BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Analisis sistem merupakan proses identifikasi dan evaluasi permasalahan yang terdapat dalam suatu sistem sehingga dapat diusulkan perbaikan-perbaikanya. Langkah yang dilakukan seperti analisis masalah, analisis sistem yang sedang berjalan, analisis arsitektur, analisis non-fungsional dan analisis fungsional.

3.1.1 Analisis Masalah

Berdasarkan penelitian yang dilakukan, permasalahan yang terdapat pada domain ini adalah sebagai berikut :

1 Penyampaian informasi melalui *website* dan penyuluhan langsung dinilai kurang optimal karena tidak bisa memberikan informasi secara cepat.
2 Kurangnya media informasi resmi yang digunakan berdampak pada kurang cepatnya berita tentang aktifitas gunungapi menyebar di masyarakat.
3 Diperlukan media informasi yang memberikan informasi letak geografis gunungapi.

3.1.2 Analisis Sistem Yang Sedang Berjalan

Prosedur yang sedang berjalan di Pusat Vulkanologi di bidang pengamatan dan penyelidikan gunungapi dalam menyampaikan informasi adalah melalui website resmi tentang status gunungapi. Selain itu, juga dilakukan penyuluhan langsung kepada masyarakat di kawasan rawan bencana gunungapi untuk memberikan pengarahan tentang kawasan rawan bencana.

Prosedur penyampaian data dasar hingga menjadi informasi adalah sebagai berikut :

1. Prosedur pelaporan data dasar gunungapi
2. Prosedur penetapan kebijakan status gunungapi

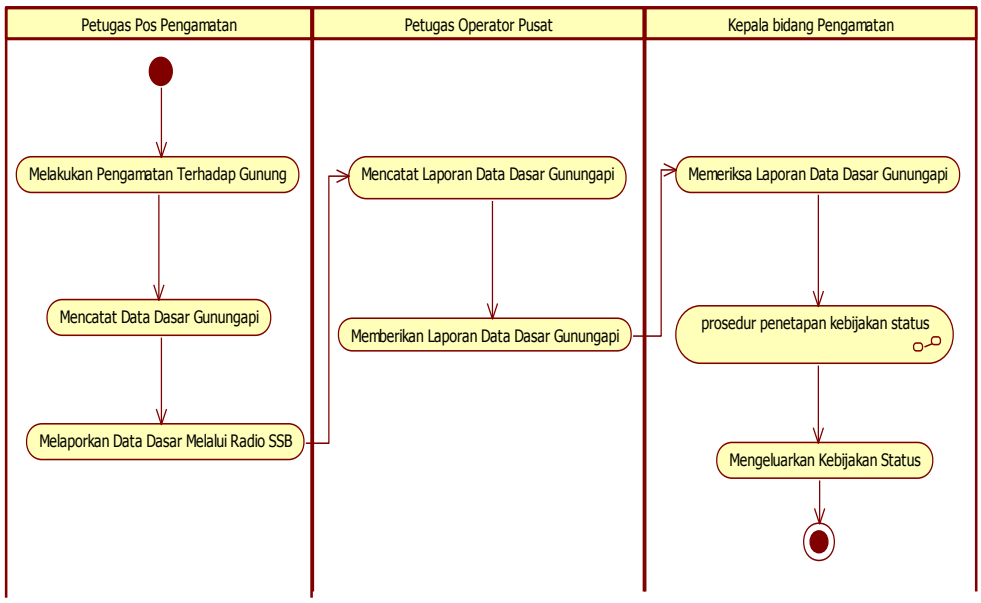
3. Prosedur penyampaian informasi melalui *website*
4. Prosedur penyuluhan kepada masyarakat

3.1.2.1 Prosedur pelaporan data dasar gunungapi

Prosedur pelaporan data dasar gunungapi melibatkan petugas pos dan petugas operator yang menerima laporan dan kepala bidang yang memberikan kebijakan status. Skenario pelaporan data dasar gunungapi adalah sebagai berikut:

1 Petugas pos melakukan pengamatan terhadap gunungapi.
2 Petugas pos mencatat data dasar gunungapi.
3 Petugas pos akan memberikan laporan harian ke pusat vulkanologi melalui radio SSB (*single sideband modulation*).
4 Petugas operator di PVMBG akan menerima laporan dan mencatat data dasar gunungapi sesuai laporan setiap pos pengamatan.
5 Petugas operator akan memberikan laporan data dasar gunungapi kepada kepala bidang.
6 Kepala bidang akan memeriksa laporan data dasar gunungapi.
7 Kepala bidang akan menjalankan prosedur penetapan kebijakan
8 Kepala bidang mengeluarkan kebijakan status gunungapi.

Proses yang terjadi bisa dilihat pada Gambar 3.1



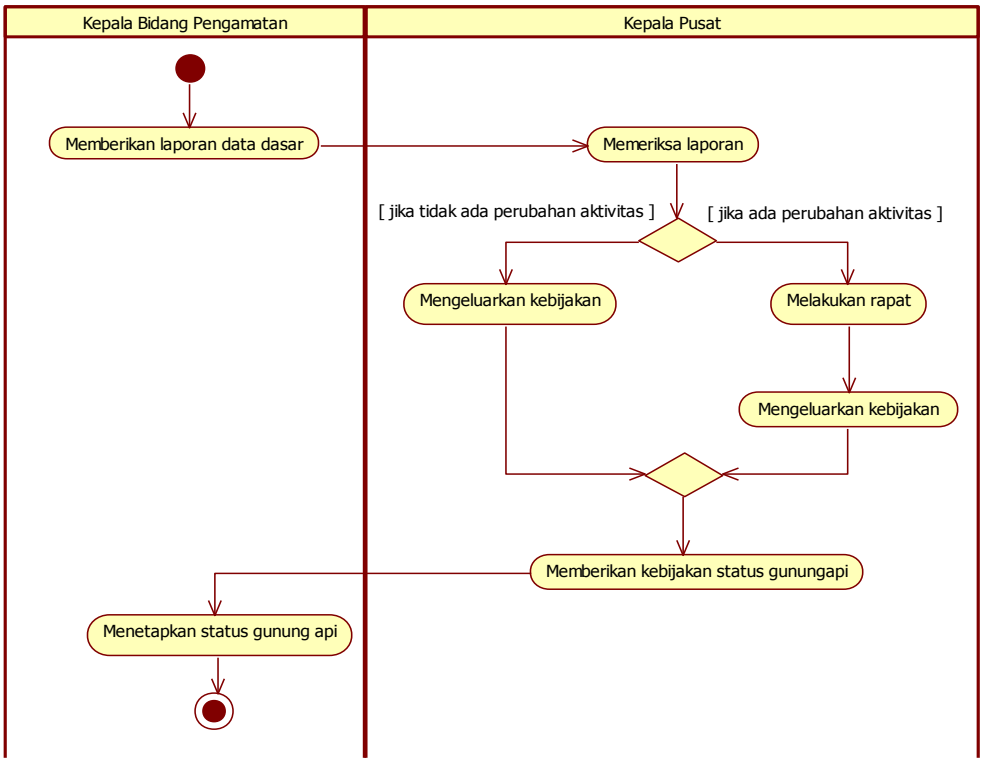
Gambar 3.1 *Activity Diagram* Pelaporan Data Dasar Gunungapi

3.1.2.2 Prosedur Penetapan Status Gunungapi

Status gunungapi ditetapkan oleh Kepala Pusat PVMBG berdasarkan data dasar hasil pengamatan setiap pos. Skenario penetapan status gunungapi oleh pihak PVMBG adalah sebagai berikut:

1. Kepala bidang akan memberikan laporan data dasar kepada kepala pusat
2. Memeriksa laporan data dasar gunungapi.
3. Jika data dasar tidak menunjukan peningkatan atau penurunan aktifitas gunung maka status gunung sama seperti laporan sebelumnya.
4. Jika ada peningkatan aktifitas gunungapi berdasarkan data dasar yang dilaporkan maka kepala pusat akan melakukan rapat bersama staf ahli yang lain sebelum mengeluarkan kebijakan.
5. Kepala bidang akan menetapkan status gunungapi berdasarkan kebijakan kepala pusat.

Proses yang terjadi bisa dilihat pada Gambar 3.2



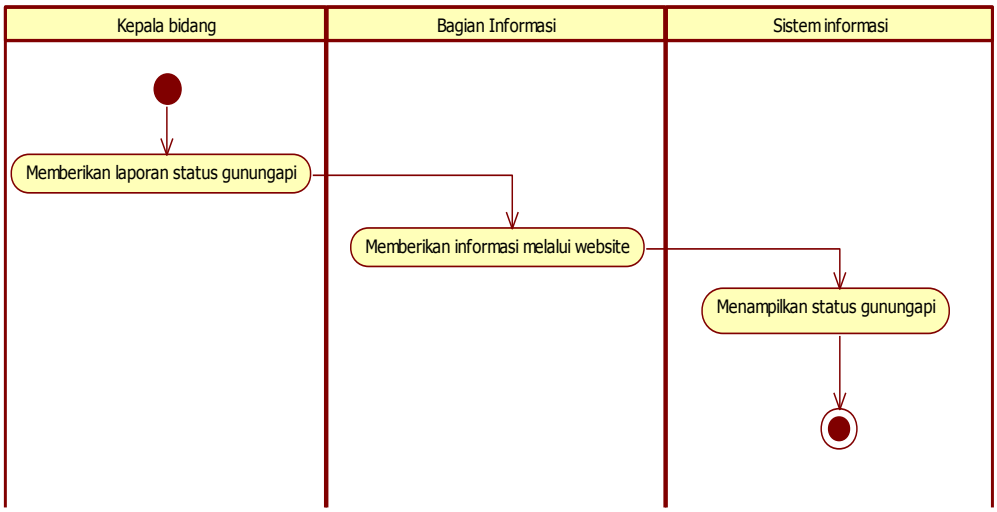
Gambar 3.2 *Activity Diagram* Penetapan Status Gunung

3.1.2.3 Prosedur Penyampaian Informasi Melalui *Website*

Status gunungapi yang telah ditetapkan oleh Kepala Bidang berdasarkan kebijakan kepala pusat akan ditampilkan di *website* resmi PVMBG sebagai informasi bagi masyarakat umum. Skenario yang berjalan adalah sebagai berikut :

1 Kepala bidang akan memberikan laporan kepada bagian informasi
2 Bagian informasi akan menyampaikan informasi kepada publik tentang status gunungapi.
3 *Website* sebagai salah satu media informasi akan menampilkan informasi status gunungapi.

Proses yang terjadi digambarkan pada Gambar 3.3



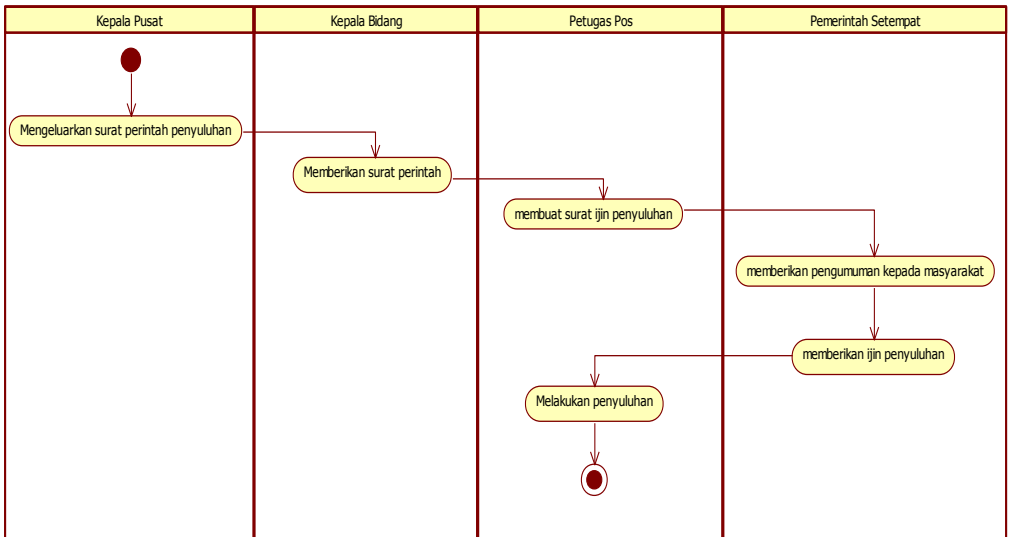
Gambar 3.3 *Activity Diagram* Penyampaian Informasi di *Website*

3.1.2.4 Prosedur Penyuluhan Kepada Masyarakat

Penyuluhan dilakukan oleh pihak PVMBG kepada Masyarakat yang berada di kawasan rawan bencana jika terjadi tanggap darurat, sebagai bentuk peringatan dini bencana gunungapi yang bisa kapan saja terjadi. Prosedur yang dilakukan untuk memberikan penyuluhan langsung kepada masyarakat adalah sebagai berikut:

1 Kepala Pusat Vullkanologi memberikan perintah pengadaan kegiatan penyuluhan.
2 Kepala Bidang akan membuat surat perintah yang ditandatangani Kepala Pusat
3 Surat Perintah diterima oleh petugas pos pengamatan.
4 Petugas pos melakukan tindak lanjut dengan mengirim surat permohonan kepada pemerintah setempat.
5 Pemerintah setempat akan mengundang masyarakat dan menyediakan tempat penyuluhan.
6 Setelah mendapat ijin dari pemerintah setempat petugas pos pengamatan akan memberikan peringatan dini tentang bahaya gunungapi.

Proses melakukan penyuluhan digambarkan pada Gambar 3.4



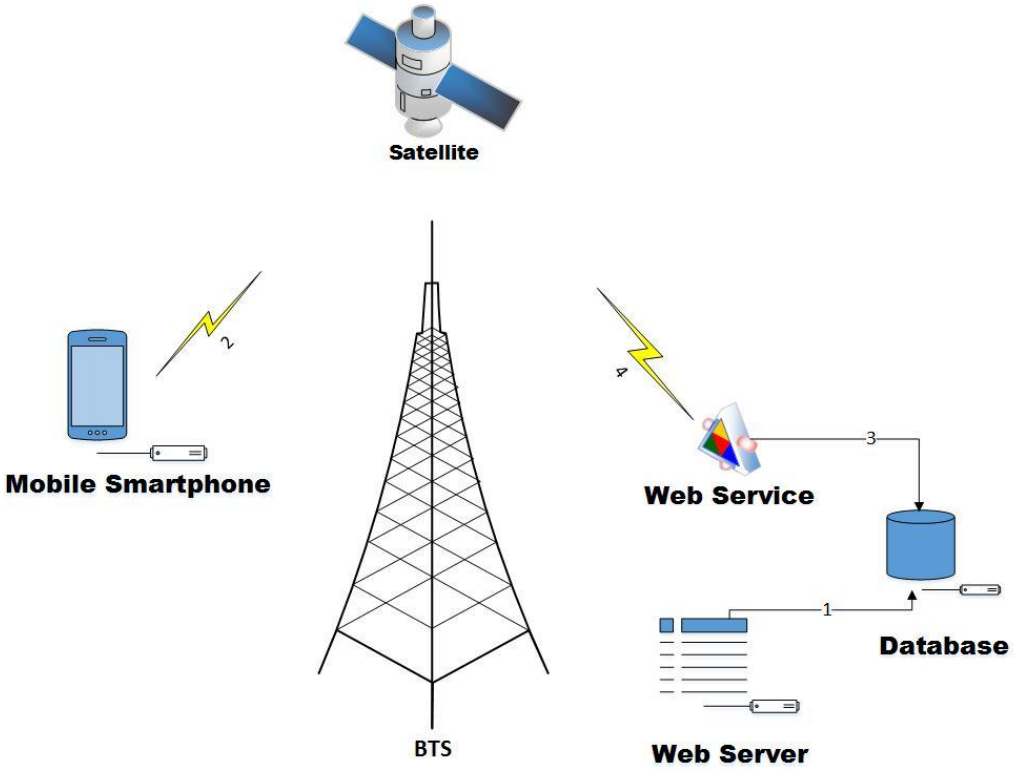
Gambar 3.4 *Activity Diagram* Penyuluhan Kepada Masyarakat

3.1.3 Analisis Arsitektur Sistem

Arsitektur fisik sistem peringatan dini bencana gunungapi terdiri dari aplikasi *mobile* dan *web service* yang dikelola oleh pihak vulkanologi. Arsitektur sistem kerja sistem yang dibangun adalah sebagai berikut :

1. *Web server* sebagai *backend* aplikasi sebagai media melakukan *update* data dalam *database* MySQL.
2. *Mobile* aplikasi sebagai *frontend* yang di akses oleh masyarakat umum, melakukan *request* informasi ke *web service* melalui jaringan internet.
3. *Web service* sebagai aplikasi yang menghubungkan perangkat dan sistem operasi yang berbeda melakukan *request* data kedalam *database*.
4. *Web service* akan mengirim data melalui internet ke perangkat *mobile*.

Gambar 3.5 berikut ini akan memberikan gambaran tentang arsitektur sistem yang dibangun.



Gambar 3.5 Arsitektur Sistem

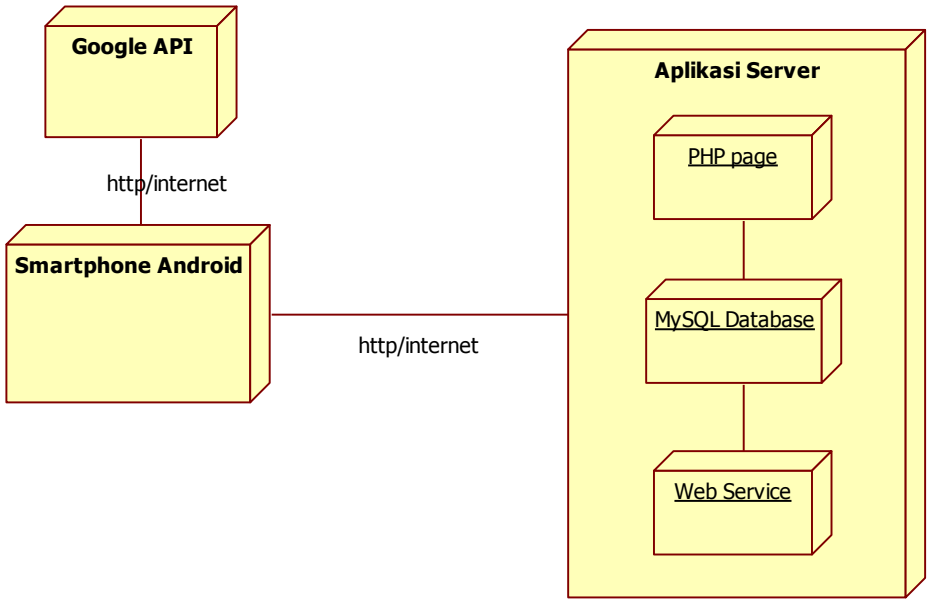
Aplikasi *Mobile smartphone* merupakan aplikasi *frontend* yang tersinstal sebagai antar muka pengguna dengan sistem. Pengembangan aplikasi meng gunakan IDE Eclipse Android *Development Tool*.

Webservice adalah aplikasi menengah (*middle application*) yang berfungsi sebagai penghubung antara aplikasi *frontend* dengan *database server* dalam proses penggunaan basis data. *Web service* tidak mempunyai tampilan, Artinya di dalam *web services* hanya tersedia fungsi-fungsi yang nantinya dapat digunakan oleh aplikasi lainnya. *Web service* menggunakan JSON sebagai *form*at dokumen dalam melakukan pertukaran data, karena JSON merupakan suatu *form*at dokumen yang berbasis teks, maka *web service* memungkinkan berlangsungnya komunikasi antar aplikasi yang berbeda dan *platform* yang berbeda pula. *Web Service* dapat diimplementasikan dalam berbagai jenis *platform* dengan menggunakan bahasa pemrograman apapun dan bisa digunakan oleh berbagai aplikasi yang menggunakan bahasa pemrograman apapun.

Web server merupakan aplikasi *backend* yang berfungsi sebagai pengelolaan informasi yang disimpan pada database *server*. Pada bagian ini aplikasi dibangun dengan menggunakan bahasa pemrograman PHP, sedangkan untuk *database* menggunakan MySQL.

Dalam arsitektur ini, GPS dimanfaatkan untuk memberitahu keberadaan posisi pengguna baik dengan koordinat *latitude* maupun *longitude* menggunakan perangkat *mobile* yang terhubung dengan jaringan.

Tata letak sistem secara fisik serta penampakan *software* yang berjalan pada bagian *hardware* dapat dilahat pada Gambar 3.6



Gambar 3.6 Diagram Deployment

3.1.4 Analisis Aturan Bisnis

Aturan bisnis yang berhubungan dengan sistem peringatan dini yang akan dibangun adalah sebagai berikut :

1. *Update* data dilakukan jika ada perubahan status gunung atau pesan rekomendasi.
2. Peringatan dini diberikan oleh bagian subbidang evaluasi potensi bencana gunungapi dan operator lapangan.

3. Informasi diberikan jika ada perubahan status ataupeningkatan aktifitas gunung.
4. *Refresh* informasi pada aplikasi mobile dilakukan setiap ada perubahan informasi.

3.1.5 Analisis Web Service

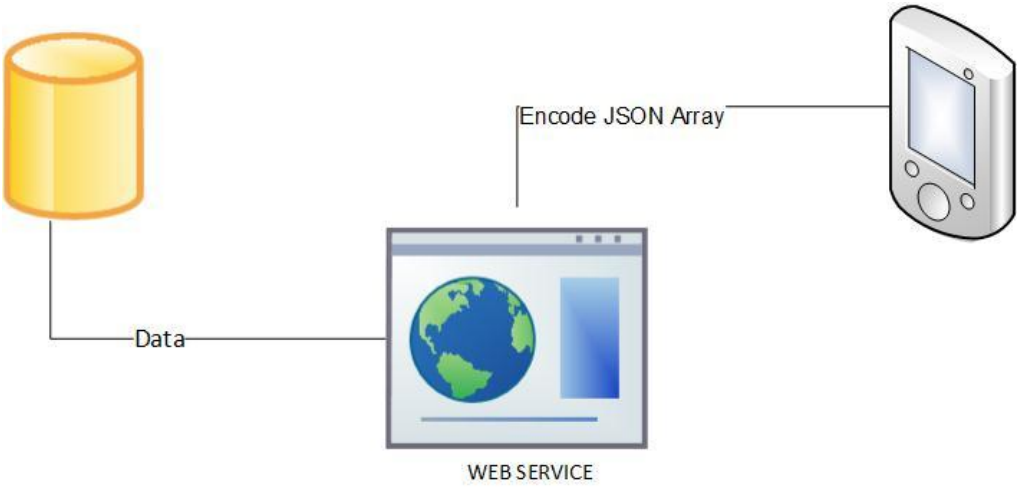
Analisis *web service* merupakan analisis salah satu bentuk sistem perangkat lunak yang didesain untuk mendukung interaksi antar perangkat melalui suatau jaringan. *Web service* memudahkan beberapa aplikasi atau komponennya untuk saling berhubungan dengan aplikasi lain dalam sebuah organisasi maupun diluar organisasi menggunakan standar yang tidak terikat platform (*platform-neutral*) dan tidak terikat akan bahasa pemrograman yang digunakan (*language-neutral*). *Web service* ini berfungsi untuk menghubungkan aplikasi *frontend* dan aplikasi *backend*. Cara kerja *web service* ini yaitu dengan mengambil data dari *database server* dan kemudian mengkonversikan data tersebut kedalam format pertukaran data. Format pertukaran data dalam *web service* ini menggunakan JSON dan menggunakan tipe data array untuk menampung data yang nantinya akan di *generate* di aplikasi *mobile*.

JSON (*Javascript Object Notation*) merupakan format ringan untuk memasukan data ke dalam variabel. Sangat mudah dimengerti dan diimplementasikan oleh manusia, dan mudah juga untuk komputer dalam melakukan parsingnya. Berikut ini adalah contoh format JSON array :

```
/* Dengan New Array */
1. var array_ke_1 = new Array('Satu','Dua','Tiga');


/* Dengan New Array dan Memasukan Satu Per-Satu */
1. var array_ke_2 = new Array();
2. array_ke_2[0] = 'Satu';
3. array_ke_2[1] = 'Dua';
4. array_ke_2[2] = 'Tiga';


/* Dengan Format JSON */
1. var array ke 3 = ['Satu','Dua','Tiga'];
```



Gambar 3.7 *Web Service*

Cara kerja *web service* dalam sistem ini adalah sebagai berikut :

1. *Web service* melakukan koneksi dengan database MySQL
2. Data yang diambil akan ditampung dalam sebuah array.
3. Data yang sudah ditampung dalam array di encode ke JSON.
4. JSON encode kemudian dikirim ke aplikasi mobile yang melakukan *request* melalui jaringan internet.
5. JSON encode nantinya akan diterima oleh aplikasi dan dilakukan decode lagi dalam bahasa java dan ditampung dalam array list.
6. Aplikasi mobile akan menampilkan data yang terdapat dalam JSON encode.

3.1.6 Spesifikasi Kebutuhan Perangkat Lunak

Spesifikasi kebutuhan perangkat lunak yang akan dibangun berdasarkan kebutuhan pengguna. Spesifikasi kebutuhan perangkat lunak akan dibagi ke dalam dua bagian yaitu SKPL-F (spesifikasi kebutuhan perangkat lunak fungsional) dan

SKPL-NF (spesifikasi kebutuhan perangkat lunak non-fungsional). Spesifikasi kebutuhan perangkat lunak dalam penelitian ini dapat dilihat pada tabel berikut.

Tabel 3.1 SKPL *User Requirement*

Kode	Kebutuhan
SKPL-F1	Aplikasi *back end* menyediakan layanan *login*
SKPL-F2	Aplikasi *back end* dapat mengolah informasi gunung
SKPL-F3	Aplikasi *back end* dapat mengolah data user (operator gunung)
SKPL-F4	Aplikasi *front end* dapat memparsing data dari *web service*
SKPL-F5	Aplikasi *front end* dapat menampilkan *update* informasi gunung
SKPL-F6	Aplikasi *front end* dapat menampilkan *update* lokasi
SKPL-F7	Aplikasi *front end* dapat menampilkan notifikasi status

Tabel 3.2 SKPL Fungsional *System requirement*

Kode	Kebutuhan
SKPL-F1	*1.* Data yang digunakan untuk *login* adalah *username* dan *password* 2. Waktu *Session login* selama 15 menit
SKPL-F2	1. Sistem dapat menambah informasi gunung 2. Sistem dapat mengubah informasi gunung 3. Sistem dapat menghapus informasi gunung
SKPL-F3	1. Sistem dapat menambah data operator 2. Sistem dapat mengubah data operator 3. Sistem dapat menghapus data operator
SKPL-F4	1. Data yang di*parsing* merupakan informasi gunung 2. *Parsing* data menggunakan format JSON
SKPL-F5	1. Sistem menampilkan daftar gunung 2. Sistem menampilkan status gunung 3. Sistem menampilkan detail gunung
SKPL-F6	1. Sistem menampilkan peta lokasi 2. Sistem menampilkan letak koordinat gunung 3. Sistem menampilkan letak perangkat 4. Sistem menampilkan kawasan rawan bencana gunungapi
SKPL-F7	1. Notifikasi aktif jika pengguna aplikasi mobile telah melakukan registrasi pengguna 2. Notifikasi tampil jika ada gunung yang menunjukan peningkatan aktivitas.

