PENGEMBANGAN RESTFUL API UNTUK APPLICATION SPECIFIC HIGH LEVEL LOCATION SERVICE

Disusun Oleh:

N a m a	: Muhammad Akbar
NIM	: 14523277

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2018

ii

HALAMAN PENGESAHAN DOSEN PEMBIMBING

PENGEMBANGAN RESTFUL API UNTUK APPLICATION SPECIFIC HIGH LEVEL LOCATION SERVICE

TUGAS AKHIR

Disusun Oleh:

N a m a : Muhammad Akbar
NIM : 14523277

Yogyakarta, 6 Juli 2018
Pembimbing,

(R. Teduh Dirgahayu, Dr., ST., M.Sc.)

iii

HALAMAN PENGESAHAN DOSEN PENGUJI

PENGEMBANGAN RESTFUL API UNTUK APPLICATION SPECIFIC HIGH LEVEL LOCATION SERVICE

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik Informatika
di Fakultas Teknologi Industri Universitas Islam Indonesia
Yogyakarta, 1 Agustus 2018

Tim Penguji

R. Teduh Dirgahayu, Dr., ST., M.Sc.

Anggota 1

Dhomas Hatta F, ST., M.Eng., Ph.D.

Anggota 2

Sri Mulyati, S.Kom., M.Kom.

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Hendrik, ST., M.Eng.)

iv

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : **Muhammad Akbar**

NIM : **14523277**

Tugas akhir dengan judul:

PENGEMBANGAN RESTFUL API UNTUK APPLICATION SPECIFIC HIGH LEVEL LOCATION SERVICE

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 1 Agustus 2018

(Muhammad Akbar)

HALAMAN PERSEMBAHAN

Dengan mengucapkan syukur *Alhamdulillah* atas berkat rahmat dan karunia Allah SWT, sehingga saya dapat menyelesaikan tugas akhir ini. Karya ini saya persembahkan kepada orang – orang yang saya sayangi:

Kedua orang tua saya yang tercinta

(Ayahanda Zulkifli Koto dan Ibunda Tetian Erni)

Saya persembahkan karya ini kepada orang tua saya tercinta yang telah menjadi motivator dalam hidup saya yang selalu memberikan doa, semangat, motivasi, nasihat, dan bimbingan atas kelancaran tugas akhir ini.

Ketiga kakak saya yang tercinta

(Reza Juanda Permana, Paramita Nauli, dan Widya Sartika)

Saya persembahkan karya ini kepada ketiga kakak saya yang senantiasa memberikan dukungan, semangat, dan doa. Serta senantiasa memberikan masukan positif kepada saya selama masa kuliah sampai penyelesaian tugas akhir ini.

HALAMAN MOTO

“Allah akan meninggikan orang-orang yang beriman di antara kamu dan orang-orang yang berilmu pengetahuan beberapa derajat. Dan Allah Maha mengetahui apa yang kamu kerjakan” (QS. Al Mujadillah:11)

“Barang siapa menempuh suatu jalan untuk mencari ilmu, maka Allah memudahkannya mendapat jalan ke surga” (HR. Muslim)

“Sebaik - baik manusia di antaramu adalah yang paling banyak manfaatnya bagi orang lain” (HR. Bukhari dan Muslim)

“Stay Hungry Stay Foolish” - Steve Jobs

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh.

Alhamdulillahirabbil'alamin, Penulis ucapkan puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat dan karunia-Nya kepada penulis sehingga penulis diberikan kekuatan dan kemudahan dalam menyelesaikan tugas akhir ini. Sholawat dan salam penulis ucapkan kepada Panglima Umat Baginda Nabi Muhammad SAW dengan hantaran kata *Allahumma Sholli'alaa Sayyidina Muhammad Wa'alaa aali Sayyidina Muhammad*. Dengan meneladani semangat Rasulullah SAW dalam menimba ilmu dan mengamalkannya sehingga memotivasi penulis dalam menyelesaikan tugas akhir ini yang berjudul "Pengembangan RESTful API untuk Application Specific High Level Location Service".

Laporan tugas akhir ini ditujukan sebagai salah satu syarat dalam menyelesaikan pendidikan pada jenjang Strata 1 (S1) Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia. Penulis menyadari bahwa penyelesaian tugas akhir ini tidak terlepas dari dukungan, bantuan dan bimbingan dari berbagai pihak. Mulai dari perancangan sistem, implementasi, pengujian, serta pembuatan laporan tugas akhir. Oleh sebab itu, penulis ingin menyampaikan terima kasih sebesar - besarnya kepada:

1. Bapak Fathul Wahid, S.T., M.Sc., Ph.D selaku Rektor Universitas Islam Indonesia.
2. Bapak Prof. Dr. Ir. Hari Purnomo, MT. selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
3. Bapak Hendrik, S.T., M.Eng. selaku Ketua Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Bapak R. Teduh Dirgahayu, Dr., ST., M.Sc. selaku Dosen Pembimbing Tugas Akhir yang telah baik membimbing, mengarahkan, dan memberi masukan kepada penulis dalam menyelesaikan tugas akhir ini.
5. Segenap Dosen Jurusan Teknik Informatika yang telah memberikan ilmunya kepada penulis selama masa studi.
6. Kedua orang tua penulis (Zulkifli Koto dan Tetian Erni) serta ketiga kakak dari penulis (Reza Juanda Permana, Paramita Nauli, dan Widya Sartika) yang senantiasa memberikan masukan, motivasi, semangat, doa, nasihat, dan dukungan kepada penulis.

7. Teman – teman angkatan 14 Teknik Informatika dan asisten laboratorium informatika terpadu yang telah memberikan bantuan dan dukungan kepada penulis selama masa studi.
8. Semua pihak yang telah banyak membantu penulis dalam menyelesaikan tugas akhir ini yang tidak dapat disebutkan satu persatu.

Penulis menyadari bahwa laporan tugas akhir ini masih memiliki banyak kekurangan dan jauh dari kata sempurna. Oleh karena itu, penulis mengharapkan kritik dan saran yang bersifat membangun untuk menjadi perbaikan agar dapat menjadi lebih baik lagi. Semoga laporan tugas akhir ini dapat memberikan manfaat bagi semua pihak. *Aamiin.*

Wassalamu'alaikum Warahmatullahi Wabarakatuh.

Yogyakarta, 6 Juli 2018

(Muhammad Akbar)

SARI

Penggunaan API (*Application Programming Interface*) dalam pengembangan suatu aplikasi dapat memberikan kemudahan kepada *developer* dalam mempercepat dan mempermudah proses pengembangan sistem. Saat ini, sudah banyak beragam jenis API yang tersedia mulai dari prakiraan cuaca, produk barang dan jasa, lokasi, dan lain sebagainya. Aplikasi LBS (*Location Based Service*) merupakan suatu aplikasi yang memanfaatkan layanan lokasi dalam memberikan informasi geografis. Saat ini sudah banyak berkembang aplikasi LBS mulai dari aplikasi transportasi *online*, pemetaan, wisata, dan lain sebagainya. Dengan peningkatan penggunaan layanan lokasi, maka dibutuhkan suatu layanan lokasi yang dapat memberikan informasi secara abstraksi level tinggi yang dapat memenuhi kebutuhan spesifik aplikasi LBS.

Pada penelitian ini akan membahas tentang bagaimana mengembangkan RESTful API yang dapat memberikan informasi area secara abstraksi level tinggi yaitu informasi yang menjelaskan dari suatu area sehingga dapat langsung dapat dipahami oleh manusia. Layanan lokasi yang dibangun memberikan kebebasan kepada *developer* untuk memetakan area atau lokasi yang diinginkan menggunakan *file* KML (*Keyhole Markup Language*) sehingga dapat memenuhi kebutuhan spesifik aplikasi LBS. Selain itu, penelitian ini akan membangun sebuah aplikasi berbasis *web* yang memudahkan *developer* dalam mengelola sumber data miliknya dan aplikasi pengguna yang akan mengakses sumber data. Layanan lokasi ini menggunakan Mysql spatial dalam mengelola data area. Layanan lokasi ini menggunakan fitur OAuth (*Open Authorization*) 2.0 untuk memudahkan *developer* dalam memberikan otoritas penggunaan sumber data kepada aplikasi pengguna.

Kata kunci : API, REST, *Location Based Service*, Abstraksi Level Tinggi, Aplikasi Spesifik, KML, OAuth, Mysql spatial.

GLOSARIUM

API	Suatu aplikasi berupa antarmuka yang bertugas sebagai kurir yang melayani permintaan dari aplikasi lainnya.
Location Based Service	Layanan yang memungkinkan perangkat bergerak untuk dapat mengetahui posisi lokasi pengguna.
Keyhole Markup Language	Suatu format *file* yang ditujukan untuk menampilkan data geografi pada perambat seperti Google Earth.
REST	Seperangkat prinsip arsitektur yang melakukan transmisi data melalui antarmuka yang terstandarisasi seperti HTTP.
RESTful	Istilah untuk layanan yang mengimplementasikan arsitektur REST.
Authorization	Proses perizinan penggunaan sumber data yang dilakukan oleh pemilik sumber data kepada pengguna data.
Client	Suatu aplikasi yang melakukan permintaan akses sumber data yang dilindungi atas nama *resource owner* dan atas wewenang otorisasi yang diberikan oleh *resource owner*.
Resource Owner	Pemilik sumber data yang memiliki kemampuan untuk memberikan otoritas kepada *client* untuk mengakses sumber data.
Resource Server	*Server* yang melakukan layanan hosting sumber data milik *resource owner* yang dilindungi oleh OAuth, serta mampu menerima dan menanggapi permintaan akses sumber data yang terlindungi menggunakan *access token*.
Authorization Server	*Server* yang mengeluarkan *access token* kepada *client* untuk mengakses sumber data setelah melakukan proses otentikasi dan memperoleh otorisasi.
Authorization Code	Kode otoritas yang diberikan oleh *resource owner* sebagai bukti diberikannya otorisasi dalam menggunakan sumber dan akan ditukarkan menjadi *access token.*
Access Token	Penanda otoritas penggunaan sumber data oleh *client*
Refresh Token	Penanda untuk melakukan pembaruan *access token*

DAFTAR ISI

HALAMAN JUDUL i
HALAMAN PENGESAHAN DOSEN PEMBIMBING ii
HALAMAN PENGESAHAN DOSEN PENGUJI iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR iv
HALAMAN PERSEMBAHAN v
HALAMAN MOTO vi
KATA PENGANTAR vii
SARI ix
GLOSARIUM x
DAFTAR ISI xi
DAFTAR TABEL xiv
DAFTAR GAMBAR xv
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan Penelitian 3
1.5 Manfaat Penelitian 3
1.6 Metodologi Penelitian 4
1.7 Sistematika Penulisan 5
BAB II LANDASAN TEORI 6
2.1 LBS (*Location Based Service*) 6
2.1.1 Definisi LBS 6
2.1.2 Komponen LBS 6
2.1.3 Penggunaan LBS 8
2.1.4 Contoh Aplikasi LBS 8
2.2 REST (*Representational State Transfer*) 9
2.2.1 Definisi REST 9
2.2.2 Perbandingan SOAP dan REST 10
2.2.3 Keunggulan REST 12
2.3 OAuth 2.0 12

2.3.1 Definisi OAuth 2.0 12
2.3.2 Pembagian Peran OAuth 2.0 13
2.3.3 Alur Protokol 13
2.4 Mysql Spatial 14
2.4.1 Pengenalan Mysql Spatial 14
2.4.2 Tipe Data Spasial 15
2.4.3 Standar Format Data Spasial 16
2.4.4 Fungsi Analisis Sistem 16
2.5 KML (*Keyhole Markup Language*) 17
2.5.1 Pengenalan KML 17
2.5.2 Dasar Dokumen KML 17
2.6 Penelitian Terkait Dengan Pengembangan RESTful API Untuk Application Specific High Level Location Service 18
BAB III ANALISIS DAN PERANCANGAN SISTEM 21
3.1 Analisis Masalah 21
3.2 Usulan Penyelesaian Masalah 23
3.3 Identifikasi Pengguna 23
3.4 Kebutuhan Proses 24
3.5 Kebutuhan Antarmuka 28
3.6 Perangkat Lunak Yang Digunakan 28
3.7 Perangkat Keras Yang Digunakan 29
3.8 Arsitektur Sistem 29
3.9 *Use Case Diagram* 30
3.10 *Activity Diagram* 31
3.11 *Entity Relationship Diagram* 50
3.12 Rancangan Antarmuka 56
3.13 Rancangan *Request* dan *Response* RESTful API 62
3.14 Rancangan Fungsi Pembuatan Dan Pengolahan Data Spasial 67
3.15 Rancangan Skenario Penggunaan RESTful API 68
3.16 Rancangan Pengujian Sistem 70
3.17 Rumus Perhitungan Yang Digunakan 75
BAB IV IMPLEMENTASI DAN PENGUJIAN SISTEM 79
4.1 Implementasi 79
4.1.1 Aplikasi berbasis *web* 79

4.1.2 RESTful API 86
4.2 Pengujian 95
BAB V KESIMPULAN DAN SARAN 106
5.1 Kesimpulan 106
5.2 Saran 106
DAFTAR PUSTAKA 107
LAMPIRAN 109

DAFTAR TABEL

Tabel 2.1 Perbandingan SOAP dan REST 11
Tabel 2.2 Rangkuman Penelitian Sebelumnya 19
Tabel 3.1 Tabel users 51
Tabel 3.2 Tabel oauth_clients 52
Tabel 3.3 Tabel areas 53
Tabel 3.4 Tabel oauth_auth_codes 53
Tabel 3.5 Tabel oauth_access_tokens 54
Tabel 3.6 Tabel oauth_refresh_tokens 55
Tabel 3.7 Tabel password_resets 55
Tabel 3.8 Rancangan *Request* dan *Response* Kode Otorisasi 62
Tabel 3.9 Rancangan *Request* dan *Response* Token Akses 63
Tabel 3.10 Rancangan *Request* dan *Response* Refresh Token 63
Tabel 3.11 Rancangan *Request* dan *Response* Informasi Daftar Area 64
Tabel 3.12 Rancangan *Request* dan *Response* Informasi Area Terkini 65
Tabel 3.13 Rancangan *Request* dan *Response* Informasi Area Terdekat 65
Tabel 3.14 Rancangan *Request* dan *Response* Informasi Area Berdasarkan Kata Kunci 66
Tabel 3.15 Rancangan Fungsi Pembuatan Dan Pengolahan Data Spasial 67
Tabel 3.16 Rancangan Kasus Uji 70
Tabel 3.17 Rancangan Pengujian *Use Case* 74
Tabel 4.1 Hasil Pengujian UC-1 (*Register*) 95
Tabel 4.2 Hasil Pengujian UC-2 (*Login*) 96
Tabel 4.3 Hasil Pengujian UC-3 (*Reset Password*) 97
Tabel 4.4 Hasil Pengujian UC-4 (Mengelola Aplikasi Pengguna) 98
Tabel 4.5 Hasil Pengujian UC-5 (Mengelola Area) 99
Tabel 4.6 Hasil Pengujian UC-6 (Melihat Dokumentasi) 102
Tabel 4.7 Hasil Pengujian UC-7 (Mendapatkan Token Akses) 102
Tabel 4.8 Hasil Pengujian UC-8 (Mendapatkan Informasi Area) 103
Tabel 4.9 Hasil Pengujian UC-9 (Memperbarui Token Akses) 105

DAFTAR GAMBAR

Gambar 2.1 Komponen Dasar LBS 7
Gambar 2.2 Alur Protokol 14
Gambar 2.3 Visualisasi Dokumen KML 17
Gambar 2.4 Dokumen KML 18
Gambar 3.1 Informasi Area Abstraksi Level Rendah & Abstraksi Level Tinggi 21
Gambar 3.2 Representasi Geo-Lokasi 22
Gambar 3.3 Kebebasan Dalam Memetakan Area 23
Gambar 3.4 Arsitektur Sistem 29
Gambar 3.5 *Use Case Diagram* 30
Gambar 3.6 *Activity Diagram Register* 31
Gambar 3.7 *Activity Diagram Login* 32
Gambar 3.8 *Activity Diagram Reset Password* 33
Gambar 3.9 *Activity Diagram* Menampilkan Daftar Aplikasi Pengguna 34
Gambar 3.10 *Activity Diagram* Mendaftarkan Aplikasi Pengguna 35
Gambar 3.11 *Activity Diagram* Mengubah Aplikasi Pengguna 36
Gambar 3.12 *Activity Diagram* Menghapus Aplikasi Pengguna 37
Gambar 3.13 *Activity Diagram* Mencabut Otoritas Aplikasi Pengguna 38
Gambar 3.14 *Activity Diagram* Menampilkan Daftar Area 39
Gambar 3.15 *Activity Diagram* Membuat Area 40
Gambar 3.16 *Activity Diagram* Mengubah Area 41
Gambar 3.17 *Activity Diagram* Menghapus Area 42
Gambar 3.18 *Activity Diagram* Menampilkan Visualisasi Area 43
Gambar 3.19 *Activity Diagram* Melihat Dokumentasi 44
Gambar 3.20 *Activity Diagram* Mendapatkan Token Akses 45
Gambar 3.21 *Activity Diagram* Mendapatkan Informasi Area Terkini 46
Gambar 3.22 *Activity Diagram* Mendapatkan Informasi Area Terdekat 47
Gambar 3.23 *Activity Diagram* Mendapatkan Informasi Daftar Area 48
Gambar 3.24 *Activity Diagram* Mendapatkan Informasi Area Berdasarkan Kata Kunci 49
Gambar 3.25 *Activity Diagram* Memperbarui Token Akses 50
Gambar 3.26 *Entity Relationship Diagram* MyHilvabs 51
Gambar 3.27 Rancangan Halaman *Register* 56

