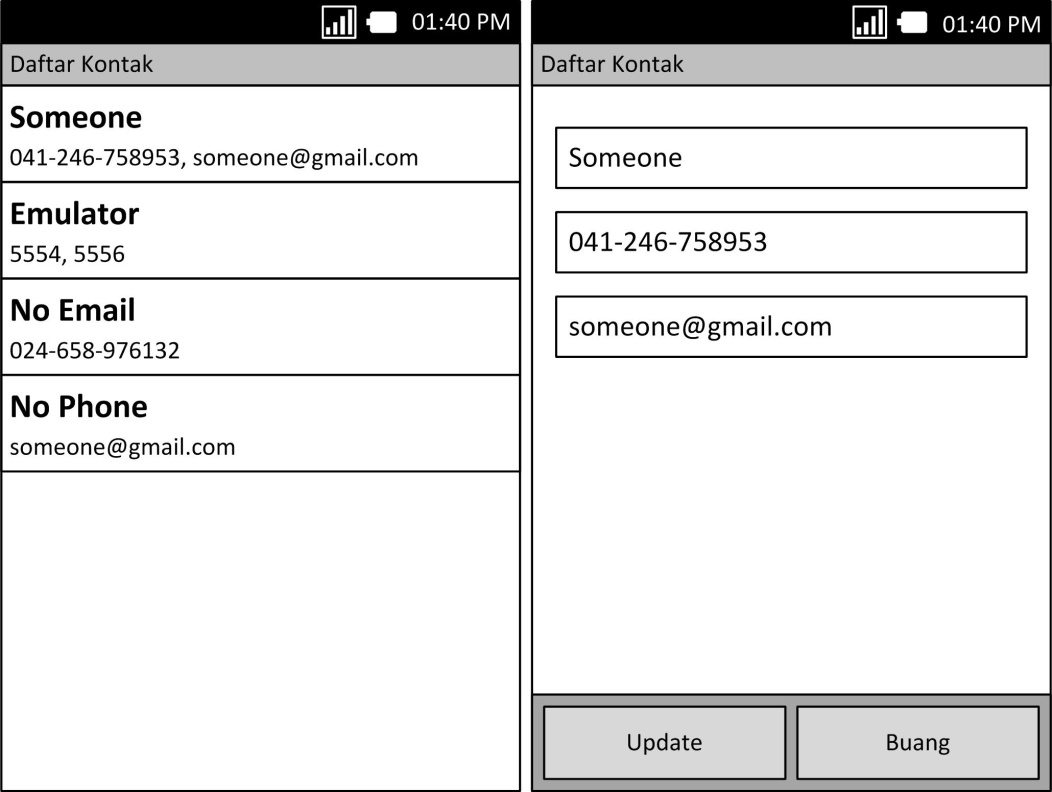
* 1. Social Connect [Preference]: Koneksi ke jejaring sosial. Akan ditampilkan *activity Social* *Connect*.



Gambar 3. Rancangan Tampilan *Social* *Connect*

*Activity helper* yang berfungsi untuk *login* atau *logout* dari Facebook dan twitter. *Activity* ini mungkin hanya akan digunakan jika ingin *logout*, karena saat posting dan belum login, akan otomatis diarahkan untuk login terlebih dahulu, baru pesan di posting.

* 1. Daftar Kontak [Preference]: Daftar kontak yang akan dikirimkan SMS dan email secara otomatis. Akan ditampilkan *activity* daftar kontak.



Gambar 3. Rancangan Tampilan Daftar Kontak

(Kiri: Daftar, Kanan: Tambah/Ubah Data)

Daftar kontak ini hanya akan digunakan pada saat mengirim SMS dan email secara otomatis. Untuk mengirim secara manual, akan digunakan daftar kontak telepon. Pada *activity* ini terdapat menu tambah, untuk menambah dari kontak telepon, dan menu input, untuk menginput manual.

1. **Group Lainnya**
   1. Start Saat Boot [CheckBoxPreference]: Start *service* secara otomatis saat sistem *boot*.
   2. Dialog [Preference]: Perlihatkan atau sembunyikan dialog.
      1. Startup Dialog [CheckBoxPreference]: Perlihatkan startup dialog.
      2. Peringatan Interval [CheckBoxPreference]: Perlihatkan peringatan bahwa interval terlalu sering akan menguras baterai.

# IMPLEMENTASI SISTEM

Setelah dilakukan analisis dan perancangan terhadap sistem, maka sistem tersebut siap diimplementasikan. Tahap implementasi terhadap sistem ini merupakan tahap meletakkan rancangan sistem ke dalam bentuk coding bahasa pemrograman.

Tahapan dalam mengimplementasikan aplikasi monitoring gempa ini meliputi perancangan antarmuka sistem dan implementasi proses mendapatkan data dari USGS, menampilkan data tersebut dan memberikan notifikasi jika terdapat gempa baru yang cukup kuat sesuai dengan pengaturan.

## Kebutuhan Implementasi

Sebelum memulai *coding*, terdapat beberapa hal yang perlu di-*install* yaitu Java, IDE (*Integrated Development Environment*), dan Android SDK (*Software Development Kit*). Berikut ini adalah kebutuhan minimal sistem dan software untuk mengembangkan aplikasi Android:

1. Sistem Operasi
2. Windows XP (32-bit), Vista (32- atau 64-bit), atau Windows 7 (32- atau 64-bit)
3. Mac OS X 10.5.8 atau lebih baru (hanya x86)
4. Linux (diuji pada Ubuntu Linux, Lucid Lynx)
5. Java JDK (*Java Development Kit*)

Minimal versi Java JDK yang diperlukan yaitu JDK 5 atau JDK 6, JRE (*Java Runtime Environment*) sendiri belum cukup.

1. Eclipse IDE

Minimal versi untuk Eclipse adalah Eclipse 3.5 (Galileo) atau lebih baru. Google telah menyediakan plug-in untuk Eclipse yaitu *Android Development Tools* (ADT) sehingga pengembangan aplikasi dapat lebih mudah.

1. Android SDK

Dimulai dari Android 2.0, Android SDK dipecah menjadi dua bagian yaitu SDK *Starter Package* dan SDK *Components*. SDK *Starter Package* dapat didownload dari website Android[[1]](#footnote-1), sedangkan SDK *Compontent* di-*download* melalui Android SDK *Starter Package* tadi.

Pada aplikasi monitoring gempa sendiri ditargetkan dengan minimal Android versi 2.1 (Eclair), artinya aplikasi hanya akan berjalan pada Android versi 2.1 atau lebih baru. Aplikasi juga memerlukan *library* pemetaan besutan Google yaitu Maps *external library* yang terdapat pada *package* com.google.android.maps. *Library* ini bukan merupakan bagian *standar library* Android, jadi ini mungkin tidak tersedia pada beberapa perangkat Android.

Untuk dapat memposting pesan ke Facebook secara otomatis, digunakan *library* Facebook Android SDK[[2]](#footnote-2), sedangkan posting ke Twitter menggunakan *library* Twitter4J[[3]](#footnote-3), dan mengirim email menggunakan *library* javamail-android[[4]](#footnote-4).

### Google Maps

*Class* MapView pada Maps *external library* merupakan *class* yang sangat berguna yang memungkinkan dengan mudah mengintegrasikan Google Map ke dalam aplikasi.

Karena MapView memberikan akses data ke Google Maps, maka perlu mendaftar pada layanan Google Maps dan menyetujui ketentuan layanan yang berlaku. Pendaftarannya mempunyai dua bagian:

1. Mendaftarkan sertifikat sidik jari MD5 yang akan digunakan untuk menandatangani aplikasi. Pertama, menghasilkan sertifikat ini pada *comman-line*.

$ keytool -list -alias alias\_name –keystore my-key.keystore

Certificate fingerprint (MD5): 94:1E:43:49:87:73:BB:E6:A6:88:D7:20:F1:8E:B5:98

Listing 4. Menghasilkan Sertifikat Sidik Jari

Selanjutnya membuka situs web pendaftaran[[5]](#footnote-5), baca dan setujui ketentuan dan layanannya, paste sertifikat sidik jari, dan klik “*Generate* API *Key*”. Server akan mengembalikan hasil *Key string*.