Tabel 3.3 SKPL Non-Fungsional

Kode	Kategori	Kebutuhan
SKPL-NF-1	*Efficiency*(*Product Requirement*)	Sistem dapat menampilkan data gunungapi secara cepat dalam bentuk informasi
SKPL-NF-2	*Dependability*(*Product Requirement*)	Sistem membutuhkan web service, koneksi GPS untuk aplikasi mobile, dan browser

Kode	Kategori	Kebutuhan
		untuk menjalankan web server.
SKPL-NF-3	*Security*(*Product Requirement*)	Password admin dalam database dienkripsi untuk menjaga keamanan informasi.
SKPL-NF-4-a	*Performance*(*Product Requirement*)	Sistem harus dapat digunakan 24 jam sehari
SKPL-NF-4-b	*Performance*(*Product Requirement*)	Jenis sistem operasi dan spesifikasi mobile akan sangat berpengaruh pada kinerja sistem.
SKPL-NF-5	*Enviromental*(*Organizatonal Requirement*)	Software ini menggunakan database yang terhubung dengan SQL server sebagai database server (DBMS).
SKPL-NF-6	*Operational(Organizatonal Requirement)*	Setiap kesalahan/error yang terjadi disertai dengan pesan error kepada pengguna
SKPL-NF-7	*Development(Organizatonal Requirement)*	Untuk menjalankan sistem yang di bangun maka dibutuhkan : a. Microsoft Windows sebagai sistem operasi untuk backend aplikasi b. Android OS versi 4.0 untuk aplikasi frontend c. Web browser

3.1.7 Analisis Kebutuhan Non-Fungsional

1. Analisis Kebutuhan Perangkat Lunak

Penguraian kebutuhan-kebutuhan non fungsional yang berhubungan dengan proses pembangunan perangkat lunak. Untuk sistem peringatan dini diperlukan beberapa perangkat lunak seperti pada tabel berikut.

Tabel 3.4 Analisis kebutuhan perangkat lunak

Kebutuhan non fungsional	Kebutuhan perangkat lunak
Aplikasi front end berbasis mobile application	1. Sistem operasi windows / linux 2. ADT eclipse bundle
Back end aplikasi berbasis *web* dinamis	1. Code editor Adobe Dreamweaver Cs 5 2. WAMP server versi 2
Sistem operasi *mobile* sebaiknya menggunakan android versi 4.0 untuk memaksimalkan kinerja sistem	Sistem operasi android minimal versi 2.4 untuk running aplikasi

2. Analisis Kebutuhan Perangkat Keras

Penguraian kebutuhan-kebutuhan non fungsional yang berhubungan dengan spesifikasi perangkat keras dan berhubungan dengan proses pembangunan perangkat lunak. Kebutuhan perangkat keras untuk sistem yang dibangun bisa dilihat pada tabel berikut.

Tabel 3.5 Analisis kebutuhan perangkat keras

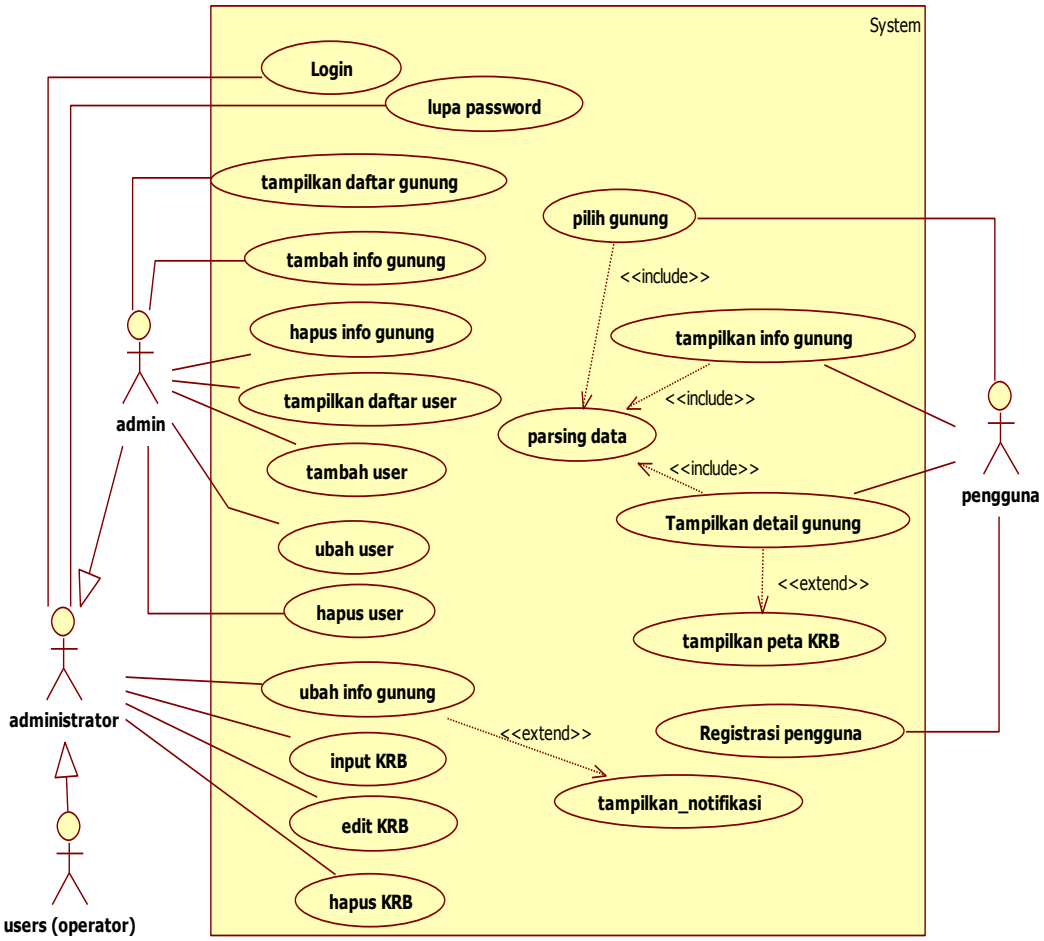
Kebutuhan non fungsional	Kebutuhan perangkat keras
Aplikasi dijalankan pada perangkat *mobile smartphone*	*Smartphone* dengan spesifikasi minimal: 1. Proccesor 600 Mhz 2. Layar resolusi 240 x 320 pixel 3. Memori 100 Mb
Back end aplikasi berbasis web dinamis	Komputer dengan spesifikasi minimal : Proccesor Intel Dual Core (2.4 GHz) / AMD Athlon X2 RAM 2 GB

3.1.8 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional merupakan alur kerja perangkat lunak yang dibangun. Aplikasi peringatan dini menggunakan metode pendekatan berorientasi objek.

3.1.8.1 *Use Case Diagram*

Use case akan menggambarkan apa yang sistem lakukan tanpa mendeskripsikan bagaimana sistem menyelesaikannya. *Use case diagram* dari sistem yang dibangun bisa dilihat pada Gambar 3.8



Gambar 3.8 *Use Case* Diagram

3.1.8.2 *Use Case Scenario*

Use case scenario merupakan hasil instansiasi dan penjelasan dari setiap *use case* yang dibangun. Berikut ini adalah *use case scenario* dari *use case* yang dibangun.

Tabel 3.6 *Use Case Scenario Login*

Usecase name	*Login*
Goal in context	Administrator mengakses halaman kelola informasi
Preconditions	Sistem menampilkan halaman login
Succesful end Condition	Sistem menampilkan halaman kelola informasi
Failed End Condition	Sistem tidak menampilkan halaman kelola informasi
Actor	Administrator

Main Flow	Step	Action
	1	Administrator mengakses halaman login
	2	Sistem menampilkan form login
	3	Administrator memasukan username dan password
	5	Jika form sudah diisi maka sistem mengecek username dan password
	6	Jika login benar sistem menampilkan halaman kelola informasi
Extension	Step	Branching action
	5.1	Jika form belum diisi sistem menampilkan pesan kesalahan
	6.1	Jika login salah sistem menampilkan pesan kesalahan

Tabel 3.7 Use Case Scenario Tambah Info Gunung

Usecase name	Tambah info gunung	
Goal in context	Admin mengakses halaman tambah gunung	
Preconditions	Admin mengakses menu manajemen gunung	
Succesful end Condition	Data gunung ditambahkan kedalam database	
Failed End Condition	Data gunung tidak berhasil ditambahkan kedalam database	
Actor	Admin	
Main Flow	Step	Action
	1	Admin memilih menu tambah gunung
	2	Sistem menampilkan form input gunung
	3	Admin mengisi data gunung
	4	Admin menekan button simpan
	5	Sistem memeriksa data masukan
	6	Sistem menyimpan data kedalam database
Extension	Step	Branching action
	5.1	Jika field belum diisi sistem menampilkan pesan kesalahan
	6.1	Data tidak disimpan dalam database

Tabel 3.8 Use Case Scenario Tampilkan Daftar Gunung

Usecase name	Tampilkan daftar gunung	
Goal in context	Sistem menampilkan data tabel bunung	
Preconditions	Admin mengakses halaman manajemen gunung	
Succesful end Condition	Data gunung dalam database ditampilkan	
Failed End Condition	Data gunung dalam database tidak tampil	
Actor	Admin	
Main Flow	Step	Action
	1	Admin mengakses manajemen gunung
	2	Sistem menampilkan data gunung

Extension	Step	Branching action
	2.1	Jika data tidak ada sistem menampilkan pesan

Tabel 3.9 Use Case Scenario Ubah Info Gunung

Usecase name	Ubah info gunung	
Goal in context	Sistem menampilkan halaman ubah gunung	
Preconditions	Administrator mengakses halaman manajemen gunung	
Succesful end Condition	Data gunung dalam database diubah	
Failed End Condition	Data gunung dalam database tidak berhasil diubah	
Actor	Administrator	
Main Flow	Step	Action
	1	Administrator memilih menu edit
	2	Sistem menampilkan form edit gunung
	3	Administrator mengisi data gunung
	4	Administrator menekan button simpan
	5	Sistem memeriksa data masukan
	6	Sistem menyimpan data kedalam database
Extension	Step	Branching action
	5.1	Jika field belum diisi sistem menampilkan pesan kesalahan

Tabel 3.10 Use Case Scenario Hapus Info Gunung

Usecase name	Hapus info gunung	
Goal in context	Admin mengakses menu hapus gunung	
Preconditions	Admin mengakses halaman manajemen gunung	
Succesful end Condition	Data gunung dalam database dihapus	
Failed End Condition	Data gunung dalam database tidak berhasil dihapus	
Actor	Admin	
Main Flow	Step	Action
	1	Admin memilih menu hapus
	2	Sistem menampilkan pesan konfirmasi
	3	Admin menyetujui pesan konfirmasi
	4	Sistem menghapus data dalam database
Extension	Step	Branching action
	4.1	Jika admin tidak menyetujui pesan konfirmasi maka proses hapus dibatalkan

Tabel 3.11 Use Case Scenario Masukan KRB

Usecase name	Masukan Kawasan Rawan Bencana (KRB)
Goal in context	Sistem menampilkan halaman tambah KRB
Preconditions	Administrator mengakses menu KRB
Succesful end	Data KRB berhasil disimpan dalam database

Condition		
Failed End Condition	Data KRB tidak tersimpan dalam database	
Actor	Administrator	
Main Flow	Step	Action
	1	Administrator memilih menu input KRB
	2	Sistem menampilkan peta
	3	Administrator mengisi data KRB
	4	Administrator menekan tombol simpan
	5	Sistem menyimpan data KRB kedalam database
Extension	Step	Branching action
	4.1	Jika administrator menekan tombol cancel maka data tidak disimpan dalam database
	5.1	Data tidak disimpan dalam database

Tabel 3.12 Use Case Scenario Ubah KRB

Usecase name	Ubah Kawasan Rawan Bencana	
Goal in context	Sistem menampilkan halaman ubah KRB	
Preconditions	Administrator mengakses menu KRB	
Succesful end Condition	Data KRB berhasil disimpan dalam database	
Failed End Condition	Data KRB tidak tersimpan dalam database	
Actor	Administrator	
Main Flow	Step	Action
	1	Administrator memilih menu edit KRB
	2	Sistem menampilkan peta kawasan rawan bencana
	3	Administrator mengubah data KRB
	4	Administrator menekan tombol simpan
	5	Sistem menyimpan data KRB kedalam database
Extension	Step	Branching action
	4.1	Jika administrator menekan tombol cancel maka data tidak disimpan dalam database
	5.1	Data tidak disimpan dalam database

Tabel 3.13 Use Case Scenario Hapus KRB

Usecase name	Hapus Kawasan Rawan Bencana	
Goal in context	Sistem menghapus data KRB	
Preconditions	Administrator mengakses menu KRB	
Succesful end Condition	Data KRB dalam database berhasil dihapus	
Failed End Condition	Data KRB dalam database tidak terhapus	
Actor	Administrator	
Main Flow	Step	Action
	1	Administrator memilih data yang akan

		dihapus
	2	Administrator menekan tombol hapus
	3	Sistem menampilkan pesan konfirmasi
	4	Administrator menyetujui pesan konfirmasi
	5	Sistem menghapus data KRB kedalam database
Extension	Step	Branching action
	4.1	Jika administrator menekan tombol cancel maka data tidak disimpan dalam database
	5.1	Sistem tidak menghapus data dalam database

Tabel 3.14 *Use Case Scenario* Tambah User (Operator)

Usecase name	Tambah user	
Goal in context	Sistem menampilkan halaman tambah user	
Preconditions	Admin mengakses manajemen user	
Succesful end Condition	Data user disimpan dalam database	
Failed End Condition	Data user tidak tersimpan dalam database	
Actor	Admin	
Main Flow	Step	Action
	1	Administrator memilih menu tambah user
	2	Sistem menampilkan form input user
	3	Administrator memasukan data user
	4	Administrator menekan tombol simpan
	5	Sistem memeriksa data masukan
	6	Sistem menyimpan data kedalam database
Extension	Step	Branching action
	6.1	Jika data masukan tidak lengkap sistem menampilkan pesan kesalahan

Tabel 3.15 *Use Case Scenario* Tampilkan Daftar User

Usecase name	Tampilkan daftar user	
Goal in context	Sistem menampilkan daftar user di tabel user	
Preconditions	Administrator mengakses manajemen user	
Succesful end Condition	Data user dalam database ditampilkan	
Failed End Condition	Data user dalam database tidak ditampilkan	
Actor	Admin	
Main Flow	Step	Action
	1	Administrator memilih menu tambah user
	2	Sistem menampilkan form input user
	3	Administrator memasukan data user
	4	Administrator menekan tombol simpan
	5	Sistem memeriksa data masukan
	6	Sistem menyimpan data kedalam database
Extension	Step	Branching action

	6.1	Jika data masukan tidak lengkap sistem menampilkan pesan kesalahan

Tabel 3.16 *Use Case Scenario* Ubah User

Usecase name	Ubah user	
Goal in context	Sistem menampilkan halaman edit user	
Preconditions	Sistem menampilakan daftar info gunung	
Succesful end Condition	Data user dalam database diubah	
Failed End Condition	Data user dalam *database* tidak berubah	
Actor	Administrator	
Main Flow	Step	Action
	1	Administrator memilih menu edit
	2	Sistem menampilkan form edit user
	3	Administrator mengubah data user
	4	Administrator menekan tombol simpan
	5	Sistem mengubah data
Extension	Step	Branching action
	4.1	Jika ubah data gagal sistem menampilkan pesan kesalahan

Tabel 3.17 Use Case Scenario Hapus User

Usecase name	Hapus user	
Goal in context	Sistem menghapus data dalam database	
Preconditions	Administrator mengakses halaman manajemen user	
Succesful end Condition	Data user dihapus	
Failed End Condition	Data user tidak terhapus	
Actor	Administrator	
Main Flow	Step	Action
	1	Administrator memilih menu hapus
	2	Sistem menampilkan pesan konfirmasi
	3	Administrator menyetujui pesan konfirmasi
	4	Sistem menghapus data dalam database
Extension	Step	Branching action
	4.1	Proses

Tabel 3.18 *Use Case Scenario* Lupa Password

Usecase name	Lupa password	
Goal in context	Sistem mereset password dalam database	
Preconditions	Administrator mengakses menu lupa password	
Succesful end Condition	Sistem mengirim password baru	
Failed End Condition	Sistem tidak menampilkan password baru	
Actor	*Administrator*	
Main Flow	Step	Action
	1	Administrator memilih menu lupa password

	2	Sistem menampilkan form input email
	3	Administrator memasukan email
	4	Sistem periksa email
	5	Sistem mengirim email konfirmasi
	6	Pengguna menyetujui reset password
	7	Sistem mengirim password baru
Extension	Step	Branching action
	5.1	Sistem menampilkan pesan kesalahan jika email tidak ditemukan

Tabel 3.19 *Use Case Scenario* Pilih Gunung

Usecase name	Pilih gunung	
Goal in context	Sistem menampilkan halaman pilih gunung yang dipantau	
Preconditions	Pengguna membuka memilih menu setting	
Succesful end Condition	Sistem menyimpan data gunung yang dipilih pengguna	
Failed End Condition	Pengguna tidak dapat memilih gunung	
Actor	Pengguna	
Main Flow	Step	Action
	1	Pengguna memilih menu setting
	2	Sistem menampilkan halaman pilih gunung
	3	Pengguna memilih gunung
	4	Sistem menyimpan gunung yang dipilih

Tabel 3.20 *Use Case Scenario* Tampilkan Info Gunung

Usecase name	Tampilkan Info Gunung	
Goal in context	Sistem menampilkan List gunungapi	
Preconditions	Pengguna membuka aplikasi	
Succesful end Condition	Sistem menampilkan daftar gunungapi	
Failed End Condition	Sistem tidak menampilkan gunungapi yang dipilih	
Actor	Pengguna	
Main Flow	Step	Action
	1	Pengguna memilih menu gunungapi
	2	Sistem melakukan parsing data
	3	Sistem menampilkan list gunung
Extension	Step	Branching action
	3.1	Sistem menampilkan pesan kesalahan jika parsing data gagal

Tabel 3.21 *Use Case Scenario* Detail Gunung

Usecase name	Tampilkan Detail Gunung
Goal in context	Sistem menampilkan Detail gunungapi
Preconditions	Pengguna mengakses Info Gunung
Succesful end Condition	Sistem menampilkan informasi detail gunungapi
Failed End Condition	Sistem tidak menampilkan informasi
Actor	Pengguna

Main Flow	Step	Action
	1	Pengguna memilih gunungapi di daftar
	2	Sistem melakukan parsing data
	3	Sistem menampilkan detail gunung
Extension	Step	Branching action
	3.1	Sistem menampilkan pesan kesalahan jika parsing data gagal

Tabel 3.22 *Use Case Scenario* Peta KRB

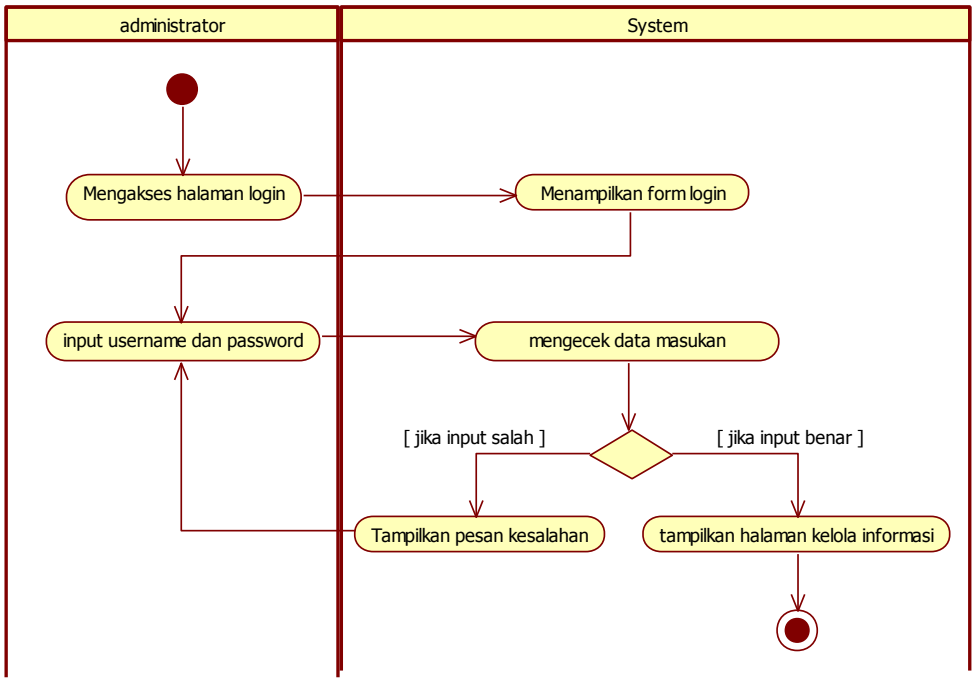
Usecase name	Tampilkan Peta KRB	
Goal in context	Sistem menampilkan peta KRB	
Preconditions	Pengguna mengakses Detail Gunung	
Succesful end Condition	Sistem menampilkan peta	
Failed End Condition	Sistem tidak menampilkan peta	
Actor	Pengguna	
Main Flow	Step	Action
	1	Pengguna memilih menu peta KRB
	2	Sistem melakukan parsing data
	3	Sistem menampilkan peta KRB gunungapi
Extension	Step	Branching action
	3.1	Sistem menampilkan pesan kesalahan jika parsing data gagal

Tabel 3.23 *Use Case Scenario* Registrasi

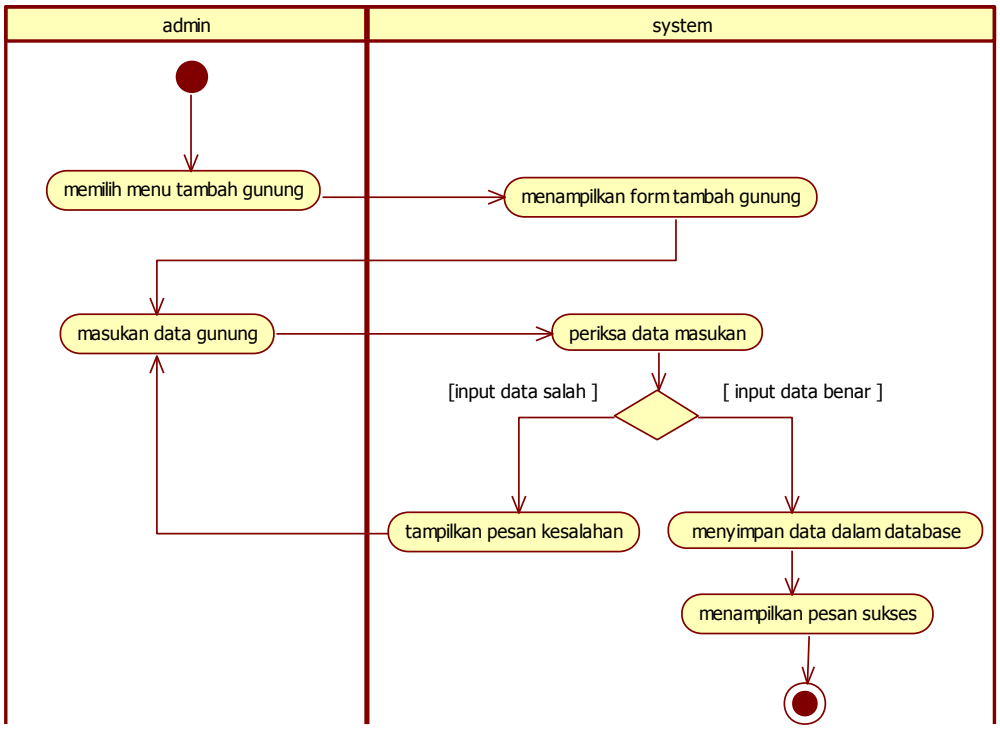
Usecase name	Registrasi Pengguna	
Goal in context	Sistem menampilkan halaman registrasi pengguna	
Preconditions	Pengguna mengakses menu notifikasi	
Succesful end Condition	Sistem menyimpan data pengguna kedalam database	
Failed End Condition	Data pengguna tidak tersimpan	
Actor	Pengguna	
Main Flow	Step	Action
	1	Pengguna memilih menu notifikasi
	2	Sistem menampilkan halaman registrasi
	3	Pengguna mengisi form registrasi
	4.	Sistem menyimpan data pengguna
Extension	Step	Branching action
	4.1	Jika masukan salah sistem menampilkan pesan kesalahan

3.1.8.3 *Activity Diagram*

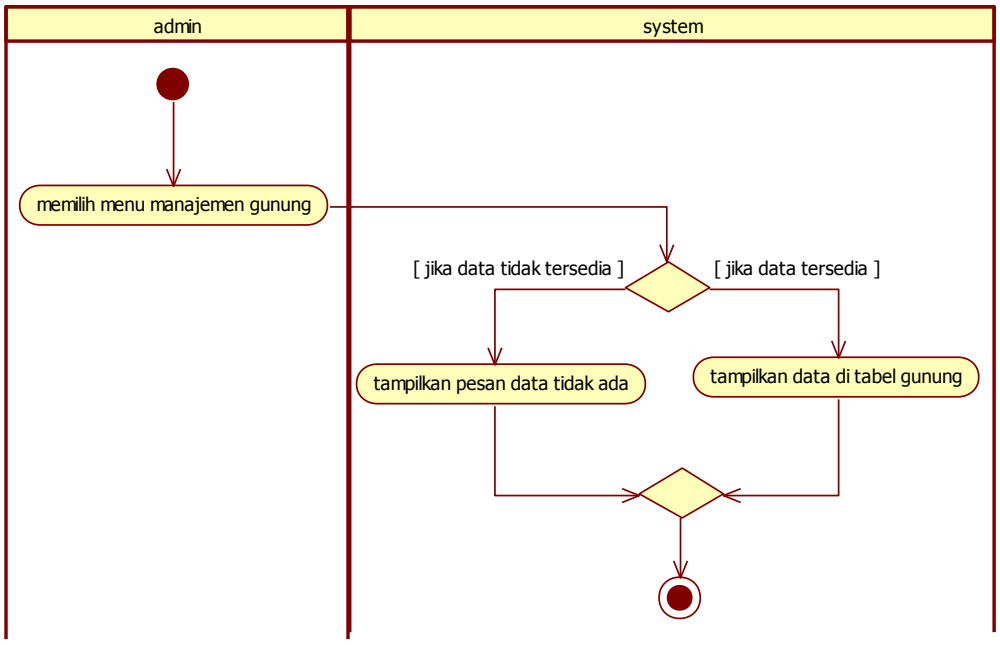
Activity diagram akan merepresentasikan alur dari aktivitas yang terjadi pada sebuah *use case*. Berikut ini adalah gambar *activity diagram* dari setiap *use case.*