Gambar 3.28 Rancangan Halaman *Login* 56
Gambar 3.29 Rancangan Halaman *Reset Password* – Pengiriman *Reset Password Link* 57
Gambar 3.30 Rancangan Halaman *Reset Password* – Pengaturan Ulang *Password* Baru 57
Gambar 3.31 Rancangan Halaman Dokumentasi 58
Gambar 3.32 Rancangan Halaman *My App* 58
Gambar 3.33 Rancangan Halaman *My App* – Pendaftaran dan Pengubahan Aplikasi Pengguna 59
Gambar 3.34 Rancangan Halaman *Resources* – Daftar Aplikasi Pengguna 60
Gambar 3.35 Rancangan Halaman *Resources* – Daftar Area 60
Gambar 3.36 Rancangan Halaman *Resources* – Pembuatan dan Pengubahan Area 61
Gambar 3.37 Rancangan Halaman *Resources* – Visualisasi Area 62
Gambar 3.38 Rancangan Skenario Pengguna RESTful API (1) 68
Gambar 3.39 Rancangan Skenario Pengguna RESTful API (2) 69
Gambar 3.40 Obyek Geometri *Linestring* Yang Dipecah 75
Gambar 3.41 Obyek Geometri *Polygon* Yang Dipecah 76
Gambar 3.42 Kode Sumber *Method* getPointProjection 76
Gambar 3.43 Titik Proyeksi Di Dalam Garis 77
Gambar 3.44 Titik Proyeksi Di Luar Garis 77
Gambar 3.45 Kode Sumber *Method* contains 77
Gambar 3.46 Kode Sumber *Method* getDistanceTwoPoints 78
Gambar 4.1 Implementasi Halaman *Register* 80
Gambar 4.2 Implementasi Halaman *Login* 80
Gambar 4.3 Implementasi Halaman *Reset Password* – Pengiriman *Reset Password Link* 81
Gambar 4.4 Token *Reset Password* 81
Gambar 4.5 Implementasi Halaman *Reset Password* – Pengaturan Ulang *Password* Baru 81
Gambar 4.6 Implementasi Halaman Dokumentasi 82
Gambar 4.7 Implementasi Halaman *My App* – Daftar Aplikasi Pengguna 83
Gambar 4.8 Implementasi Halaman My App – Form Pendaftaraan Aplikasi Pengguna 83
Gambar 4.9 Implementasi Halaman *My App* – Pesan Konfirmasi Hapus Aplikasi Pengguna 83
Gambar 4.10 Implementasi Halaman *Resources* – Daftar Aplikasi Pengguna 84
Gambar 4.11 Implementasi Halaman *Resources* – Daftar Area 85
Gambar 4.12 Implementasi Halaman *Resources* – Pembuatan Area 85
Gambar 4.13 Implementasi Halaman *Resources* – Pesan Konfirmasi Hapus Area 85
Gambar 4.14 Implementasi Halaman *Resources* – Visualisasi Area 86

Gambar 4.15 Kode Sumber HTTP *Request* Kode Otorisasi 86
Gambar 4.16 Persetujuan Otoritas Pengaksesan Sumber Data 87
Gambar 4.17 Kode Sumber HTTP *Request* Token Akses 88
Gambar 4.18 Respons Permintaan Token Akses 88
Gambar 4.19 Halaman *My App* – Daftar Aplikasi Yang Sudah Mendapatkan Otoritas 89
Gambar 4.20 *Request* dan *Response* Informasi Area Terkini 90
Gambar 4.21 *Request* dan *Response* Informasi Area Terdekat 91
Gambar 4.22 *Request* dan *Response* Informasi Daftar Area 92
Gambar 4.23 *Request* dan *Response* Informasi Area Berdasarkan Kata Kunci 93
Gambar 4.24 Kode Sumber HTTP *Request* Permintaan Perbaruan Token Akses 94
Gambar 4.25 Respons Perbaruan Token Akses 95

BAB I
PENDAHULUAN

1.1 Latar Belakang

API (*Application Programming Interface*) adalah suatu aplikasi berupa antarmuka yang berperan sebagai kurir yang melayani permintaan dari aplikasi lain dan mengembalikan respons ke penerima (MuleSoft Videos, 2015). API biasanya digunakan ketika terdapat dua atau lebih sistem terpisah yang membutuhkan informasi atau fitur yang tidak bisa mereka kerjakan sendiri. API dibangun oleh suatu organisasi pengembang perangkat lunak yang dirilis ke publik agar para pengembang aplikasi lainnya dapat menikmati layanan yang diberikan oleh penyedia API. Sehingga *developer* dapat merancang dan mendesain produk yang didukung oleh penyedia layanan.

API memiliki berbagai jenis arsitektur salah satu di antaranya ialah REST. REST (*Representational State Transfer*) merupakan seperangkat prinsip arsitektur yang melakukan transmisi data melalui antarmuka yang terstandarisasi seperti HTTP (Dhingra, 2013). RESTful sendiri merupakan istilah yang dipakai untuk layanan *web* yang mengimplementasikan arsitektur REST (Adamczyk, Smith, Johnson, & Hafiz, 2011). Untuk mengakses suatu *resource*, ada beberapa hal yang harus diperhatikan oleh *developer* yaitu URI (*Uniform Resource Identifier*) sebagai identitas lokasi *resource*, HTTP *method* (GET, POST, DELETE, PUT), *header* sebagai meta informasi suatu *request*, dan *body* sebagai isi data yang ingin dikirimkan. Keluaran dari RESTful API dapat berupa format pertukaran JSON (*JavaScript Object Notation*) atau XML (*Extensible Markup Language*). Kedua format tersebut dapat digunakan pada berbagai bahasa pemrograman seperti PHP, Java, Python, dan lain sebagainya. Penggunaan API saat ini membantu *developer* dalam menyingkat waktu pengembangan aplikasi. Sehingga *developer* tidak perlu membuat fitur yang sudah ada, tetapi langsung menggunakan layanan yang menyediakan fitur tersebut. Layanan API yang diberikan dapat berupa layanan cuaca, keuangan, lokasi, barang, otentikasi, dan lain sebagainya. Seperti halnya layanan lokasi yang banyak diterapkan oleh berbagai aplikasi LBS (*Location Based Service*).

Aplikasi LBS merupakan aplikasi yang memanfaatkan layanan lokasi untuk menyediakan informasi geografis kepada pengguna seperti lokasi fasilitas umum, sarana transportasi, wisata, dan lain - lain. Aplikasi LBS memanfaatkan modul GPS (*Global*

Positioning System) yang tertanam dalam perangkat komputer untuk mengetahui posisi berdasarkan *latitude* dan *longitude* pengguna. Selanjutnya, aplikasi LBS dapat menampilkan posisi lokasi dengan menggunakan fitur visualisasi yang diperoleh dari layanan lokasi. Saat ini semakin banyak aplikasi yang memanfaatkan layanan lokasi dengan kebutuhan yang berbeda – beda. Sehingga dibutuhkan layanan yang dapat menyediakan informasi lokasi pada abstraksi level tinggi untuk memenuhi kebutuhan spesifik aplikasi LBS.

Informasi lokasi abstraksi level tinggi merupakan informasi lokasi yang dapat secara langsung dipahami oleh manusia (Dirgahayu, 2016). Contohnya adalah nama dan deskripsi suatu area atau lokasi. Sedangkan, informasi lokasi abstraksi level rendah merupakan informasi lokasi yang sulit dipahami secara langsung oleh manusia. Contohnya adalah koordinat geo-lokasi pengguna. Layanan lokasi seharusnya dapat menyediakan informasi lokasi secara spesifik menyesuaikan dengan aplikasi LBS. Koordinat geo-lokasi yang sama, seharusnya dapat diinterpretasikan sebagai nama area atau lokasi yang berbeda oleh aplikasi LBS yang berbeda. Contohnya adalah koordinat (-7.768470, 110.373480) diinterpretasikan oleh aplikasi pencari rumah sakit sebagai area Rumah Sakit Umum Pusat Dr. Sardjito, sedangkan koordinat tersebut diinterpretasikan oleh aplikasi pemetaan kampus sebagai area dari Universitas Gadjah Mada. Selanjutnya, layanan lokasi yang diharapkan mampu memberikan kebebasan kepada *developer* dalam memetakan area atau lokasi dalam berbagai bentuk baik berupa titik, garis, maupun banyak sisi. Sehingga area atau lokasi dapat menyesuaikan dengan kebutuhan aplikasi LBS.

Berdasarkan permasalahan di atas, maka dibutuhkan layanan lokasi yang menyediakan informasi area atau lokasi abstraksi level tinggi untuk memenuhi kebutuhan spesifik aplikasi LBS. Informasi area atau lokasi abstraksi level tinggi yaitu informasi area atau lokasi yang mendeskripsikan suatu obyek area atau lokasi seperti nama area, deskripsi area, dan lain sebagainya sehingga langsung dipahami oleh manusia. Pengembangan layanan lokasi dibuat untuk memudahkan *developer* dalam mengelola data area dan aplikasi pengguna layanan lokasi. Selain itu, layanan lokasi tersebut dibuat agar mampu memberikan kebebasan dalam memetakan area yang disesuaikan dengan kebutuhan aplikasi LBS. Serta, layanan lokasi dibuat agar mampu mengelola aplikasi pengguna baik proses pendaftaran aplikasi maupun pencabutan hak akses terhadap informasi area atau lokasi yang diberikan oleh layanan lokasi.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang dijelaskan di atas, maka permasalahan yang dapat dirumuskan pada penelitian ini adalah bagaimana mengembangkan sebuah RESTful API yang dapat memberikan informasi area atau lokasi pada abstraksi level tinggi untuk memenuhi kebutuhan spesifik aplikasi LBS?

1.3 Batasan Masalah

Adapun batasan masalah yang ditentukan oleh penulis pada penelitian ini adalah sebagai berikut:

a. Produk yang dikembangkan ialah RESTful API dan aplikasi berbasis *web*.
b. RESTful API menyediakan fitur otorisasi aplikasi pengguna dan fitur mendapatkan informasi area.
c. Aplikasi berbasis *web* menyediakan fitur pembuatan akun, mengelola aplikasi pengguna, mengelola area, dan dokumentasi penggunaan RESTful API.
d. RESTful API diperuntukkan kepada aplikasi pengguna dan aplikasi berbasis *web* diperuntukkan kepada *developer*.
e. Peta area didefinisikan dalam *file* KML (*Keyhole Markup Language*).
f. Pengambilan koordinat untuk memetakan suatu area dilakukan dengan cara mengambil nilai koordinat *longitude* dan *latitude* area yang dipilih dengan menggunakan Google Maps.
g. Sebuah *file* KML hanya memuat satu obyek area.
h. Obyek area yang dapat disimpan berupa *point*, *linestring*, dan *polygon*.
i. Keluaran dari RESTful API ialah format JSON (*JavaScript Object Notation*).

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah menghasilkan sebuah RESTful API yang dapat digunakan oleh *developer* dalam menyajikan informasi area atau lokasi pada abstraksi level tinggi untuk memenuhi kebutuhan spesifik aplikasi LBS.

1.5 Manfaat Penelitian

Adapun manfaat pada penelitian ini adalah sebagai berikut:

a. Memudahkan *developer* dalam mengelola data area.

b. Memudahkan *developer* dalam menentukan area secara bebas sesuai dengan aplikasi LBS yang dikembangkannya.
c. Memudahkan *developer* dalam memberikan hak akses kepada aplikasi LBS yang dikembangkannya untuk mengakses sumber data.
d. Memudahkan aplikasi LBS dalam menyajikan informasi yang berbeda sesuai lokasi pengguna.

1.6 Metodologi Penelitian

Metodologi penelitian dilakukan supaya proses pelaksanaan penelitian ini dapat berjalan sesuai rencana dan tahapan sehingga memperoleh hasil yang diharapkan. Adapun metodologi yang ditetapkan pada penelitian ini adalah sebagai berikut:

a. Studi Literatur

   Pada tahapan ini, penulis mempelajari rancangan yang telah dibuat oleh (Dirgahayu, 2016) dan mempelajari sumber pendukung yang relevan lainnya.

b. Analisis Kebutuhan

   Pada tahapan ini, penulis menyiapkan kebutuhan dalam pengembangan layanan lokasi. Analisis kebutuhan yang dilakukan meliputi analisis masalah, usulan penyelesaian, identifikasi pengguna, kebutuhan proses, kebutuhan antarmuka dan kebutuhan perangkat lunak dan perangkat keras yang digunakan.

c. Perancangan Sistem

   Pada tahapan ini, penulis menyiapkan rancangan yang nantinya akan dipakai pada proses implementasi. Rancangan sistem yang akan disiapkan adalah arsitektur sistem, *use case diagram*, *activity diagram*, *entity relationship diagram*, rancangan antarmuka, rancangan *request* dan *response* RESTful API, rancangan fungsi pembuatan dan pengolahan data spasial, rancangan skenario penggunaan RESTful API, rancangan pengujian, dan rumus perhitungan yang digunakan.

d. Implementasi

   Pada tahapan ini, penulis mengaplikasikan hasil rancangan sistem yang telah dibuat. Selain itu, tahapan implementasi menerapkan sejumlah materi yang telah dijelaskan pada tahapan sebelumnya.

e. Pengujian

   Pada tahapan ini, penulis melakukan pengujian terhadap RESTful API dan aplikasi berbasis *web*. Pengujian ini dilakukan untuk memeriksa apakah terdapat suatu kesalahan

(*bugs*) atau ketidaksesuaian sistem terhadap analisis yang telah dilakukan sebelumnya. Metode pengujian yang digunakan adalah *black box* dengan teknik *equivalence partitioning*.

1.7 Sistematika Penulisan

Sistematika penulisan adalah uraian singkat sistematis dan pengorganisasian isi laporan penelitian yang bertujuan untuk memudahkan pembaca dalam memahami isi laporan secara keseluruhan. Adapun pembagian penulisan laporan penelitian menjadi lima BAB adalah sebagai berikut:

a. BAB I PENDAHULUAN

   BAB ini berisi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

b. BAB II LANDASAN TEORI

   BAB ini berisi materi – materi yang digunakan dalam penelitian. Materi – materi tersebut adalah LBS (*Location Based Service*), REST (*Representational State Transfer*), OAuth (*Open Authorization*) 2.0, Mysql spatial, KML (*Keyhole Markup Language*), dan penelitian sebelumnya yang terkait dengan penelitian saat ini.

c. BAB III ANALISIS DAN PERANCANGAN SISTEM

   BAB ini berisi analisis masalah, usulan penyelesaian masalah, identifikasi pengguna, kebutuhan proses, kebutuhan antarmuka, kebutuhan perangkat lunak dan perangkat keras yang digunakan, arsitektur sistem, *use case diagram*, *activity diagram*, *entity relationship diagram*, rancangan antarmuka, rancangan *request* dan *response* RESTful API, rancangan fungsi pembuatan dan pengolahan data spasial, rancangan skenario penggunaan RESTful API, rancangan pengujian, dan rumus perhitungan yang digunakan.

d. BAB IV IMPLEMENTASI DAN PENGUJIAN

   BAB ini berisi penjelasan implementasi sistem baik RESTful API maupun aplikasi berbasis *web* dan pengujian sistem.

e. BAB V KESIMPULAN DAN SARAN

   BAB ini berisi penjelasan kesimpulan dan saran pada penelitian.

BAB II
LANDASAN TEORI

2.1 LBS (*Location Based Service*)

2.1.1 Definisi LBS

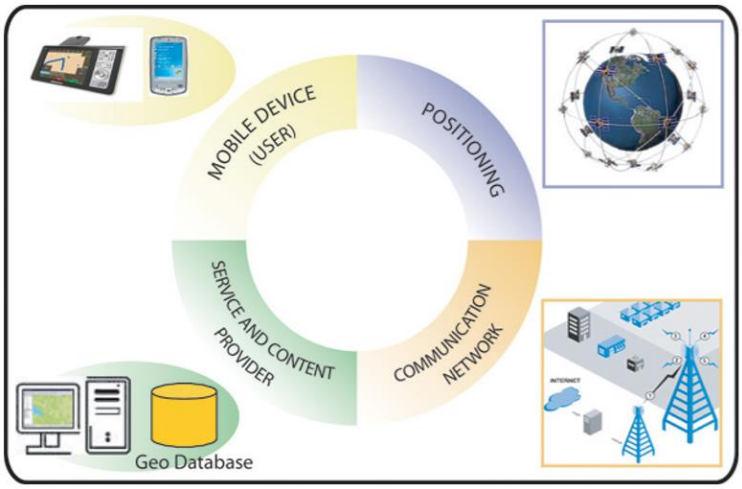
Berikut di bawah ini merupakan definisi dari LBS (*Location Based Service*) menurut para ahli:

a. Menurut (Virrantaus et al., 2001), LBS merupakan sebuah layanan informasi yang dapat diakses oleh perangkat seluler melalui jaringan seluler dan memanfaatkan kemampuan untuk menggunakan lokasi perangkat seluler.
b. Berdasarkan (Open Geospatial Consortium (OGC), 2005), LBS merupakan layanan IP (*Internet Protocol*) nirkabel yang menggunakan informasi geografis untuk melayani pengguna perangkat bergerak sehingga layanan aplikasi tersebut dapat mengeksploitasi posisi perangkat bergerak.

Berdasarkan definisi yang telah dijabarkan di atas, dapat diambil definisi LBS (*Location Based Service*) secara umum yaitu suatu layanan yang memungkinkan pengguna perangkat bergerak atau komputer dapat mengetahui posisi lokasi pengguna dengan memanfaatkan jaringan internet. Sedangkan aplikasi LBS merupakan suatu aplikasi yang menerapkan layanan berbasis lokasi untuk menyajikan informasi geografis kepada pengguna aplikasi tersebut.