1. Menambahkan *referensi* Maps API Key pada MapView. Ada dua cara yang dapat dilakukan, melalui deklarasi pada *layout* XML atau melalui kode.

Untuk elemen <MapView> dideklarasikan pada *file layout* XML:

<com.google.android.maps.MapView  
 android:layout\_width=*"fill\_parent"*

android:layout\_height=*"fill\_parent"*

android:clickable=*"true"*

**android:apiKey=*"Maps\_ApiKey\_String"***

/>

Listing 4. *Reference* Maps API Key pada *Layout*

Untuk MapView melalui kode:

mapView = **new** MapView(**this**, "**Maps\_ApiKey\_String**");

Listing 4. *Reference* Maps API Key pada Kode Program

Langkah akhir untuk mengaktifkan elemen MapView agar berjalan dengan baik yaitu dengan menambahkan elemen <uses-library> yang mengacu pada *library* com.google.android.maps pada Android Manifest.xml:

<manifest  
 xmlns:android=*"http://schemas.android.com/apk/res/android"*

package=*"com.adisayoga.earthquake"*>

...

<application>

**<uses-library android:name=*"com.google.android.maps"*/>**

...

</manifest>

Listing 4. Elemen <uses-linrary> pada AndroidManifest.xml

### Facebook

Facebook *Platform* memungkinkan untuk mengintegrasikan aplikasi Android dengan Facebook *login* dan API (*Application Programming Interface*), dan juga SSO (*Single Sign-On*) untuk membiarkan *user sign-in* ke aplikasi menggunakan identitas mereka. *User* masuk dengan segera tanpa harus memasukkan *username* dan *password*, jika sudah masuk ke Facebook Android *app* sebelumnya. Adapun langkah-langkah mengintegrasikan Android dengan Facebook yaitu:

1. Pertama, dengan membuat aplikasi baru pada Facebook[[6]](#footnote-6), kemudian mengisi informasi dasar aplikasi.
2. Menambahkan tanda tangan aplikasi ke pengaturan Facebook App. Facebook memerlukan tambahan lapisan keamanan untuk aplikasi *mobile* dalam bentuk tanda tangan aplikasi. Untuk menghasilkan tanda tangan, digunakan Keytool. Selain itu juga diperlukan Openssl[[7]](#footnote-7):

$ keytool -exportcert -alias androiddebugkey –keystore \

debug.keystore > debug.txt

$ openssl sha1 -binary debug.txt > debug\_sha.txt

$ openssl base64 -in debug\_sha.txt > debug\_base64.txt

Listing 4. Menghasilkan Tanda Tangan Aplikasi untuk Facebook

File debug\_base64.txt berisi nilai hash, nilai ini disalin ke pengaturan aplikasi pada Facebook.

1. Membuat *project* Facebook SDK[[8]](#footnote-8). Kemudian pada *project* monitoring gempa ditambahkan referensi ke *project* Facebook SDK ini.
2. Selanjutnya inisiasi objek Facebook pada kode program dengan menggunakan *app* ID Facebook:

facebook = **new** Facebook(**"YOUR\_APP\_ID"**);

Listing 4. Inisiasi Objek Facebook

### Twitter

Untuk dapat memposting pesan ke Twitter digunakan *library* Twitter4J. Twitter4J sendiri bukan merupakan Java *library* resmi untuk Twitter API. Langkah untuk mengintegrasikannya yaitu:

1. Membuat aplikasi baru pada Twitter[[9]](#footnote-9)
2. Menambahkan *reference* Twitter4J pada *project*
3. Implementasi pada kode program dengan melewatkan *cunsomer key* dan *cunsomer secret* aplikasi Twitter:

twitter = new TwitterFactory().getInstance();

twitter.setOAuthConsumer(**"CONSUMER\_KEY"**, **"CONSUMER\_SECRET"**);

Listing 4. Inisiasi Objek Twitter

## Kebutuhan Perangkat Keras

Android SDK memerlukan ruang penyimpanan untuk setiap komponen yang di-*install*. Tabel berikut menyediakan perkiraan keperluan ruang penyimpanan.

|  |  |  |
| --- | --- | --- |
| **Tipe  Komponen** | **Ukuran Perkiraan** | **Keterangan** |
| SDK Tools | 35 MB | Diperlukan |
| SDK Platform-tools | 6 MB | Diperlukan |
| Android platform (masing-masing) | 150 MB | Setidaknya satu platform diperlukan |
| SDK Add-on (masing-masing) | 100 MB | Optional |
| USB Driver untuk Windows | 10 MB | Optional. Hanya untuk Windows |
| Sample (per platform) | 10 MB | Optional |
| Dokumentasi *offline* | 250 MB | Optional |

Tabel 4. Keperluan Ruang Penyimpanan Android SDK

Android berjalan pada perangkat yang bervariasi yang menyediakan ukuran dan densitas layar berbeda-beda. Pembuatan aplikasi seharusnya difokuskan hanya pada ukuran layar dan densitas. Android membagi rentang ukuran layar dan densitas menjadi:

1. Serangkaian dari empat generalisasi ukuran fisik layar: *small*, *normal*, *large*, dan *xlarge*.
2. Serangkaian dari empat generalisasi densitas: ldpi (*low*), mdpi (*medium*), hdpi (*high*), dan xhdpi (*extra high*).

Pada tugas akhir ini, aplikasi hanya dirancang untuk ukuran layar *handphone* yaitu ukuran layar *small* atau *normal* dan densitas ldpi, mdpi, dan hdpi. Aplikasi belum dioptimalkan untuk layar tablet. Selain itu, aplikasi juga memerlukan GPS untuk menentukan lokasi *user*, dan semua perangkat Android seharusnya sudah dilengkapi GPS.

Seperti dijelaskan pada landasan teori, setiap aplikasi Android berjalan pada proses linux berbeda. *Hardware* melarang satu proses mengakses memori proses lain. Akses ke operasi kritis tertentu dibatasi dan harus secara khusus meminta ijin untuk menggunakannya melalui AndroidManifest.xml. Berikut ini adalah ijin yang diperlukan untuk aplikasi monitoring gempa:

<uses-permission

android:name=*"android.permission.INTERNET"* />

<uses-permission

android:name=*"android.permission.ACCESS\_COARSE\_LOCATION"* />

<uses-permission

android:name=*"android.permission.ACCESS\_FINE\_LOCATION"* />

<uses-permission

android:name=*"android.permission.VIBRATE"* />

<uses-permission

android:name=*"android.permission.WAKE\_LOCK"* />

<uses-permission

android:name=*"android.permission.RECEIVE\_BOOT\_COMPLETED"* />

<uses-permission

android:name=*"android.permission.READ\_CONTACTS"* />

<uses-permission  
 android:name=*"android.permission.SEND\_SMS"* />

Listing 4. Ijin Aplikasi pada AndroidManifest.xml

## Komponen Sistem

### Service

Aplikasi monitoring gempa ini memanfaatkan service untuk mengambil data dari USGS dan menyimpannya ke *database* dan juga untuk mengirim notifikasi, mengirim SMS, mengirim email, posting ke Facebook maupun posting ke Twitter secara otomatis.

**protected** **void** onHandleIntent(Intent intent) {

deleteOldQuakes(prefs.getMaxAge());

**try** {

...

quakes = UsgsSource.*read*(lastUpdate, minMagnitude);

...

**if** (quakes != **null** && quakes.size() > 0) {

quakes = getNewQuakes(quakes);

}

**if** (quakes != **null** && quakes.size() > 0) {

addNewQuakes(quakes);

notifyNewQuake(quakes);

sendBroadcast(**new** Intent(*NEW\_QUAKE\_FOUND*));

} **else** {

sendBroadcast(**new** Intent(*NO\_NEW\_QUAKE*));

}

} **catch** (IOException e) {

sendBroadcast(**new** Intent(*NETWORK\_ERROR*));

}

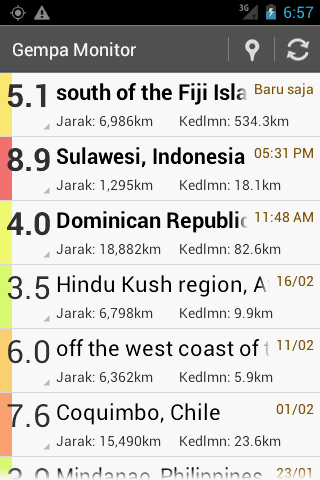
}

Listing 4. Potongan Kode Program pada *Service*

### Daftar Gempa

Pada daftar gempa, terdapat dua layout yang akan ditampilkan atau disembunyikan satu sama lainnya. Satu untuk menampilkan daftar gempa, dan satunya lagi untuk menampilkan keterangan tidak ada data.