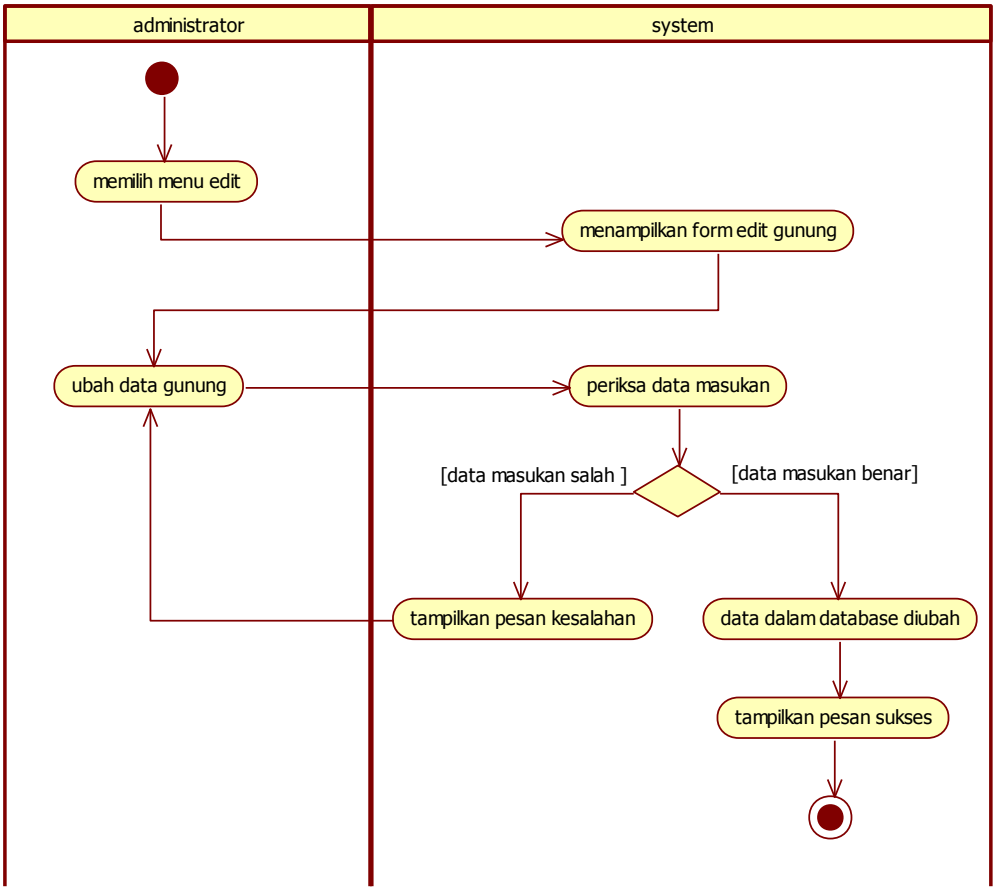
Gambar 3.9 *Activity Diagram Login*



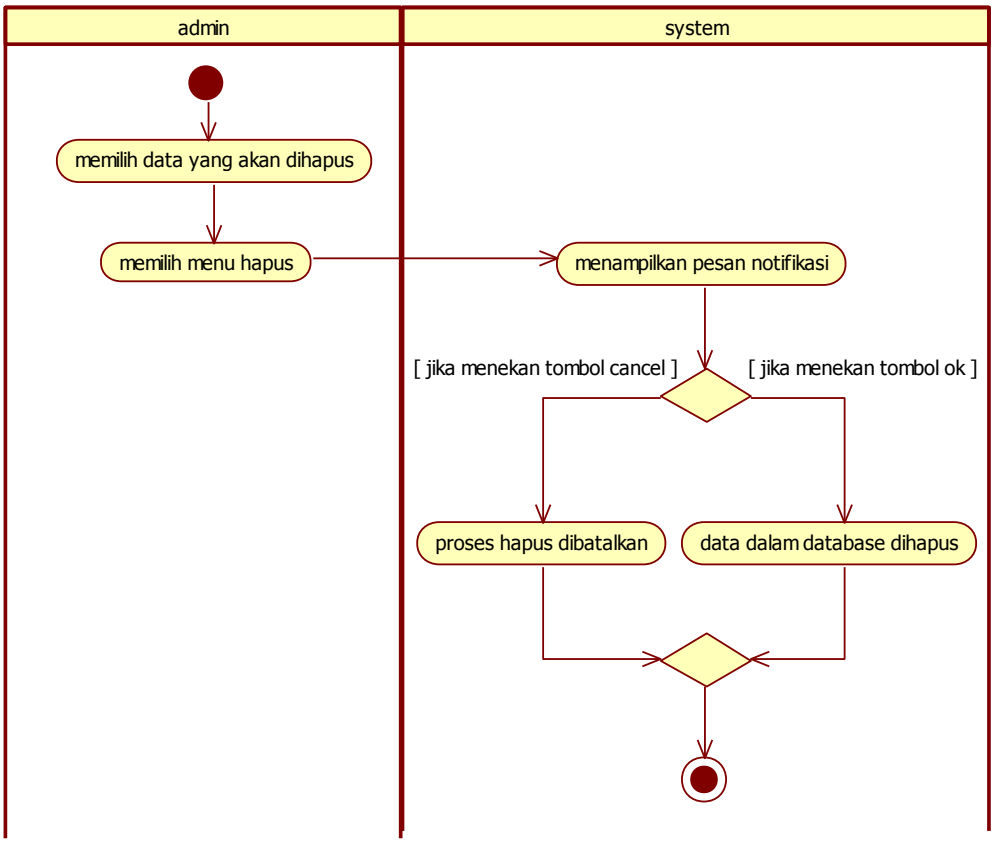
Gambar 3.10 *Activity Diagram* Tambah Info Gunung



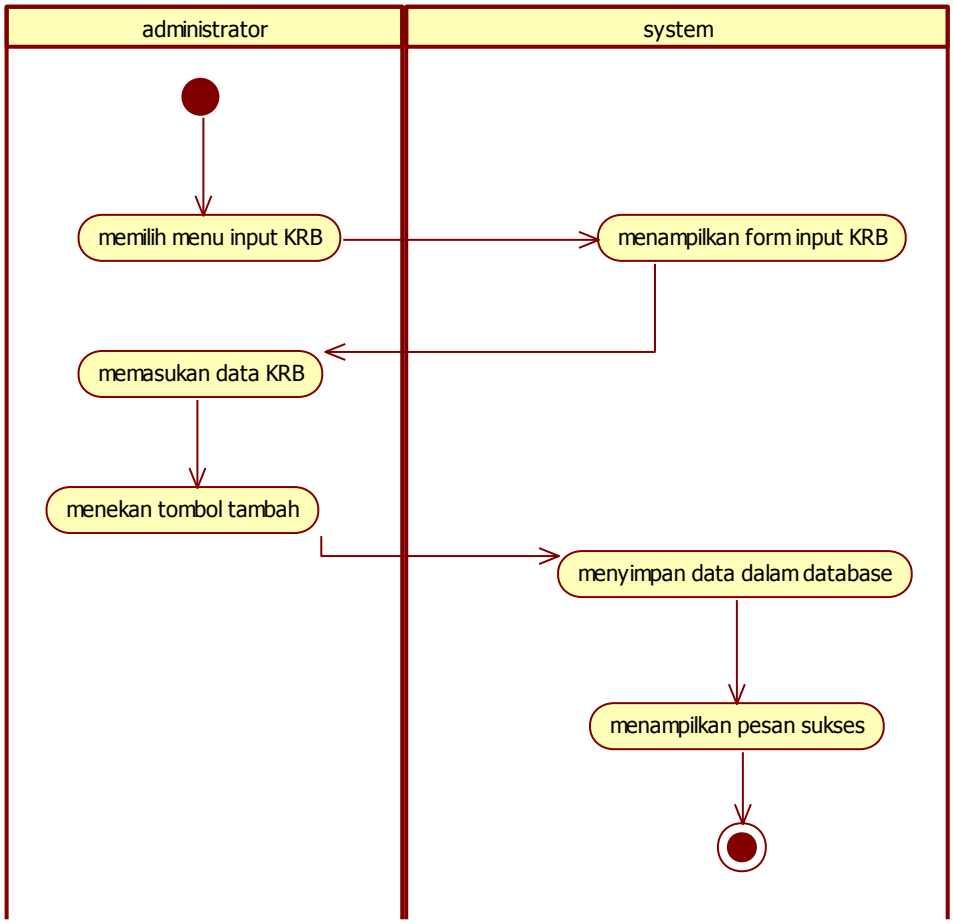
Gambar 3.11 *Activity Diagram* Tampilkan Daftar Gunung



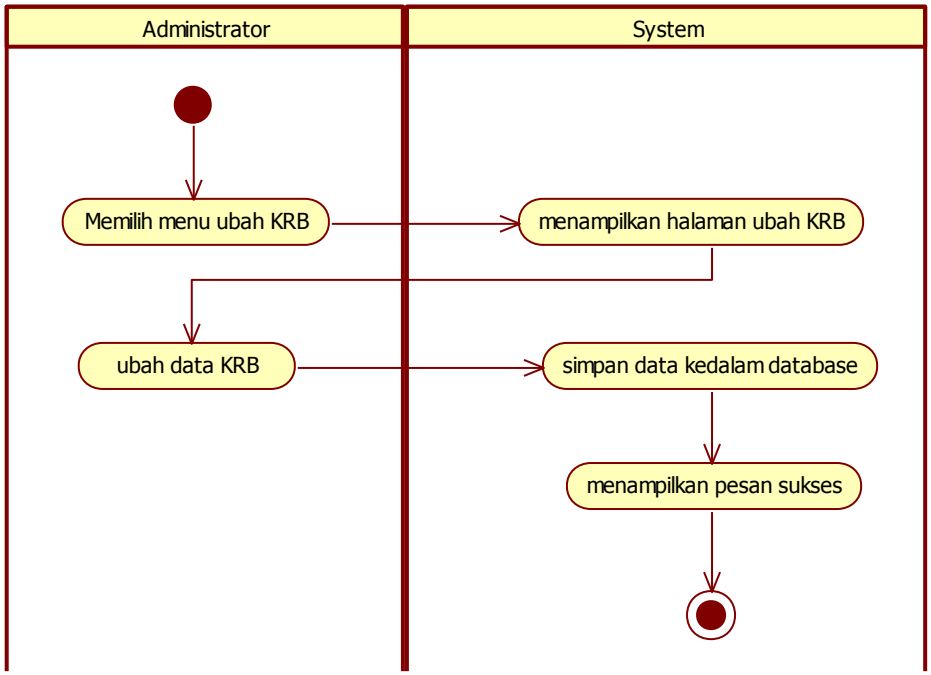
Gambar 3.12 *Activity Diagram* Ubah Info Gunung



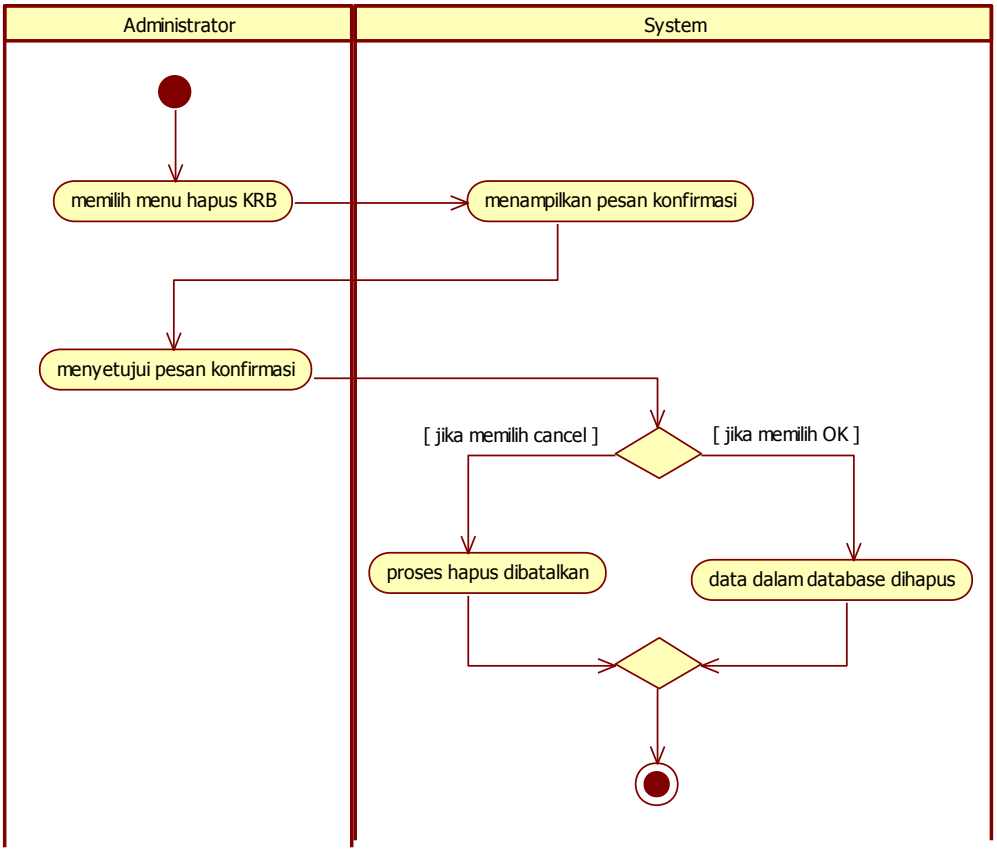
Gambar 3.13 *Activity Diagram* Hapus Info Gunung



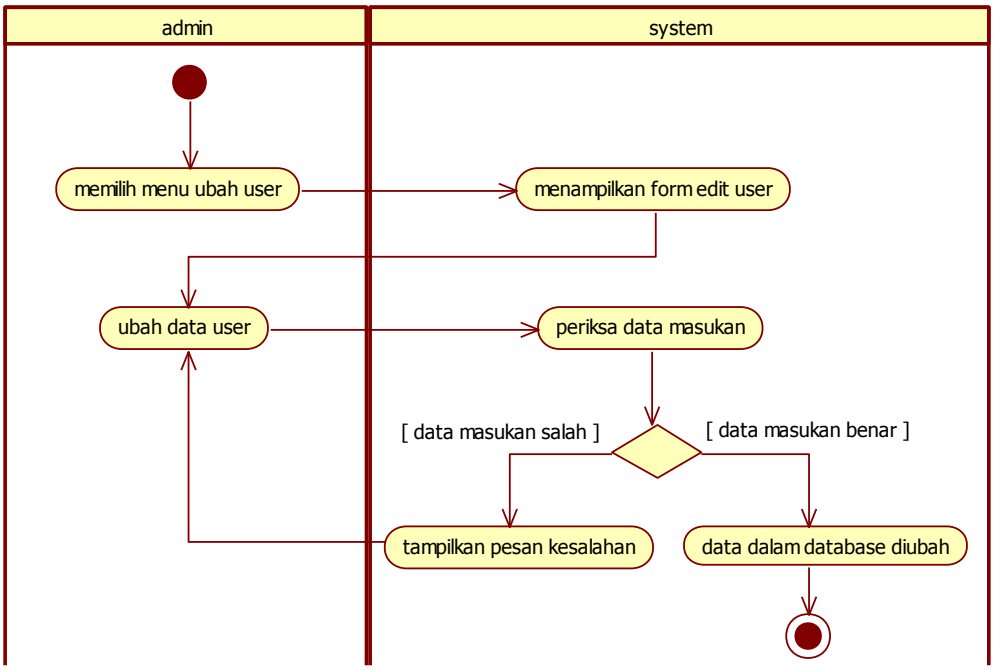
Gambar 3.14 *Activity Diagram* Input KRB



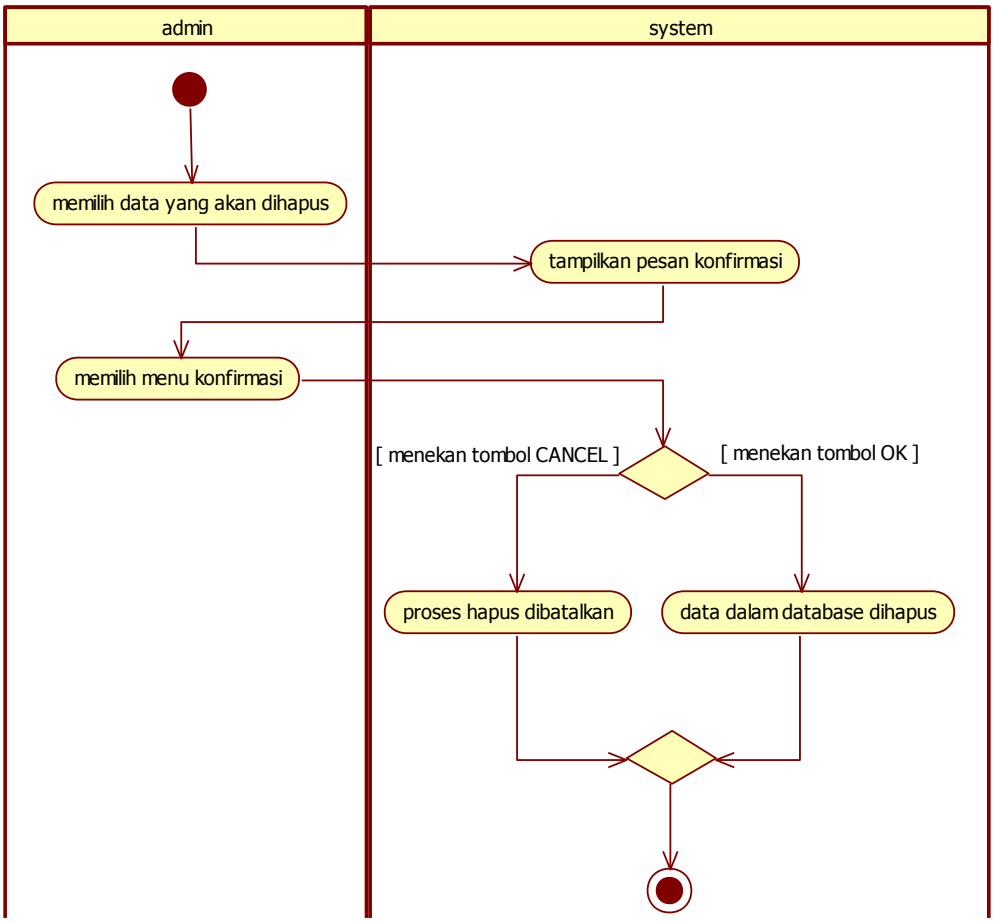
Gambar 3.15 *Activity Diagram* Ubah KRB



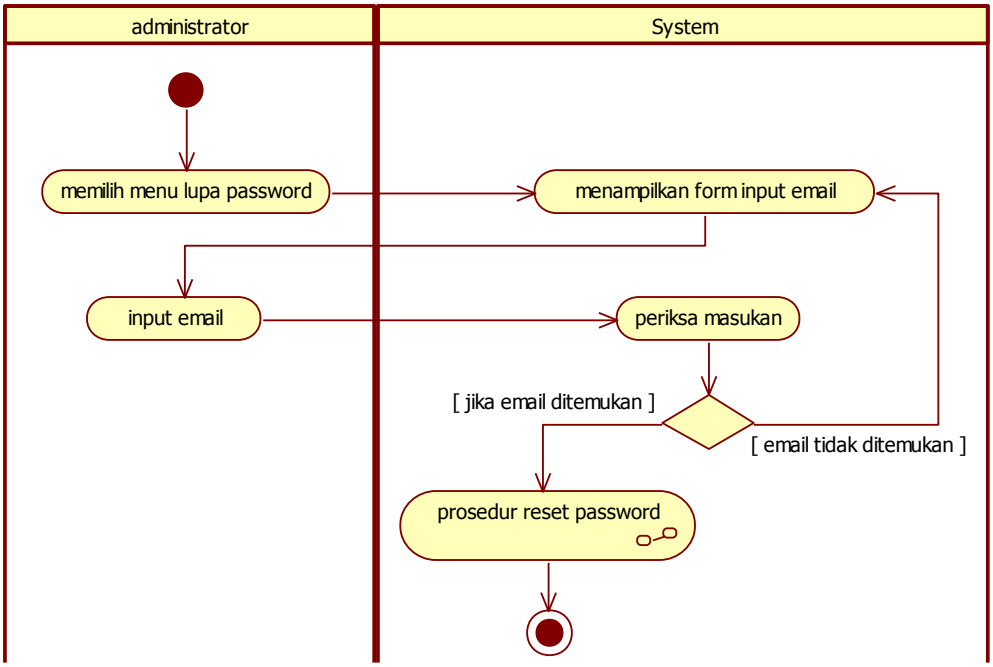
Gambar 3.16 *Activity Diagram* Hapus KRB



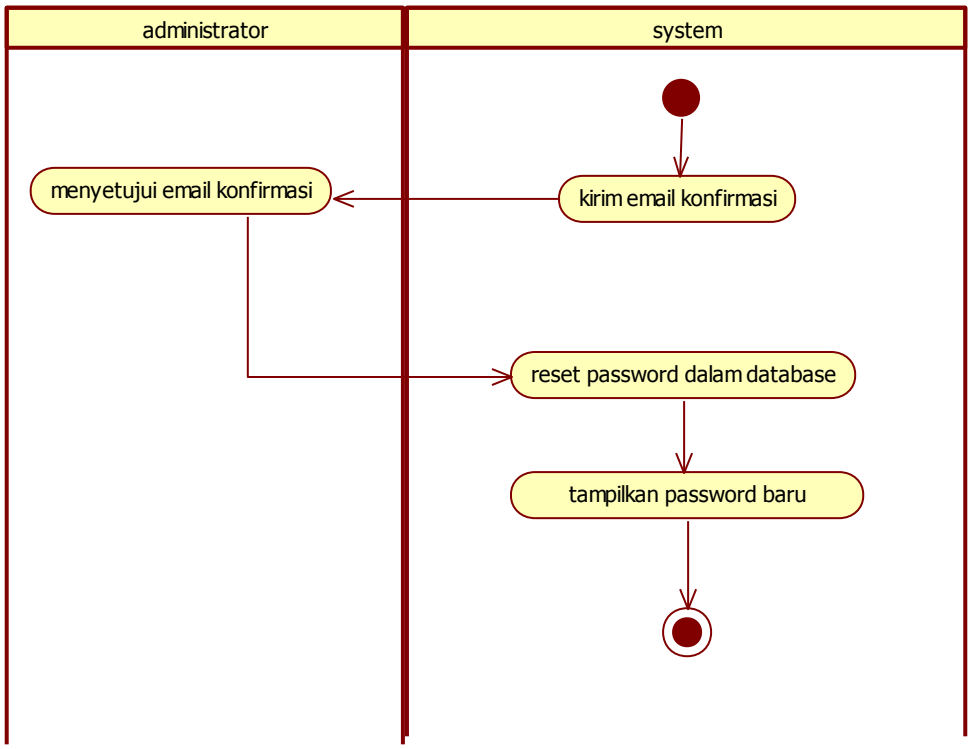
Gambar 3.17 *Activity Diagram* Ubah User



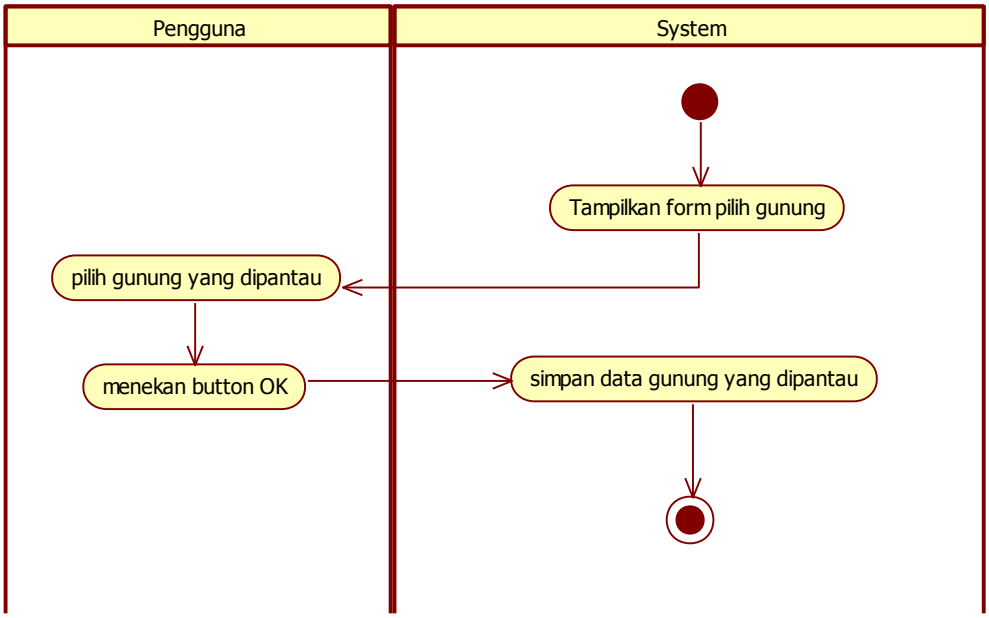
Gambar 3.18 *Activity Diagram* Hapus User



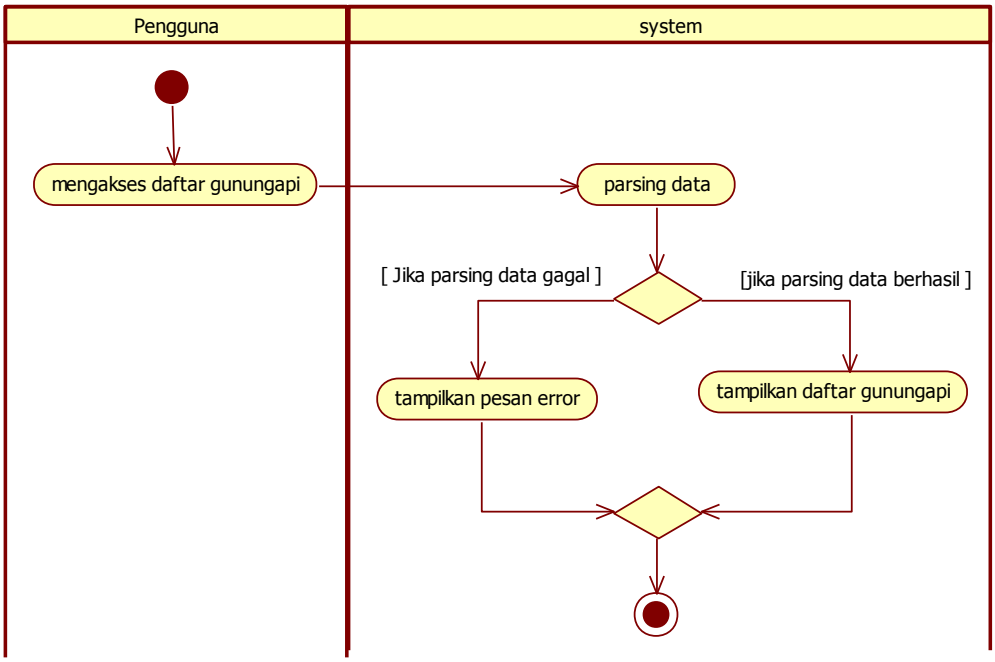
Gambar 3.19 *Activity Diagram* Lupa Password



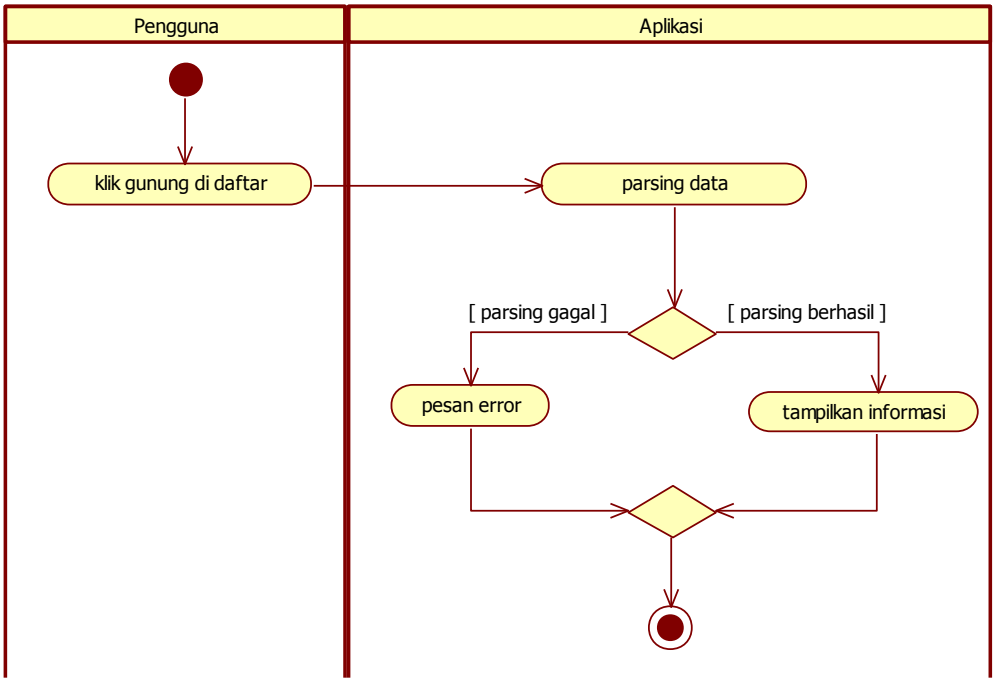
Gambar 3.20 *Activity Diagram* Reset Password