2.1.2 Komponen LBS

Terdapat beberapa komponen dasar yang menyusun suatu layanan berbasis lokasi yang ditunjukkan pada Gambar 2.1 adalah sebagai berikut (Steiniger et al., 2012):



Gambar 2.1 Komponen Dasar LBS

a. Perangkat Bergerak

Perangkat bergerak merupakan suatu alat yang digunakan oleh pengguna untuk melakukan suatu permintaan informasi. Melalui perangkat bergerak, pengguna dapat melihat informasi beserta visualisasi yang disajikan oleh penyedia layanan.

b. Jaringan Komunikasi

Jaringan komunikasi merupakan komponen penghubung yang mentransmisikan permintaan pengguna kepada penyedia layanan dan mengembalikan hasil permintaan ke perangkat bergerak pengguna.

c. Komponen Pemosisian

Komponen pemosisian merupakan komponen yang digunakan sebagai alat untuk mengetahui lokasi pengguna. Komponen pemosisian memperoleh posisi seorang pengguna melalui jaringan komunikasi perangkat bergerak atau dapat melalui GPS (*Global Positioning System*).

d. Penyedia Layanan

Penyedia layanan merupakan komponen yang menyediakan berbagai layanan yang dapat digunakan oleh pengguna. Layanan yang diberikan dapat berupa pencarian rute terdekat, pencarian suatu lokasi, menghitung jarak antar dua lokasi, dan lain sebagainya.

e. Penyedia Data dan Konten

Penyedia data dan konten merupakan komponen yang berhubungan dengan data dan konten yang dimiliki oleh penyedia. Data dan konten yang dimiliki oleh penyedia ini yang akan disajikan kepada pengguna berdasarkan permintaan. Data dan konten yang dimiliki dapat berupa koordinat dan detail suatu lokasi.

2.1.3 Penggunaan LBS

Terdapat beberapa penggunaan layanan berbasis lokasi yang umum digunakan adalah sebagai berikut (Steiniger et al., 2012):

a. *Locating*

Locating merupakan aksi yang digunakan untuk menentukan posisi dari pengguna maupun obyek lainnya. Contohnya ialah menentukan posisi pengguna saat ini.

b. *Searching*

Searching merupakan aksi yang digunakan untuk mencari suatu obyek. Contohnya ialah mencari Rumah Sakit, Perpustakaan, dan lain sebagainya.

c. *Navigating*

Navigating merupakan aksi yang digunakan untuk mencari cara untuk mencapai suatu titik atau lokasi. Contohnya ialah mencari jarak terpendek untuk menuju Rumah Sakit.

d. *Identifying*

Identifying merupakan aksi yang digunakan untuk menentukan properti yang dimiliki oleh suatu obyek.

e. *Checking*

Checking merupakan aksi yang digunakan untuk mencari suatu informasi di atau dekat lokasi tertentu. Contohnya ialah mencari acara yang berada di dekat Candi Prambanan.

2.1.4 Contoh Aplikasi LBS

Dengan munculnya layanan berbasis lokasi, maka semakin banyak aplikasi yang tertarik menggunakan layanan tersebut. Ditambah dengan perkembangan perangkat bergerak yang pesat, sehingga pengolahan informasi geografis semakin beragam. Adapun kategori aplikasi yang menerapkan layanan berbasis lokasi adalah sebagai berikut (Steiniger et al., 2012):

a. Layanan Darurat

Kategori layanan darurat yang diterapkan oleh aplikasi LBS bertujuan untuk mendapatkan posisi pengguna pada saat terjadi kejadian darurat seperti kecelakaan, kerusakan kendaraan, dan lain sebagainya. Sehingga *admin* dari suatu aplikasi dapat mengirimkan bantuan secara langsung berdasarkan koordinat. Adapun jenis layanan darurat yang dapat berupa panggilan darurat dan asisten otomotif.

b. Permainan

Kategori permainan menerapkan layanan berbasis lokasi untuk mendukung fitur *augmented reality* dalam memberikan konten permainan yang berbeda pada setiap lokasi.

PokemonGO merupakan suatu permainan yang menerapkan LBS dalam memberikan konten yang disesuaikan dengan posisi pengguna.

c. Informasi

Kategori informasi menerapkan layanan berbasis lokasi untuk memberikan informasi tertentu yang dapat disesuaikan dengan posisi pengguna. Contoh informasi yang diberikan ialah panduan wisata, informasi daerah pemetaan, dan lain sebagainya.

d. Navigasi

Kategori navigasi menerapkan layanan berbasis lokasi untuk memberikan informasi arah menuju suatu lokasi yang ditentukan. Salah satu aplikasi yang -menerapkan fitur navigasi dalam layanan berbasis lokasi adalah Google Maps. Aplikasi tersebut memberikan berbagai alternatif arah untuk mencapai tujuan lokasi.

2.2 REST (*Representational State Transfer*)

2.2.1 Definisi REST

REST (*Representational State Transfer*) merupakan seperangkat prinsip arsitektur yang melakukan transmisi data melalui antarmuka yang terstandarisasi seperti HTTP (Dhingra, 2013). REST API bekerja layaknya seperti aplikasi *web* biasa. *Client* dapat mengirimkan permintaan kepada *server* melalui protokol HTTP dan kemudian *server* memberikan respons balik kepada klien. REST dikembangkan oleh Roy Fielding yang merupakan *co-founder* dari Apache HTTP Server Project. Arsitektur REST menjelaskan enam batasan. Adapun keenam batasan arsitektur REST adalah sebagai berikut (RestApiTutorial, n.d.):

a. *Client Server*

Batasan *client server* menjelaskan suatu antarmuka yang memisahkan bagian *client* dan *server*. Melalui pemisahan antarmuka, REST memberikan keuntungan yaitu *client* tidak perlu berurusan dengan masalah penyimpanan. Hal tersebut meningkatkan portabilitas antarmuka pengguna di berbagai *platform* dan meningkatkan skalabilitas dengan menyederhanakan komponen *server*. Pemisahan antarmuka pengguna memungkinkan komponen dapat terus berkembang secara independen. *Uniform Interface* yang menghubungkan antara *client* dan *server*.

b. *Stateless*

Batasan ini menjadi hal penting pada arsitektur REST. Batasan ini menjelaskan bahwa *server* tidak menyimpan *state* atau penanda *client*. Maksudnya adalah setiap pesan yang dikirimkan oleh *client* bersifat *self-descriptive* atau dengan kata lain setiap pesan yang

dikirimkan memiliki informasi atau konteks yang cukup untuk *server* dalam memeroses pesan tersebut. Setiap *state* atau penanda *session* tersimpan pada pihak *client*.

c. *Cacheable*

Batasan ini menjelaskan bahwa respons *server* bersifat *cacheable*. Setiap respons *server* dapat disimpan secara implisit, eksplisit, atau *negotiated*.

d. *Uniform Interface*

Unform interface menjelaskan antarmuka antara *client* dan *server*. Hal ini menyederhanakan dan memisahkan arsitektur kedua pihak, yang memungkinkan setiap bagian dapat berkembang secara independen. Batasan ini merupakan hal fundamental untuk desain RESTful. RESTful menggunakan HTTP *method* (GET, POST, PUT, DELETE) untuk menjelaskan metode permintaan, URI (*Uniform Resource Identifier*) untuk mengidentifikasi nama sumber, HTTP *response* (*status*, *body*) untuk menjelaskan informasi yang dikembalikan oleh *server*.

e. *Layered System*

Batasan ini menjelaskan bahwa pihak *client* tidak bisa secara langsung terhubung ke *server*. Terdapat perantara antara *server* dan *client*. Hal tersebut berkaitan dengan poin pemisahan antara *client* dan *server*. Perantara tersebut meningkatkan skalabilitas dengan memungkinkan *load-balancing* dan dengan menyediakan *cache* bersama. Serta *layered system* dapat menerapkan kebijakan keamanan.

f. *Code On Demand* (Opsional)

Batasan ini menjelaskan bahwa *server* dapat memberikan atau menyesuaikan fungsionalitas *client* secara sementara dengan mentransferkan *logic* ke dalamnya sehingga dapat dijalankan. Contohnya yaitu komponen yang dikompilasi seperti Java applet dan Javascript.

2.2.2 Perbandingan SOAP dan REST

Terdapat *framework* yang populer yang digunakan untuk membuat suatu layanan *web* yaitu SOAP (*Simple Object Access Protocol*) dan REST (*Representational State Transfer*). Kedua *framework* tersebut memiliki ciri khasnya yang berbeda. Adapun perbandingan antara SOAP dan REST dijabarkan pada Tabel 2.1 (Wagh & Thool, 2012).

Tabel 2.1 Perbandingan SOAP dan REST

SOAP	**REST**
Merupakan teknologi lama	Merupakan teknologi baru
Masih cocok untuk *enterprise* dan B2B	Belum cocok untuk *enterprise*, namun dapat diimplementasikan pada aplikasi perbankan
Ketat aturan penulisan pada interaksi *client-server*	Lemah aturan pada interaksi *client-server*
Dalam hal implementasi, SOAP lebih unggul sebab sudah ada perangkat pengembangan	Namun *developer* REST berpendapat bahwa, REST memiliki fleksibilitas antarmuka
Memiliki muatan yang berat dalam transfer data jika dibandingkan dengan REST	Transfer data sangat ringan karena melalui *interface* atau yang umum dikenal sebagai URI
Perubahan layanan dalam kasus SOAP, maka akan terjadi perubahan pada sisi *Clint*	Perubahan layanan dalam kasus REST, maka tidak memerlukan perubahan apapun dalam kode milik *client*
Membutuhkan *parsing* lampiran binary	Mendukung semua tipe data secara langsung
SOAP bukan infrastruktur nirkabel	REST merupakan infrastruktur nirkabel
Selalu mengembalikan data dalam bentuk XML	Fleksibilitas dalam hal jenis data yang dikembalikan
Menggunakan lebih banyak *bandwidth* sebab respons SOAP membutuhkan 10 kali lebih besar byte	Menggunakan lebih sedikit *bandwidth* sebab respons yang ringan
Permintaan SOAP menggunakan POST dan membutuhkan permintaan XML yang kompleks untuk dibuat, yang mana membuat respons *cache* sulit	Dapat menggunakan permintaan GET sederhana, *intermediate proxy servers* / *reserve-proxies* dapat menyimpan respons dengan sangat mudah
Dirancang untuk menangani lingkungan komputasi terdistribusi	Mengasumsikan sebuah model komunikasi *point to point* dan bukan untuk lingkungan komputasi terdistribusi
Lebih susah untuk dikembangkan, membutuhkan *tools*	Jauh lebih sederhana untuk dikembangkan
SOAP menggunakan WS-Security. WS-Security dibuat karena spesifikasi SOAP merupakan *transport-independent* dan tidak ada asumsi mengenai keamanan yang tersedia pada lapisan transport	REST beranggapan bahwa transportasi akan berupa HTTP (atau HTTPS) dan mekanisme keamanan yang ada di dalam protokol akan tersedia
Memiliki dukungan yang lebih baik dari standar lain (WSDL, WS) dan *tools* dari vendor	Kurangnya dukungan standar untuk keamanan, kebijakan, dan lain – lain, sehingga layanan yang memiliki persyaratan lebih canggih maka lebih sulit untuk dikembangkan

2.2.3 Keunggulan REST

Adapun beberapa keunggulan yang dimiliki oleh REST adalah sebagai berikut (Dhingra, 2013):

a. REST menyediakan infrastruktur yang bagus dalam proses *caching* melalui metode HTTP GET. Hal ini dapat meningkatkan performa jika informasi tidak diubah dan tidak dinamis.
b. REST memisahkan perspektif *server* dan *client* melalui interaksi yang menggunakan HTTP.
c. REST dapat mengembalikan respons dalam format yang beragam dan sesuai dengan permintaan *client*.
d. REST dapat dikembangkan menggunakan bahasa pemrograman manapun selama bahasa tersebut dapat membuat permintaan berbasis *web* melalui HTTP.
e. REST cocok digunakan pada aplikasi perangkat bergerak.

2.3 OAuth 2.0

2.3.1 Definisi OAuth 2.0

OAuth (*Open Authentication*) 2.0 merupakan sebuah *framework* otorisasi yang mengizinkan aplikasi pihak ketiga untuk memperoleh akses terbatas ke layanan *web* (Hardt, 2012). OAuth 2.0 berfokus pada kemudahan pengembangan aplikasi milik klien serta memberikan otoritas khusus untuk aplikasi *web*, aplikasi *desktop*, aplikasi perangkat bergerak, dan lain sebagainya. Otorisasi sendiri merupakan suatu proses untuk memberikan hak akses kepada pihak ketiga dalam mengakses sumber data yang dimiliki oleh pemilik sumber. OAuth 2.0 ditulis ulang dari awal untuk menggantikan versi sebelumnya. Adapun beberapa kendala yang terjadi pada OAuth 1.0 adalah sebagai berikut (Parecki, 2012):

a. Kesulitan dalam persyaratan kriptografi untuk menandatangani permintaan dengan ID dan *secret* klien. Serta kehilangan kemampuan untuk menyalin dan menempelkan cURL dengan mudah. OAuth 2.0 mengganti tanda tangan dengan membutuhkan HTTPS untuk seluruh komunikasi antar perambat, klien, dan API.
b. OAuth 1.0 berfungsi baik untuk aplikasi *web* dan *desktop*, namun sulit untuk aplikasi perangkat bergerak.
c. Kesulitan dalam melakukan pemisahan peran untuk mendapatkan otorisasi pengguna.

2.3.2 Pembagian Peran OAuth 2.0

Adapun beberapa aktor yang berperan dalam protokol alur OAuth adalah sebagai berikut (Boyd, 2012):

a. *Resource Owner*

Resource Owner adalah pemilik sumber data. *Resource owner* memiliki kemampuan untuk memberikan otoritas kepada *client* untuk mengakses sumber data. Contohnya adalah *developer*.

b. *Client*

Client merupakan suatu aplikasi yang melakukan permintaan akses sumber data yang dilindungi atas nama r*esource owner* dan atas wewenang otorisasi yang diberikan oleh *resource owner*. Contohnya adalah aplikasi LBS.

c. *Resource Server*

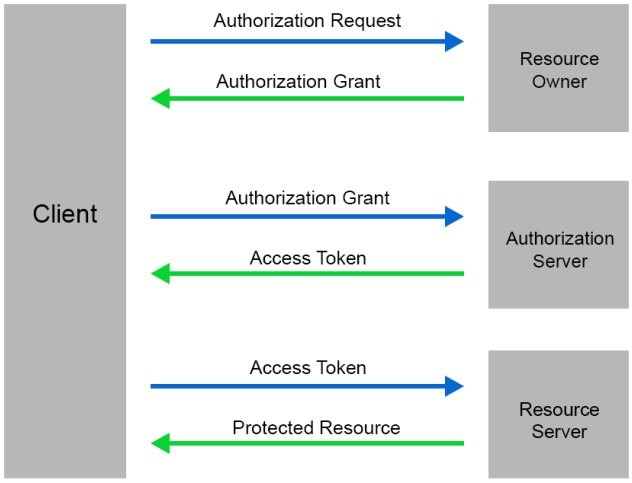
Resource Server merupakan *server* yang melakukan layanan hosting sumber data milik *resource owner* yang dilindungi oleh OAuth, serta mampu menerima dan menanggapi permintaan akses sumber data yang terlindungi menggunakan token akses.

d. *Authorization Server*

Authorization Server merupakan *server* yang mengeluarkan token akses kepada *client* untuk mengakses sumber data setelah melakukan proses otentikasi dan memperoleh otorisasi.

2.3.3 Alur Protokol

Alur protokol merupakan alur yang memuat langkah – langkah *client* untuk mengakses sumber data. Terdapat aliran abstrak yang menggambarkan interaksi antar 4 peran dan mencakup beberapa langkah di antaranya yang ditunjukkan pada Gambar 2.2 adalah sebagai berikut (Hardt, 2012):



Gambar 2.2 Alur Protokol

a. *Client* melakukan permintaan otorisasi dari r*esource owner*. Permintaan otoritas dapat dilakukan secara langsung melalui r*esource owner* ataupun secara tidak langsung melalui a*uthorization server* sebagai perantara.
b. *Client* menerima wewenang otorisasi dari r*esource owner*, yang merupakan kredensial yang mewakili otorisasi r*esource owner*.
c. *Client* meminta token akses dengan melakukan otentikasi terhadap *authorization server* dengan menyajikan wewenang otorisasi yang didapatkan pada langkah sebelumnya.
d. *Authorization server* melakukan otentikasi c*lient* dan validasi terhadap wewenang otorisasi. Jika hasil proses tersebut bernilai valid, maka token akses akan diterbitkan.
e. *Client* melakukan permintaan sumber data terlindungi dari r*esource server* dan melakukan otentikasi dengan memberikan token akses.
f. *Resource server* melakukan validasi terhadap token akses yang diberikan c*lient*. Apabila hasilnya valid, maka r*esource server* mengembalikan respons terhadap permintaan c*lient*.

2.4 Mysql Spatial

2.4.1 Pengenalan Mysql Spatial

Data spasial merupakan suatu data yang mereferensikan pada komponen – komponen permukaan bumi (Kusuma R. et al., 2013). Contoh data spasial adalah kota, pemukiman penduduk, jalan, dan lain sebagainya. Dalam perkembangan pengolahan data spasial, diadakan sebuah kesepakatan internasional yang diikuti oleh lebih dari 250 perusahaan, agensi, dan universitas yang terlibat dalam pengembangan solusi konseptual yang tersedia untuk umum yang dapat berguna untuk semua jenis aplikasi yang mengelola data spasial yang

diberi nama Open Geospatial Consortium (OGC). Kesepakatan OGC mempublikasikan OpenGIS *Implementation Standard for Geographic Information – Simple Feature Access* yang merupakan sebuah dokumen yang mengusulkan beberapa cara konseptual untuk memperluas SQL RMDBS untuk mendukung data spasial.

Mysql mengimplementasikan ekstensi spasial sebagai bagian dari SQL yang berjenis lingkungan geometri. Ekstensi tersebut mengizinkan pembuatan, penyimpanan, dan analisis fitur geografis. Fitur geografis meliputi tipe data yang merepresentasikan nilai spasial, fungsi untuk memanipulasi nilai spasial, dan pengindeksan spasial untuk meningkatkan waktu akses ke kolom spasial. Mysql telah menyediakan berbagai fungsi yang dapat digunakan dalam operasi pada data spasial untuk kebutuhan analisis. Fungsi tersebut dapat dikategorikan menjadi berbagai kelompok besar seperti fungsi untuk membuat geometri dalam beragam format, fungsi untuk mengubah geometri antar format, fungsi yang dapat menggambarkan hubungan antar dua geometri, fungsi yang digunakan untuk membuat geometri baru dari yang sudah ada sebelumnya, dan fungsi untuk melihat kualitatif dan kuantitatif dari suatu geometri.