Saat terdapat data baru, service akan mengirim *broadcast* terdapat data baru, dan *activity* daftar gempa ini mendengarkan/ menerima *broadcast* melalui *handler* sehingga *list* dapat diupdate.

Gambar 4. Implementasi Tampilan Daftar Gempa

(Kiri: Terdapat Data, Kanan: Tidak Terdapat Data)

**public** **void** onReceive(Context context, Intent intent) {

String action = intent.getAction();

**if** (action.equals(*NEW\_QUAKE\_FOUND*)) {

handler.sendEmptyMessage(*NEW\_QUAKE\_FOUND\_WHAT*);

} **else** **if** (action.equals(*NO\_NEW\_QUAKE*)) {

handler.sendEmptyMessage(*NO\_NEW\_QUAKE\_WHAT*);

} **else** **if** (action.equals(*NETWORK\_ERROR*)) {  
 handler.sendEmptyMessage(*NETWORK\_ERROR\_WHAT*);  
 }

}

Listing 4. *Receiver* dari *Broadcast* Daftar Gempa

**private** **final** Handler handler = **new** Handler() {

**public** **void** handleMessage(Message message) {

**switch** (message.what) {

**case** *NEW\_QUAKE\_FOUND\_WHAT*:

updateQuakes(...);

**break**;

**case** *NO\_NEW\_QUAKE\_WHAT*:

...

**break**;

**case** *NETWORK\_ERROR\_WHAT*:

...

showDialog(*NETWORK\_ERROR\_DIALOG*);

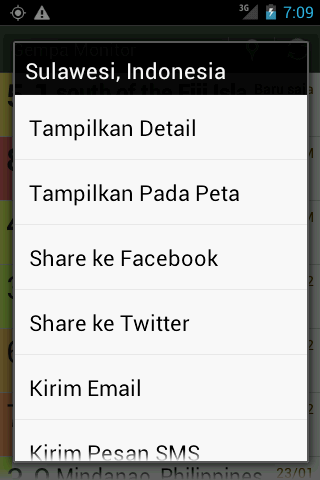
**break**;

}

}

};

Listing 4. *Handler* Daftar Gempa untuk Dilewatkan ke *Receiver*

Gambar 4. Implementasi Tampilan *Context Menu* dan *Quick Action*

Pada Android 2.1 belum mendukung *quick action* secara ekplisit, sehingga pada tugas akhir ini mengunakan *quick action* buatan pihak ketiga[[10]](#footnote-10) dengan sedikit penyesuaian.

### Detail Gempa

Detail gempa ini digunakan untuk menampilkan informasi gempa yang lebih lengkap. Sebelum menampilkannya, data didapatkan dari *extra* yang dilewatkan oleh daftar gempa pada saat memulai *activity* detail gempa ini.

|  |  |
| --- | --- |
|  |  |

Gambar 4. Implementasi Tampilan Detail Gempa

(Kiri: Portrait, Kanan: Landscape)

Intent intent = **new** Intent(**this**, EarthquakeDetailActivity.**class**);

**if** (quake != **null**) intent.putExtra(*TABLE\_NAME*, quake);

startActivity(intent);

Listing 4. Memulai *Activity* Detail Gempa

Intent intent = getIntent();

**if** (intent.hasExtra(*TABLE\_NAME*)) {

quake = (EarthquakeDTO) intent.getExtras().get(*TABLE\_NAME*);

bindView(quake, ...);

}

Listing 4. Mendapatkan Data Gempa dari *Intent*

Pada Android, *sharing* konten ke aplikasi lain dapat dilakukan dengan cukup sederhana yaitu dengan menggunakan Intent, tetapi cara ini tidak dapat diterapkan tanpa interaksi *user*. Cara ini juga diketahui bermasalah dengan Facebook dan Twitter.

Intent intent = new Intent();

intent.setAction(Intent.ACTION\_SEND);

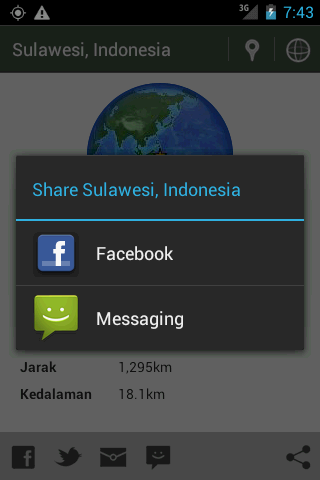
intent.setType("text/plain");

intent.putExtra(Intent.EXTRA\_TEXT, message);

intent.putExtra(Intent.EXTRA\_SUBJECT, subject);

startActivity(Intent.createChooser(intent, title));

Listing 4. *Share* konten menggunakan Intent



Gambar 4. Implementasi Tampilan Share Menggunakan Intent

Gambar 4. Implementasi Tampilan Kirim SMS dan Kirim Email

Gambar 4. Implementasi Tampilan *Share* ke Facebook dan Twitter



Gambar 4. Implementasi Tampilan pada Halaman Facebook



Gambar 4. Implementasi Tampilan pada Halaman Twitter

### Peta Gempa

Gambar 4. Implementasi Tampilan Peta Gempa

(Kiri: Zoom Dibawah 6, Kanan: Zoom 6 Keatas)

Sesuai dengan perancangan, data gempa ditampilkan berbeda pada tingkat zoom tertentu dan gempa yang cukup lama akan ditampilkan lebih transparan dari gempa baru. Label magnitudonya hanya akan ditampilkan jika melebihi atau sama dengan 4 SR dan merupakan gempa baru, serta pada tingkat zoom 6 keatas.

...

DrawValues.*prepare*(...);

...

**int** radius = (zoom >= 6) ? 4 : DrawValues.*markRadius*;

...

canvas.drawCircle(point.x, point.y, radius,   
 DrawValues.*markPaint*);

**int** dmgRadius = (**int**) projection.metersToEquatorPixels(

DrawValues.*dmgMeters*);

**if** (dmgRadius > 3 \* DrawValues.*markRadius*) {

...

canvas.drawCircle(point.x, point.y, dmgRadius,   
 DrawValues.*feelPaint*);

}

**if** (zoom >= 6 && DrawValues.*text* != "") {

...

canvas.drawText(DrawValues.*text*, x, y, DrawValues.*textPaint*);

}

Listing 4. Potongan Program Menggambar Lokasi Gempa

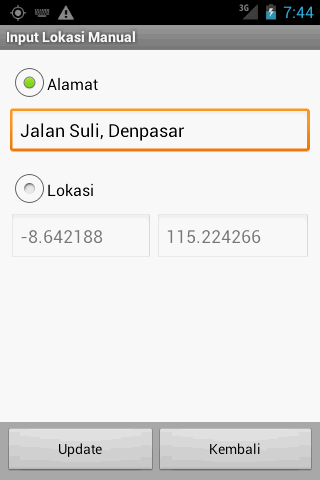


Gambar 4. Implementasi Tampilan *Layers* Peta

### Pengaturan



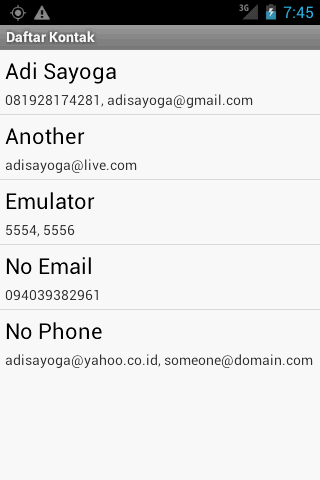
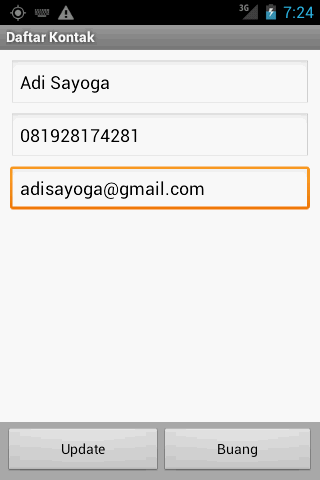
Gambar 4.11 Implementasi Tampilan Pengaturan



Gambar 4.12 Implementasi Tampilan Input Lokasi Manual



Gambar 4.13 Implementasi Tampilan Template Pesan

Gambar 4.14 Implementasi Tampilan Daftar Kontak

(Kiri: Daftar, Kanan: Tambah/Ubah Data)



Gambar 4.15 Implementasi Tampilan *Social* *Connect*

## Pengujian Sistem

Pengujian merupakan serangkaian aktivitas untuk mengintegrasi-kan model desain *test case* perangkat lunak kedalam sederetan langkah-langkah yang direncanakan dengan baik, dan hasilnya merupakan konstruksi perangkat lunak yang berhasil. Tahap pengujian ini akan meliputi verifikasi untuk memastikan apakah program telah terbebas dari kesalahan.