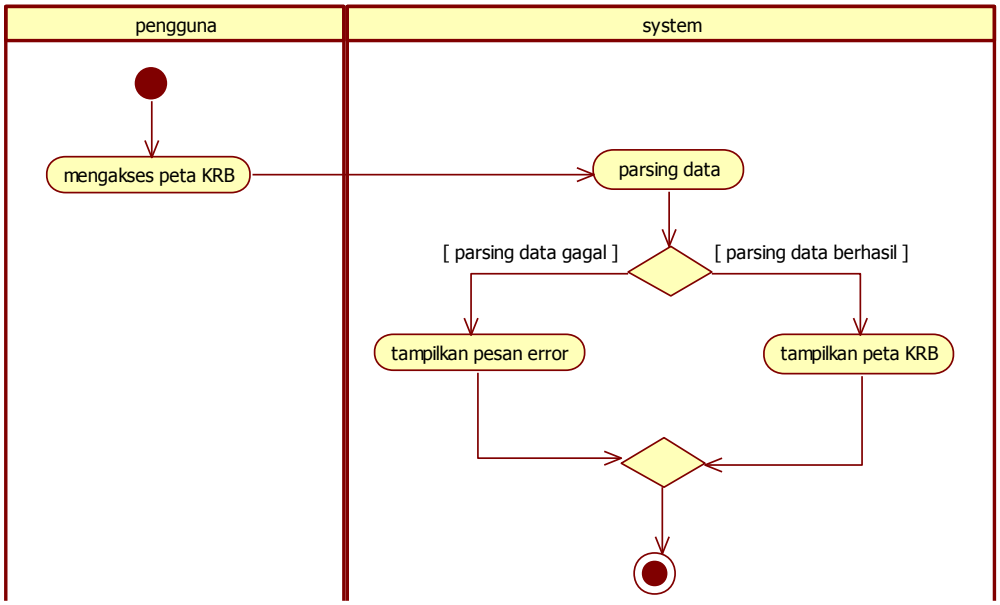
Gambar 3.21 *Activity Diagram* Pilih Gunung Yang Dipantau



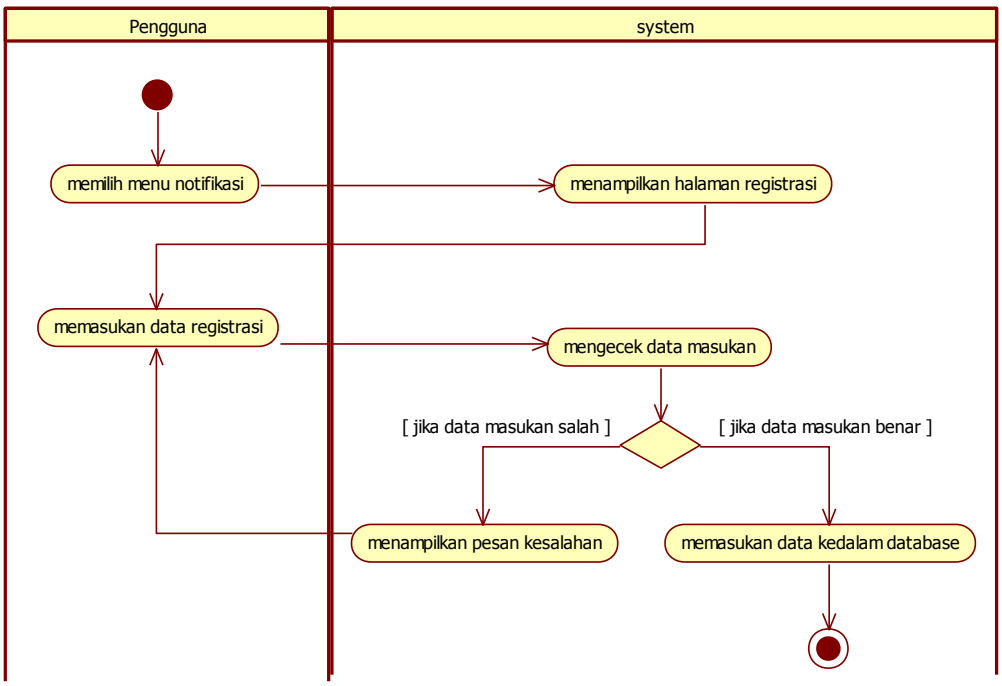
Gambar 3.22 *Activity Diagram* Tampilkan Daftar Gunungapi



Gambar 3.23 *Activity Diagram* Detail Gunung



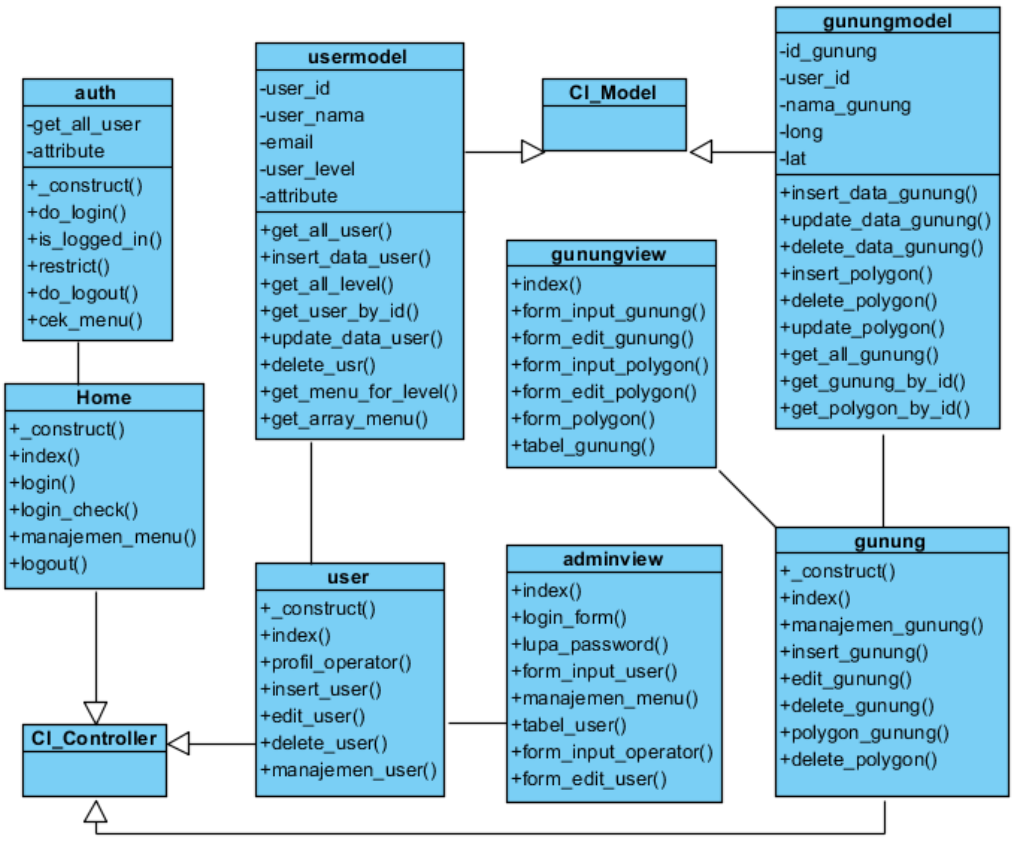
Gambar 3.24 *Activity Diagram* Tampilkan Peta KRB



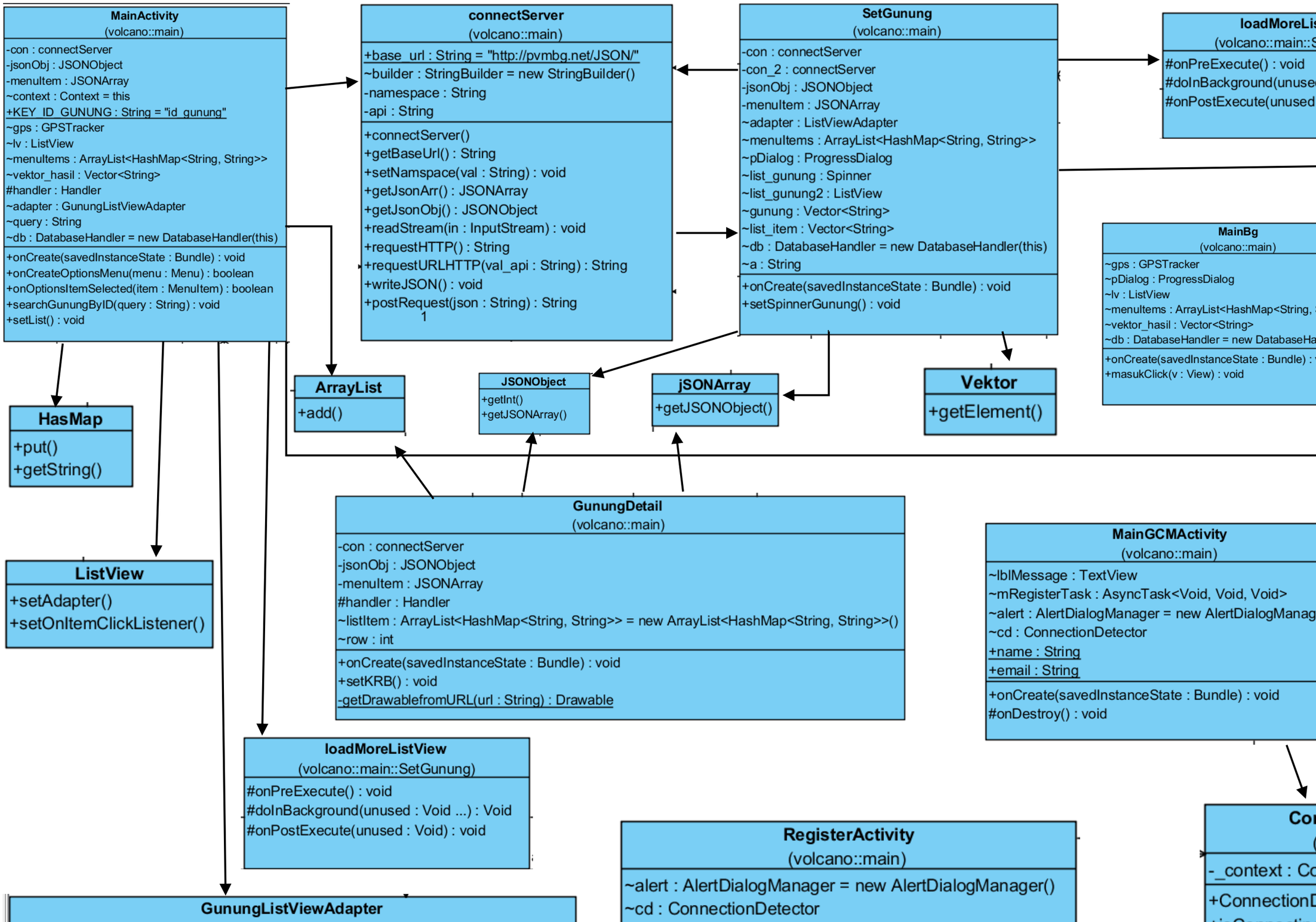
Gambar 3.25 *Activity Diagram* Registrasi Pengguna

3.1.8.4 *Class Diagram*

Class diagram mendeskripsikan jenis–jenis objek dalam sistem dan berbagai macam hubungan statis yang ada. Berikut adalah *Class diagram* dari sistem yang dibangun.

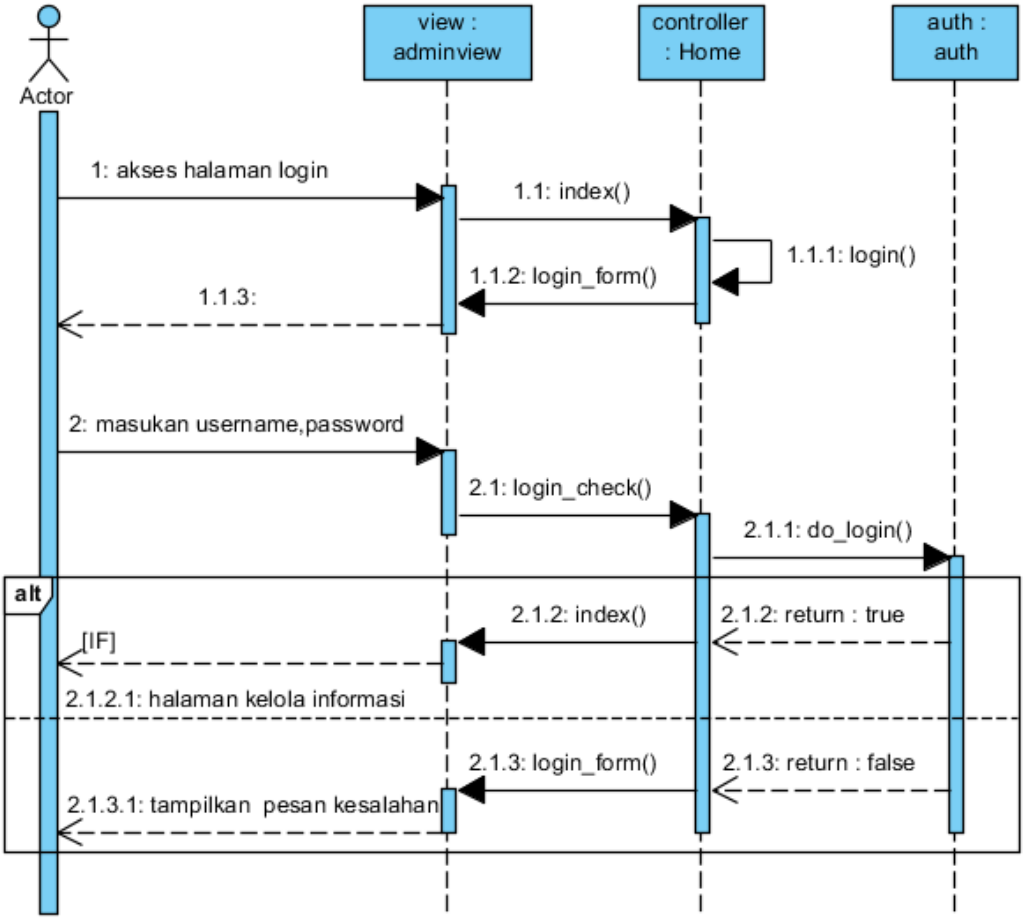


Gambar 3.26 *Class Diagram Back end*

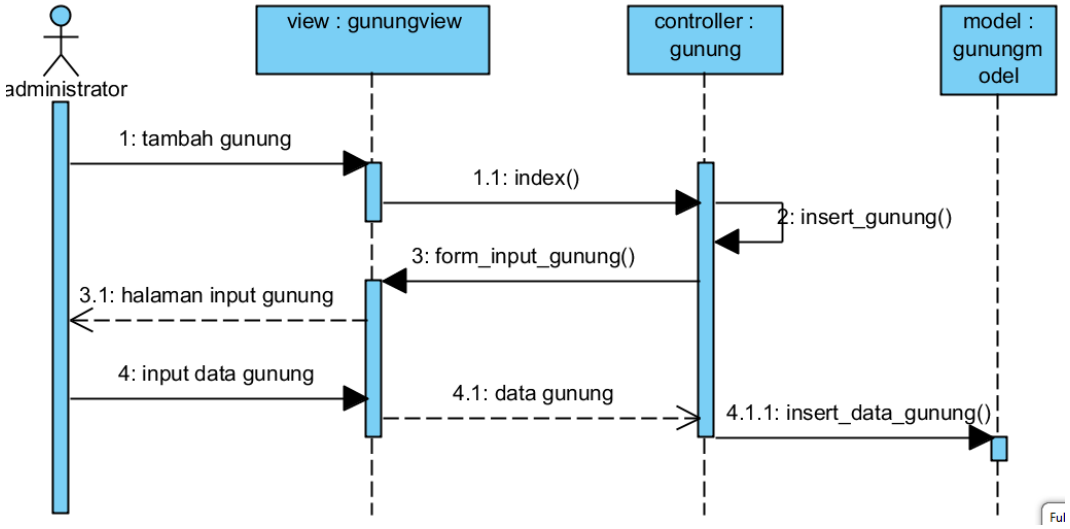

MainActivity
(volcano::main)
-con : connectServer
-jsonObj : JSONObject
-menuItem : JSONArray
~context : Context = this
+KEY_ID_GUNUNG : String = "id_gunung"
~gps : GPSTracker
~lv : ListView
~menuItems : ArrayList<HashMap<String, String>>
~vektor_hasil : Vector<String>
#handler : Handler
~adapter : GunungListViewAdapter
~query : String
~db : DatabaseHandler = new DatabaseHandler(this)
+onCreate(savedInstanceState : Bundle) : void
+onCreateOptionsMenu(menu : Menu) : boolean
+onOptionsItemSelected(item : MenuItem) : boolean
+searchGunungByID(query : String) : void
+setList() : void
connectServer
(volcano::main)
+base_url : String = "http://pvmbg.net/JSON/"
~builder : StringBuilder = new StringBuilder()
-namespace : String
-api : String
+connectServer()
+getBaseUrl() : String
+setNamspace(val : String) : void
+getJsonArr() : JSONArray
+getJsonObj() : JSONObject
+readStream(in : InputStream) : void
+requestHTTP() : String
+requestURLHTTP(val_api : String) : String
+writeJSON() : void
+postRequest(json : String) : String
1
SetGunung
(volcano::main)
-con : connectServer
-con_2 : connectServer
-jsonObj : JSONObject
-menuItem : JSONArray
~adapter : ListViewAdapter
~menuItems : ArrayList<HashMap<String, String>>
~pDialog : ProgressDialog
~list_gunung : Spinner
~list_gunung2 : ListView
~gunung : Vector<String>
~list_item : Vector<String>
~db : DatabaseHandler = new DatabaseHandler(this)
~a : String
+onCreate(savedInstanceState : Bundle) : void
+setSpinnerGunung() : void
loadMoreLis
(volcano::main::S
#onPreExecute() : void
#doInBackground(unuse
#onPostExecute(unused
MainBg
(volcano::main)
~gps : GPSTracker
~pDialog : ProgressDialog
~lv : ListView
~menuItems : ArrayList<HashMap<String,
~vektor_hasil : Vector<String>
~db : DatabaseHandler = new DatabaseHa
+onCreate(savedInstanceState : Bundle) :
+masukClick(v : View) : void
HasMap
+put()
+getString()
ArrayList
+add()
JSONObject
+getInt()
+getJSONArray()
jSONArray
+getJSONObject()
Vektor
+getElement()
ListView
+setAdapter()
+setOnItemClickListener()
GunungDetail
(volcano::main)
-con : connectServer
-jsonObj : JSONObject
-menuItem : JSONArray
#handler : Handler
~listItem : ArrayList<HashMap<String, String>> = new ArrayList<HashMap<String, String>>()
~row : int
+onCreate(savedInstanceState : Bundle) : void
+setKRB() : void
-getDrawablefromURL(url : String) : Drawable
MainGCMActivity
(volcano::main)
~lblMessage : TextView
~mRegisterTask : AsyncTask<Void, Void, Void>
~alert : AlertDialogManager = new AlertDialogManag
~cd : ConnectionDetector
+name : String
+email : String
+onCreate(savedInstanceState : Bundle) : void
#onDestroy() : void
loadMoreListView
(volcano::main::SetGunung)
#onPreExecute() : void
#doInBackground(unused : Void ...) : Void
#onPostExecute(unused : Void) : void
RegisterActivity
(volcano::main)
~alert : AlertDialogManager = new AlertDialogManager()
~cd : ConnectionDetector
Co
-_context : Co
+ConnectionD
GunungListViewAdapter

3.1.8.5 Sequence Diagram

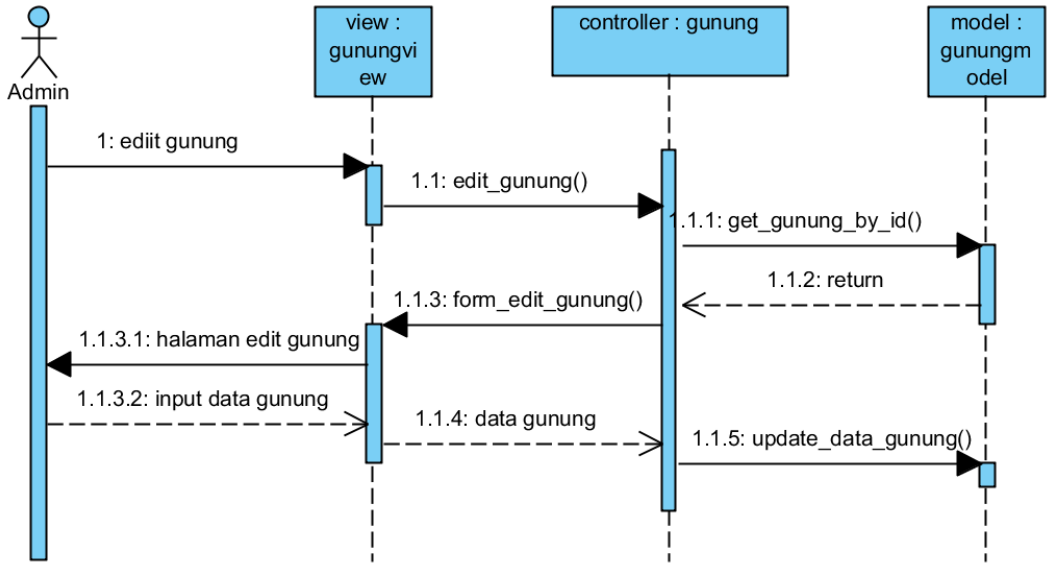
Diagram sequence menggambarkan interaksi antar objek di dalam dan di sekitar sistem berupa pesan (*message*) yang digambarkan terhadap waktu. Diagram sequence dalam sistem ini adalah sebagai berikut :



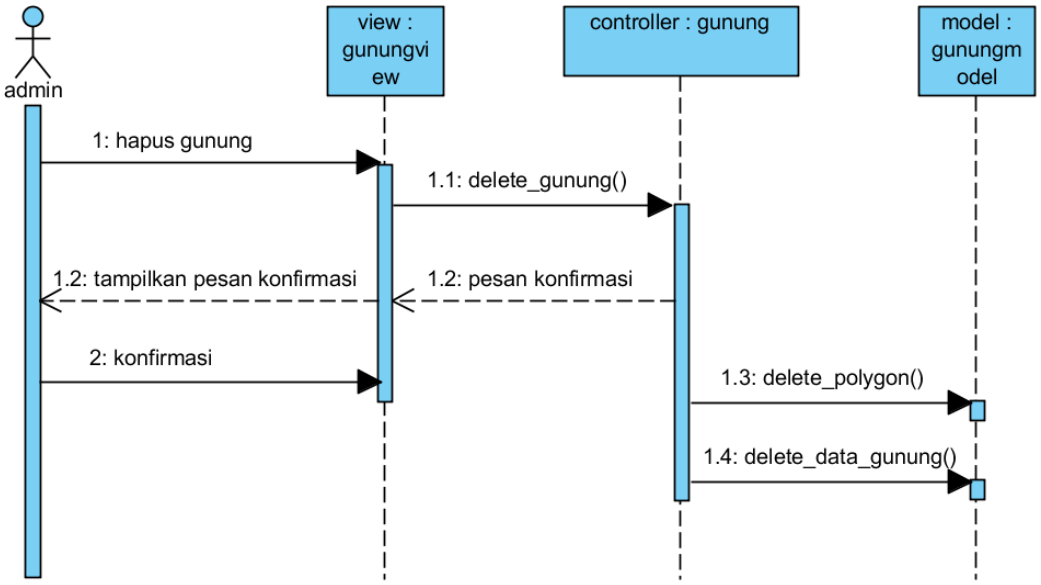
Gambar 3.28 *Sequence Diagram Login*



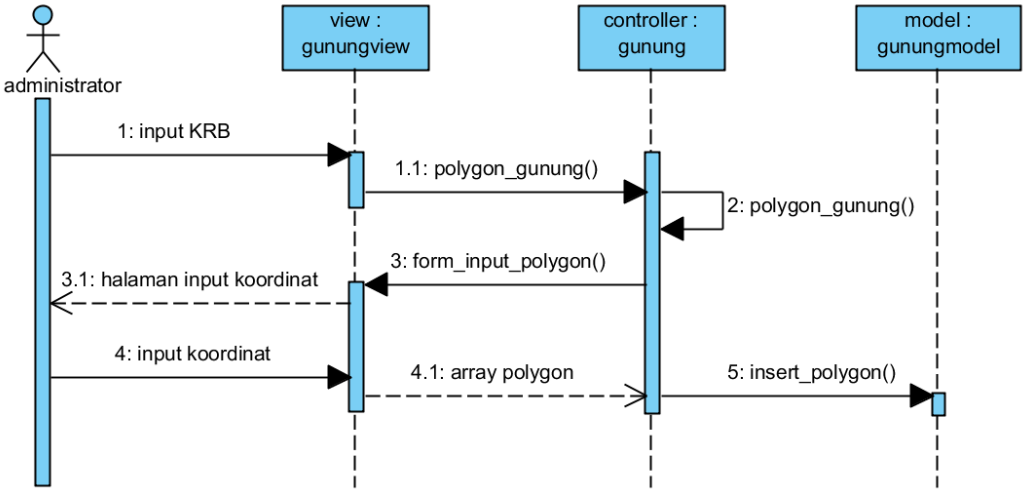
Gambar 3.29 *Sequence Diagram* Tambah Info Gunung



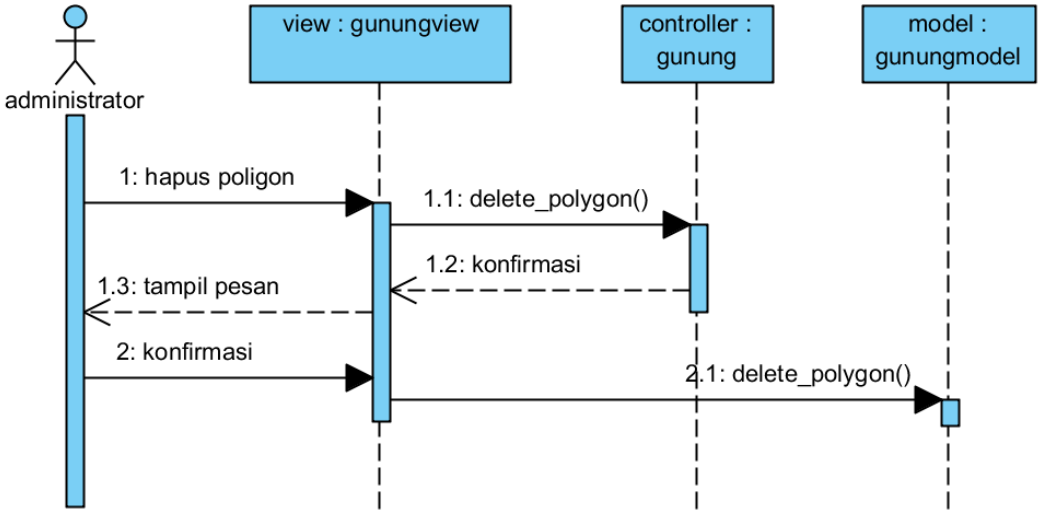
Gambar 3.30 *Sequence Diagram* Ubah Info Gunung



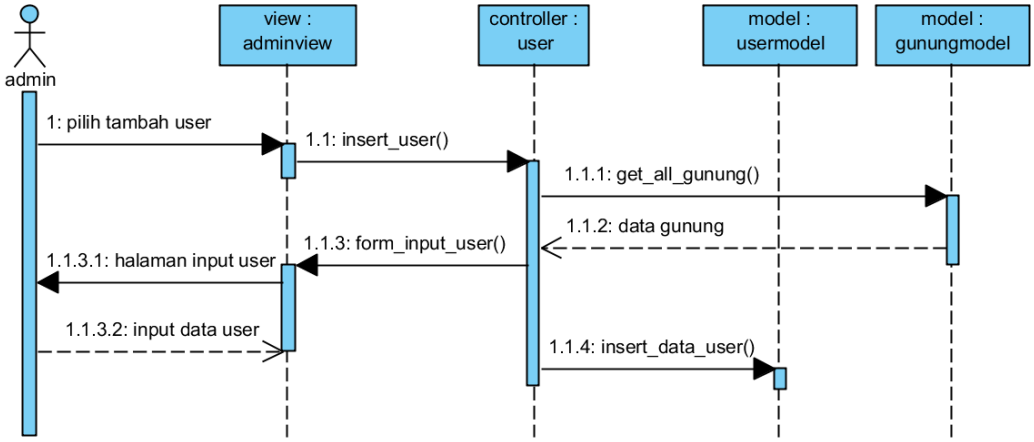
Gambar 3.31 *Sequence Diagram* Hapus Info Gunung