2.4.2 Tipe Data Spasial

Mysql spatial memiliki tipe data yang sesuai dengan kelas OpenGIS. Berikut di bawah ini merupakan tipe data spasial dasar yang diimplementasikan oleh mysql spatial (Oracle, n.d.):

a. *Point* merupakan obyek yang menyimpan nilai geometri untuk satu lokasi. *Point* merupakan obyek nol dimensi. *Point* dapat dicontohkan sebagai sebuah halte bis, pos polisi, monumen, dan lain sebagainya. Properti yang dimiliki oleh obyek *point* ialah nilai koordinat x dan y.
b. *Linestring* merupakan obyek garis yang disusun oleh titik titik yang berhubungan. Obyek *linestring* setidaknya memiliki dua titik. *Linestring* dapat dicontohkan pada sungai, jalur kereta api, dan lain sebagainya.
c. *Polygon* merupakan obyek permukaan planar yang mewakili geometri sisi banyak. Selain itu, *polygon* merupakan sekumpulan obyek linearring atau *linestring* yang tertutup yang membentuk cincin. Obyek *polygon* harus memiliki titik awal dan titik akhir yang sama. Suatu obyek *polygon* dapat memiliki cincin dalam atau lubang. Contoh dari *polygon* adalah hutan, taman, dan lain sebagainya.

2.4.3 Standar Format Data Spasial

Mysql spatial memberikan beberapa standar format penulisan dalam menggunakan fitur pengolahan data spasial. Salah satu format penulisan adalah format WKT (*Well-Known Text*). WKT merupakan format yang dirancang untuk melakukan pertukaran nilai geometri menggunakan format ASCII (*American Standard Code for Information Interchange*). Format ini sangat cocok digunakan sebab menggunakan teks yang dapat langsung dipahami. Berikut di bawah ini merupakan contoh penulisan dalam format WKT:

a. *Point* : POINT(15 20)
b. *Linestring* : LINESTRING(0 0, 10 10, 20 25, 50 60)
c. *Polygon* : POLYGON((0 0,10 0,10 10,0 10,0 0),(5 5,7 5,7 7,5 7, 5 5))

Pada contoh *admin*, setiap obyek geometri ditulis dengan nama tipe data spasial dan diikuti dengan nilai koordinat (*longitude latitude*).

2.4.4 Fungsi Analisis Sistem

Mysql spatial telah memberikan berbagai fungsi yang dapat digunakan dalam melakukan fungsi analisis. Adapun beberapa fungsi analisis yang digunakan dalam penelitian ini adalah sebagai berikut:

a. ST_GeomFromText(WKT Geometri)

   ST_GeomFromText(WKT Geometri) merupakan fungsi untuk membuat obyek geometri dari tipe data spasial apapun dengan menggunakan format WKT (*Well-Known Text*). Obyek geometri dapat berupa point, linestring, polygon, dan lain sebagainya. Contoh penulisan fungsi adalah ST_GeomFromText ('POINT(1 1)').

b. MBRContains(Geometri, Geometri)

   MBRContains(Geometri, Geometri) merupakan fungsi yang digunakan untuk memeriksa apakah suatu koordinat lokasi yang diberikan berada dalam suatu obyek geometri. Fungsi tersebut akan mengembalikan nilai 1 apabila suatu obyek berada di suatu obyek dan mengembalikan nilai 0 apabila suatu obyek tidak berada di suatu obyek. Contoh penulisan fungsi adalah MBRContains(ST_GeomFromText(‘POLYGON((0 0, 0 1, 2 2, 0 0))’), ST_GeomFromText(‘POINT(1 2)’)).

c. ST_Distance_Sphere(Geometri, Geometri)

   ST_Distance_Sphere(Geometri, Geometri) merupakan fungsi yang digunakan untuk menghitung jarak minimum antara dua titik pada permukaan bumi. Fungsi ini memberikan nilai jarak terdekat dalam satuan meter. Obyek geometri yang dapat

diterima hanya *point* dan *multipoint*. Contoh penulisan fungsi adalah ST_Distance_Sphere(ST_GeomFromText('POINT(0 0)'), ST_GeomFromText('POINT(180 0)')).

2.5 KML (*Keyhole Markup Language*)

2.5.1 Pengenalan KML

KML (*Keyhole Markup Language*) merupakan suatu format *file* yang digunakan untuk menampilkan data geografis pada suatu *earth browser* seperti Google Earth (Google, 2017). KML menggunakan struktur berbasis *tag* dengan elemen dan atribut bersarang serta berdasarkan pada standar XML (*Extensible Markup Language*). KML mengizinkan *developer* untuk membuat obyek dengan menggunakan *tag* yang telah didefinisikan. Semua *tag* yang ada pada KML bersifat *case sensitive*, yang mana harus disesuaikan dengan referensi KML. Obyek – obyek yang dapat dibuat oleh *developer* adalah *point*, *linestring*, *polygon*, dan lain sebagainya. Selain itu obyek tersebut dapat mempercantik tampilan obyek menggunakan *tag* styles, yang mana propertinya sama seperti CSS (*Cascading Style Sheet*). Untuk menampilkan peta yang dibuat menggunakan *file* KML, *developer* dapat mengunggah *file* KML tersebut ke Google Earth. Contoh peta yang diunggah ke Google Earth dapat dilihat pada Gambar 2.3.

Gambar 2.3 Visualisasi Dokumen KML

2.5.2 Dasar Dokumen KML

Dokumen KML memiliki standar struktur yang harus diikuti. Adapun penjelasan struktur sederhana dari suatu dokumen KML adalah sebagai berikut:

a. *Header* XML

Tag ini wajib ada pada setiap dokumen KML.

b. *Tag* KML

Tag KML digunakan untuk mendeklarasikan bahwa *file* tersebut menggunakan elemen atau atribut KML.

c. *Tag* Placemark

Tag Placemark merupakan fitur yang paling umum digunakan. Placemark digunakan untuk menandai suatu posisi di permukaan bumi. Placemark dapat menyimpan obyek *point*, *linestring*, *polygon*, dan lain sebagainya.

d. *Tag* Name

Tag Name digunakan untuk menyimpan nama lokasi yang dipetakan.

e. *Tag* Description

Tag Description digunakan untuk menyimpan deskripsi lokasi yang dipetakan.

f. Geometri

Bagian geometri ini digunakan untuk menyimpan tipe obyek geometri yang dipetakan. Bagian geometri dapat berupa *tag* Point, LineString, Polygon, dan lain sebagainya.

g. *Tag* Coordinates

Tag Coordinates digunakan untuk menyimpan koordinat lokasi yang dipetakan.