Lingkungan pengembangan Android mencakup *framework* pengujian terintegrasi yang membantu dalam menguji aspek dalam aplikasi. Pengujian Android ini didasarkan pada Junit dan diatur dalam project tersendiri. Fitur kunci dari *framework* pengujian Android adalah kelas *test case* komponen-khusus diantaranya: pengujian *activity*, pengujian *content provider*, dan pengujian *service*. Android SDK juga menyediakan *monkeyrunner*, sebuah perangkat pengujian API menggunakan program Python, dan UI/*Application Exerciser Monkey*, sebuah alat berbasis *command-line* untuk pengujian stres UI dengan mengirimkan event acak ke perangkat.

Karena keterbatasan pengetahuan dan waktu, pada tugas akhir ini tidak menggunakan *tools* diatas. Pengujian dilakukan dengan manual dan mengamati *output* log pada LogCat. Dalam pengujian ini, sebagai simulasi, digunakan:

1. server <http://10.0.2.2> (*localhost*),
2. Minimal magnitudo adalah 3SR, minimal magnitudo notifikasi 5SR, minimal magnitudo mengirim SMS 7SR, dan minimal magnitudo mengirim Email, *share* ke Facebook dan *share* ke Twitter masing-masing 6SR.
3. Nomor telepon pada daftar kontak 5556, dan alamat email [adisayoga@gmail.com](mailto:adisayoga@gmail.com).

Berikut adalah contoh data server:

Src,Eqid,Version,Datetime,Lat,Lon,Magnitude,Depth,NST,Region

us,b00082nd,6,"Monday, February 20, 2012 02:28:17 UTC",1.8307,99.5828,6.2,188.60,155,"northern Sumatra, Indonesia"

ak,10416900,1,"Monday, February 20, 2012 01:43:46 UTC",60.9023,  
-146.8564,2.8,23.60,38,"Southern Alaska"

us,b00082mr,6,"Monday, February 20, 2012 00:50:03 UTC",38.4286,43.3762,3.1,10.20,30,"eastern Turkey"

us,b00082mp,7,"Monday, February 20, 2012 00:35:53 UTC",-9.0155,-109.5065,4.8,10.00,85,"central East Pacific Rise"

us,b00082mm,6,"Monday, February 20, 2012 00:33:49 UTC",  
-3.1092,100.8862,7.6,47.30,37,"Kepulauan Mentawai region, Indonesia"

us,b00082md,7,"Sunday, February 19, 2012 23:33:08 UTC",-17.6143,  
-178.7346,5.1,552.70,138,"Fiji region"

Listing 4. Data Gempa pada Server

Saat pertama kali dijalankan, output pada LogCat sebagai berikut:

EarthquakeModel: Menghapus data...

EarthquakeModel: Data lama dihapus 0 items

EarthquakeService: Merefresh data... Last update=01-21 12:37:29

UsgsSource: Mendapatkan data dari server...

UsgsSource: Magnitude=3.0 interval=2592001163, url dipilih: http://10.0.2.2/earthquake/eqs7day-M2.5.txt

EarthquakeListActivity: Tidak terdapat data, menyembunyikan list

UsgsSource: Selesai mengambil data: 6 items

EarthquakeService: Data pada server: 6 items

EarthquakeService: Terdapat data baru: 6 items

EarthquakeService: Menyimpan data...

EarthquakeService: Mengirim notifikasi...

EarthquakeNotification: Alert sent: M7.6 Kepulauan Mentawai region, Indonesia, dan 2 lainnya

EarthquakeService: Share ke Facebook...

EarthquakeFacebook: Post sent: Lokasi (1° 49' 50.0"LU, 99° 34' 58.0"BT), Kedalaman 188.6km, Jarak 2,086.6km dari Jalan Suli, Denpasar

EarthquakeFacebook: Post sent: Lokasi (-3° 6' 33.0"LS, 100° 53' 10.0"BT), Kedalaman 47.3km, Jarak 1,701km dari Jalan Suli, Denpasar

EarthquakeService: Share ke Twitter...

EarthquakeTwitter: Login authorized user...

EarthquakeTwitter: Post sent: Gempa 2 jam lalu, 10:28 AM M6.2 northern Sumatra, Indonesia (1.8307LU, 99.5828BT), Gempa 4 jam lalu 08:33 AM M7.6 Kepulauan Mentawai region, Ind...

EarthquakeService: Mengirim Email...

ContactModel: Mengambil data dari provider...

ContactModel: Selesai mengambil data, 1 items

EarthquakeMail: Pesan telah dikirim ke: adisayoga@gmail.com

EarthquakeMail: Telah terjadi gempa:

EarthquakeMail: 2 jam lalu, 10:28 AM M6.2 northern Sumatra, Indonesia (1.8307LU, 99.5828BT) Kedlmn 188.6km Jarak 2,086.6km

EarthquakeMail: 4 jam lalu M7.6 Kepulauan Mentawai region, Indonesia (3.1092LS, 100.8862BT) Kedlmn 47.3km Jarak 1,701km

EarthquakeService: Mengirim sms...

ContactModel: Mengambil data dari provider...

ContactModel: Selesai mengambil data, 1 items

EarthquakeSMS: Pesan telah dikirim ke: 5556

EarthquakeSMS: Telah terjadi gempa:

EarthquakeSMS: 4 jam lalu, 08:33 AM M7.6 Kepulauan Mentawai region, Indonesia (3.1092LS, 100.8862BT) Kedlmn 47.3km Jarak 1,701km

EarthquakeReceiver: action=new\_quake\_found

EarthquakeListActivity: Handler: message.what=1

EarthquakeModel: Mengambil data dari provider...

EarthquakeModel: Selesai mengambil data, 5 items

EarthquakeListActivity: Terdapat data, memperlihatkan list

Listing 4. Output LogCat *Start* Awal

Kemudian pada saat tombol *refresh* ditekan:

RefreshReceiver: action=refresh

EarthquakeModel: Menghapus data...

EarthquakeModel: Data lama dihapus 0 items

EarthquakeService: Merefresh data... Last update=02-20 12:37:31

UsgsSource: Mendapatkan data dari server...

UsgsSource: Magnitude=3.0 interval=38158, url dipilih: http://10.0.2.2/earthquake/eqs1hour-M1.txt

UsgsSource: Selesai mengambil data: 0 items

EarthquakeService: Tidak ada data yang perlu di-update

EarthquakeReceiver: action=no\_new\_quake

Listing 4. Output LogCat *Refresh Data*

# PENUTUP

## Kesimpulan

Aplikasi mengecek gempa baru ke situs USGS secara periodik. Pengecekan dilakukan pada *background service* sehingga dapat berjalan kapan saja tanpa harus membuka aplikasi. Jika terdapat gempa besar, maka *service* akan mengirim pemberitahuan baik berupa notifikasi status bar, mengirim SMS, mengirim Email, dan/atau *posting* ke Facebook dan Twitter sesuai dengan pengaturan.

*Service* juga mengirim *broadcast* setiap kali pengecekan. Pada *activity* daftar gempa, saat menerima *broadcast* gempa baru, *list* akan langsung di-*update* sehingga menampilkan gempa terbaru. *Broadcast* ini hanya akan diterima jika aplikasi berada di permukaan, artinya jika aplikasi tidak aktif, tidak ada yang menerima *broadcast*.