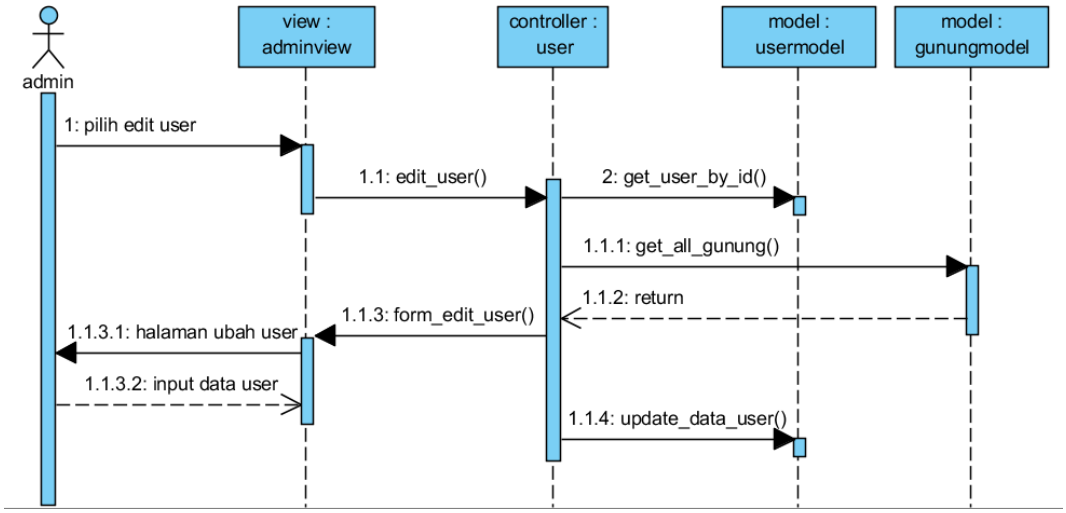
Gambar 3.32 *Sequence Diagram* Input KRB



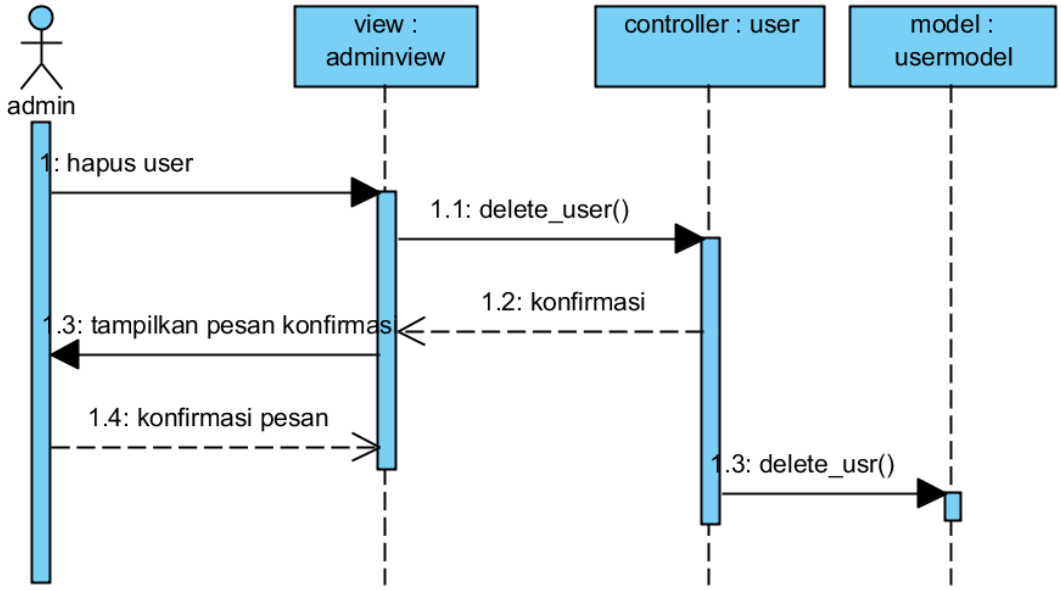
Gambar 3.33 *Sequence Diagram* Hapus KRB



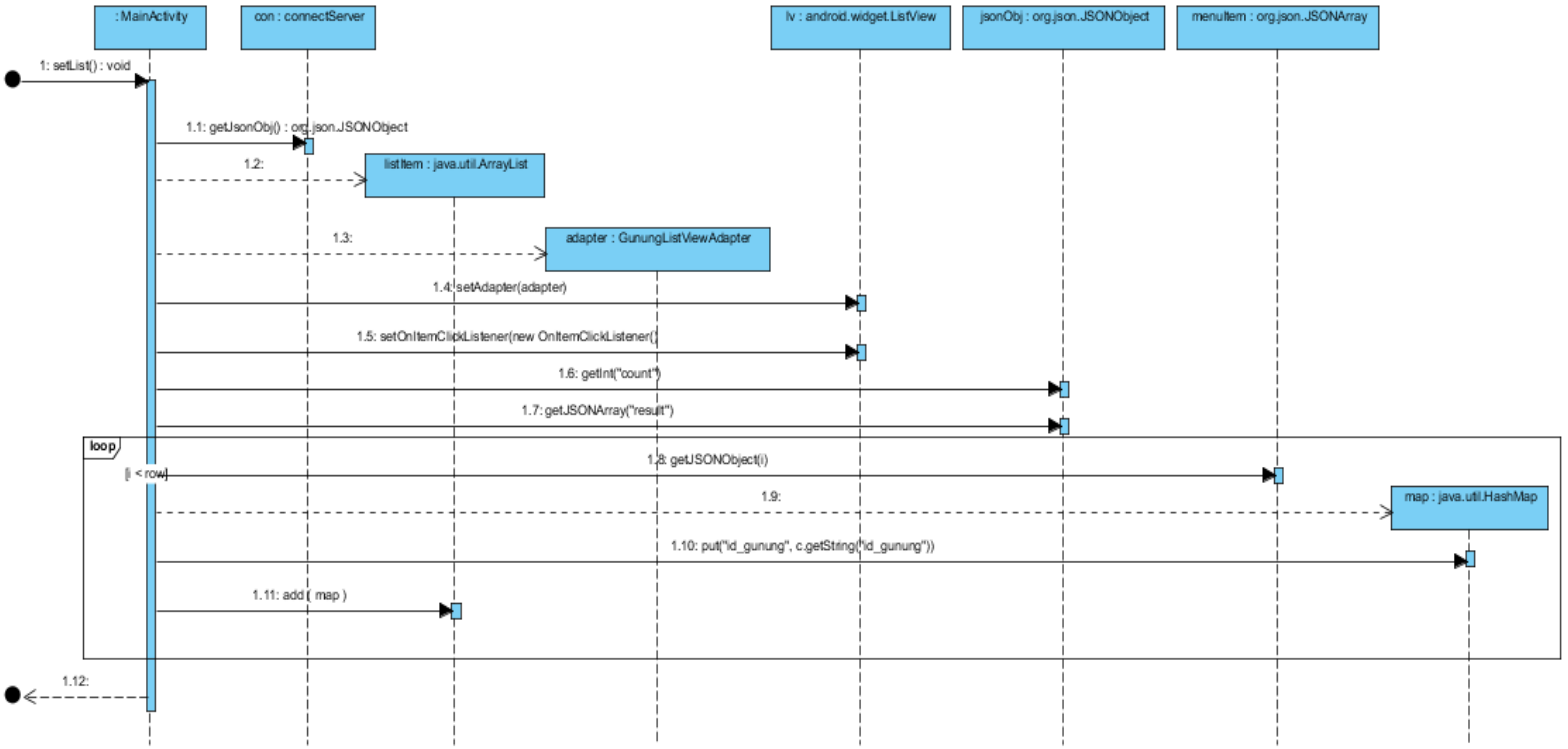
Gambar 3.34 *Activity Diagram* Tambah User



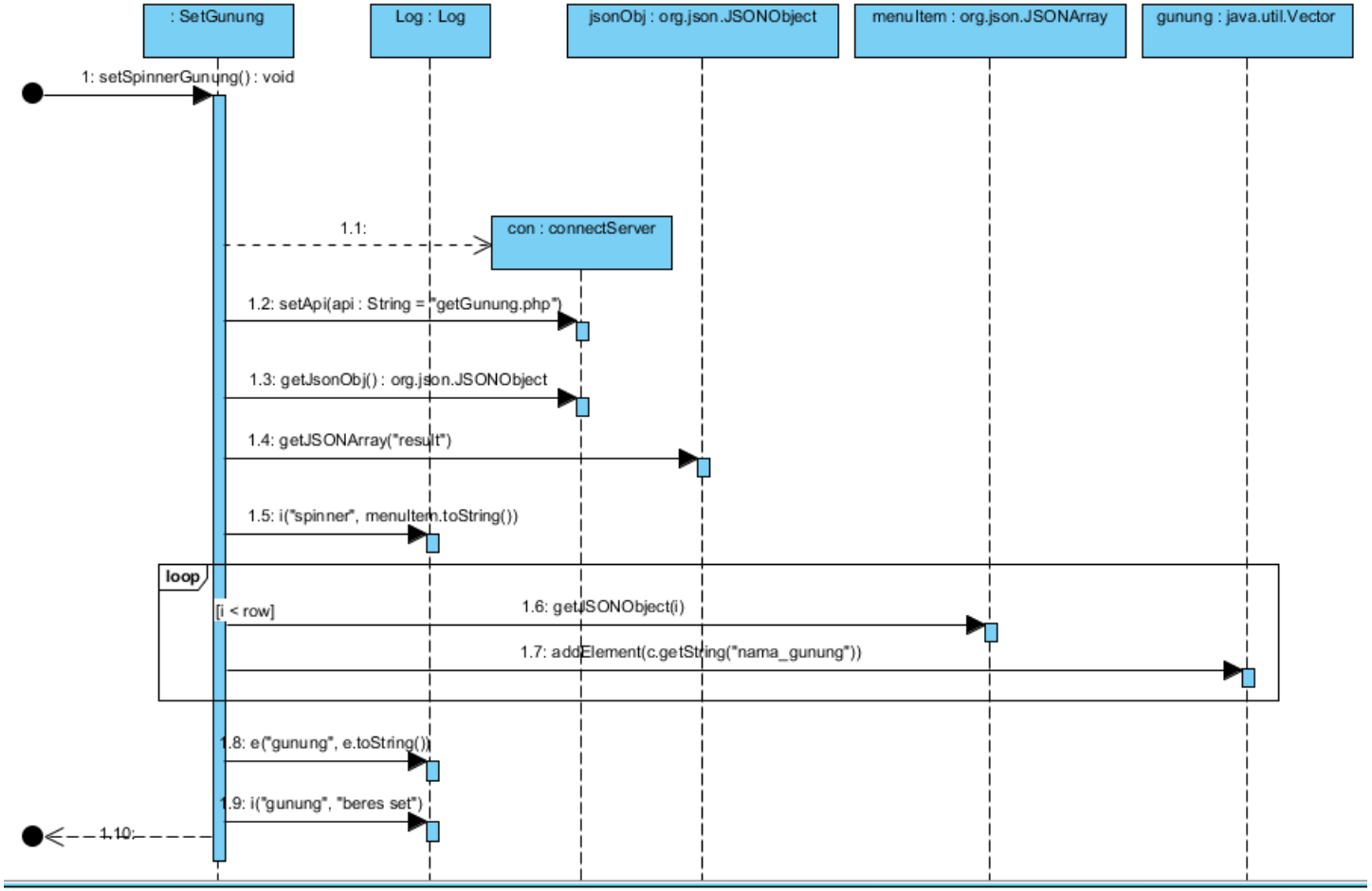
Gambar 3.35 *Sequence Diagram* Ubah User



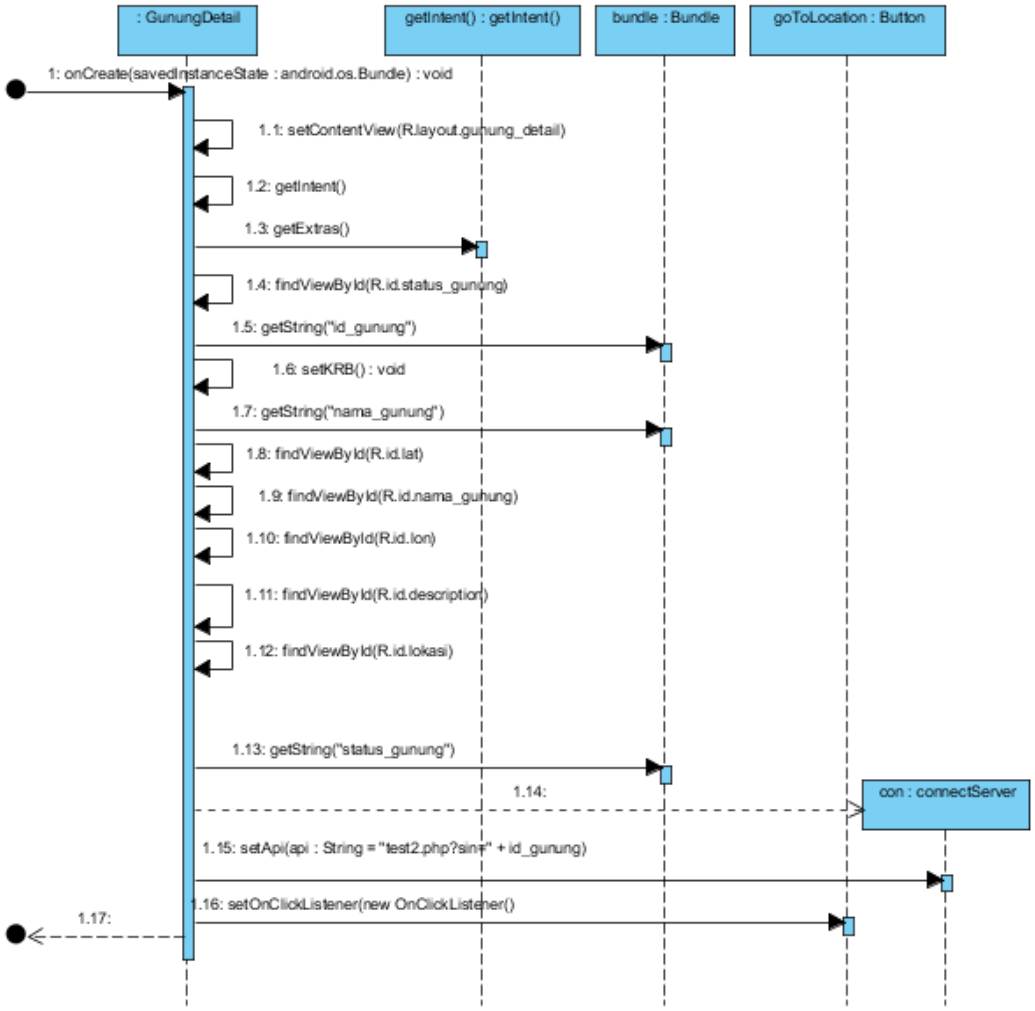
Gambar 3.36 *Sequence Diagram* Hapus User



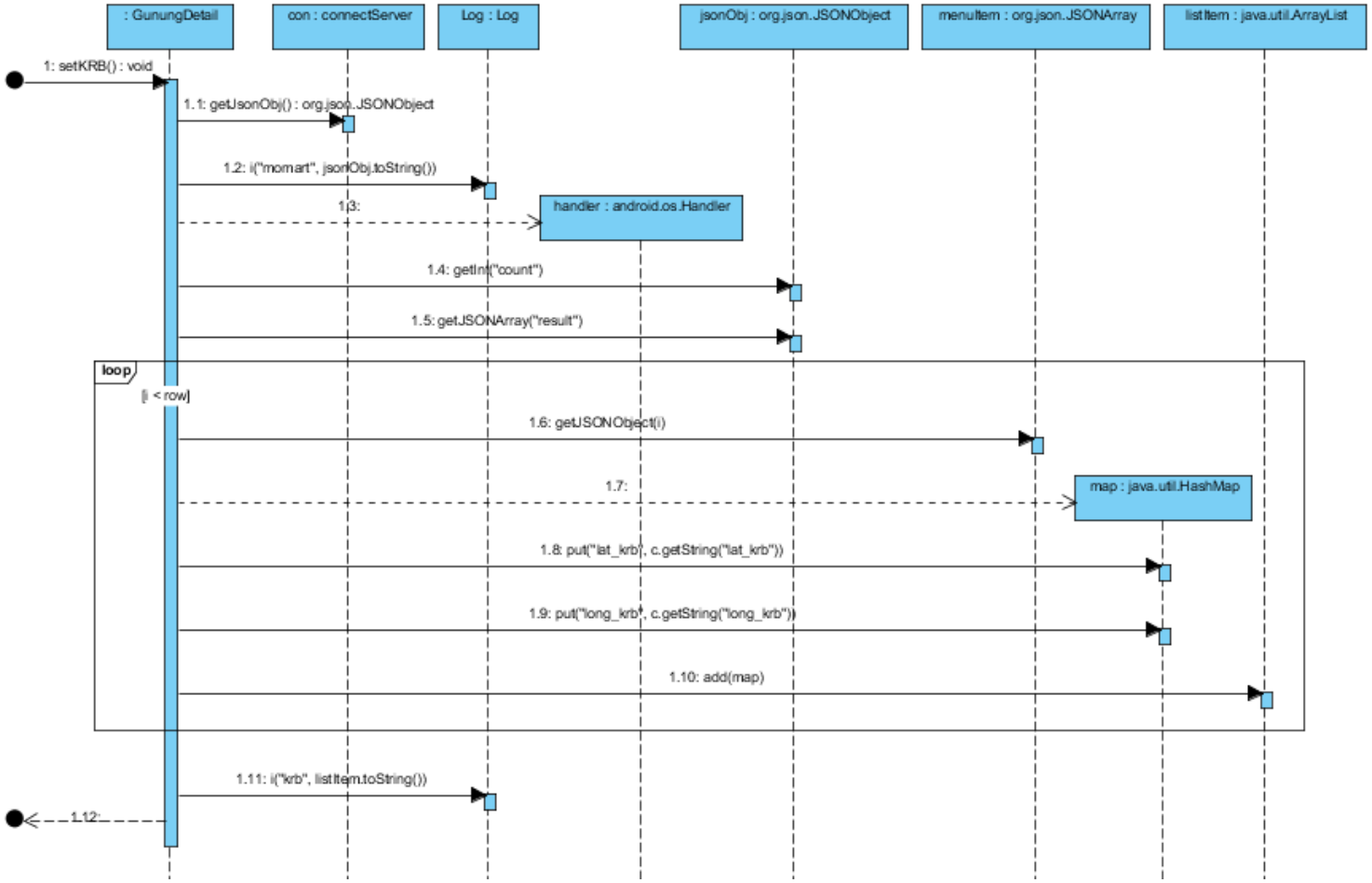
Gambar 3.37 *Sequence Diagram* Tampilkan Info Gunung



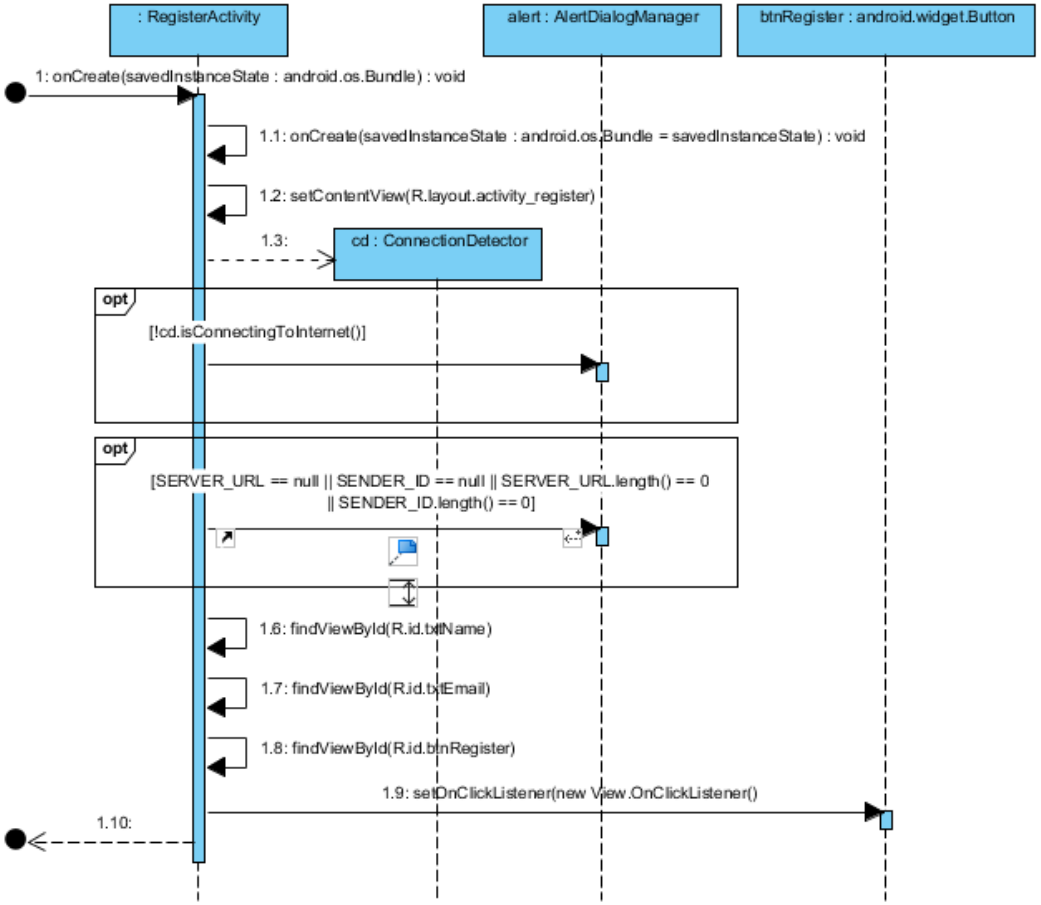
Gambar 3.38 *Sequence Diagram* Pilih Gunung



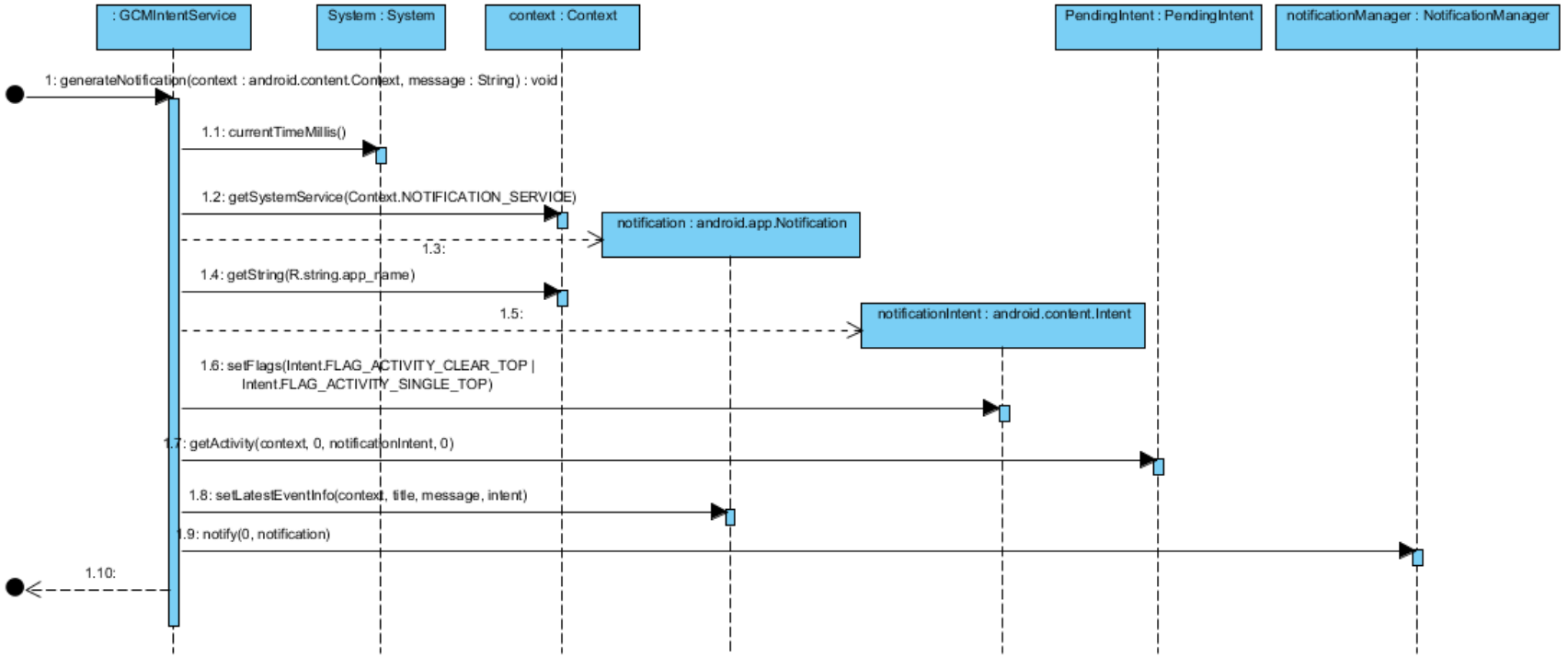
Gambar 3.39 *Sequence Diagram* Tampilkan Detail Gunung



Gambar 3.40 *Sequence Diagram* Tampilkan Peta KRB



Gambar 3.41 *Sequence Diagram* Registrasi Pengguna



Gambar 3.42 *Sequence Diagram* Tampilkan Notifikasi

3.2 Perancangan Sistem

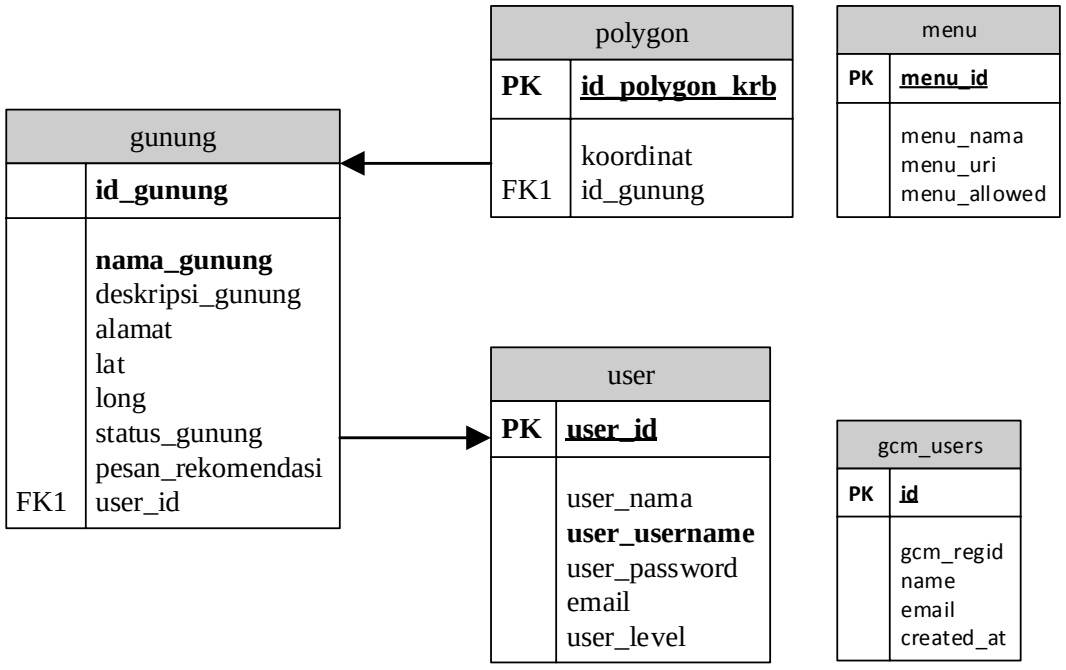
Perancangan merupakan tahapan setelah dilakukan analisis, Pada bagian ini akan dijelaskan perancangan aplikasi yang akan dikembangkan untuk menggambarkan perbedaan antara sistem yang sedang berjalan dengan sistem yang akan dikembangkan.