Adapun contoh dari dokumen KML ditunjukkan pada Gambar 2.4.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Placemark>
    <name>Simple placemark</name>
    <description>Attached to the ground. Intelligently places itself
       at the height of the underlying terrain.</description>
    <Point>
      <coordinates>-122.0822035425683,37.42228990140251,0</coordinates>
    </Point>
  </Placemark>
</kml>
```

Gambar 2.4 Dokumen KML

2.6 Penelitian Terkait Dengan Pengembangan RESTful API Untuk Application Specific High Level Location Service

Pada penelitian kali ini, penulis membuat suatu RESTful API untuk layanan berbasis lokasi. Sampai saat ini, banyak penelitian yang pernah dilakukan sebelumnya yang membahas tentang pengembangan REST API dan layanan berbasis lokasi. Adapun rangkuman dari penelitian sebelumnya ditunjukkan pada Tabel 2.2.

Tabel 2.2 Rangkuman Penelitian Sebelumnya

Pengarang	Bahasan	Obyek	Hasil
(Dirgahayu, 2016)	*Application-Specific High-Level Location Service*	Pemetaan area menggunakan dokumen KML (*Keyhole Markup Language*) dan Pengolahan data spasial menggunakan Mysql spatial	Rancangan Layanan (*Architecture at large, Internal Architecture, Interface Service, Implementation Service, Example of Service Use)*
(Sumadikarta & Nugroho, 2017)	Perancangan Aplikasi *Location Based Service* Pencarian Rumah Sakit Bermitra Dengan BPJS Kesehatan Berbasis Android	Pencarian Rumah Sakit bermitra dengan BPJS	Aplikasi LBS Pencarian Rumah Sakit bermitra dengan BPJS Kesehatan berbasis android
(Prasetyo, Hamzah, & Sutanta, 2016)	Penyedia Aplikasi Layanan Lokasi Berbasis *Location Based Service* (LBS)	Pengelolaan dan pencarian lokasi	Aplikasi *web* digunakan oleh *admin* untuk mengelola data *member*, data wilayah, data kategori wilayah, data layanan lokasi, dan laporan. Aplikasi *mobile* digunakan oleh *member* untuk mengelola data layanan lokasi dan data pengguna. Kedua aplikasi tersebut memuat fitur pencarian lokasi

Penelitian (Dirgahayu, 2016) membahas tentang perancangan layanan berbasis lokasi. Layanan tersebut diajukan untuk memenuhi kebutuhan informasi lokasi yang dapat disesuaikan dengan beragam aplikasi LBS. Selain itu, rancangan layanan ini sebagai acuan dalam pengembangan lanjutan. Obyek dari penelitian ini adalah pemetaan area dengan menggunakan dokumen KML dan pengolahan data spasial dengan menggunakan Mysql spatial. Hasil dari penelitian ini adalah suatu rancangan layanan yang meliputi arsitektur secara luas, arsitektur internal, antarmuka layanan, implementasi layanan, dan contoh penggunaan layanan.

Penelitian (Sumadikarta & Nugroho, 2017) membahas tentang pembuatan aplikasi yang digunakan untuk mencari lokasi rumah sakit yang bermitra dengan BPJS kesehatan yang

disesuaikan dengan lokasi pengguna. Pada aplikasi ini, terdapat dua tipe pengguna yaitu *admin* dan *user*. Seorang *admin* dapat memanajemen data rumah sakit dan melakukan pencarian rumah sakit. Sedangkan, *user* hanya dapat mencari rumah sakit terdekat sesuai dengan radius yang diberikan *user*. Pada penelitian ini menggunakan perhitungan haversein formula yang digunakan untuk menghitung jarak rumah sakit dari lokasi pengguna. Obyek dari penelitian ini adalah pencarian rumah sakit bermitra BPJS kesehatan. Hasil dari penelitian ini adalah aplikasi LBS pencarian rumah sakit bermitra dengan BPJS Kesehatan berbasis android.

Penelitian (Prasetyo et al., 2016) membahas tentang pembuatan aplikasi yang digunakan untuk mengelola data lokasi dan dapat digunakan untuk melakukan pencarian lokasi. Fitur untuk menambahkan suatu lokasi menggunakan *geo tagging* dengan mengisikan nilai *longitude* dan *latitude*. Lokasi yang dapat ditambahkan berupa *point*. Obyek penelitian ini adalah pengelolaan dan pencarian lokasi. Terdapat dua produk yang dihasilkan pada penelitian ini adalah aplikasi *web* dan aplikasi *mobile*. Aplikasi *web* digunakan oleh *admin* untuk mengelola semua data seperti data *member*, data wilayah, data kategori wilayah, data layanan lokasi, dan laporan. Aplikasi *mobile* digunakan oleh *member* untuk mengelola data layanan lokasi dan data pengguna.

Berdasarkan penelitian sebelumnya yang telah dipilih, penulis mengembangkan suatu layanan lokasi yang digunakan untuk memenuhi kebutuhan spesifik dari aplikasi LBS. Layanan lokasi yang dikembangkan berdasarkan rancangan yang telah dibuat oleh (Dirgahayu, 2016). Layanan lokasi ini dibangun dengan menggunakan arsitektur REST yang mana memudahkan penggunaan layanan baik untuk komputer maupun perangkat bergerak. Layanan lokasi ini menggunakan *framework* OAuth 2.0. Fitur tersebut mengizinkan *developer* untuk dapat mengelola aplikasi pengguna secara mudah dan aman dalam mengatur hak akses ke sumber data. Layanan lokasi ini dapat menyimpan data area yang diunggah oleh *developer* menggunakan *file* KML. *File* KML tersebut menyimpan data area yang ditentukan secara bebas oleh *developer*, namun tetap mengikuti aturan penulisan KML. Layanan lokasi ini menggunakan mysql spatial untuk melakukan fungsi analisis terhadap data area.

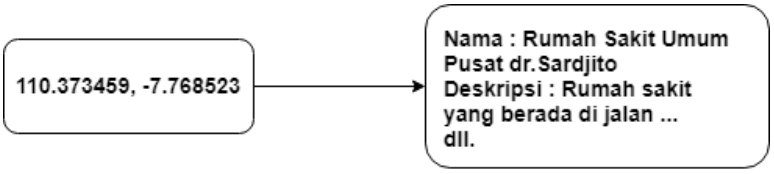
BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Masalah

Pengembangan aplikasi LBS membutuhkan layanan lokasi yang dapat memenuhi kebutuhan informasi area atau lokasi untuk menunjang proses bisnis aplikasi LBS. Pada penelitian ini terdapat beberapa poin yang menjadi masalah dalam pengembangan layanan lokasi yaitu:

a. Informasi area atau lokasi abstraksi level rendah

Informasi area atau lokasi abstraksi level rendah merupakan informasi area atau lokasi yang berbentuk geo-lokasi (nilai koordinat) dari suatu obyek area seperti 110.373459, -7.768523. Koordinat geo-lokasi tersebut tidak secara langsung menjelaskan deskripsi dari suatu area atau lokasi. Koordinat geo-lokasi tersebut bisa didapatkan dengan memanfaatkan modul GPS yang tertanam pada perangkat bergerak atau komputer. Pada penelitian ini akan membahas bagaimana memanfaatkan informasi area atau lokasi abstraksi level rendah menjadi abstraksi level tinggi yang ditunjukkan pada Gambar 3.1.



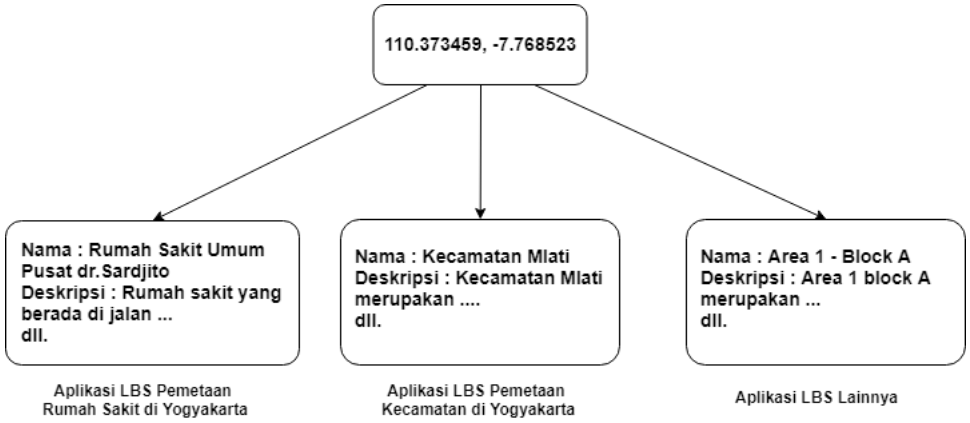
Gambar 3.1 Informasi Area Abstraksi Level Rendah & Abstraksi Level Tinggi

Informasi area atau lokasi abstraksi level tinggi merupakan informasi area atau lokasi yang mengembalikan nilai suatu area atau lokasi yang dapat dipahami secara langsung oleh manusia seperti nama area, deskripsi area, dan lain sebagainya.

b. Pemetaan Area Yang Belum Dapat Disesuaikan Dengan Aplikasi LBS