Aplikasi dikhususkan/ditargetkan untuk sistem operasi Android. Android dipilih karena perkembangannya yang begitu pesat dan yang tidak kalah pentingya, Android merupakan proyek *open* source. Pengembangan aplikasi ini menggunakan bahasa pemrograman Java dengan *tools* Eclipse IDE. Walaupun dimungkinkan untuk menggunakan IDE lain, Eclipse mendapat dukungan penuh dari Google.

## Saran

Aplikasi monitoring gempa yang dikembangkan dalam tugas akhir ini belum cukup sempurna. Beberapa hal yang perlu ditingkatkan adalah:

1. Dalam mendapatkan data terbaru, aplikasi ini mengecek data ke *server* secara periodik. Sebenarnya ada cara yang lebih baik, salah satunya dengan menggunakan C2DM (*Cloud to Device Messaging*), semacam *push message*, yaitu data akan di-*update* hanya jika terdapat data baru pada *server*.
2. Saat ini aplikasi dirancang untuk smartphone, kedepannya dapat dikembangkan untuk mendukung *form factor* tablet dan/atau lainnya.
3. Android sudah menyediakan *tools* pengujian terintegrasi. Aplikasi perlu dikembangkan agar menggunakan *tools* ini, sehingga pengujian aplikasi dapat lebih baik.
4. Pada beberapa *activity* hanya menyediakan fitur dasar sehingga perlu ditingkatkan. Misalnya mengirim SMS dan Email perlu ditambahkan fitur *auto-complete* saat memasukkan nomor telephone atau Email, dan SMS yang dikirim perlu disimpan ke *database* (*database* kontak telepon), sehingga *user* tahu apa saja yang sudah dikirim.

# DAFTAR PUSTAKA

Burnette, Ed. 2010. *Hello, Android: Introducing Google’s Mobile Development Platform, Third Edition*. United States of America: Pragmatic Programmers

Brandon, Dan. 2006. *Project Management for Modern Information Systems*. United States of America: IRM Press

Meier, Reto. 2010. *Proffessional Androidtm 2 Application Development*. United States of America: Wiley Publishing.

Holdener, Anthony T. 2011. *HTML 5 Geolocation*. United States of America: O’Reilly.

Utami, Endah Tri. 2011. *Kupas Tuntas Android dari Nol Sampai Mahir*. Jakarta: Gudang Ilmu.

Hamilton, Kim, dan Russell Miles. 2006. *Learning UML 2.0*. United States of America: O’Reilly.

U.S. Geological Survey Earthquake Hazards Program. http://earthquake. usgs.gov. Diakses tanggal 20 Juni 2011 pukul 9.50 Wita.

Google I/O 2011. <http://www.google.com/events/io/2011>. Diakses tang-gal 20 Juni 2011 pukul 9.25 Wita.

Android Developers. <http://developer.android.com>. Diakses tanggal 20 Juni 2011 pukul 9.20 Wita.

Facebook Developers. <http://developers.facebook.com>. Diakses tanggal 27 Juni 2011 pukul 10.15 Wita.

Twitter Developers. <https://devs.twitter.com>. Diakses tanggal 23 Desem-ber pukul 15.04 Wita.

Twitter4J - A Java library for the Twitter API. <http://twitter4j.org>. Diakses tanggal 23 Desember pukul 15.20 Wita.

javamail-android. <http://code.google.com/p/javamail-android/>. Diakses tanggal 24 Desember pukul 10.00 Wita.

# LAMPIRAN KODE PROGRAM

## Service

**public** **class** EarthquakeService **extends** IntentService {

...

@Override

**protected** **void** onHandleIntent(Intent intent) {

*// Hapus data lama*

deleteOldQuakes(prefs.getMaxAge());

**try** {

*// Mendapatkan data dari USGS*

**long** lastUpdate = prefs.getLastUpdate();

**float** minMagnitude = prefs.getMinMagnitude();

List<EarthquakeDTO> quakes = UsgsSource.*read*(lastUpdate,   
 minMagnitude);

*// Kita sudah selesai mendapatkan data, simpan terakhir* *// kali diupdate*

prefs.setLastUpdate(System.*currentTimeMillis*());

**if** (quakes != **null** && quakes.size() > 0) {

*// Terdapat data pada server, filter data ini sehingga* *// data yang didapat merupakan data yang benar-benar*

*// baru*

quakes = getNewQuakes(quakes);

}

**if** (quakes != **null** && quakes.size() > 0) {

*// Terdapat data, simpan ke provider, beritahukan ada* *// gempa baru (jika sesuai dengan minimal magnitudo*

*// pengaturan), dan kirim broadcast terdapat data baru*

addNewQuakes(quakes);

notifyNewQuake(quakes);

sendBroadcast(**new** Intent(*NEW\_QUAKE\_FOUND*));

} **else** {  
 *// Tidak ada data baru, kirim broadcast tidak ada data baru*

sendBroadcast(**new** Intent(*NO\_NEW\_QUAKE*));

}

} **catch** (IOException e) {

*// Terdapat error, kirim broadcast jaringan error*

sendBroadcast(**new** Intent(*NETWORK\_ERROR*));

}

}

...

}

**private** **void** notifyNewQuake(List<EarthquakeDTO> quakes) {

...

*// Variable untuk menyimpan data*

quakeNotify = **null**;

quakesFacebook = **new** ArrayList<EarthquakeDTO>();

quakeTwitter = **new** ArrayList<EarthquakeDTO>();

quakeMail = **new** ArrayList<EarthquakeDTO>();

quakeSms = **new** ArrayList<EarthquakeDTO>();

**for** (EarthquakeDTO quake : quakes) {

**float** magnitude = quake.magnitude;

**if** (magnitude < prefMinMag) **continue**;

*// Jarak dari masing-masing gempa pada range regional/tidak*

**boolean** inRange = **false**;

**if** (location != **null**) {

**float** distance = quake.getLocation().distanceTo(location);

inRange = distance <= prefRange;

}

**if** (isNotify) { *// Notifikasi, prioritas regional*

**if** (inRange && notifyMinMagReg <= magnitude) {

quakeCount++;

**if** (lastMagReg < magnitude) {

quakeNotify = quake;

lastMagReg = magnitude;

}

} **else** **if** (notifyMinMagGlobal <= magnitude) {

quakeCount++;

**if** (lastMagGlobal < magnitude) {

*// Jika belum ada notifikasi regional maka ini yang*

*// dipakai, jika tidak maka biarkan yang regional*

**if** (lastMagReg == 0) quakeNotify = quake;

lastMagGlobal = magnitude;

}

}

}

*// Share ke Facebook*

**if** (isFacebookSend) addQuakeNotify(quakesFacebook, quake,   
 inRange, magnitude, facebookMinMagReg, facebookMinMagGlobal);

*// Share ke Twitter*

**if** (isTwitterSend) addQuakeNotify(quakesTwitter, quake,   
 inRange, magnitude, twitterMinMagReg, twitterMinMagGlobal);

*// Kirim email*

**if** (isMailSend) addQuakeNotify(quakesMail, quake, inRange,   
 magnitude, mailMinMagReg, mailMinMagGlobal);

*// SMS*

**if** (isSmsSend) addQuakeNotify(quakesSms, quake, inRange,   
 magnitude, smsMinMagReg, smsMinMagGlobal);

}

*// Tampilkan notifikasi, kirim pesan, dan/atau share*

**if** (quakeCount > 0)

sendNotification(quakeNotify, quakeCount);

**if** (quakesFacebook != **null** && quakesFacebook.size() > 0)   
 shareToFacebook(quakesFacebook);

**if** (quakesTwitter != **null** && quakesTwitter.size() > 0)   
 shareToTwitter(quakesTwitter);

**if** (quakesMail != **null** && quakesMail.size() > 0)   
 sendMail(quakesMail);

**if** (quakesSms != **null** && quakesSms.size() > 0)   
 sendSms(quakesSms);

}

**private** **void** sendNotification(EarthquakeDTO quake, **int** quakeCount) {  
 **boolean** isFlash = prefs.isNotifyFlash();