3.2.1 Perancangan Data

Perancangan data merupakan tahapan membangun keterkaitan antar data sehingga tercipta basis data. Untuk menggambarkan digunakan skema relasi dan struktur tabel.

3.2.1.1 Skema Relasi

Skema relasi merupakan rangkaian hubungan antara dua tabel atau lebih pada sistem basis data. Skema relasi pada *database* aplikasi backend bisa dilihat pada Gambar 3.43



Gambar 3.43 Skema Relasi

3.2.1.2 Struktur Tabel

Struktur tabel menggambarkan detail tabel yang berisi *field*, tipe data, panjang data, dan keterangann lainnya, berikut adalah struktur tabel yang akan dibangun.

1. Struktur Tabel Aplikasi *back end*

Tabel 3.24 Struktur Tabel gunung

Nama Field	Data Type	Size	Description
id_gunung	int	11	Primary Key
nama_gunung	varchar	35	Unique
deskripsi_gunung	text	-	
Lat	decimal	-	
Long	decimal	-	
status_gunung	varchar	10	
pesan_rekomendasi	text	-	
user_id	int	11	Foreign Key dari tabel user (user_id)

Tabel 3.25 Struktur Tabel User

Nama Field	Data Type	Size	Description
user_id	int	11	Primary Key
user_nama	varchar	100	
user_username	varchar	100	Unique
user_password	varchar	100	
Email	varchar	200	
user_level	varchar	8	

Tabel 3.26 Struktur Tabel polygon

Nama Field	Data Type	Size	Description
id_polygon_krb	int	5	Primary Key
koordinat	polygon	-	
id_gunung	int	8	Foreign Key dari tabel gunung (id_gunung)

Tabel 3.27 Struktur Tabel gcm_users

Nama Field	Data Type	Size	Description
Id	int	11	Foreign Key
gcm_regid	text	-	
name	varchar	50	
email	varchar	255	
created_at	timestamp	-	
id_gunung	int	8	Foreign Key dari tabel gunung (id_gunung)

Tabel 3.28 Struktur Tabel Menu

Nama Field	Data Type	Size	Description
menu_id	int	11	
menu_nama	varchar	100	
menu_uri	varchar	100	
menu_allowed	varchar	100	

2. Struktur Tabel Aplikasi *Front end*

Tabel 3.29 Struktur Tabel gunung_table

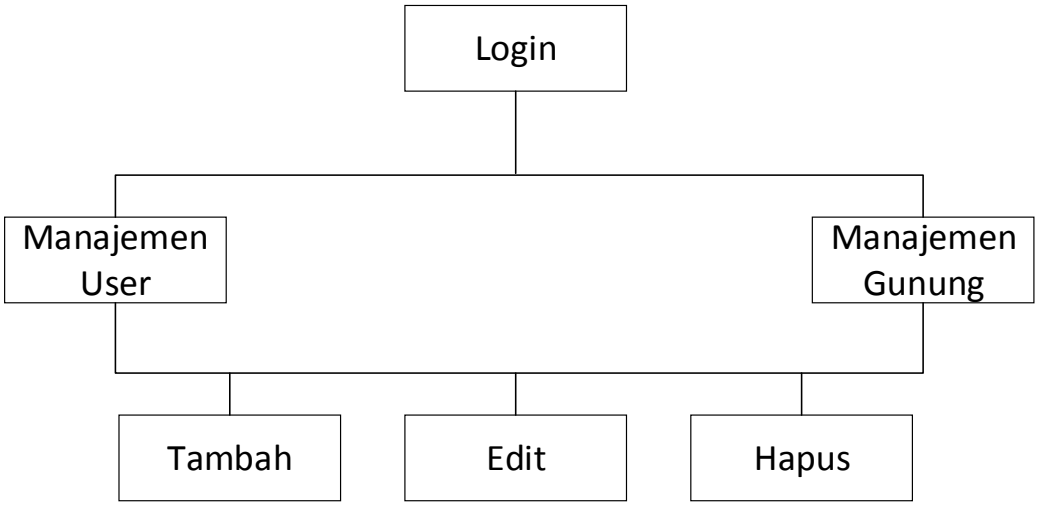
Nama Field	Data Type	Size	Description
KEY_ID	int	11	Primary Key
KEY_SQL	text	-	

3.2.2 Perancangan Struktur Menu

Perancangan struktur menu merupakan gambaran jalur pemakaian aplikasi sehingga aplikasi yang mudah dipahami dan mudah digunakan.

1. Struktur Menu Admin

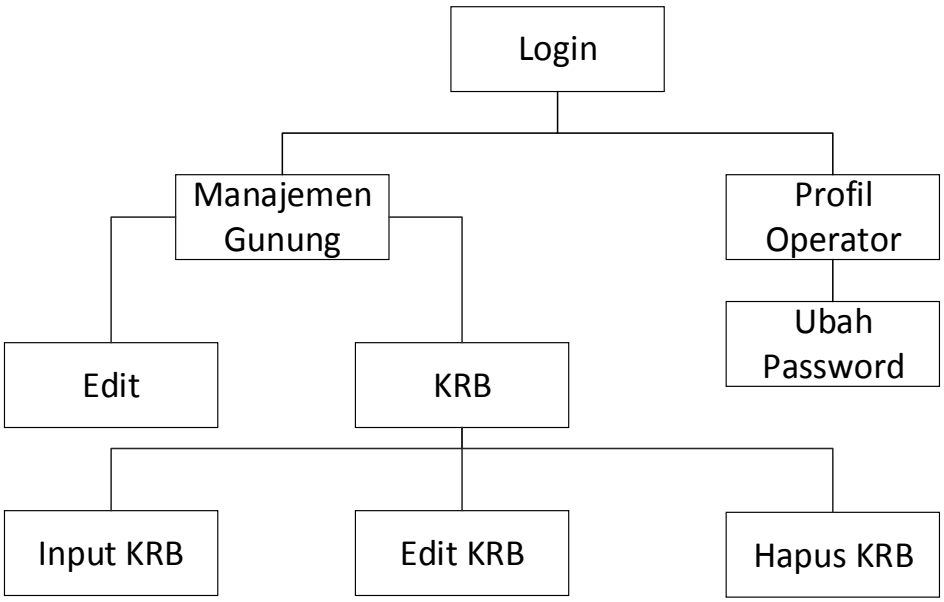
Perancangan menu admin untuk *back end* aplikasi bisa dilihat pada Gambar 3.44



Gambar 3.44 Struktur Menu Admin

2. Struktur Menu Operator

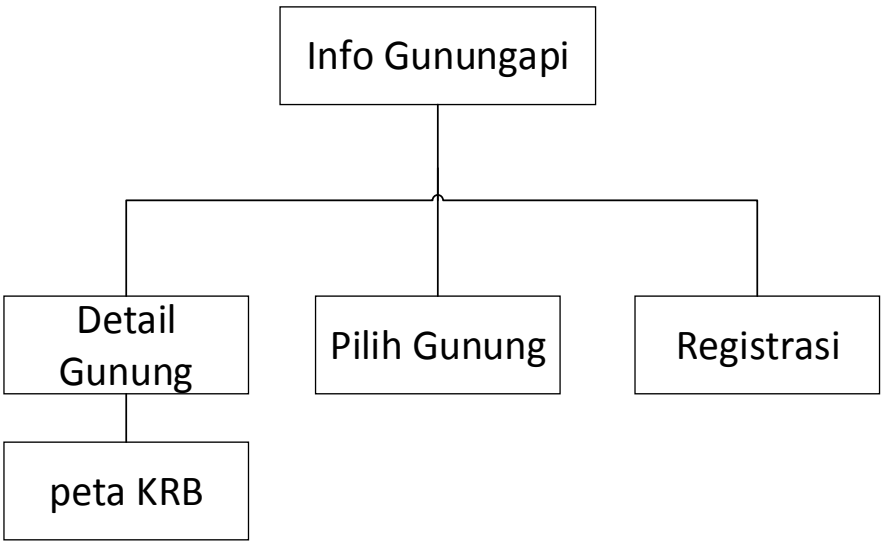
Perancangan menu operator untuk *back end* aplikasi bisa dilihat pada Gambar 3.45



Gambar 3.45 Struktur Menu User (Operator)

3. Struktur Menu Pengguna

Perancangan menu admin untuk *back end* aplikasi bisa dilihat pada Gambar 3.46



Gambar 3.46 Struktur Menu Pengguna Umum

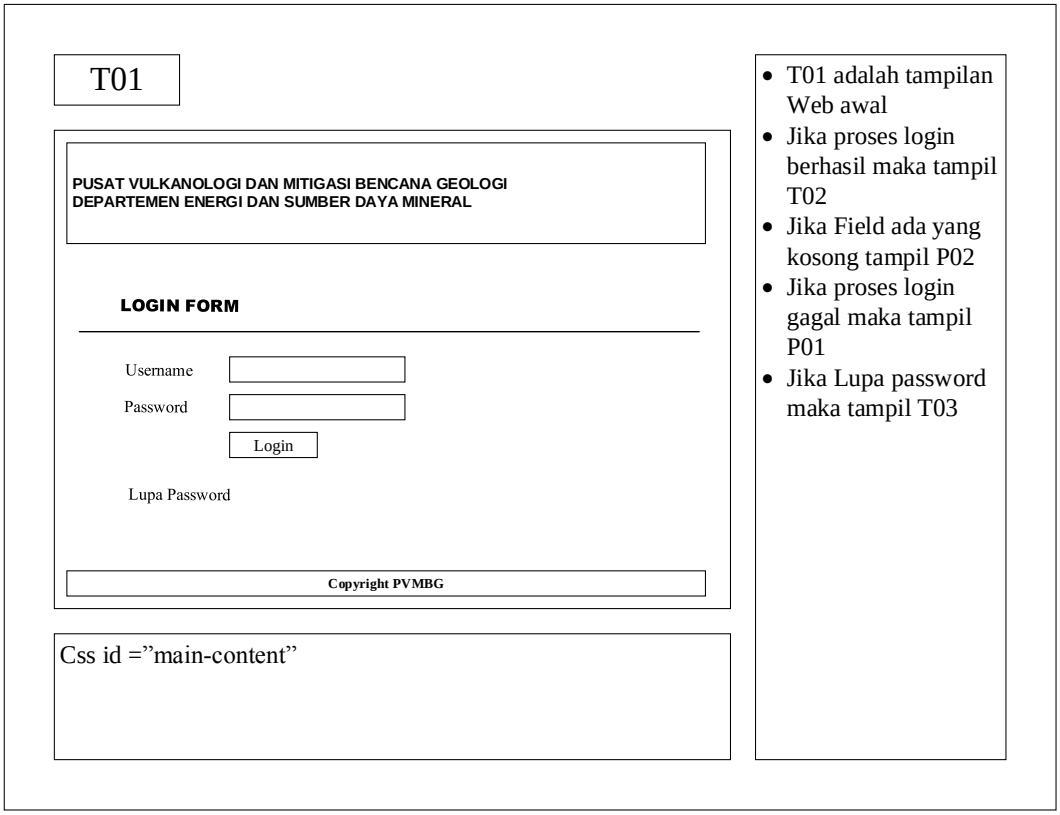
3.2.3 Perancangan Antarmuka

Interface atau antar muka merupakan tampilan dari suatu aplikasi yangberperan sebagai media komunikasi antara aplikasi dan pengguna.

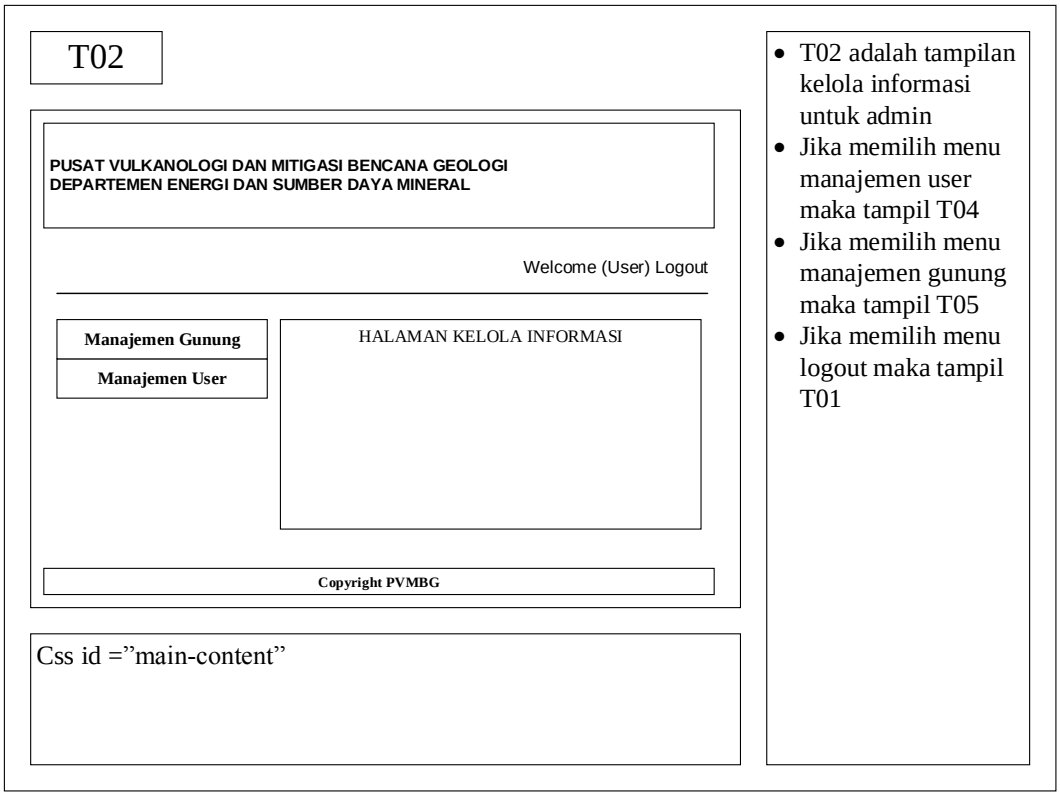
3.2.3.1 Perancangan *Form*

Sistem yang akan dibangun diharapkan menyediakan *interface* yang mudah dipahami dan digunakan oleh pengguna, berikut perancangan *interface* bagi tiap pengguna.

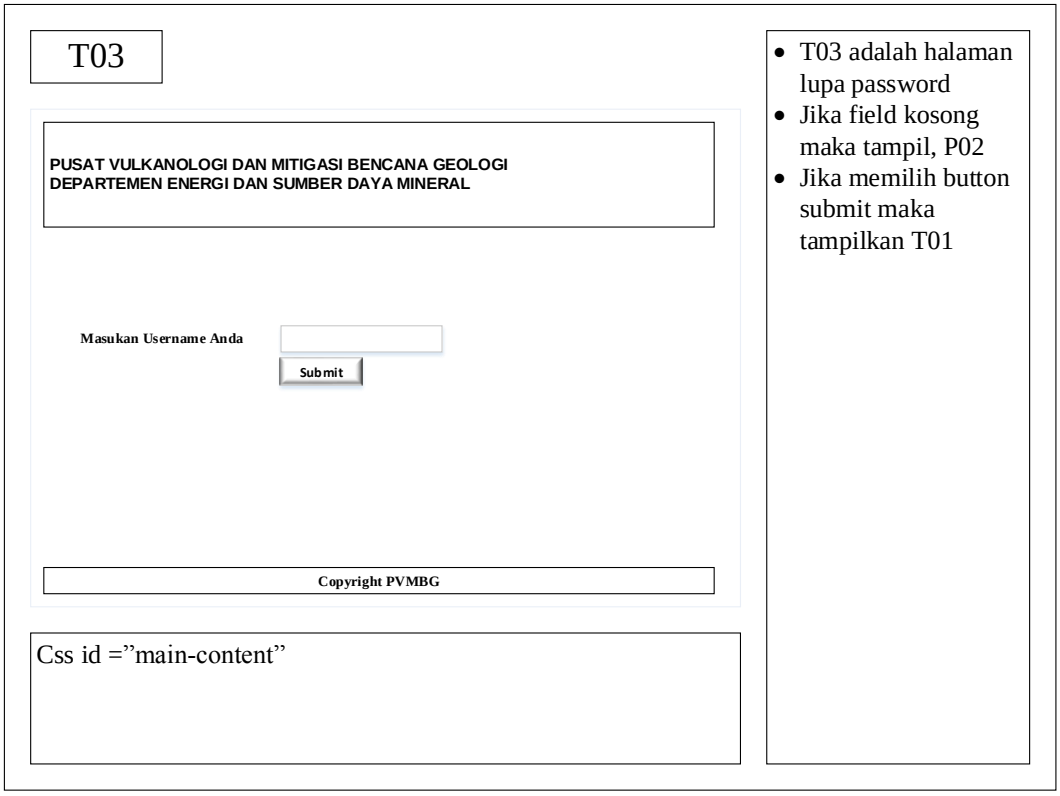
1. Peranccangan Antarmuka *Backend*



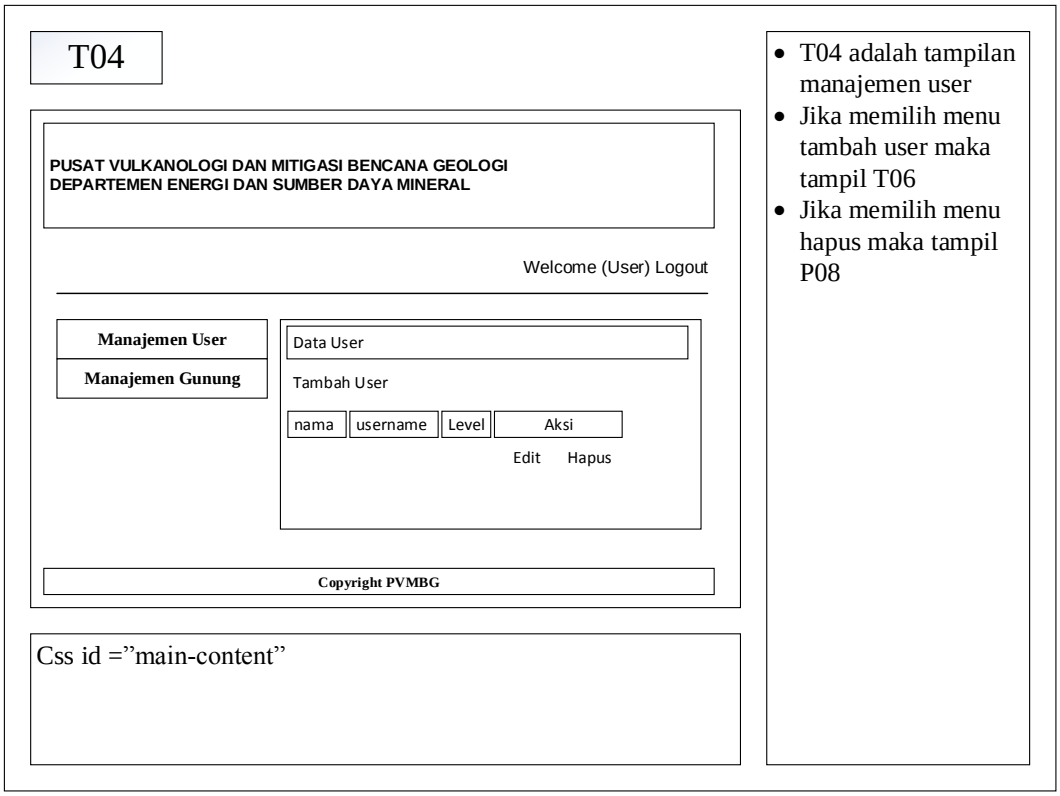
Gambar 3.47 Perancangan Form Login



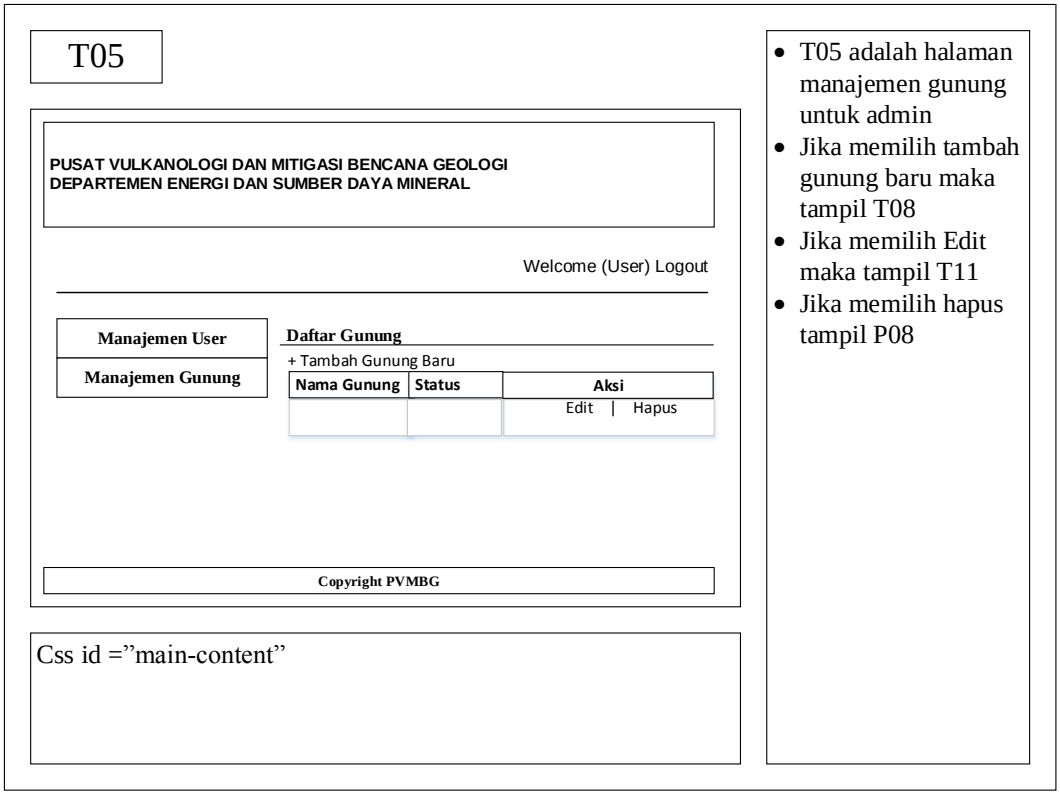
Gambar 3.48 Perancangan Kelola Informasi Admin



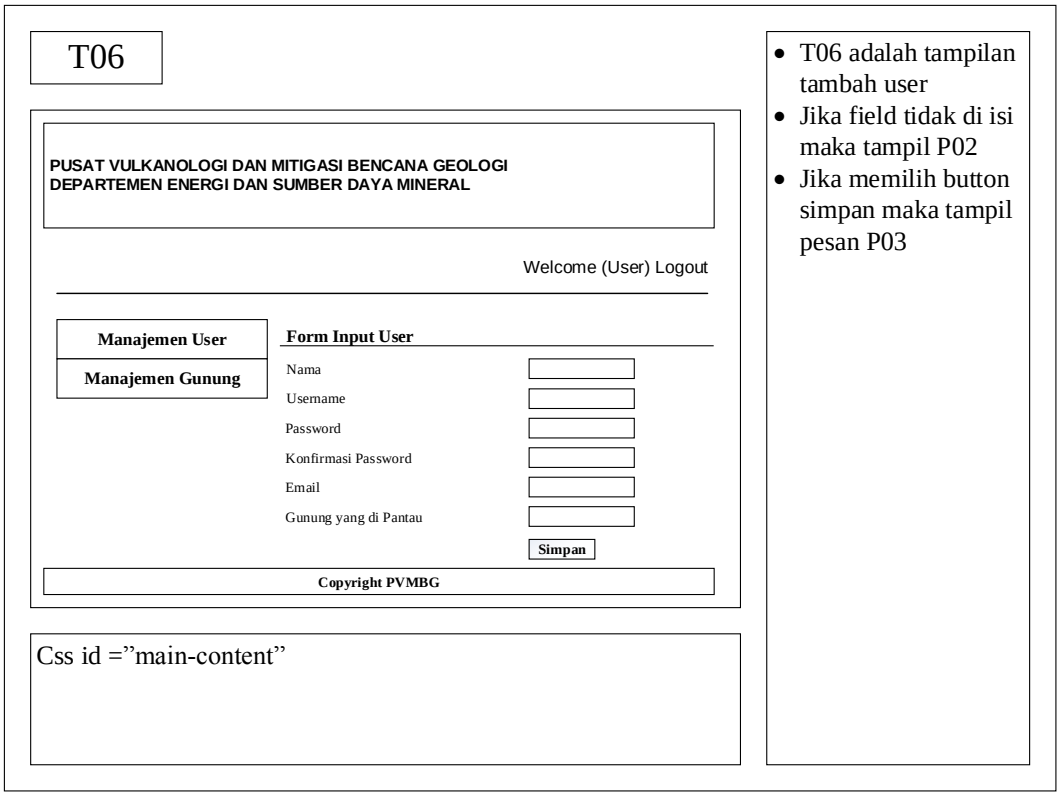
Gambar 3.49 Perancangan Lupa Password



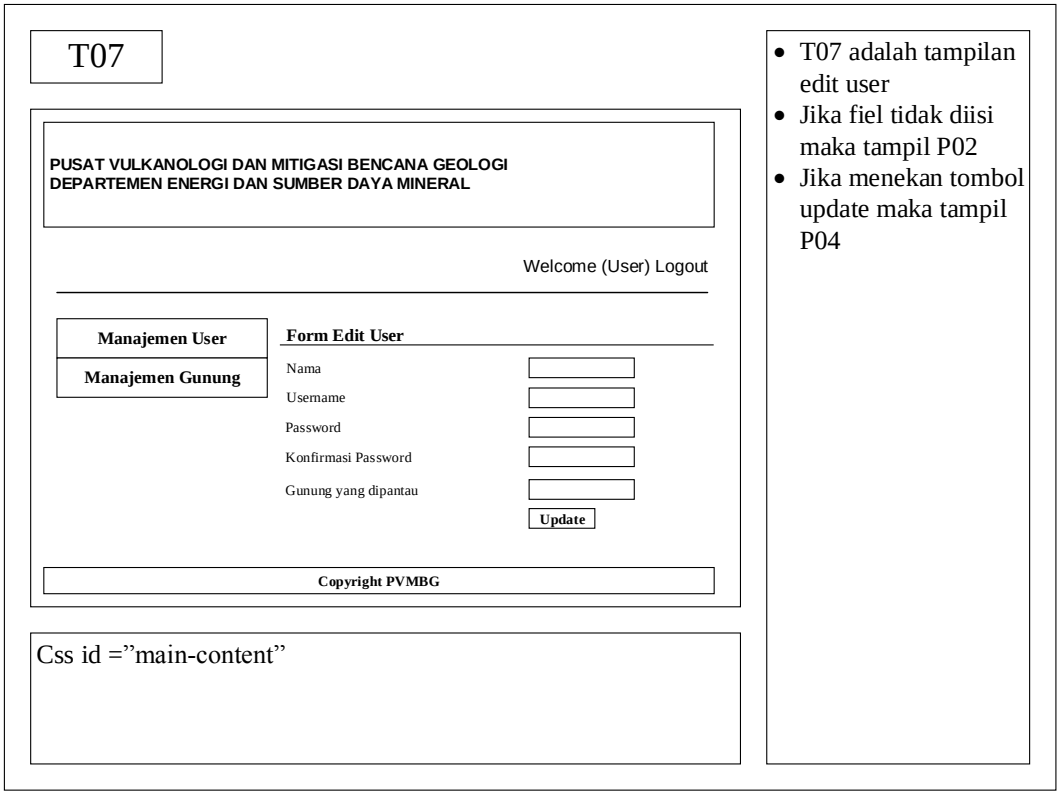
Gambar 3.50 Perancangan Manajemen User



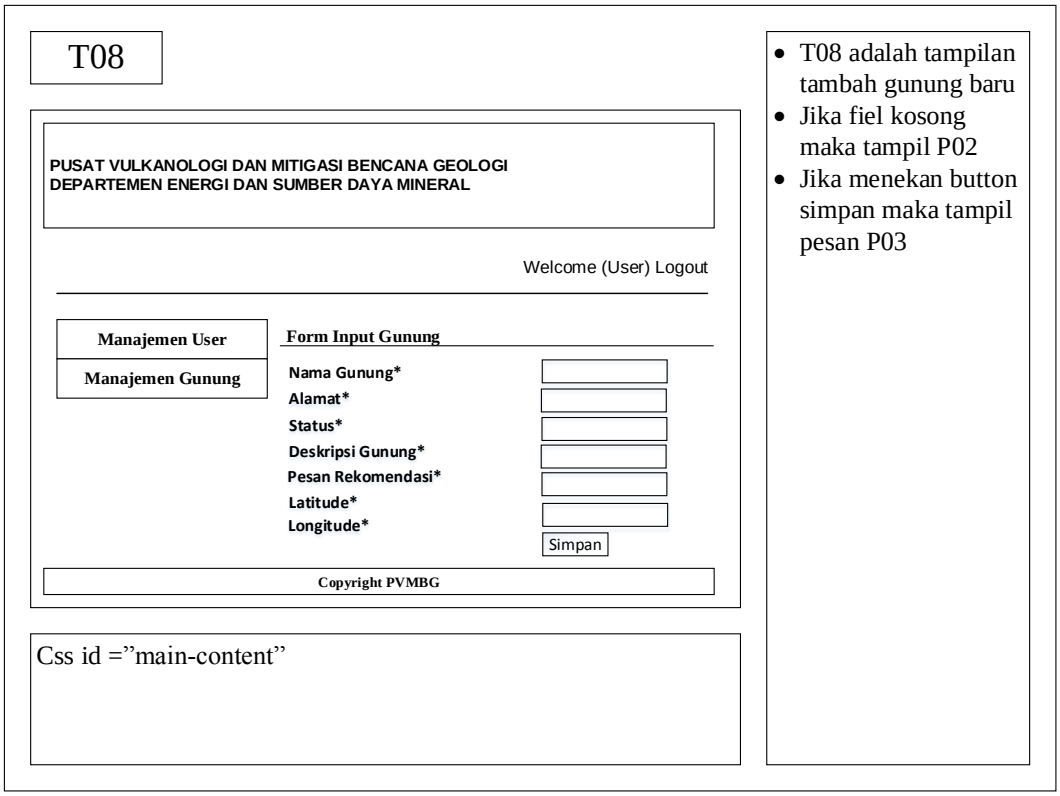
Gambar 3.51 Perancangan Manajemen Gunung