Informasi area atau lokasi yang diberikan oleh layanan lokasi belum dapat diakomodasikan ke dalam berbagai kebutuhan spesifik aplikasi LBS. Informasi area atau lokasi hanya direpresentasikan dalam satu koordinat saja. Contohnya adalah koordinat geo-lokasi (-7.768523, 110.373459) direpresentasikan oleh Google Maps sebagai Rumah Sakit Umum Pusat dr.Sardjito. Tentu hal tersebut cocok untuk digunakan oleh aplikasi

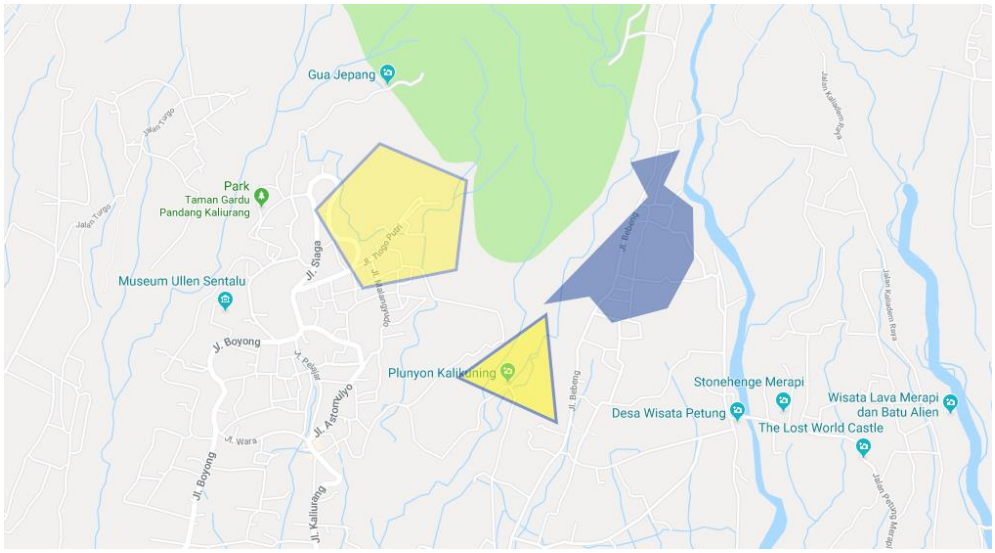
LBS pemetaan rumah sakit Yogyakarta. Apabila ada aplikasi LBS lain yang ingin menggunakan koordinat geo-lokasi tersebut, maka informasi yang diberikan kurang cocok untuk digunakan. Sehingga pada penelitian ini mengangkat bagaimana layanan lokasi dapat memberikan informasi area yang sesuai dengan aplikasi LBS yang ditunjukkan pada Gambar 3.2.



Gambar 3.2 Representasi Geo-Lokasi

c. Kebebasan Dalam Memetakan Area

Suatu area atau lokasi dapat berbentuk berbagai macam baik berupa *point*, *linestring*, maupun *polygon*. Area atau lokasi yang beragam tersebut dapat memenuhi kebutuhan spesifik aplikasi LBS. Sebab, area yang dipetakan sesuai dengan contoh dalam dunia nyata. *Point* dalam dunia nyata dapat berupa halte bis, pos polisi, tugu, dan lain sebagainya. *Linestring* dalam dunia nyata dapat berupa sungai, jalur kereta, lintasan balap, dan lain sebagainya. *Polygon* dalam dunia nyata dapat berupa hutan, taman, gedung, dan lain sebagainya. Pada penelitian ini mengangkat bagaimana membuat suatu area atau lokasi dapat dibentuk sesuai dengan kebutuhan aplikasi LBS yang ditunjukkan pada Gambar 3.3.



Gambar 3.3 Kebebasan Dalam Memetakan Area

3.2 Usulan Penyelesaian Masalah

Pada penjabaran permasalahan di atas, maka penulis mengusulkan solusi permasalahan yaitu mengembangkan sebuah layanan lokasi yang dapat membantu *developer* dalam memberikan informasi area atau lokasi abstraksi level tinggi untuk dapat memenuhi kebutuhan spesifik aplikasi LBS. Layanan lokasi ini mengusung arsitektur REST yang mana memudahkan *developer* dalam menggunakan layanan di berbagai *platform*. Layanan lokasi ini memberikan kemudahan dan kebebasan *developer* dalam mengelola data area. *Developer* dapat memetakan area secara bebas dengan menggunakan *file* KML yang populer dalam pengembangan GIS (*Geographic Information System*). Dengan mengusung kebebasan mengelola area, layanan lokasi dapat memberikan informasi area yang disesuaikan dengan kebutuhan spesifik aplikasi LBS. Selain itu, layanan lokasi juga menerapkan *framework* OAuth 2.0 yang digunakan untuk memudahkan *developer* dalam memberikan otoritas kepada aplikasi pengguna dalam mengakses sumber data. Pada penelitian kali ini menghasilkan dua produk yaitu RESTful API dan aplikasi berbasis *web*. RESTful API digunakan oleh aplikasi LBS untuk mengakses informasi area atau lokasi, sedangkan aplikasi berbasis *web* digunakan oleh *developer* untuk mengelola area, mengelola aplikasi, dan melihat dokumentasi penggunaan layanan.

3.3 Identifikasi Pengguna

Pengembangan sistem pada penelitian ini akan merilis dua produk, yaitu RESTful API dan aplikasi berbasis *web*. Kedua produk tersebut memiliki pengguna yang berbeda. Pada

penelitian ini, penulis mengidentifikasikan pengguna menjadi dua, yaitu *developer* dan aplikasi pengguna. Adapun penjelasan tiap pengguna sistem adalah sebagai berikut:

a. *Developer*

Developer merupakan jenis pengguna yang memiliki akun sistem. *Developer* hanya dapat mengakses aplikasi berbasis *web*. *Developer* dapat mendaftarkan aplikasi LBS atau aplikasi pengguna miliknya. *Developer* juga disebut sebagai *user*. Dalam istilah yang dipakai pada OAuth 2.0, *developer* disebut sebagai *resource owner* yang merupakan pemilik sumber data.

b. Aplikasi Pengguna

Aplikasi pengguna merupakan jenis pengguna yang didaftarkan oleh *developer*. Aplikasi pengguna hanya dapat mengakses RESTful API. Aplikasi pengguna dapat menikmati layanan lokasi seperti memperoleh informasi area terkini dan terdekat berdasarkan koordinat lokasi. Dalam istilah yang dipakai pada OAuth 2.0, aplikasi pengguna disebut sebagai *client* yang mana merupakan pemakai sumber data milik *resource owner*.

3.4 Kebutuhan Proses

Terdapat beberapa proses yang terjadi pada layanan lokasi yang akan dikembangkan. Adapun proses – proses yang dilakukan oleh layanan lokasi adalah sebagai berikut:

a. Proses *Register*

Proses *register* bertujuan untuk mendapatkan akun sistem bagi *developer* yang akan menggunakan layanan lokasi ini. Sehingga, akun tersebut dapat digunakan dalam mengelola aplikasi, mengelola area, dan lain sebagainya. Proses *register* membutuhkan data masukkan berupa nama, *email*, dan *password*. Data – data tersebut akan diproses oleh sistem dan disimpan ke basis data. Setelah proses *register* berjalan lancar, maka akun tersebut dapat digunakan oleh *developer*.

b. Proses *Login*

Proses *login* dibutuhkan sebagai proses otentikasi akun sistem. Sistem mewajibkan *developer* untuk *login* terlebih dahulu sebelum melakukan pengolahan aplikasi pengguna atau area. Proses *login* memerlukan data masukan berupa *email* dan *password*. Data masukan tersebut akan diproses oleh sistem. Setelah proses *login* berjalan lancar, maka sistem akan mengarahkan ke halaman home.

c. Proses *Reset Password*

Proses *reset password* berfungsi untuk mengatur ulang *password* akun sistem. Sebelum dapat melakukan pengaturan ulang *password*, maka sistem akan mengirimkan token *reset password* dalam bentuk tautan kepada *developer* melalui alamat *email* yang diberikan. Tautan tersebut akan mengarahkan *developer* ke halaman pengaturan ulang *password* baru. Pada halaman tersebut *developer* dapat memasukkan *password* baru.

d. Proses Mengelola Aplikasi Pengguna

Proses mengelola aplikasi pengguna terdiri dari proses mendaftar aplikasi pengguna, mengubah aplikasi pengguna, menghapus aplikasi pengguna, menampilkan daftar aplikasi pengguna, dan mencabut otoritas aplikasi pengguna. Adapun penjelasan tiap proses mengelola aplikasi adalah sebagai berikut:

1. Proses mendaftarkan aplikasi pengguna bertujuan agar aplikasi pengguna milik *developer* dapat mengakses sumber data. Proses mendaftar aplikasi pengguna memerlukan data masukkan berupa nama dan URL (*Uniform Resource Locator*) aplikasi. Masukan URL tersebut sebagai alamat penerima otorisasi kode yang diberikan oleh sistem. Setelah melakukan proses mendaftar aplikasi pengguna, maka *developer* akan mendapat keluaran seperti *client* ID, *client secret*, nama, dan URL aplikasi. Kedua data keluaran tersebut digunakan dalam proses mendapatkan token akses.
2. Proses mengubah data aplikasi pengguna memerlukan data masukkan berupa *client* ID, nama, dan URL aplikasi. Data masukkan *client* ID merupakan penanda spesifik data aplikasi pengguna yang akan diubah. Data keluaran dari proses ini sama seperti pada proses sebelumnya.