**boolean** isAlert = prefs.isNotifyAlert();

Uri alertSound = prefs.getNotifyAlertSound();

**boolean** isVibrate = prefs.isNotifyVibrate();

EarthquakeNotification notifier = **new** EarthquakeNotification(

**this**, quake, quakeCount, isAlert, alertSound, isFlash, isVibrate);

notifier.alert();

}

**private** **void** shareToFacebook(List<EarthquakeDTO> quakes) {

**if** (!facebook.isSessionValid()) **return**;

*// Post setiap list quake*

**boolean** postSent = **false**;

**for** (EarthquakeDTO quake : quakes) {

Bundle params = facebook.genereateParams(quake, **null**, *location*);

postSent |= facebook.postMessage(params);

}

showMessage((postSent) ? R.string.*facebook\_post\_sent*

: R.string.*facebook\_post\_fail*);

}

**private** **void** shareToTwitter(**final** List<EarthquakeDTO> quakes) {

twitter.login(**null**, **new** EarthquakeTwitter.AuthListener() {

@Override

**public** **void** onAuthComplete() {

String message = "";

**for** (EarthquakeDTO quake : quakes) {

**if** (message != "") message += ", ";

message += twitter.getPostMessage(quake, location);

}

*// Message maksimal 140 karakter*

message = message.substring(0, 137) + "...";

**boolean** postSent = twitter.postMessage(message);

showMessage((postSent) ? R.string.twitter\_post\_sent

: R.string.twitter\_post\_fail);

}

@Override

**public void** onAuthFail() {

showMessage(R.string.twitter\_post\_fail);

}

});

}

**private** **void** sendMail(List<EarthquakeDTO> quakes) {

**try** {

Context context = this;

mail.setFrom(prefs.getMailUsername());

ContactModel table = new ContactModel(this);

String[] mails = table.getMails();

mail.setTo(mails);

mail.setSubject(context.getString(R.string.app\_name));

String message = EarthquakeTemplate.getInstance(context).getMessage(  
 prefs.getMailTemplate(context),

prefs.getMailTemplateDetail(context),

quakes, location);

mail.setBody(message);

boolean mailSent = mail.send();

showMessage((mailSent) ? R.string.mail\_sent : R.string.mail\_fail);

} **catch** (Exception e) {

showMessage(R.string.*mail\_fail*);

}

}

**private** **void** sendSms(List<EarthquakeDTO> quakes) {

Context context = **this**;

String message = EarthquakeTemplate.*getInstance*(context)  
 .getMessage(prefs.getSmsTemplate(context),   
 prefs.getSmsTemplateDetail(context), quakes, *location*);

ContactModel table = **new** ContactModel(**this**);

String[] phones = table.getPhones();

EarthquakeSms sms = **new** EarthquakeSms(**this**);

List<String> phonesSent = sms.sendTextMessage(phones, message,

EarthquakeSms.*SPLIT\_SMS\_MESSAGE*);

**boolean** smsSent = phonesSent != **null** && phonesSent.size() > 0;

showMessage((smsSent) ? R.string.*sms\_sent* : R.string.*sms\_fail*);

}

## Daftar Gempa

Layout *action bar*:

<LinearLayout style=*"@style/ActionBar"*>

<TextView style=*"@style/ActionBarTitle"* ... />

<ImageView style=*"@style/ActionBarSeparator"* />

<LinearLayout android:layout\_width=*"wrap\_content"* ... />

<ImageView style=*"@style/ActionBarButton"*   
 android:id=*"@+id/show\_map"* ... />

<ImageView style=*"@style/ActionBarSeparator"* />

<ImageView style=*"@style/ActionBarButton"*   
 android:id=*"@+id/refresh"* ... />

</LinearLayout>

</LinearLayout>

Layout daftar:

<LinearLayout android:id=*"@+id/no\_quake"* ...>

<TextView android:text=*"@string/no\_quake\_sum"* ... />

<TextView android:text=*"@string/no\_quake\_msg"* ... />

</LinearLayout>

<ListView android:id=*"@+id/listview"* ... />

Kode program perlihatkan/sembunyikan list:

**private** **void** updateQuakes(**boolean** finishLoading) {

...

*// Perlihatkan atau sembunyikan view*

**if** (quakes.size() > 0) {

noQuake.setVisibility(View.*GONE*);

list.setVisibility(View.*VISIBLE*);

} **else** {

noQuake.setVisibility(View.*VISIBLE*);

list.setVisibility(View.*GONE*);

}

...

}

Menampilkan ke list:

**public** **class** EarthquakeReceiver **extends** BroadcastReceiver {

...

@Override

**public** **void** onReceive(Context context, Intent intent) {

String action = intent.getAction();  
 **if** (action.equals(*NEW\_QUAKE\_FOUND*)) {

handler.sendEmptyMessage(*NEW\_QUAKE\_FOUND\_WHAT*);

} **else** **if** (action.equals(*NO\_NEW\_QUAKE*)) {  
 handler.sendEmptyMessage(*NO\_NEW\_QUAKE\_WHAT*);

} **else** **if** (action.equals(*NETWORK\_ERROR*)) {  
 handler.sendEmptyMessage(*NETWORK\_ERROR\_WHAT*);  
 }

}

}

**public** **class** EarthquakeListActivity **extends** Activity {

...

**private** **final** Handler handler = **new** Handler() {

@Override

**public** **void** handleMessage(Message message) {

**switch** (message.what) {

**case** *NEW\_QUAKE\_FOUND\_WHAT*:

updateQuakes(**true**);

**break**;

**case** *NO\_NEW\_QUAKE\_WHAT*:

...

**break**;

**case** *NETWORK\_ERROR\_WHAT*:

...

showDialog(*NETWORK\_ERROR\_DIALOG*);

**break**;

}

}

};

**private** **void** UpdateQuakes(**boolean** finishLoading) {

**new** AsyncTask<Boolean, Void, List<EarthquakeDTO>>() {

**private** **boolean** finishLoading;

@Override  
 **protected** List<EarthquakeDTO> doInBackground(Boolean... params) {

**this**.finishLoading = params[0];

*// Load data dari provider*

EarthquakeModel table = **new** EarthquakeModel(  
 EarthquakeListActivity.**this**);

List<EarthquakeDTO> newQuakes = table.getMatchQuakes(

prefs.getMinMagnitude(), prefs.getMaxAge());

**return** newQuakes;

}

@Override

**protected** **void** onPostExecute(List<EarthquakeDTO> newQuakes) {

*// Tambahkan ke quakes*

quakes.clear();

**if** (newQuakes != **null**) {

**for** (EarthquakeDTO quake : newQuakes) {

quakes.add(quake);

}

}

...

adapter.notifyDataSetChanged();

}

}.execute(finishLoading);

}

}

Context menu:

**private** **void** setupListAdapter() {

...

list.setOnItemLongClickListener(itemLongClickListener);

}

**private** **final** OnItemLongClickListener itemLongClickListener =

**new** AdapterView.OnItemLongClickListener() {

@Override

**public** **boolean** onItemLongClick(AdapterView<?> parent,   
 View view, **int** position, **long** id) {

**final** EarthquakeDTO selectedQuake = quakes.get(position);

Builder builder = **new** Builder(EarthquakeListActivity.**this**);

builder.setTitle(selectedQuake.region);

builder.setItems(R.array.*list\_options*,   
 **new** OnClickListener() {

@Override  
 **public** **void** onClick(DialogInterface dialog, **int** which) {

**switch** (which) {

**case** 0: *// Tampilkan detail gempa bumi*

showQuakeDetails(selectedQuake);

**break**;

**case** 1: *// Tampilkan pada peta*

showQuakeMap(selectedQuake);

**break**;

**case** 2: *// Share ke Facebook*

shareToFacebook(selectedQuake);

**break**;

**case** 3: *// Share ke Twitter*

shareToTwitter(selectedQuake);

**break**;

**case** 4: *// Kirim email*

sendMail(selectedQuake);

**break**;

**case** 5: *// Kirim sms*

sendSms(selectedQuake);

**break**;

**case** 6: *// Share lainnya*

shareOthers(selectedQuake);

**break**;

}

}

});

builder.show();

**return** **true**;

}

};

*Quick action*:

**private** **void** setupQuickAction() {

quickAction = **new** QuickAction(context);

Resources resources = context.getResources();

*// Share ke Facebook*

QuickactionItem facebookAction = **new** QuickactionItem();

facebookAction.setIcon(resources.getDrawable(  
 R.drawable.*ic\_quickaction\_facebook*));

facebookAction.setTitle((String) context.getText(  
 R.string.*share\_to\_facebook\_short*));

quickAction.addActionItem(facebookAction);

*// Share ke Twitter*

QuickactionItem twitterAction = **new** QuickactionItem();

twitterAction.setIcon(resources.getDrawable(  
 R.drawable.*ic\_quickaction\_twitter*));

twitterAction.setTitle((String) context.getText(  
 R.string.*share\_to\_twitter\_short*));

quickAction.addActionItem(twitterAction);

*// Kirim email*

QuickactionItem mailAction = **new** QuickactionItem();

mailAction.setIcon(resources.getDrawable(  
 R.drawable.*ic\_quickaction\_mail*));

mailAction.setTitle((String) context.getText(  
 R.string.*send\_mail\_short*));

quickAction.addActionItem(mailAction);

*// Kirim SMS*

QuickactionItem smsAction = **new** QuickactionItem();

smsAction.setIcon(resources.getDrawable(  
 R.drawable.*ic\_quickaction\_sms*));

smsAction.setTitle((String) context.getText(  
 R.string.*send\_sms\_short*));

quickAction.addActionItem(smsAction);

*// Share lainnya*

QuickactionItem shareAction = **new** QuickactionItem();

shareAction.setIcon(resources.getDrawable(  
 R.drawable.*ic\_quickaction\_detail*));

shareAction.setTitle((String) context.getText(  
 R.string.*others*));

quickAction.addActionItem(shareAction);

*//setup the action item click listener*

quickAction.setOnActionItemClickListener(  
 **new** OnActionItemClickListener() {

@Override

**public** **void** onItemClick(**int** quickActionPos) {

quickActionListener.onItemClick(quakePos, quickActionPos);

}

});

}

## Detail Gempa

*Layout* (*portrait*):

<RelativeLayout ...>

<LinearLayout style=*"@style/ActionBar"*>

<TextView android:id=*"@+id/title"* .../>

<ImageView style=*"@style/ActionBarSeparator"* />

<LinearLayout style=*"@style/ActionBarButtonLayout"*>

<ImageView android:id=*"@+id/show\_map"* .../>

<ImageView style=*"@style/ActionBarSeparator"* />

<ImageView android:id=*"@+id/usgs\_detail"* .../>

</LinearLayout>

</LinearLayout>

<ScrollView ...>

<RelativeLayout ...>

<ImageView android:id=*"@+id/map"* ... />

<ImageView android:id=*"@+id/map\_mask"* ... />

<ProgressBar android:id=*"@+id/map\_progress"* ... />

<ImageView style=*"@style/DividerHorizontal"* ... />

<TableLayout android:id=*"@+id/map\_progress"* ...>

<TableRow ...>

<TextView android:text=*"@string/region"* ... />

<TextView android:id=*"@+id/region"* ... />

</TableRow>

<TableRow ...>

<TextView android:text=*"@string/location"* ... />

<TextView android:id=*"@+id/location"* ... />

</TableRow>

<TableRow ...>

<TextView android:text=*"@string/magnitude"* ... />

<TextView android:id=*"@+id/magnitude"* ... />

</TableRow>

<TableRow ...>

<TextView android:text=*"@string/quake\_time"* ... />

<TextView android:id=*"@+id/date"* ... />

</TableRow>

<TableRow ...>

<TextView android:text=*"@string/distance"* ... />

<TextView android:id=*"@+id/distance"* ... />

</TableRow>

<TableRow ...>

<TextView android:text=*"@string/depth"* ... />

<TextView android:id=*"@+id/depth"* ... />

</TableRow>

</TableLayout>

</RelativeLayout>

</ScrollView>

<LinearLayout ...>

<LinearLayout ...>

<ImageButton android:id=*"@+id/share\_to\_facebook"* ... />

<ImageButton android:id=*"@+id/share\_to\_twitter"*... />

<ImageButton android:id=*"@+id/send\_mail"*... />

<ImageButton android:id=*"@+id/send\_sms"*... />

</LinearLayout>

<ImageButton android:id=*"@+id/share\_others"*... />

</LinearLayout>

</RelativeLayout>

Kode program:

**public** **class** EarthquakeDetailActivity **extends** Activity {

...

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

...

Intent intent = getIntent();

**if** (intent.hasExtra(EarthquakeColumns.*TABLE\_NAME*)) {

quake = (EarthquakeDTO) intent.getExtras().get(  
 EarthquakeColumns.*TABLE\_NAME*);

bindView(quake, theme);

}

}

**private** **void** bindView(EarthquakeDTO quake, EarthquakeTheme theme) {

**if** (quake == **null**) **return**;

titleTextView.setText(quake.region);

TextView regionTextView = (TextView) findViewById(R.id.*region*);

regionTextView.setText(quake.region);

TextView locationTextView = (TextView) findViewById(R.id.*location*);

locationTextView.setText(LocationUtils.*getInstance*(**this**)  
 .formatLocation(quake.latitude, quake.longitude, **false**));

TextView magnitudeTextView = (TextView) findViewById(  
 R.id.*magnitude*);

magnitudeTextView.setText(Float.*toString*(quake.magnitude) + " "   
 + getString(R.string.*sr*));

TextView dateTextView = (TextView) findViewById(R.id.*date*);

dateTextView.setText(TimeUtils.*getInstance*(**this**).toHumanReadable(  
 quake.time));

Unit unit = prefs.getUnit();

**float** distance = 0;

**if** (location != **null**) distance = quake.getLocation().distanceTo(  
 location);

distanceTextView.setText(unit.formatNumber(distance, EarthquakeDTO

.*FRACTION\_DISTANCE*));

TextView depthTextView = (TextView) findViewById(R.id.*depth*);

depthTextView.setText(unit.formatNumber(quake.depth, EarthquakeDTO

.*FRACTION\_DEPTH*));

loadGlobe(UsgsSource.*getGlobeURL*(quake));

}

## Peta Gempa

Menggambar lokasi gempa:

@Override

**public** **void** draw(Canvas canvas, MapView mapView, **boolean** shadow) {

**if** (shadow) **return**;

**for** (**int** i = items.size() - 1; i >= 0; i--) {

drawQuake(canvas, mapView, quakes.get(i), theme);

}

}

**private** **void** drawQuake(Canvas canvas, MapView mapView, EarthquakeDTO   
 quake, EarthquakeTheme theme) {

Projection projection = mapView.getProjection();

projection.toPixels(quake.getPoint(), point);

**int** height = mapView.getHeight();

**int** width = mapView.getWidth();

**int** scrollX = mapView.getScrollX();

**int** scrollY = mapView.getScrollY();

**if** (point.x < scrollX || point.x > scrollX + width) **return**;

**if** (point.y < scrollY || point.y > scrollY + height) **return**;

DrawValues.*prepare*(theme, *TEXT\_SIZE*, quake.magnitude, quake.time);

**int** zoom = mapView.getZoomLevel();

**int** radius = (zoom >= 6) ? 4 : DrawValues.*markRadius*;

canvas.drawCircle(point.x, point.y, radius + 2,   
 DrawValues.*markGlowPaint*);

canvas.drawCircle(point.x, point.y, radius + 1,   
 DrawValues.*markOutlinePaint*);

canvas.drawCircle(point.x, point.y, radius,   
 DrawValues.*markPaint*);

**int** dmgRadius = (**int**) projection.metersToEquatorPixels(

DrawValues.*dmgMeters*);

**if** (dmgRadius > 3 \* DrawValues.*markRadius*) {

canvas.drawCircle(point.x, point.y, dmgRadius,   
 DrawValues.*feelOutlinePaint*);

canvas.drawCircle(point.x, point.y, dmgRadius,   
 DrawValues.*feelPaint*);

}

**if** (zoom >= 6 && DrawValues.*text* != "") {

**int** x = point.x;

**int** y = point.y + radius + *TEXT\_SIZE* + *PADDING* \* 3;

canvas.drawText(DrawValues.*text*, x, y, DrawValues.*textOutlinePaint*);

canvas.drawText(DrawValues.*text*, x, y, DrawValues.*textPaint*);

}

}

*Balloon view*:

@Override

**protected** **boolean** onTap(**int** index) {

**if** (balloonView == **null**) {

balloonView = **new** EarthquakeOverlayView(mapView.getContext(),   
 viewOffset);

mapView.addView(balloonView);

...