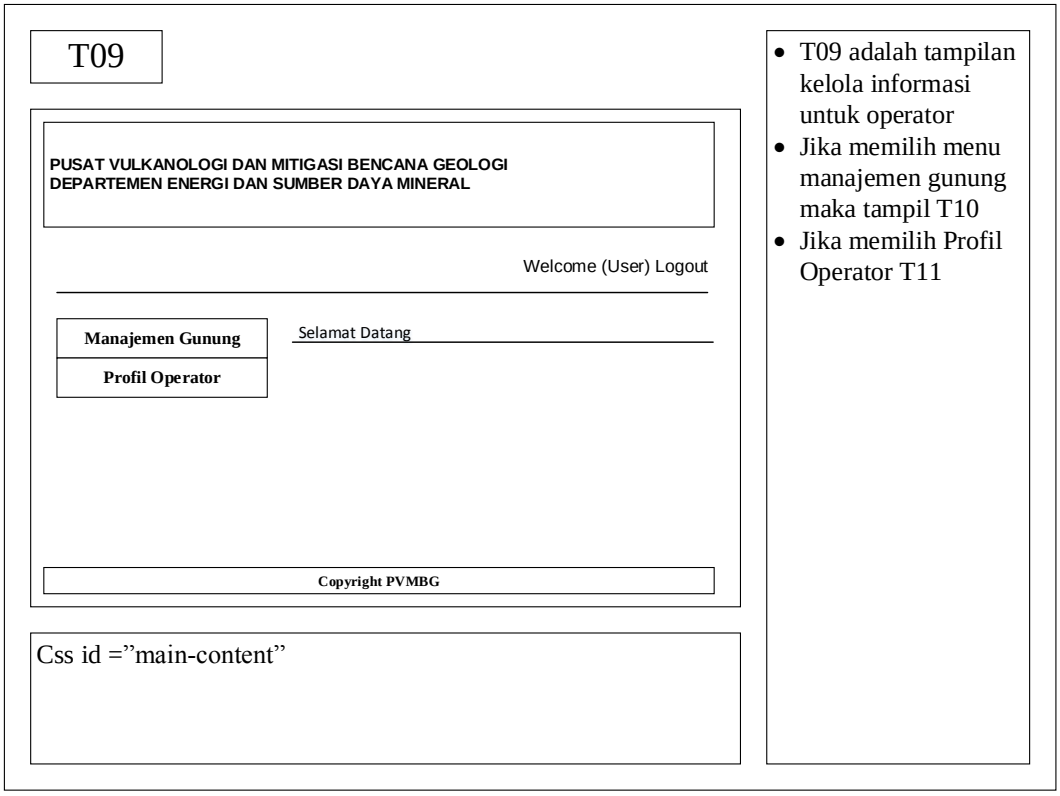
Gambar 3.52 Perancangan Tambah User



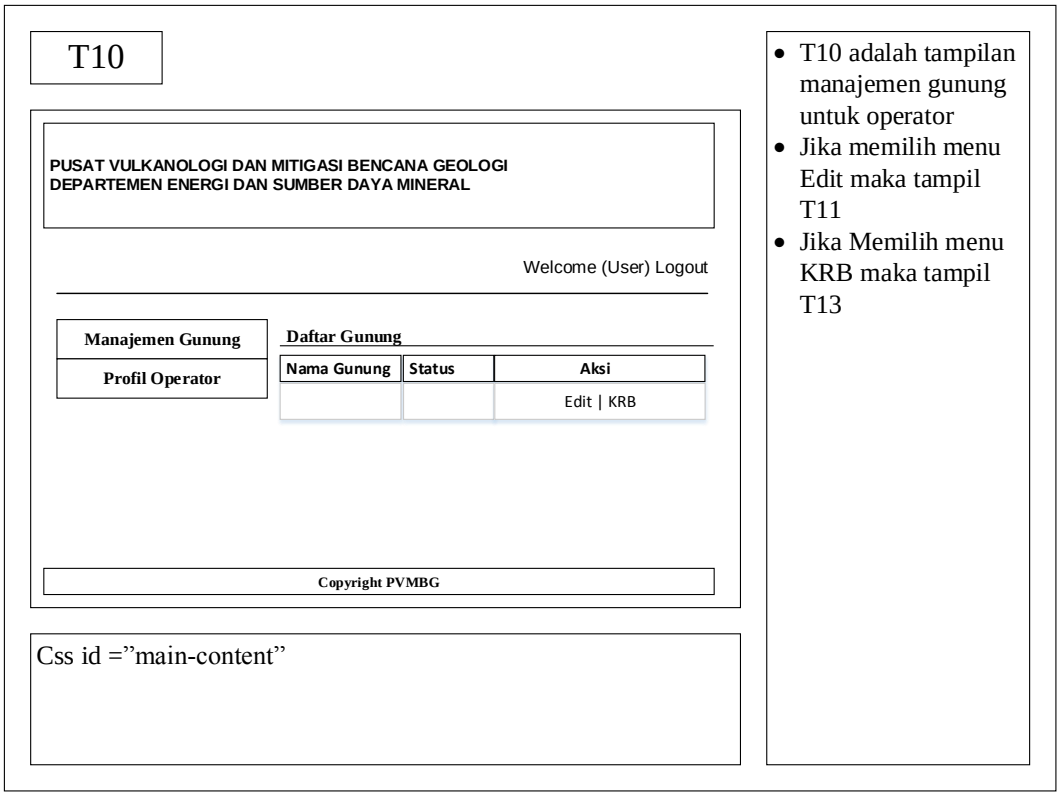
Gambar 3.53 Perancangan Ubah User



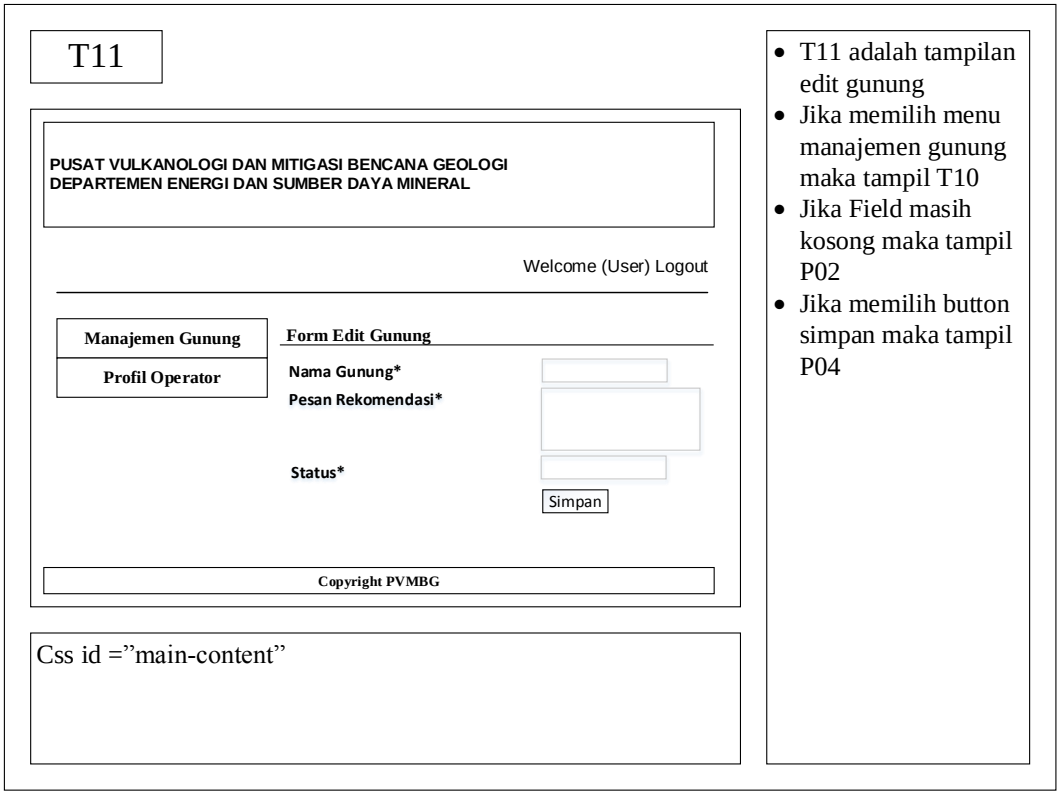
Gambar 3.54 Perancangan Tambah Gunung



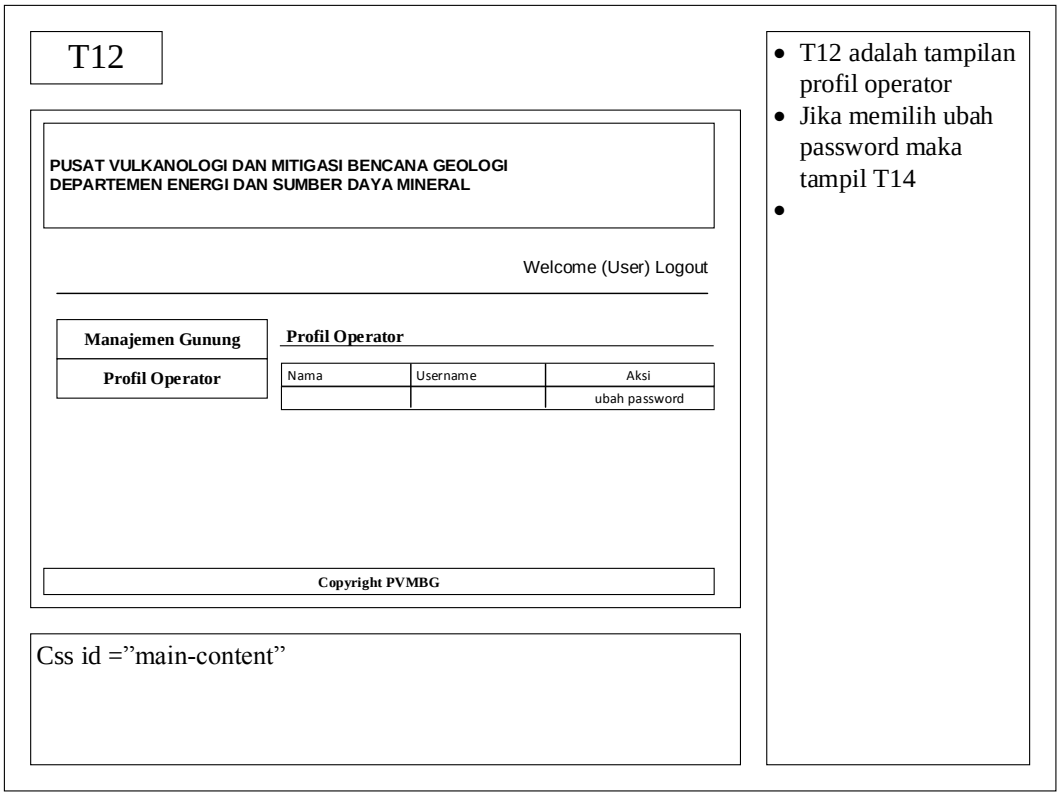
Gambar 3.55 Perancangan Kelola Informasi Operator



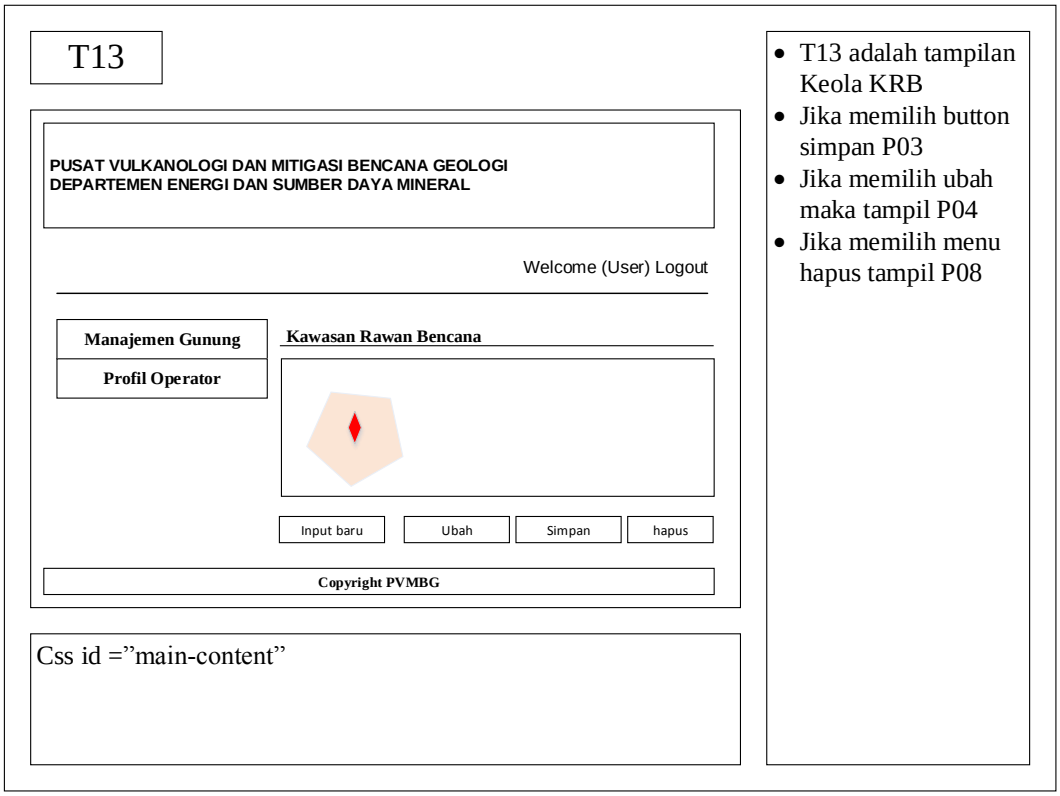
Gambar 3.56 Perancangan Manajemen Gunung Operator



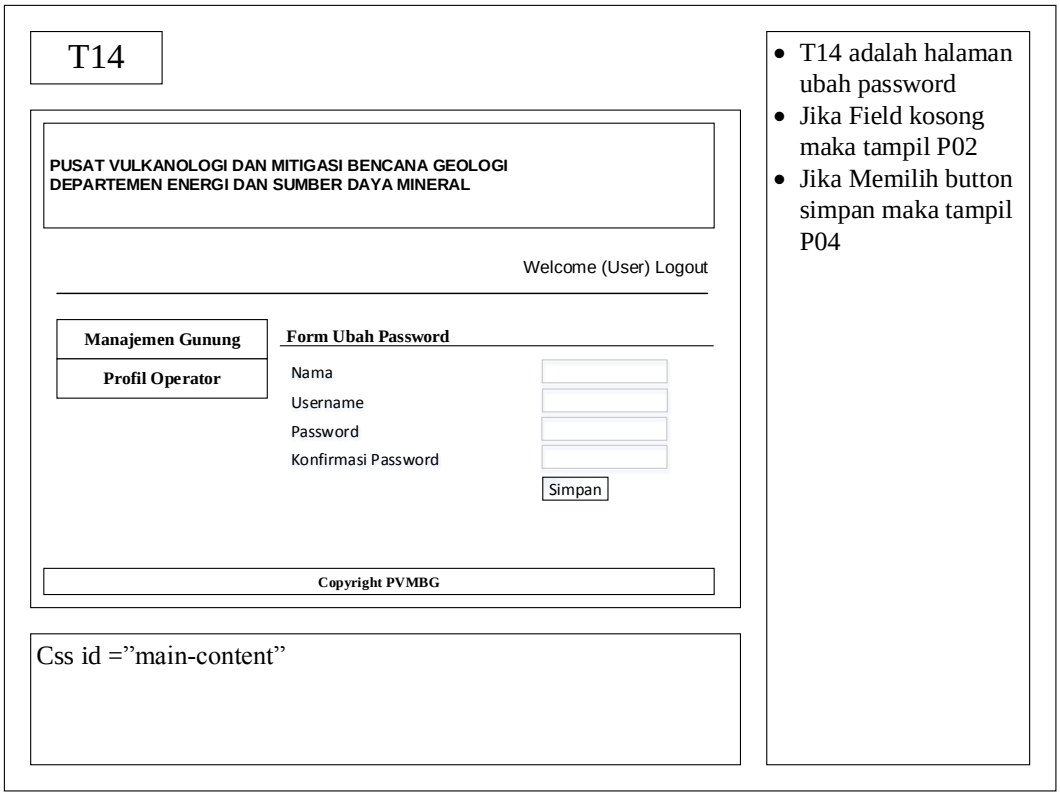
Gambar 3.57 Perancangan Ubah Gunung Operator



Gambar 3.58 Perancangan Profil Operator

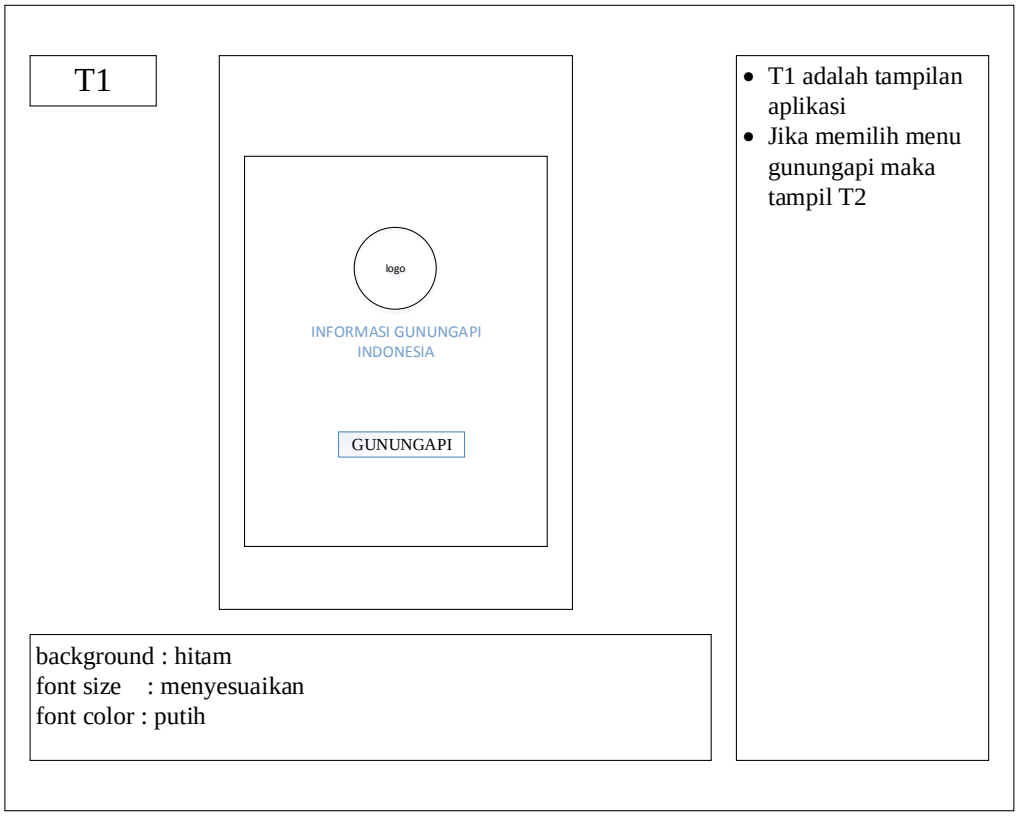


Gambar 3.59 Perancangan Kelola KRB

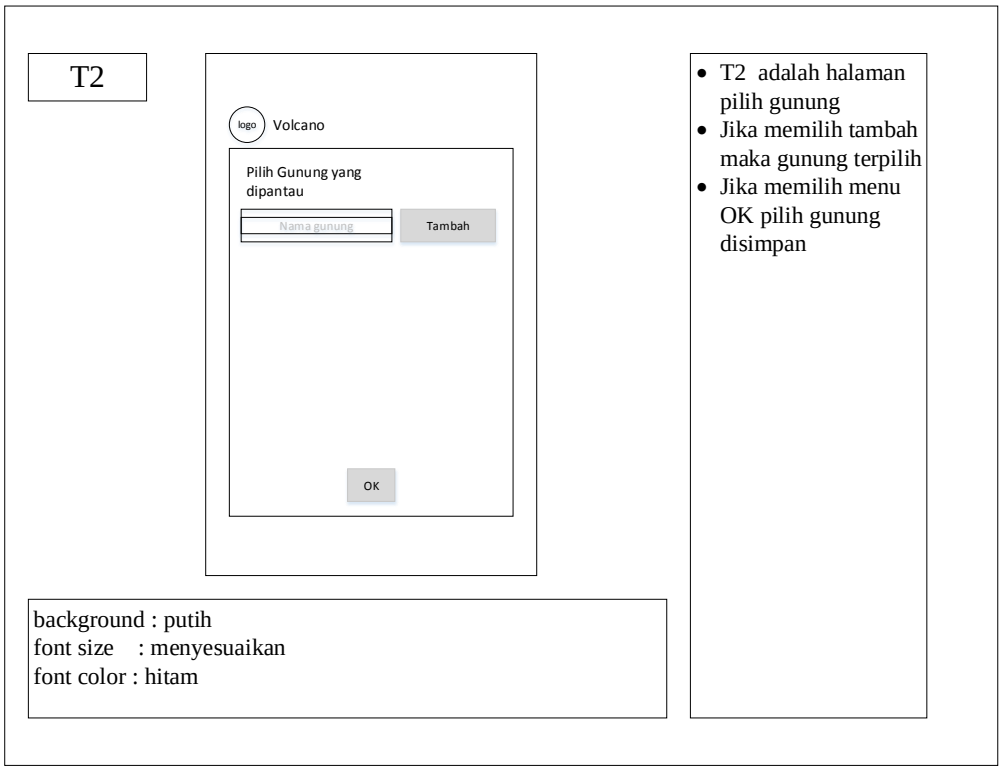


Gambar 3.60 Perancangan Ubah Password

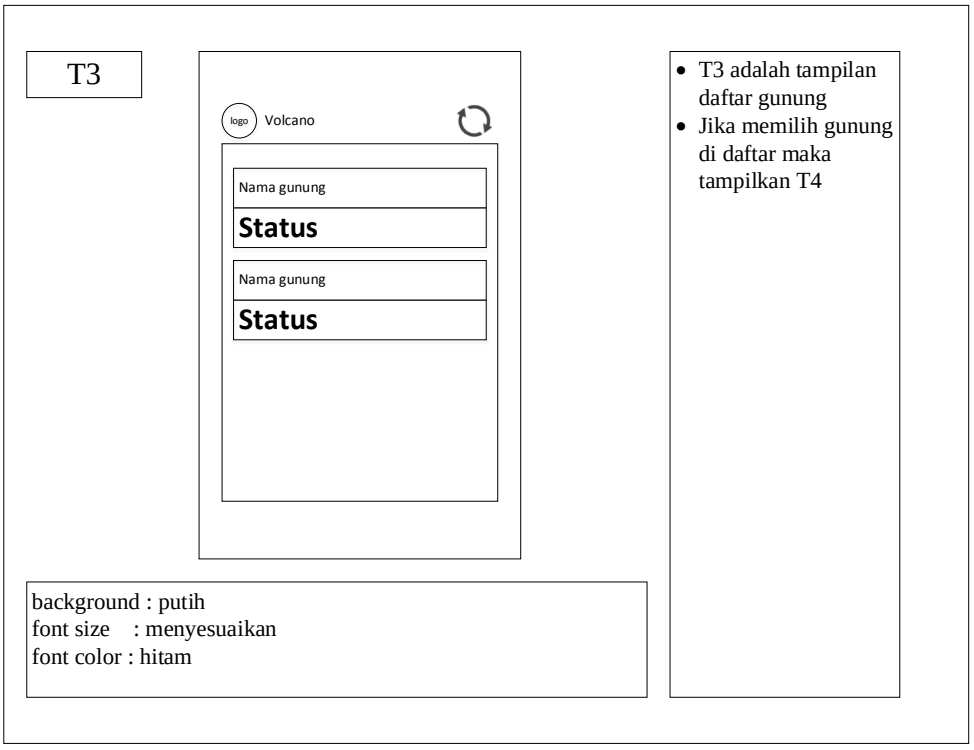
2. Tampilan Antarmuka *Frontend*



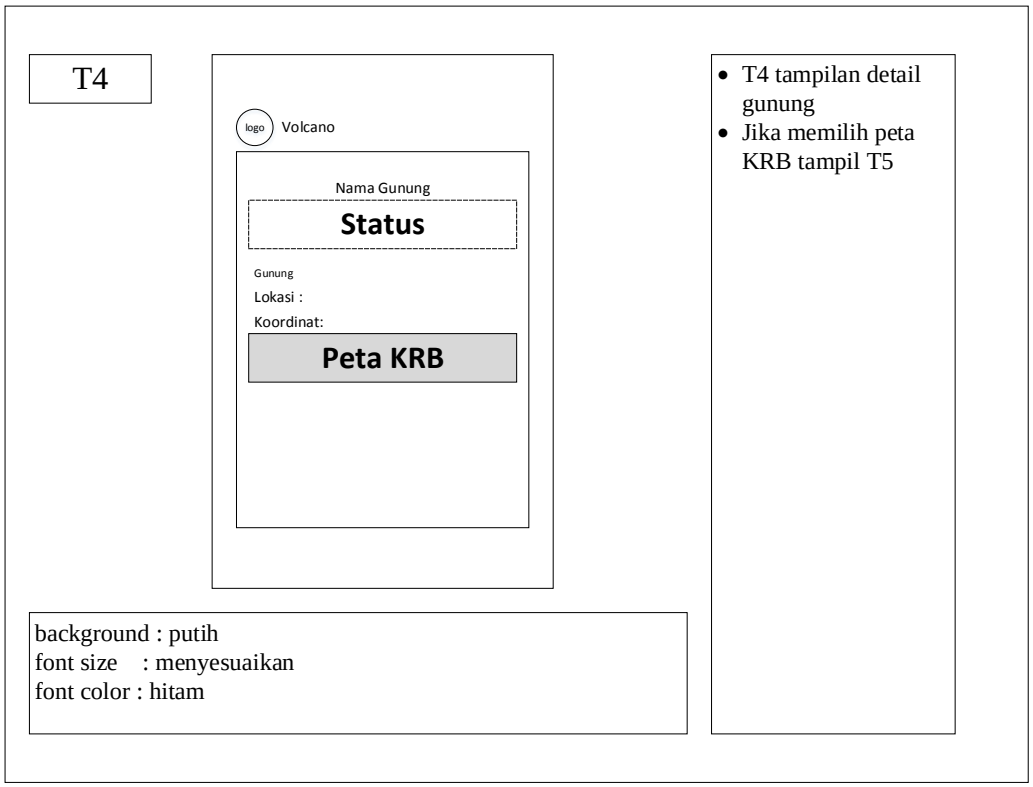
Gambar 3.61 Perancangan Tampilan Awal Aplikasi