}

...

GeoPoint point = createItem(index).getPoint();

MapView.LayoutParams params = **new** MapView.LayoutParams(  
 LayoutParams.*WRAP\_CONTENT*, LayoutParams.*WRAP\_CONTENT*, point,   
 LayoutParams.*BOTTOM\_CENTER*);

params.mode = LayoutParams.*MODE\_MAP*;

balloonView.setLayoutParams(params);

balloonView.setData(quakes.get(index), location);

List<Overlay> overlays = mapView.getOverlays();

**if** (overlays.size() > 1) hideOtherBalloons(overlays);

balloonView.setVisibility(View.*VISIBLE*);

controller.animateTo(point);

**return** **true**;

}

*Layer*:

@Override

**public** **boolean** onOptionsItemSelected(MenuItem item) {

**super**.onOptionsItemSelected(item);

Intent intent;

**switch** (item.getItemId()) {  
 **case** R.id.*layers*:

AlertDialog.Builder layersBuilder = **new** AlertDialog.Builder(**this**);

layersBuilder.setTitle(R.string.*menu\_layers*);

layersBuilder.setMultiChoiceItems(R.array.*layers*, **new** **boolean**[] {

layers[*LAYER\_SATTELITE\_INDEX*],

layers[*LAYER\_STREET\_VIEW\_INDEX*],

layers[*LAYER\_TRAFFIC\_INDEX*]

},

**new** DialogInterface.OnMultiChoiceClickListener() {  
 @Override

**public** **void** onClick(DialogInterface dialog, **int** which,   
 **boolean** isChecked) {

layers[which] = isChecked;

**switch** (which) {

**case** *LAYER\_SATTELITE\_INDEX*:

mapView.setSatellite(isChecked);

prefs.setLayerUsed(*LAYER\_SATTELITE\_KEY*, isChecked);

**break**;

**case** *LAYER\_STREET\_VIEW\_INDEX*:

mapView.setStreetView(isChecked);

prefs.setLayerUsed(*LAYER\_STREET\_VIEW\_KEY*, isChecked);

**break**;

**case** *LAYER\_TRAFFIC\_INDEX*:

mapView.setTraffic(isChecked);  
 prefs.setLayerUsed(*LAYER\_TRAFFIC\_KEY*, isChecked);

**break**;

}

}

});

layersBuilder.show();

**break**;

...

}

**return** **false**;

}

## Input Lokasi Manual

**private** **void** getFromLocationName(String addressName) {

**new** AsyncTask<String, Void, List<Address>>() {

@Override

**protected** List<Address> doInBackground(String... params) {

**try** {

Geocoder geocoder = **new** Geocoder(getApplicationContext(),

Locale.*getDefault*());

List<Address> addresses = geocoder.getFromLocationName(

params[0], 5);

**return** addresses;

} **catch** (Exception e) {

**return** **null**;

}

}

@Override  
 **protected** **void** onPostExecute(List<Address> result) {  
 updateLocation(result);

}

}.execute(addressName);

}

**private** **void** getFromLocation(**double** latitude, **double** longitude) {  
 **new** AsyncTask<Double, Void, List<Address>>() {

@Override

**protected** List<Address> doInBackground(Double... params) {

**try** {

Geocoder geocoder = **new** Geocoder(ManualLocationActivity.**this**,   
 Locale.*getDefault*());

List<Address> addresses = geocoder.getFromLocation(  
 params[0], params[1], 5);

**return** addresses;

} **catch** (Exception e) {

**return** **null**;

}

}

@Override

**protected** **void** onPostExecute(List<Address> result) {

updateLocation(result);

}

}.execute(latitude, longitude);

}

**private** **void** updateLocation(List<Address> addresses) {  
 **if** (addresses == **null** || addresses.size() == 0) {

showUnknownLocation();

**return**;

}

**if** (addresses.size() == 1) {

*// Hanya terdapat satu alamat, langsung update lokasi*

saveLocation(addresses.get(0));

} **else** {

*// Terdapat alamat lebih dari satu, tampilkan dialog pilihan* *// alamat*

showAddressDialog(addresses);

}

}

## Template Pesan

**private** **void** bindView() {

templateText = (EditText) findViewById(R.id.*template\_text*);

templateText.setText(prefs.getSmsTemplate(**this**));

detailText = (EditText) findViewById(R.id.*detail\_text*);

detailText.setText(prefs.getSmsTemplateDetail(**this**));

}

...

**private** **void** saveSettings() {

prefs.setSmsTemplate(templateText.getText().toString());

prefs.setSmsTemplateDetail(detailText.getText().toString());

}

## Daftar Kontak

**private** **void** updateListContact() {

**new** AsyncTask<Void, Void, List<ContactDTO>>() {  
 @Override

**protected** List<ContactDTO> doInBackground(Void... params) {

*// Load data dari provider*

ContactModel table = **new** ContactModel(ContactActivity.**this**);

List<ContactDTO> newContacts = table.getContacts();

**return** newContacts;

}

@Override

**protected** **void** onPostExecute(List<ContactDTO> newContacts) {

*// Tambahkan ke contacts*

contacts.clear();

**if** (newContacts != **null**) {

**for** (ContactDTO quake : newContacts) {

contacts.add(quake);

}

}

*// Perlihatkan atau sembunyikan view*  
 **if** (contacts.size() > 0) {

noContact.setVisibility(View.*GONE*);

list.setVisibility(View.*VISIBLE*);

} **else** {

noContact.setVisibility(View.*VISIBLE*);

list.setVisibility(View.*GONE*);

}

adapter.notifyDataSetChanged();

}

}.execute();

}

## *Social* *Connect*

**public** **class** SocialConnectActivity **extends** Activity {

...

**private** **void** facebookLogin() {

SessionEvents.AuthListener listener = **new** SessionEvents  
 .AuthListener() {

@Override

**public** **void** onAuthSucceed() {

updateFacebookView();

}

@Override

**public** **void** onAuthFail(String error) {

Toast.*makeText*(FacebookConnectActivity.**this**,   
 R.string.*auth\_fail*, Toast.*LENGTH\_LONG*).show();

loginStatus.setText(R.string.*auth\_fail*);

}

};

SessionEvents.*addAuthListener*(listener);

quakeFb.login(**this**);

}

}

**private** **void** twitterLogin() {

twitter.login(**this**, **new** EarthquakeTwitter.AuthListener() {

@Override

**public** **void** onAuthComplete() {

twitterLoggedIn = **true**;

updateTwitterView();

}

@Override

**public** **void** onAuthFail() {

twitterLoggedIn = **false**;

updateTwitterView();

}

});

}

Source lebih lengkap (menggunakan *version control* SVN):

svn checkout <http://android-earthquake.googlecode.com/svn/trunk/> android-earthquake-read-only

Copyright © 2011 Adi Sayoga.

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

1. <http://developer.android.com/sdk/index.html> [↑](#footnote-ref-1)
2. <https://github.com/facebook/facebook-android-sdk> [↑](#footnote-ref-2)
3. <http://twitter4j.org> [↑](#footnote-ref-3)
4. <http://code.google.com/p/javamail-android/> [↑](#footnote-ref-4)
5. <http://code.google.com/android/maps-api-signup.html> [↑](#footnote-ref-5)
6. <https://www.facebook.com/developers/createapp.php> [↑](#footnote-ref-6)
7. <http://code.google.com/p/openssl-for-windows/downloads/list> [↑](#footnote-ref-7)
8. Source project dapat didapat di <https://github.com/facebook/facebook-android-sdk> [↑](#footnote-ref-8)
9. <https://dev.twitter.com/apps> [↑](#footnote-ref-9)
10. *Copyright*©Lorensius W. L. T <[lorenz@londatiga.net](mailto:lorenz@londatiga.net)> [↑](#footnote-ref-10)