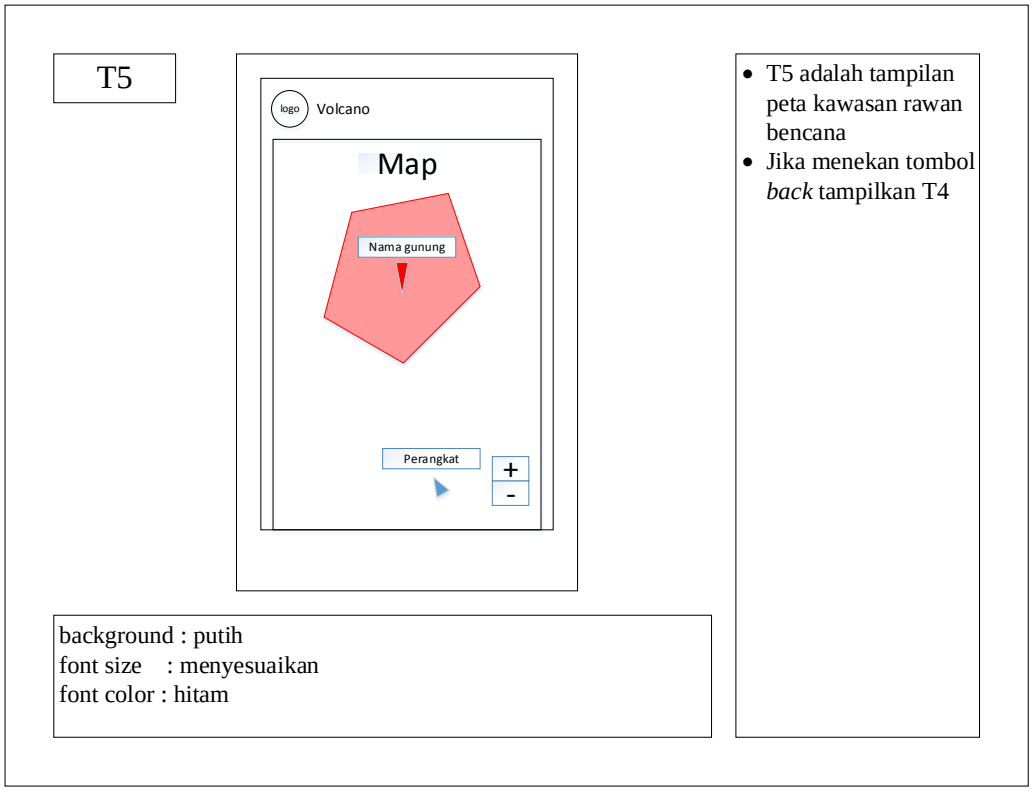
Gambar 3.62 Perancangan Pilih Gunung



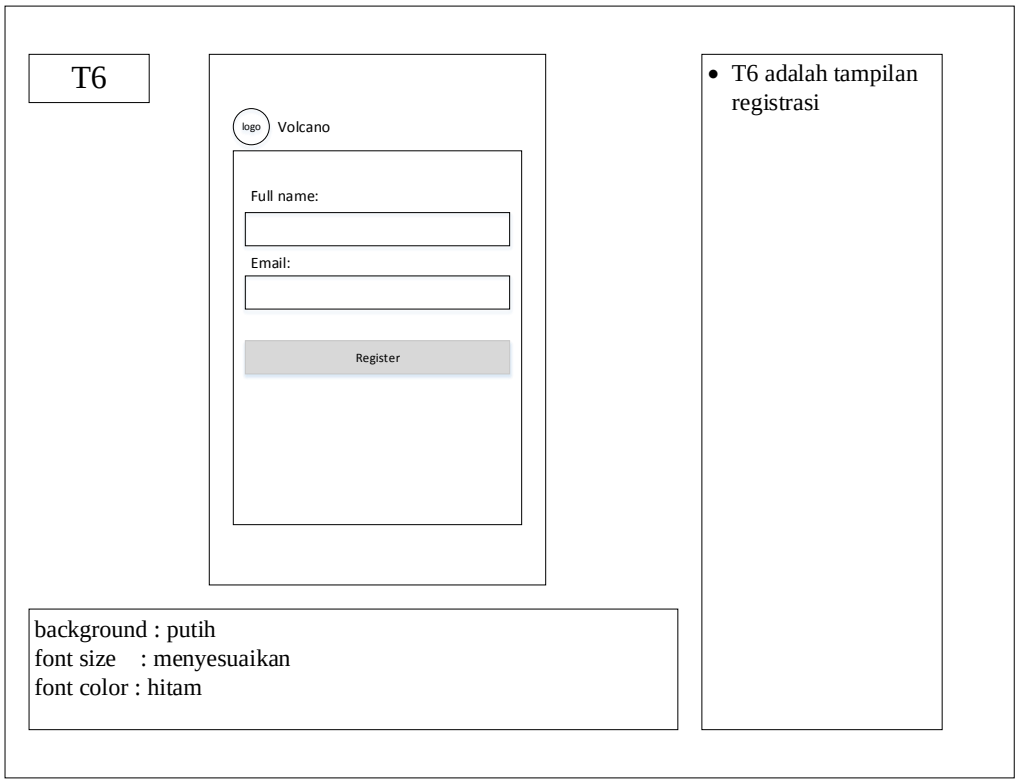
Gambar 3.63 Perancangan Tampil Info Gunung



Gambar 3.64 Perancangan Detail Gunung



Gambar 3.65 Perancangan Peta KRB



Gambar 3.66 Perancangan Registrasi Pengguna

3.2.3.2 Perancangan Pesan

1. Pesan Teks

Tabel 3.30 adalah perancangan pesan yang berbentuk teks.

Tabel 3.30 Perancangan Pesan

No	Pesan	Kode	Jenis
1.	Maaf, username dan password salah	P01	Teks
2.	“field” tidak boleh kosong	P02	Teks
3.	Sukses, data berhasil ditambah !	P03	Teks
4.	Sukses, data berhasil diubah !	P04	Teks
5.	Sukses, data berhasil dihapus !	P05	Teks
6.	Sukses, KRB berhasil ditambah!	P06	Teks
7.	Sukses, KRB berhasil diubah !	P07	Teks

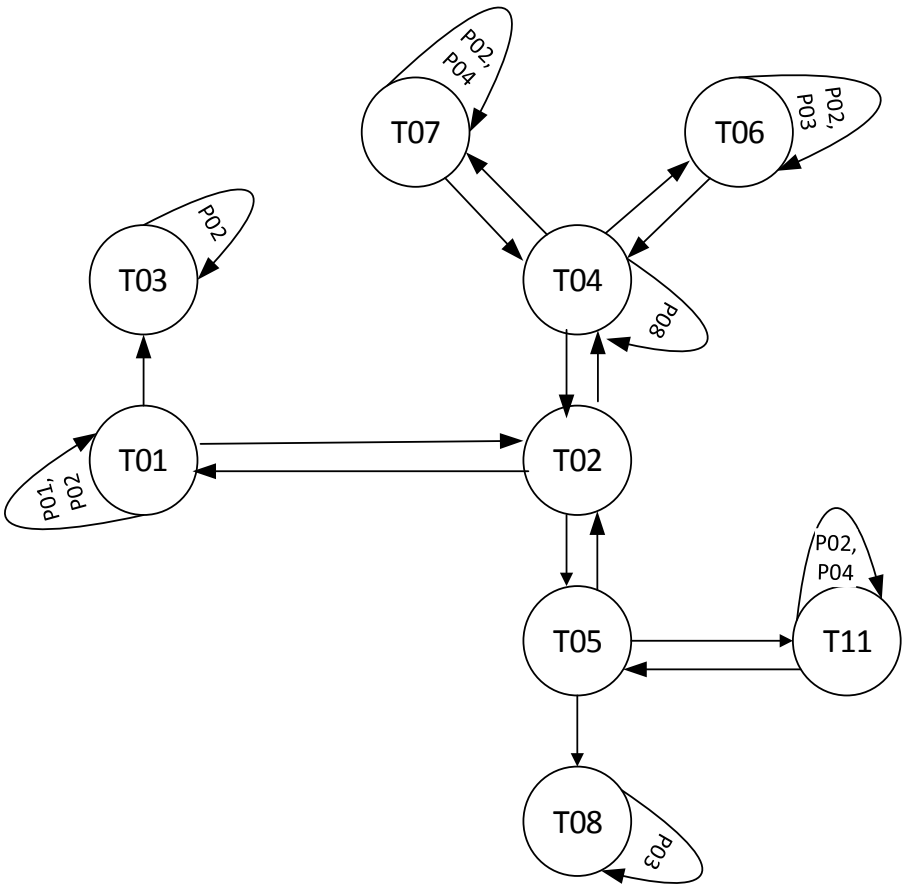
2. Pesan Dialog

Pesan dibawah ini merupakan pesan yang berbentuk dialog yang ada pada aplikasi

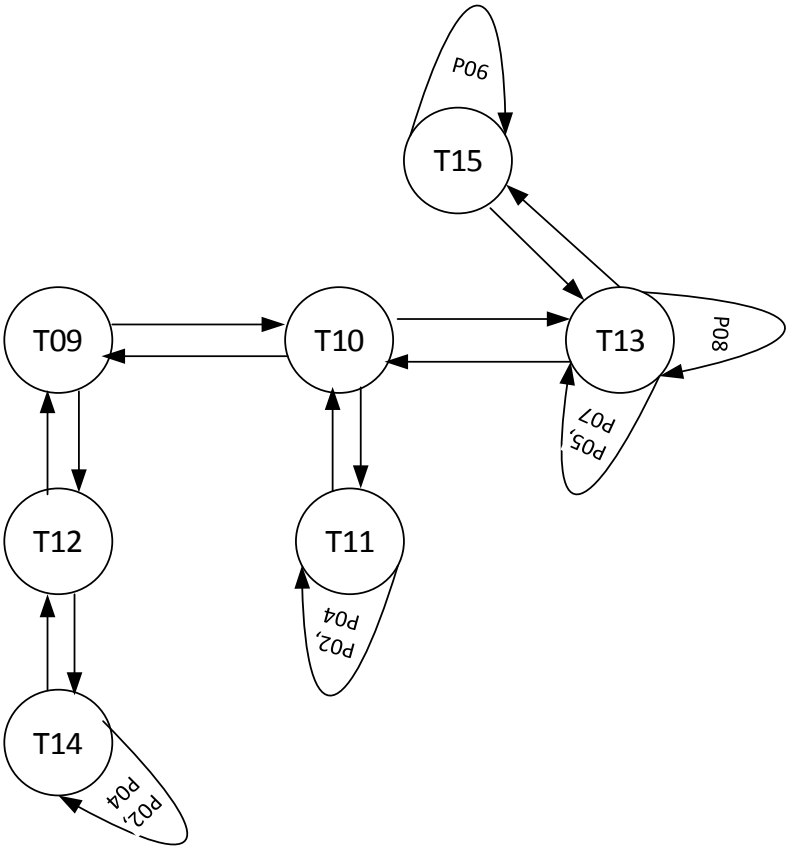
	X
Apakah anda yakin menghapus ini ? **[OK] [cancel]**	

Gambar 3.67 P08 Pesan Dialog

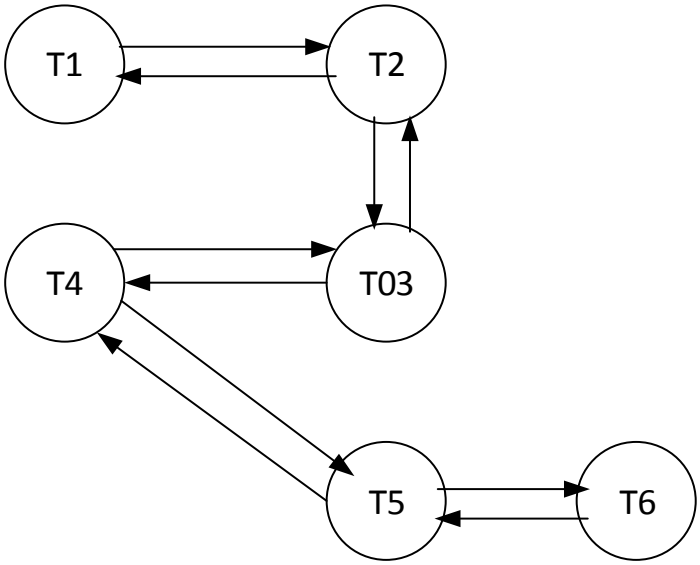
3.2.4 Jaringan Semantik



Gambar 3.68 Jaringan Semantik *Back end* Admin



Gambar 3.69 Jaringan Semantik *Back end* Operator



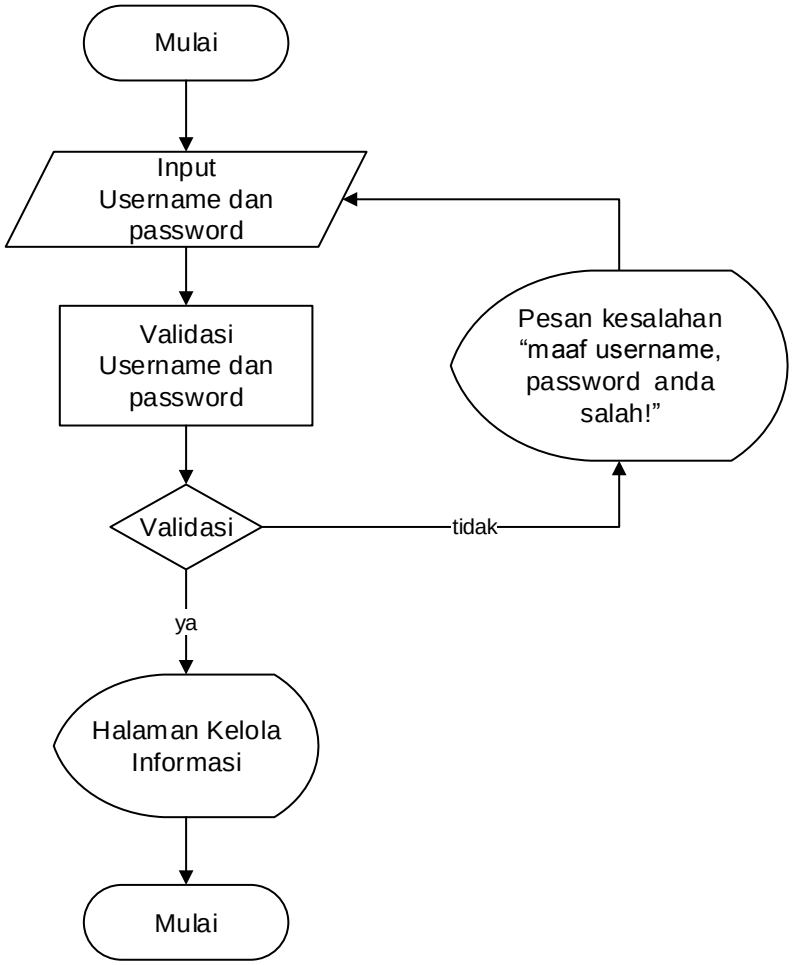
Gambar 3.70 Jaringan Semantik *Mobile*

3.2.5 Perancangan *Method*

Perancangan *method* berfungsi untuk mendeskripsikan *method* yang berada didalam sistem. Adapun *method* yang terdapat dalam sistem yang akan dibangun adalah sebagai berikut :

1. Perancangan *Method login*

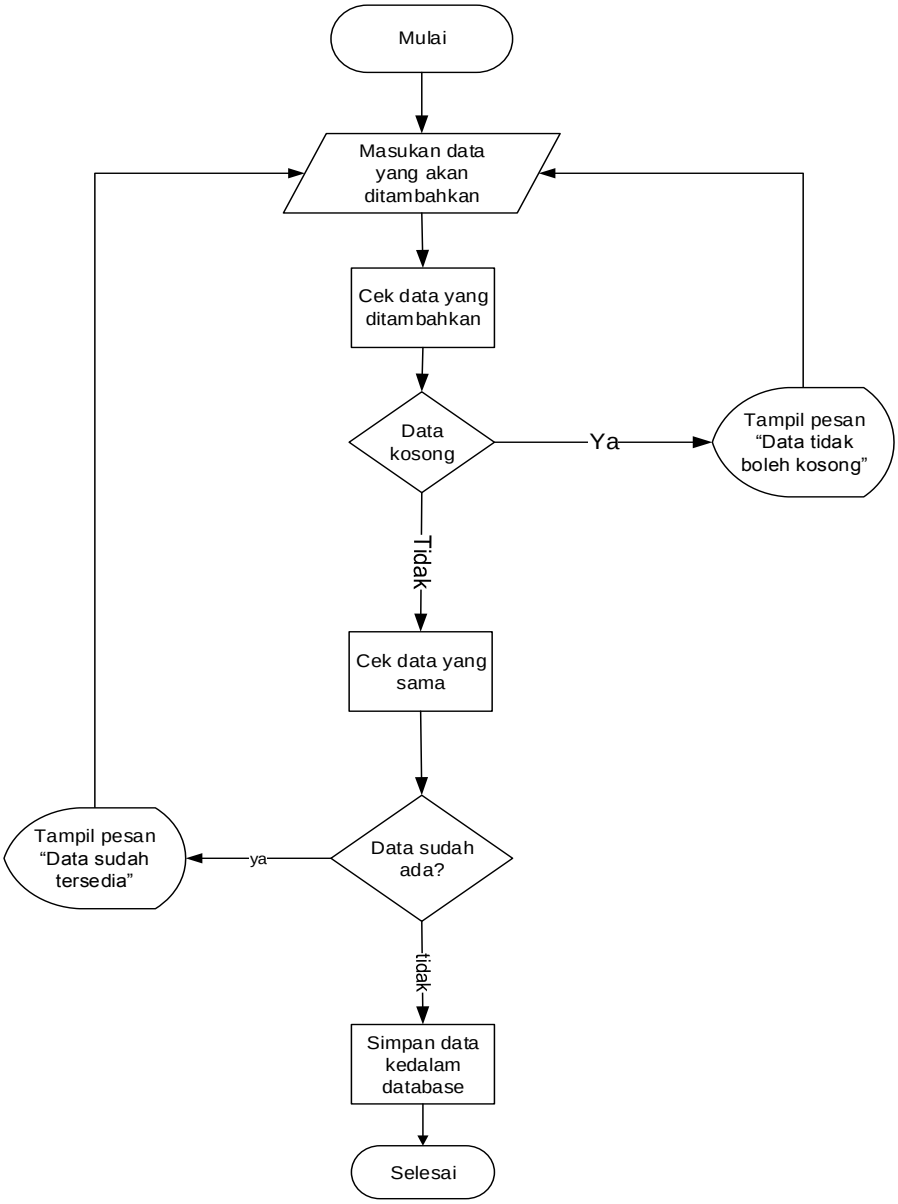
Method login() pada *class home* digunakan untuk melakukan proses login dengan membaca *username* dan *password* yang telah dimasukan *user*. *Method* Login digambarkan dalam bentuk *flowchart* yang terlihat pada Gambar 3.71



Gambar 3.71 *Flowchart* Login

2. Perancangan Method Tambah

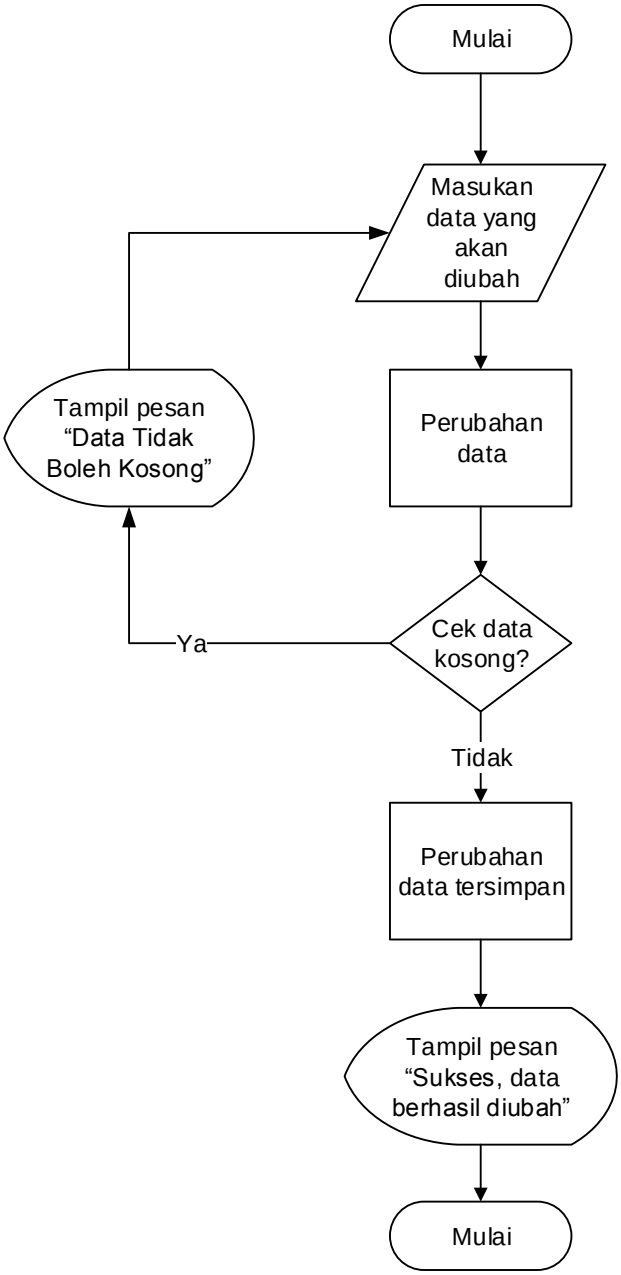
Method tambah digunakan pada pengolahan data user pada *class user* yaitu *insert_user(),* dan pengolahan data gunung pada *class gunung* yaitu *insert_gunung()* dan *insert_polygon(), method* tambah digambarkan dalam bentuk *flowchart* yang terlihat pada Gambar 3.72



Gambar 3.72 *Flowchart Insert*

3. Perancangan Method Ubah

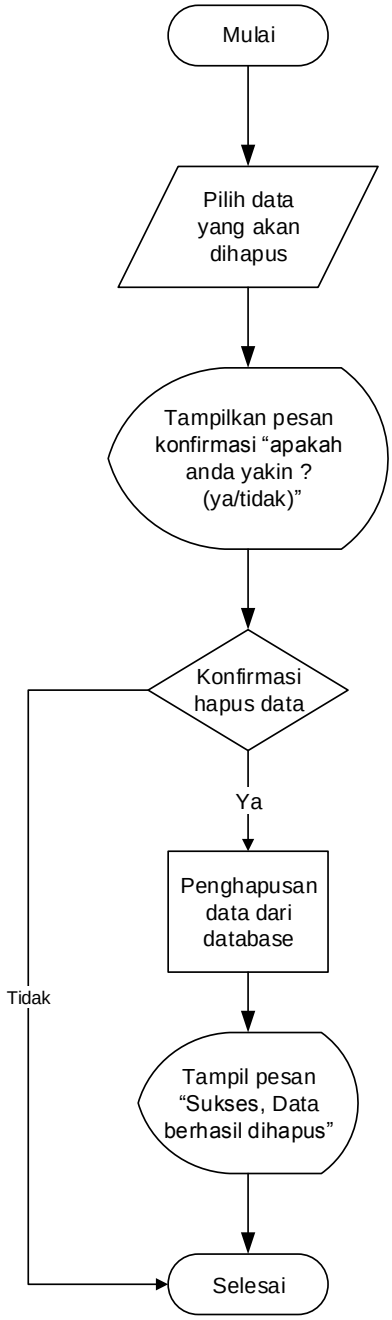
Method tambah digunakan pada pengolahan data user pada class user yaitu *insert_user*, dan pengolahan data gunung pada class gunung yaitu *insert_gunung*, dan *insert_polygon*, *method* tambah digambarkan dalam bentuk *flowchart* yang terlihat pada Gambar 3.73



Gambar 3.73 *Flowchart* Ubah

4. *Method* Hapus

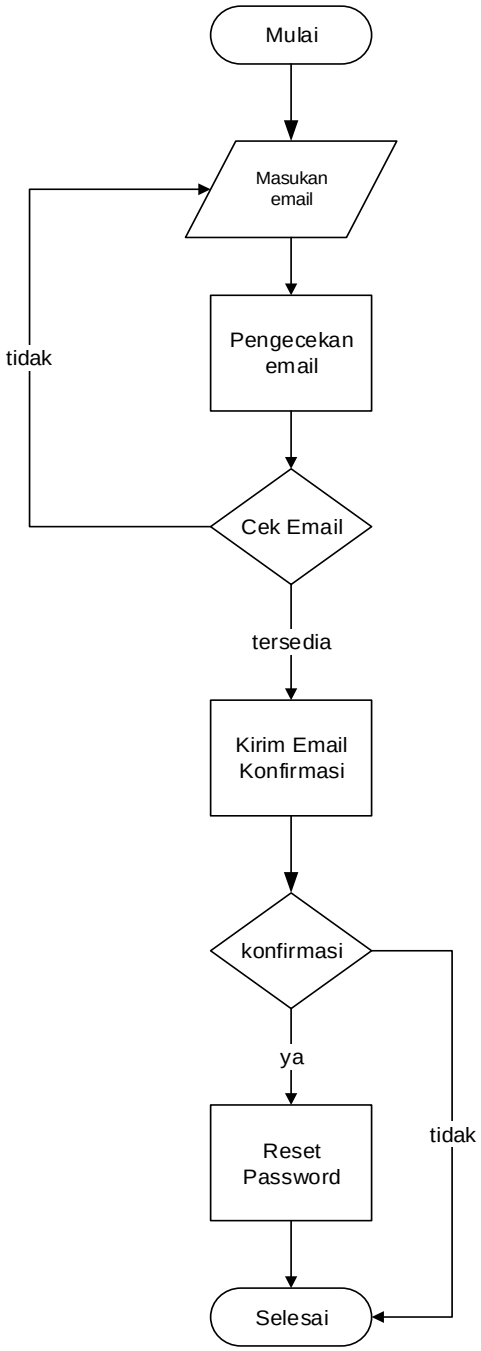
Method hapus yang digunakan pada pengolahan data gunung, *user*, dan kawasan rawan bencana mempunyai fungsi yang sama, dimana *method* tersebut terbagi atas *method delete_data_gunung*, *delete_data_user*, *delete_data_polygon*. Proses *method* hapus digambarkan dalam bentuk *flowchart* yang terlihat pada Gambar 3.74



Gambar 3.74 *Flowchart* Hapus

5. *Method* Ubah Password

Method tambah digunakan pada pengolahan data user pada class user yaitu insert_user, dan pengolahan data gunung pada *class* gunung yaitu insert_gunung() dan insert_polygon(), *method* tambah digambarkan dalam bentuk *flowchart* yang terlihat pada Gambar 3.75



Gambar 3.75 *Flowchart* Lupa password