3. Proses menghapus data aplikasi pengguna memerlukan data masukkan berupa *client* ID. Data tersebut merupakan penanda spesifik data aplikasi pengguna yang akan dihapus.
4. Proses menampilkan daftar aplikasi pengguna memerlukan data masukkan berupa *user* ID. *User* ID didapatkan ketika *developer* melakukan *login* dan secara otomatis *user* ID sudah disimpan. Sistem akan menampilkan daftar aplikasi pengguna berdasarkan kepemilikkan *developer* yang ditunjukkan dengan *user* ID. Hasil keluaran dari proses ini ialah *client* ID, *client secret*, nama, dan URL aplikasi.
5. Proses mencabut otoritas aplikasi pengguna bertujuan agar aplikasi tersebut tidak memiliki hak akses ke sumber data. Proses mencabut otoritas aplikasi pengguna memerlukan data masukan berupa *client* ID. Proses ini dapat dilakukan apabila

aplikasi pengguna tersebut telah memiliki token akses. Setelah melakukan proses mencabut otoritas aplikasi pengguna, maka token akses tersebut tidak berlaku lagi.

e. Proses Mengelola Area

Area merupakan data inti dari layanan lokasi pada penelitian ini. Data tersebut yang akan digunakan oleh aplikasi pengguna untuk kebutuhan informasi lokasi. Dalam proses mengelola area terdiri dari proses membuat, mengubah, menghapus, visualisasi, dan menampilkan daftar area. Adapun penjelasan tiap proses mengelola area adalah sebagai berikut:

1. Proses membuat area memerlukan data masukkan berupa *file* KML. Sistem mengambil data nama, deskripsi, tipe dan koordinat area dari hasil penguraian isi *file* KML untuk disimpan ke dalam basis data. Hasil keluaran dari proses ini ialah nama, deskripsi, dan tipe area.
2. Proses mengubah area memerlukan data masukkan berupa area ID dan *file* KML. Area ID merupakan penanda unik untuk menunjukkan data area yang akan diubah. Sistem mengambil data nama, deskripsi, tipe, dan koordinat area dari hasil penguraian isi *file* KML untuk menggantikan data area yang lama. Hasil keluaran dari proses ini ialah nama, deskripsi, dan tipe area.
3. Proses menghapus area memerlukan data masukkan berupa area ID. Area ID tersebut digunakan sebagai penanda unik untuk menunjukkan data area yang akan dihapus.
4. Proses visualisasi area memerlukan data masukkan berupa *file* KML. Proses ini melibatkan penggunaan Google Maps API untuk menampilkan visualisasi area dengan menggunakan *file* KML.
5. Proses menampilkan daftar area memerlukan data masukkan berupa *client* ID. Sistem akan menampilkan daftar area berdasarkan kepemilikan aplikasi pengguna yang ditunjukkan dengan *client* ID. Hasil keluaran dari proses ini ialah data nama, deskripsi, dan tipe area.

f. Proses Melihat Dokumentasi

Proses ini bertujuan untuk memberikan penjelasan penggunaan layanan lokasi yang diberikan. Dokumentasi berisi penjelasan cara penggunaan layanan. Hal ini memudahkan *developer* dalam menggunakan layanan lokasi.

g. Proses Mendapatkan Token Akses

Proses ini bertujuan untuk mendapatkan token akses yang digunakan oleh aplikasi pengguna dalam mengakses sumber data dari suatu layanan. Token akses diberikan oleh

pemilik sumber data atau *resource owner* kepada *client* yang dibuat oleh *authorization server*. Token merepresentasikan jangkauan kewenangan *client* dalam mengakses sumber data dan lama durasi akses (Hardt, 2012). Token akses biasanya disertakan saat melakukan permintaan mengakses sumber data. Dalam proses mendapatkan token akses, diperlukan data masukkan berupa *client* ID, *client secret*, URL aplikasi, dan kode otorisasi. Hasil keluaran dari proses ini ialah *access token*, *refresh token*, dan masa berlaku token.

h. Proses Mendapatkan Informasi Area

Proses mendapatkan informasi area terbagi menjadi proses mendapatkan informasi area terkini berdasarkan koordinat lokasi, proses mendapatkan informasi area terdekat berdasarkan koordinat lokasi, proses mendapatkan informasi daftar area, dan proses mendapatkan informasi area berdasarkan kata kunci. Adapun penjelasan tiap proses mendapatkan informasi area adalah sebagai berikut:

1. Untuk mendapatkan informasi area terkini berdasarkan koordinat lokasi, maka aplikasi pengguna harus memberikan data masukkan berupa koordinat lokasi dan token akses. Token akses merupakan penanda otentikasi suatu aplikasi pengguna yang wajib disertakan dalam mengakses sumber data. Hasil keluaran dari proses ini ialah area ID, nama, deskripsi, tanggal pembuatan, tanggal perubahan, URL dari *file* KML, dan tipe area.
2. Untuk mendapatkan informasi area terdekat berdasarkan koordinat lokasi, maka aplikasi pengguna harus memberikan data masukkan berupa koordinat lokasi dan token akses. Token akses merupakan penanda otentikasi suatu aplikasi pengguna yang wajib disertakan dalam mengakses sumber data. Hasil keluaran dari proses ini ialah area ID, nama, deskripsi, tanggal pembuatan, tanggal perubahan, URL dari *file* KML, tipe area, dan jarak antara koordinat yang diberikan dengan lokasi area.
3. Untuk mendapatkan informasi daftar area, maka aplikasi pengguna harus memberikan data masukan berupa token akses. Token akses merupakan penanda otentikasi suatu aplikasi yang wajib disertakan dalam mengakses sumber data. Hasil keluaran dari proses ini ialah area ID, nama, deskripsi, tanggal pembuatan, tanggal perubahan, URL dari *file* KML, dan tipe area.
4. Untuk mendapatkan informasi area berdasarkan kata kunci, maka aplikasi pengguna harus memberikan data kata kunci pada parameter *endpoint* dan token akses. Kata kunci berisi teks yang mengandung area yang akan dicari. Token akses merupakan

penanda otentikasi suatu aplikasi yang wajib disertakan dalam mengakses sumber data. Hasil keluaran dari proses ini ialah area ID, nama, deskripsi, tanggal pembuatan, tanggal perubahan, URL dari *file* KML, dan tipe area.

i. Proses Memperbarui Token Akses

Proses ini bertujuan untuk mengganti token akses lama dengan token akses baru. Token akses memiliki masa berlaku penggunaan. Sehingga aplikasi pengguna dapat mengganti token akses jika sudah mendekati masa berlaku tanpa perlu melakukan otorisasi lagi. Proses memperbarui token akses memerlukan data masukkan berupa *refresh token*, *client* ID, dan *client secret*. Hasil keluaran dari proses ini ialah *access token*, *refresh token*, dan masa berlaku token.

3.5 Kebutuhan Antarmuka

Layanan lokasi yang dikembangkan pada penelitian ini memerlukan aplikasi berbasis *web* untuk memudahkan *developer* dalam mengelola sumber data. Dalam pengembangan aplikasi berbasis *web*, penulis membuat daftar antarmuka yang akan diimplementasikan. Adapun kebutuhan antarmuka yang akan diimplementasikan adalah sebagai berikut:

a. Antarmuka halaman register untuk melakukan pendaftaran akun.
b. Antarmuka halaman login untuk melakukan proses otentikasi akun.
c. Antarmuka halaman reset password untuk melakukan proses mengatur ulang *password* akun.
d. Antarmuka halaman dokumentasi untuk menampilkan penjelasan penggunaan layanan.
e. Antarmuka halaman my app untuk melakukan pengelolaan terhadap aplikasi pengguna layanan.
f. Antarmuka halaman resources untuk melakukan pengelolaan terhadap data area milik aplikasi pengguna dan visualisasi.

3.6 Perangkat Lunak Yang Digunakan

Berikut di bawah ini merupakan daftar perangkat lunak yang digunakan dalam penelitian:

a. Windows 10 sebagai sistem operasi yang digunakan dalam pengembangan RESTful API.
b. Laravel 5.6 sebagai *framework* PHP untuk pengembangan RESTful API.
c. Laravel Passport sebagai modul implementasi OAuth 2.0.
d. Google Chrome sebagai perambat untuk mengakses aplikasi berbasis *web* yang dibuat.

e. Aplikasi Postman untuk menguji *endpoint* RESTful API.
f. Mysql 5.7 sebagai sistem manajemen basis data dan pengolahan data spasial.
g. Sublime Text sebagai teks editor dalam penulisan kode.
h. Apache sebagai *web server*.
i. Gitlab sebagai respositori *online* kode sumber RESTful API.

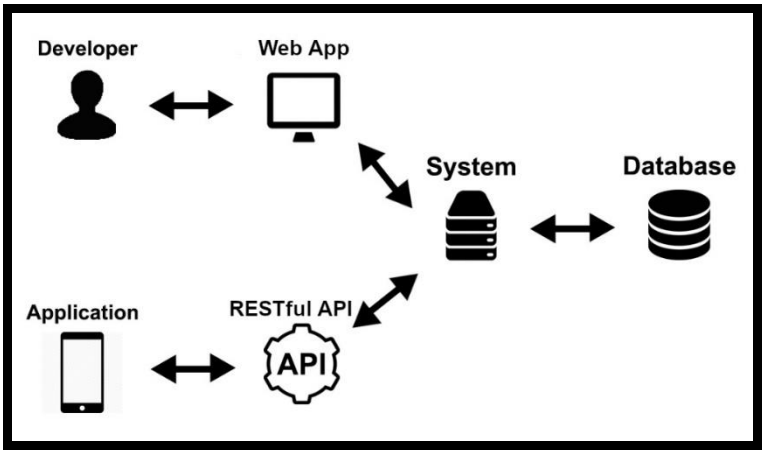
3.7 Perangkat Keras Yang Digunakan

Adapun perangkat keras yang digunakan dalam penelitian ini adalah sebuah laptop dengan spesifikasi:

a. Processor Core i3 (Minimum)
b. RAM 4 GB
c. Harddisk 500 GB

3.8 Arsitektur Sistem

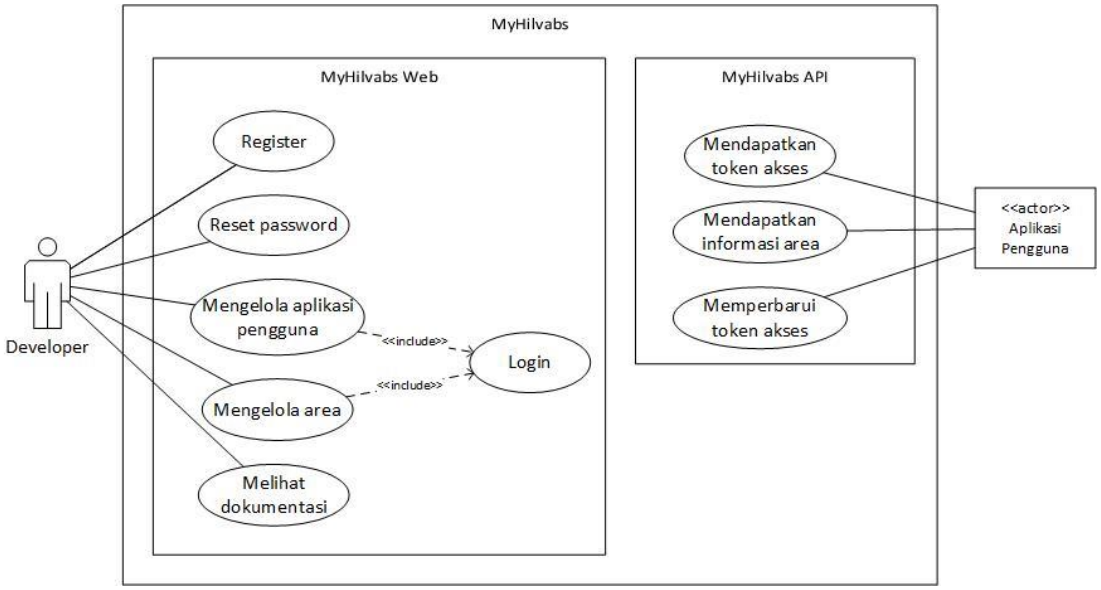
Sistem yang dikembangkan memiliki beberapa bagian yaitu RESTful API, aplikasi berbasis *web*, dan basis data. Setiap bagiannya memiliki fungsi yang berbeda. Aplikasi berbasis *web* digunakan oleh *developer* untuk mengelola aplikasi pengguna layanan, mengelola koleksi data area, dan melihat dokumentasi penggunaan RESTful API. RESTful API digunakan oleh aplikasi pengguna layanan untuk mengakses informasi area. Basis data digunakan sebagai tempat menyimpan data. Arsitektur sistem dapat dilihat pada Gambar 3.4.



Gambar 3.4 Arsitektur Sistem

3.9 *Use Case Diagram*

Use Case Diagram merupakan suatu diagram yang menggambarkan interaksi antara pengguna sistem dengan sistem itu sendiri. Pada penelitian kali ini, perancangan *use case diagram* didasarkan pada hasil analisis sistem yang telah dilakukan sebelumnya. Perancangan *use case diagram* dapat dilihat pada Gambar 3.5.



Gambar 3.5 *Use Case Diagram*

Rancangan *admin* memiliki dua bagian yaitu aplikasi berbasis *web* dan RESTful API. Setiap bagian memiliki aktor yang berbeda. Terdapat dua aktor yang terlibat pada sistem ini yaitu *developer* dan aplikasi pengguna. *Developer* memiliki interaksi pada aplikasi berbasis *web*, sedangkan aplikasi memiliki interaksi pada RESTful API. Gambar 3.5 memiliki sembilan *use case*, di antaranya ialah:

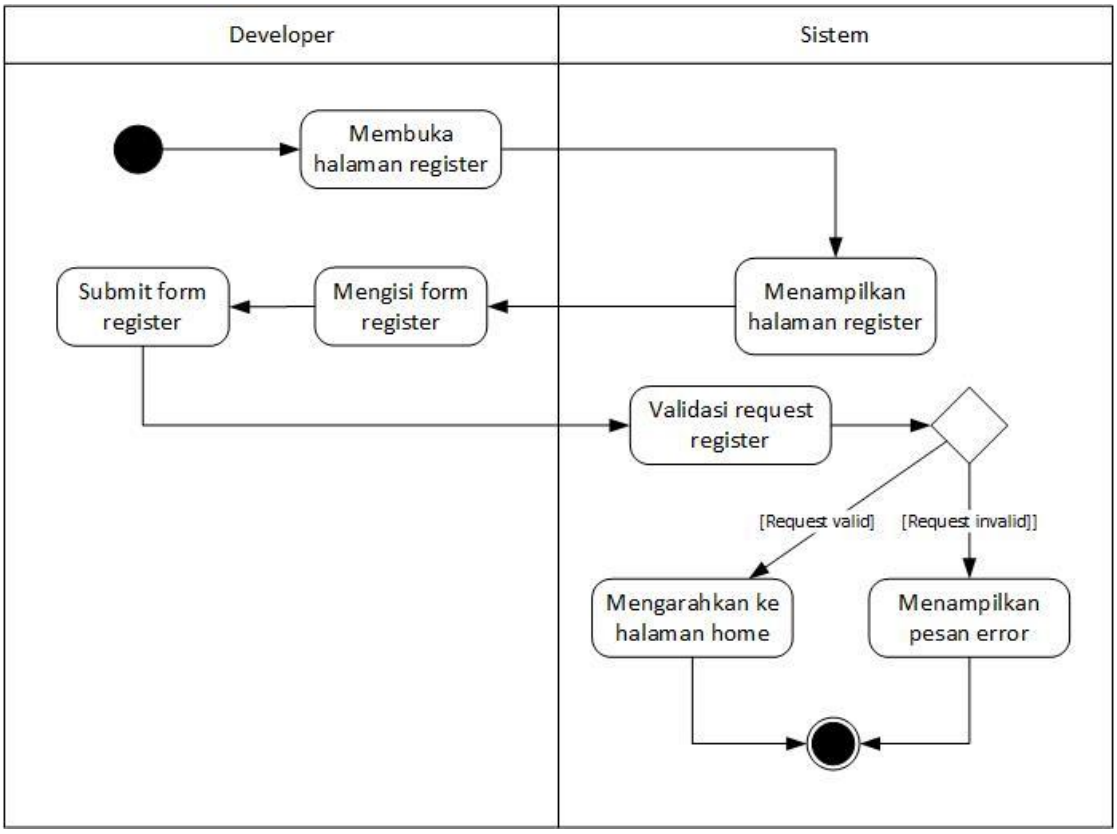
a. UC-1 : *Register*
b. UC-2 : *Login*
c. UC-3 : *Reset Password*
d. UC-4 : Mengelola Aplikasi Pengguna
e. UC-5 : Mengelola Area
f. UC-6 : Melihat Dokumentasi
g. UC-7 : Mendapatkan Token Akses
h. UC-8 : Mendapatkan Informasi Area
i. UC-9 : Memperbarui Token Akses

3.10 *Activity Diagram*

Setelah membuat desain *use case diagram*, penulis membuat desain *activity diagram* berdasarkan daftar *use case* yang telah dibuat sebelumnya. *Activity diagram* merupakan salah satu diagram UML (*Unified Modeling Language*) yang menjelaskan alur kerja dari sistem yang akan dibuat. Adapun *activity diagram* yang dibuat berdasarkan daftar *use case* adalah sebagai berikut:

a. Activity Diagram UC-1 (*Register*)

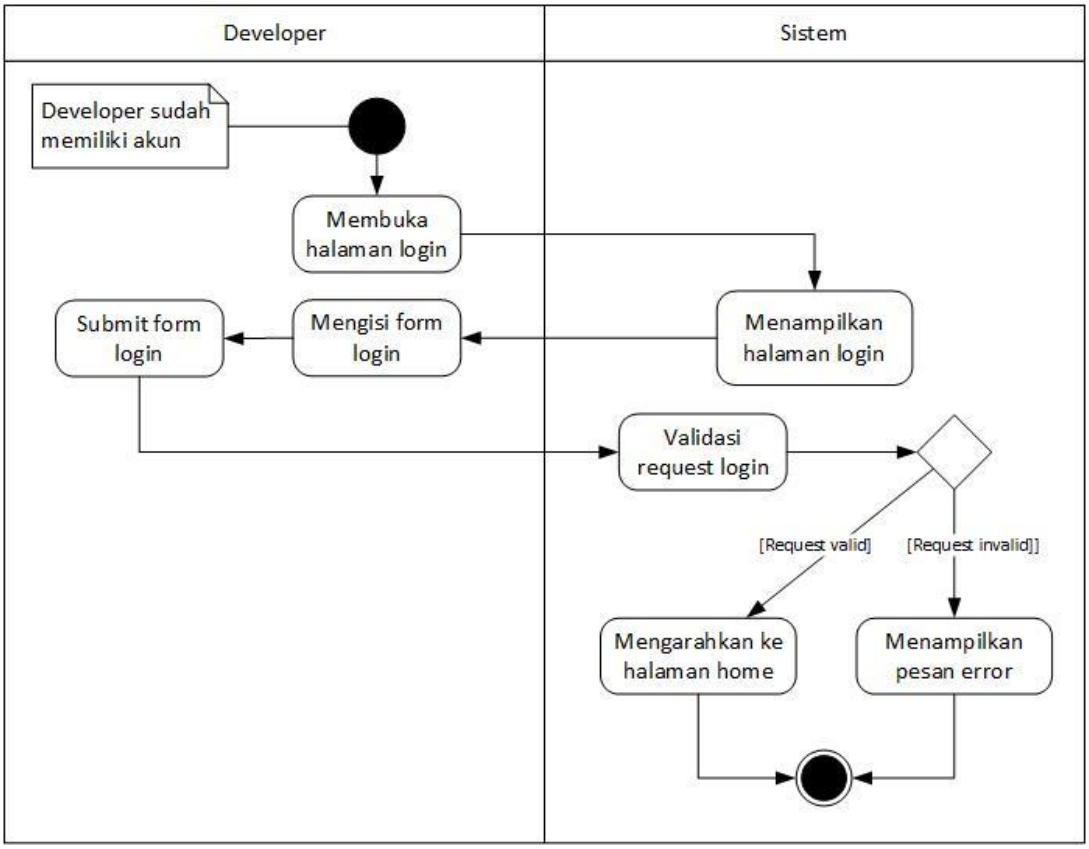
Register merupakan proses untuk membuat akun sistem. Pada diagram ini, menjelaskan alur kerja bagaimana *developer* dalam melakukan registrasi akun. *Activity diagram* untuk proses *register* ditunjukkan pada Gambar 3.6.



Gambar 3.6 *Activity Diagram Register*

b. Activity Diagram UC-2 (*Login*)

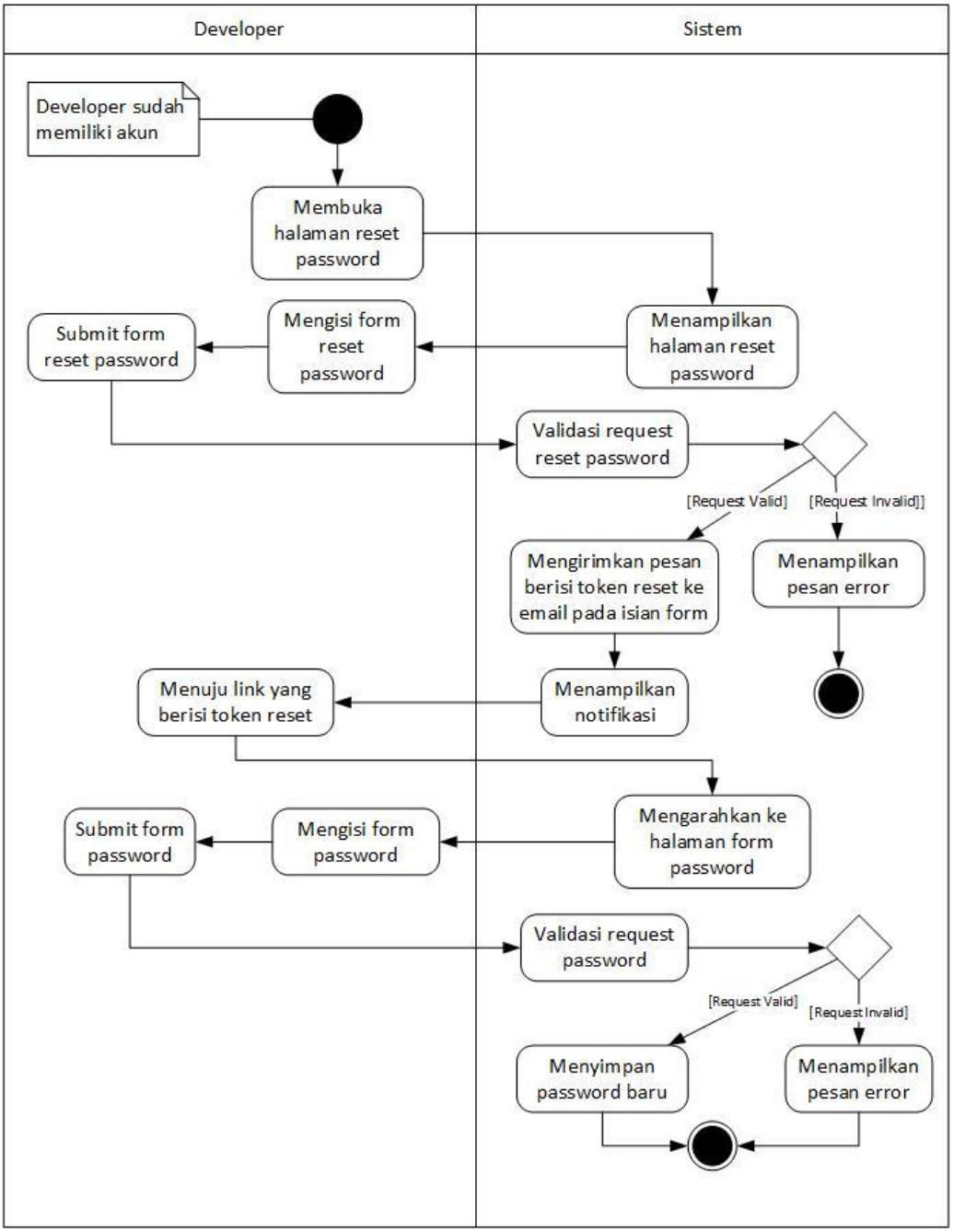
Login merupakan proses yang dilakukan oleh *developer* sebagai pemilik akun untuk masuk ke dalam sistem. Pada diagram ini, menjelaskan bagaimana alur kerja proses *login*. Sebelum melakukan proses *login*, *developer* harus memiliki akun sistem. *Activity diagram* untuk proses *login* ditunjukkan pada Gambar 3.7.



Gambar 3.7 *Activity Diagram Login*

c. Activity Diagram UC-3 (*Reset Password*)

Reset Password merupakan proses untuk mengatur ulang *password* akun. Pada diagram ini, menjelaskan alur kerja bagaimana *developer* dalam melakukan *reset password*. *Activity diagram* untuk proses *reset password* ditunjukkan pada Gambar 3.8.

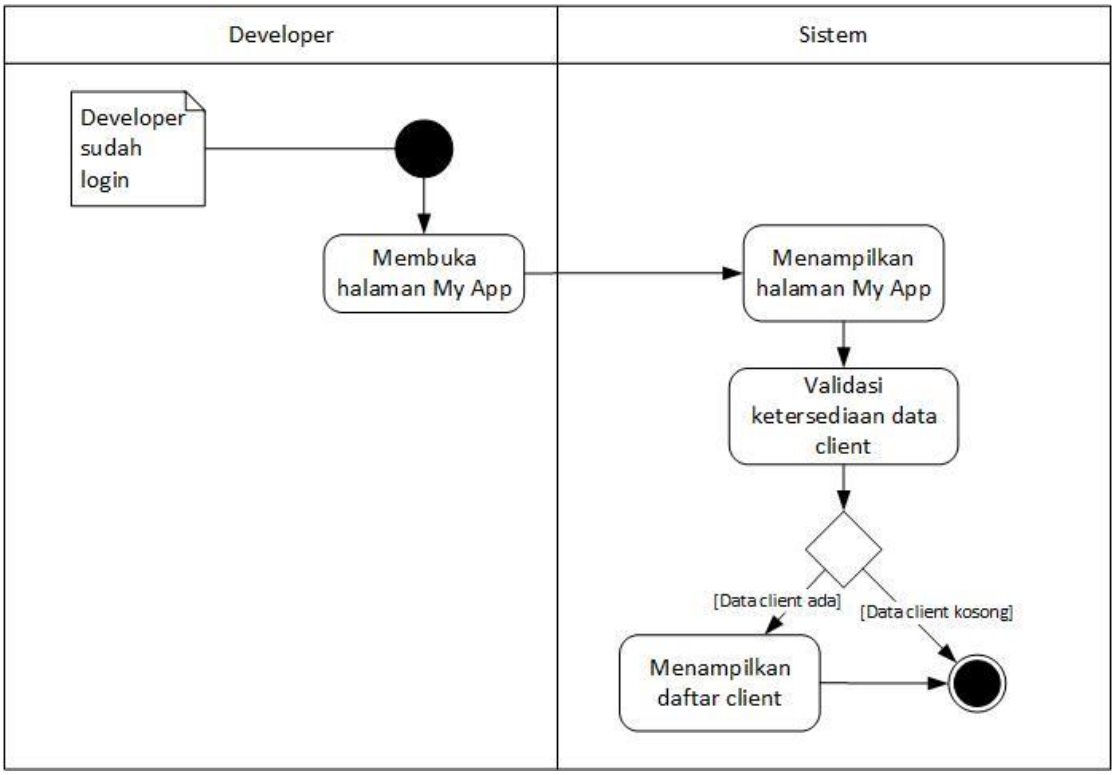


Gambar 3.8 *Activity Diagram Reset Password*

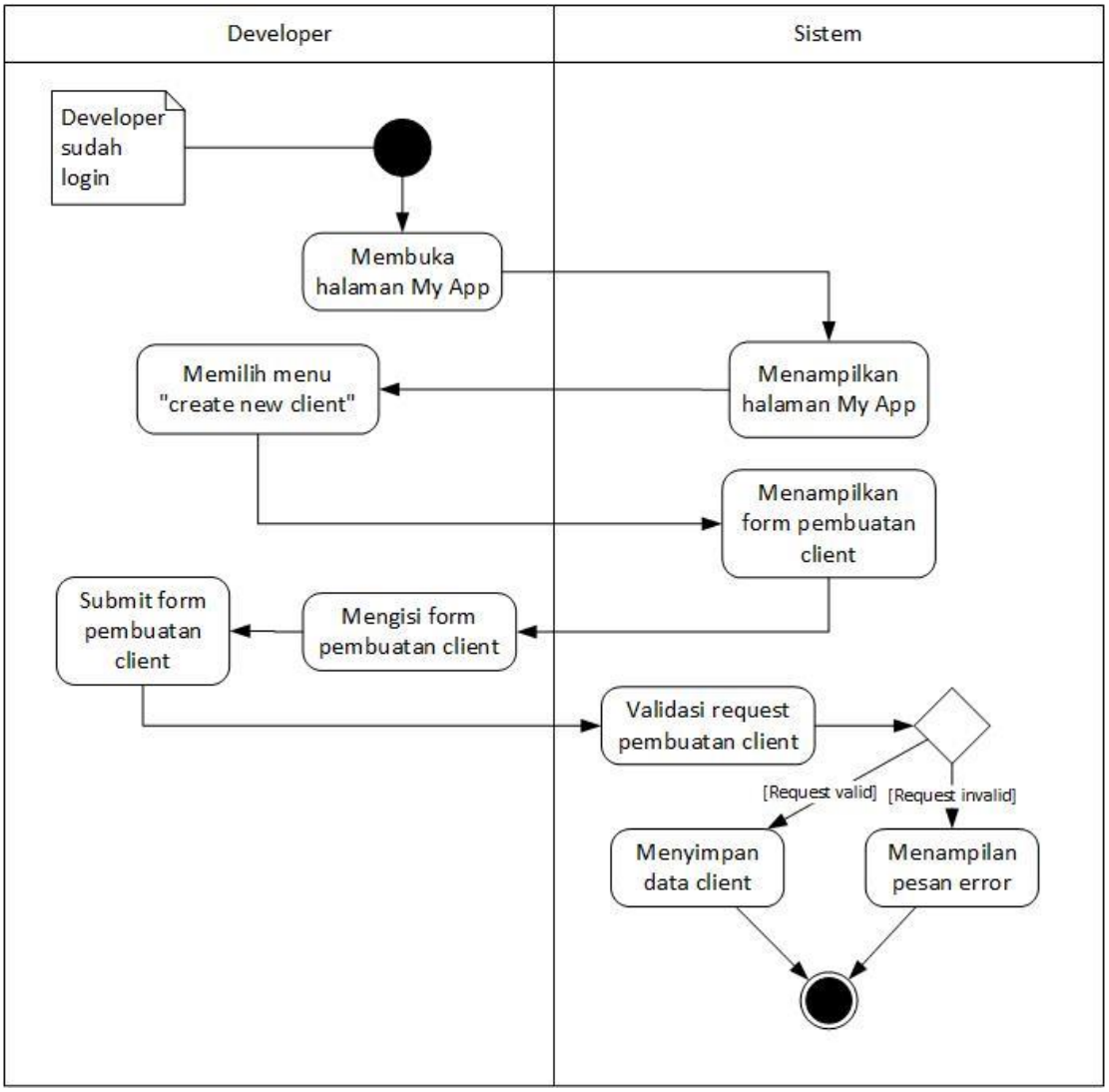
d. *Activity Diagram* UC-4 (Mengelola Aplikasi Pengguna)

Proses mengelola aplikasi terdiri dari proses mendaftar, mengubah, menghapus, menampilkan daftar, dan mencabut otoritas aplikasi pengguna. Pada diagram ini, menjelaskan alur kerja dari subproses mengelola aplikasi. *Activity diagram* untuk proses mengelola aplikasi pengguna ditunjukkan pada Gambar 3.9 (menampilkan daftar aplikasi pengguna), Gambar 3.10 (mendaftarkan aplikasi pengguna), Gambar 3.11 (mengubah

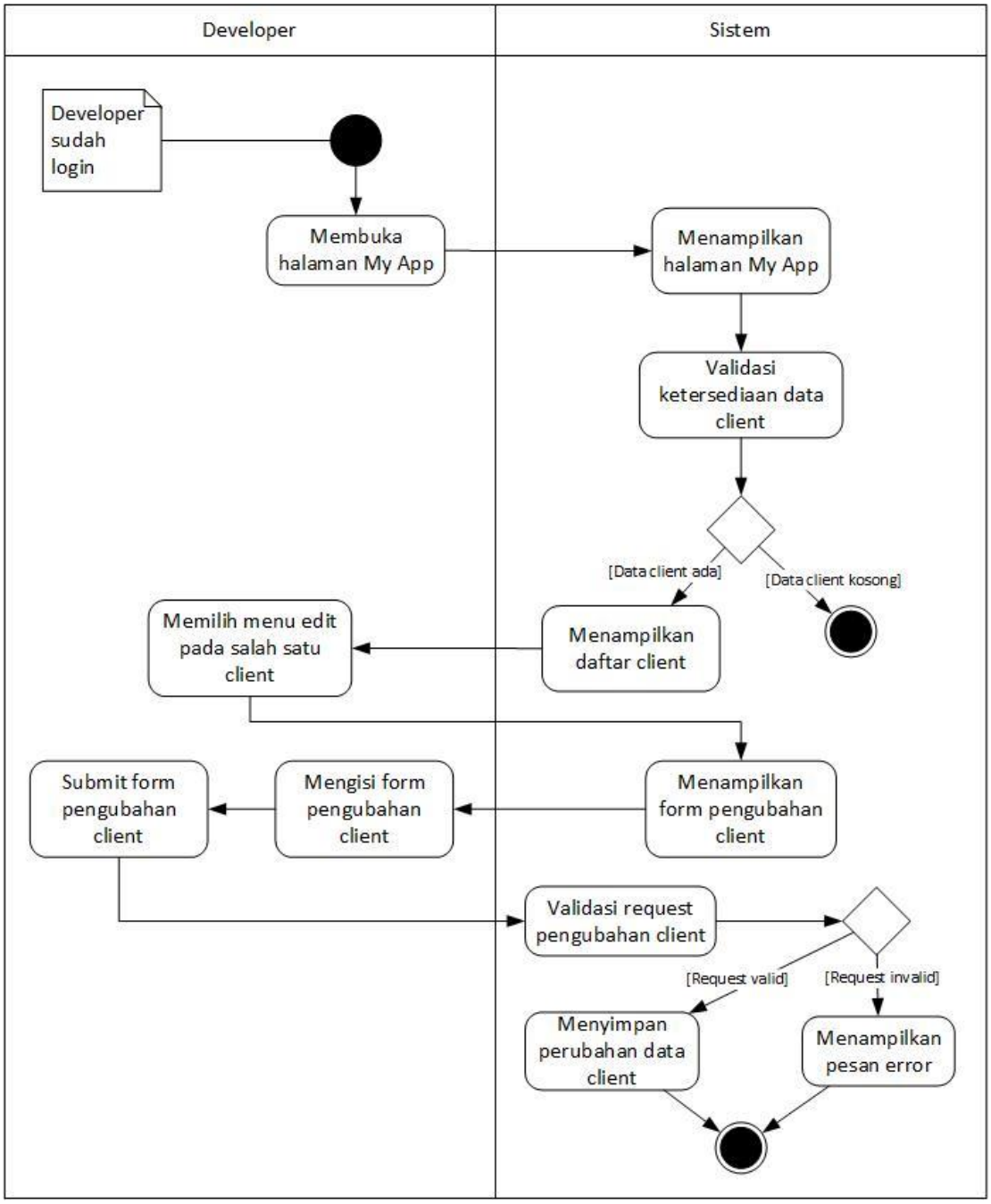
aplikasi pengguna), Gambar 3.12 (menghapus aplikasi pengguna), dan Gambar 3.13 (mencabut otoritas aplikasi pengguna).



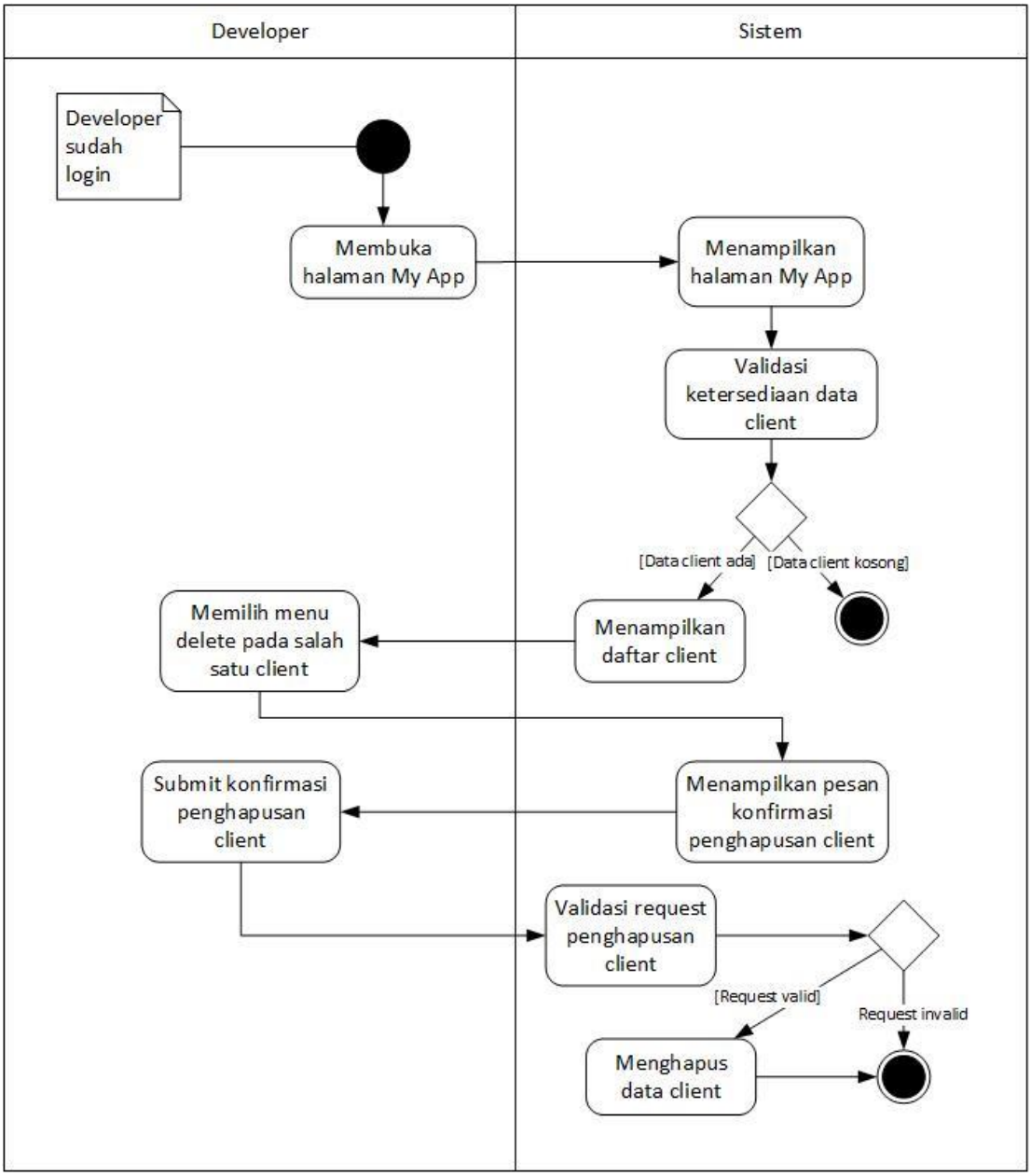
Gambar 3.9 *Activity Diagram* Menampilkan Daftar Aplikasi Pengguna



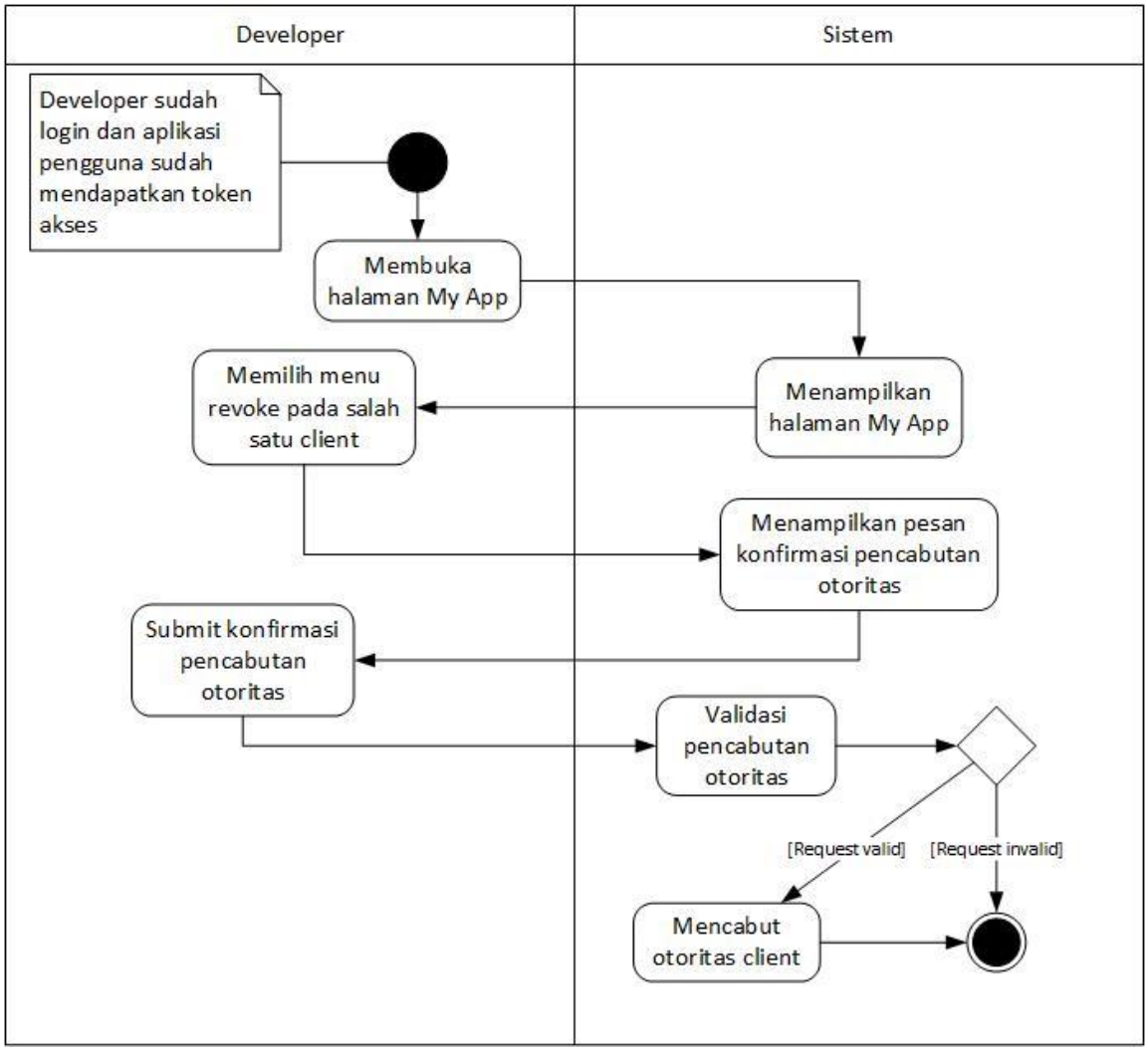
Gambar 3.10 *Activity Diagram* Mendaftarkan Aplikasi Pengguna



Gambar 3.11 *Activity Diagram* Mengubah Aplikasi Pengguna



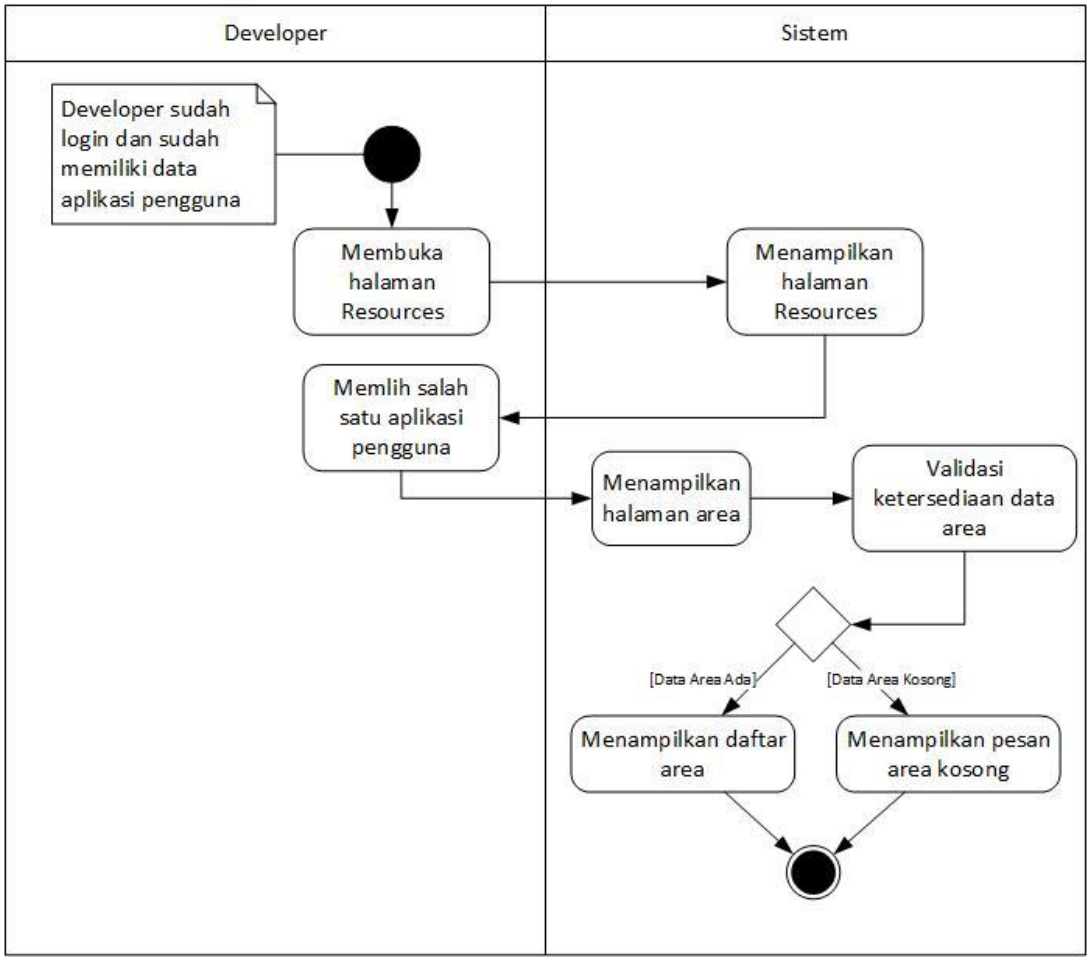
Gambar 3.12 *Activity Diagram* Menghapus Aplikasi Pengguna



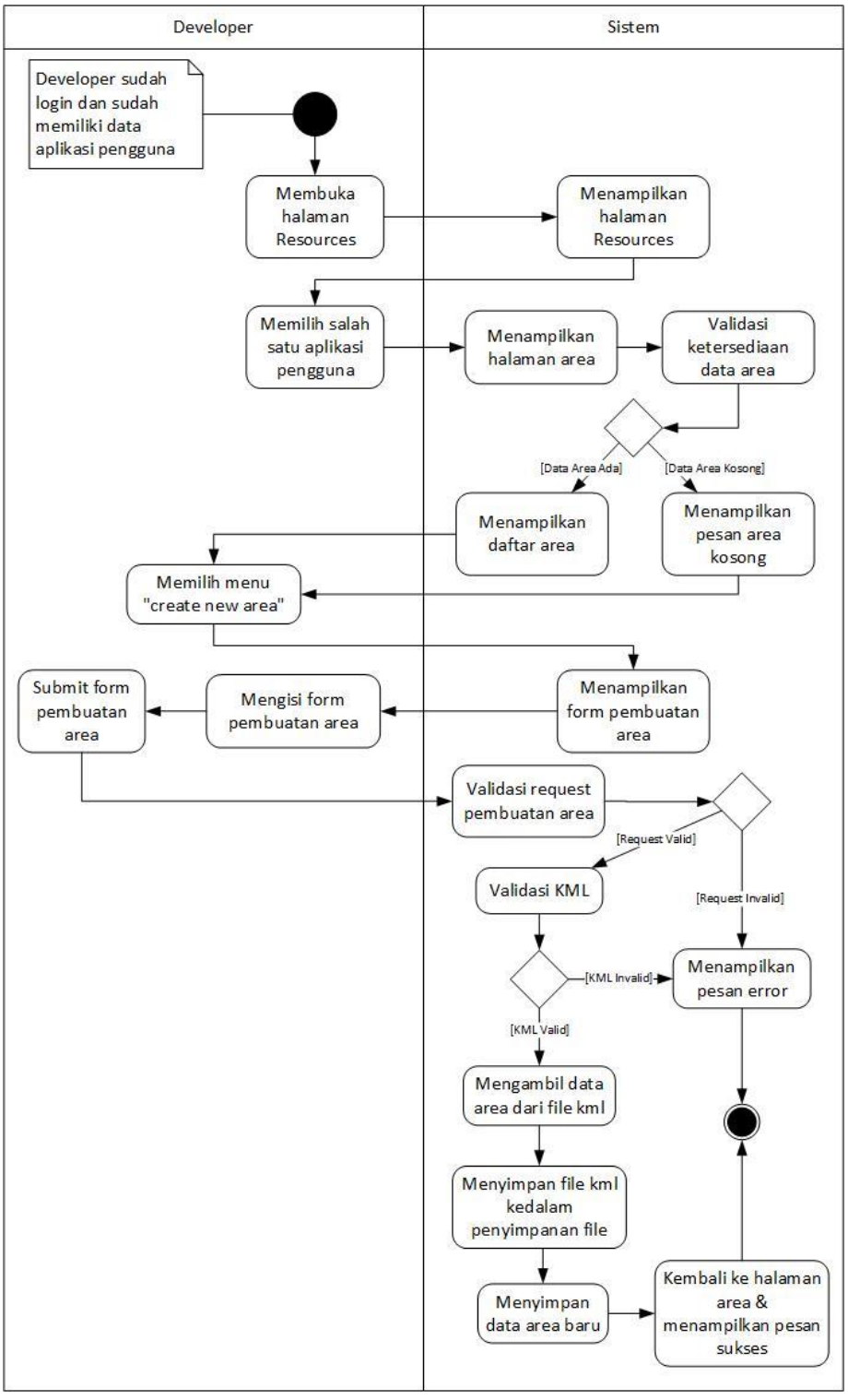
Gambar 3.13 *Activity Diagram* Mencabut Otoritas Aplikasi Pengguna

e. *Activity Diagram* UC-5 (Mengelola Area)

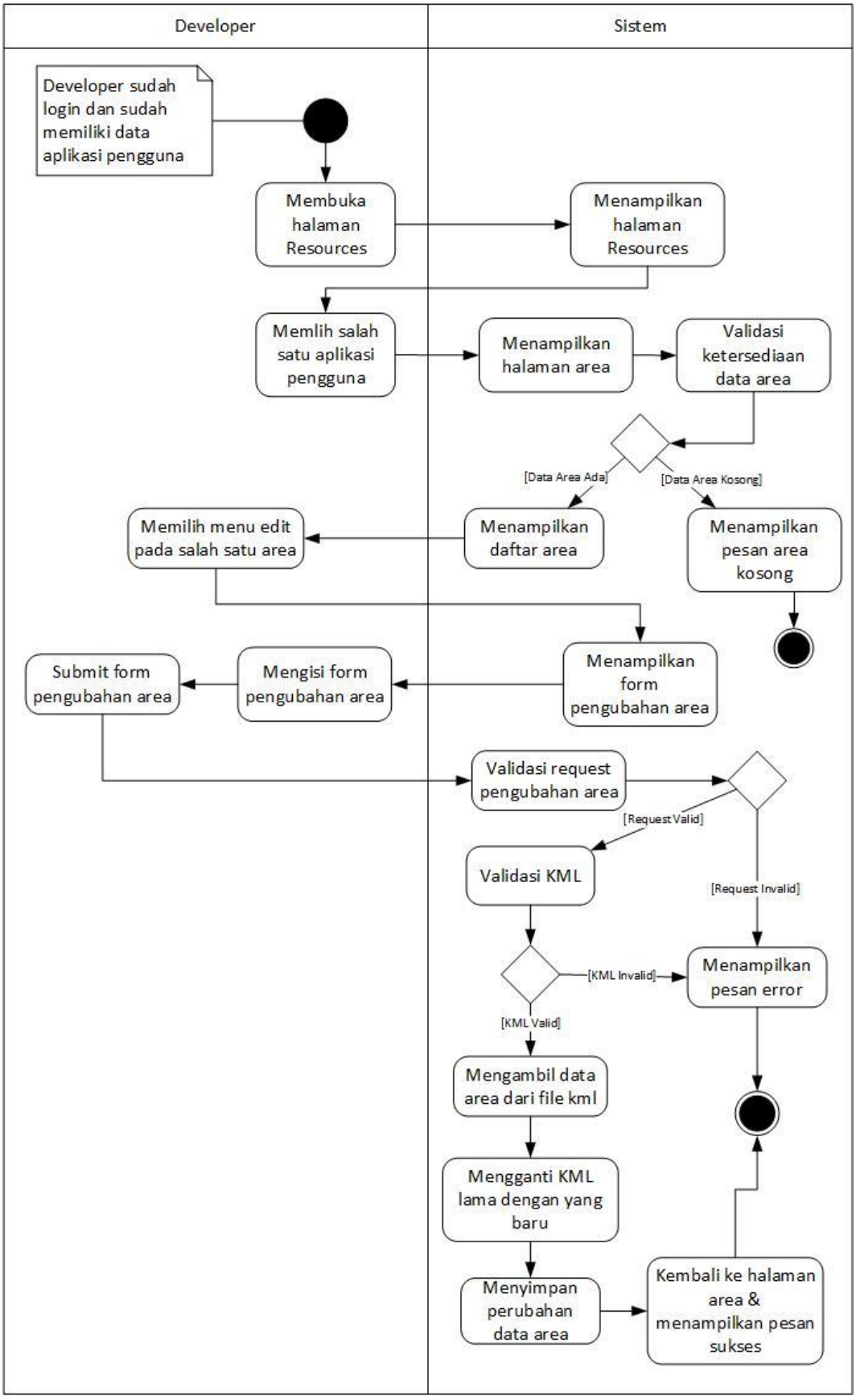
Proses mengelola area terdiri dari proses membuat, mengubah, menghapus, menampilkan, dan visualisasi area. Pada diagram ini, menjelaskan alur kerja dari subproses mengelola area. *Activty diagram* untuk mengelola area ditunjukkan pada Gambar 3.14 (menampilkan daftar area), Gambar 3.15 (membuat area), Gambar 3.16 (mengubah area), Gambar 3.17 (menghapus area) dan Gambar 3.18 (menampilkan visualisasi area).



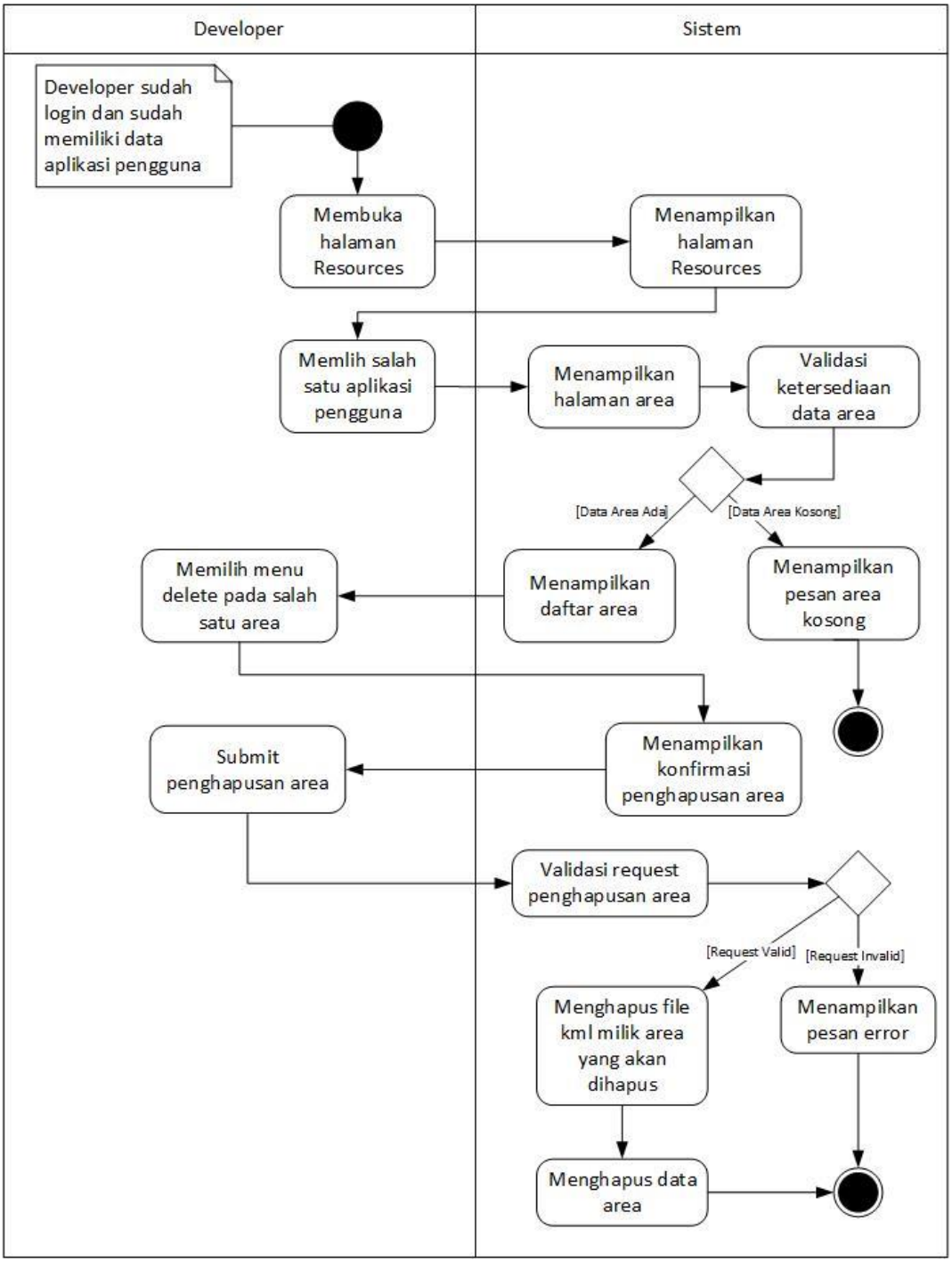
Gambar 3.14 *Activity Diagram* Menampilkan Daftar Area



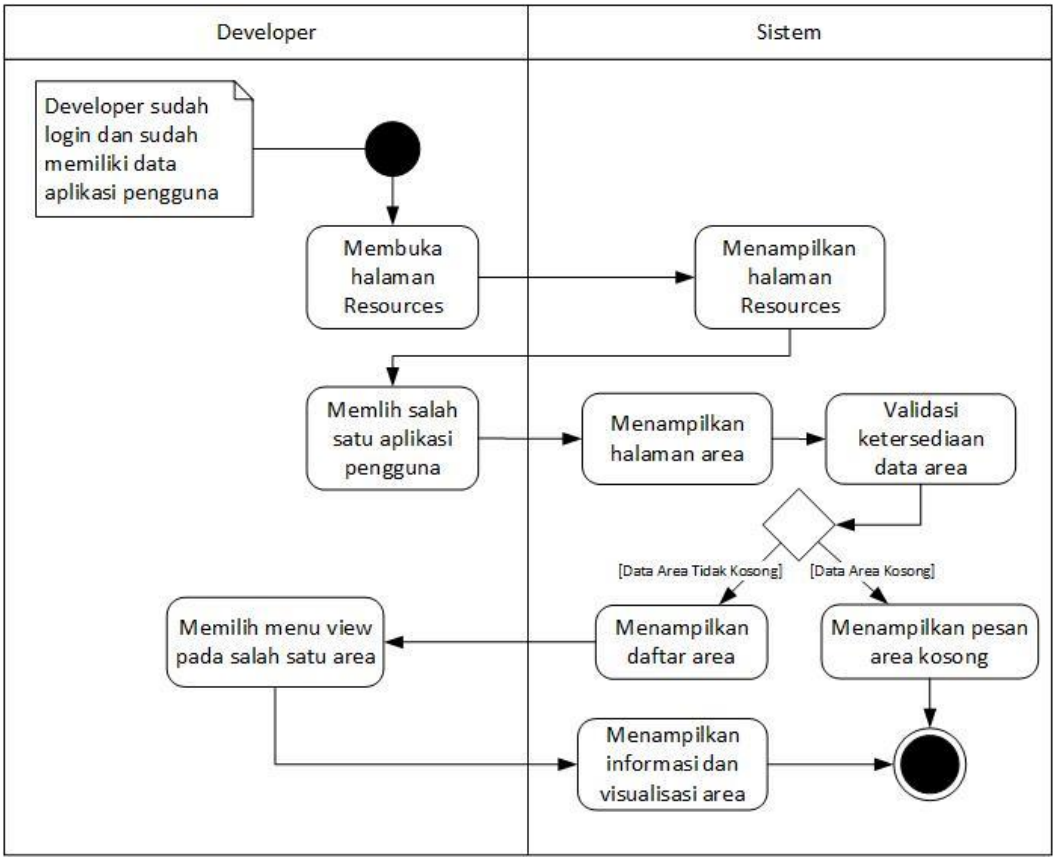
Gambar 3.15 *Activity Diagram* Membuat Area



Gambar 3.16 *Activity Diagram* Mengubah Area



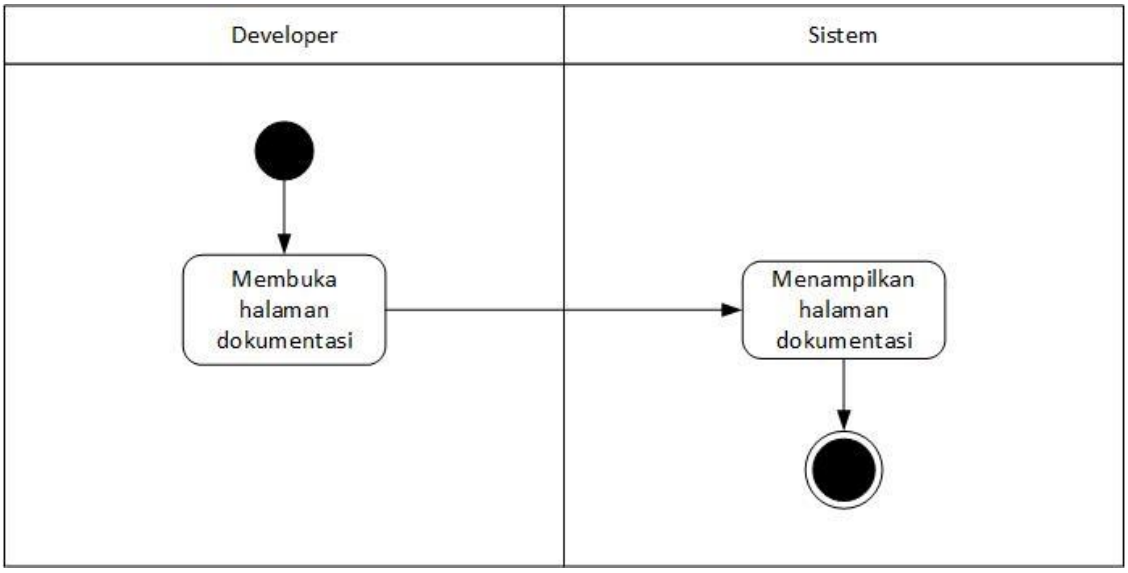
Gambar 3.17 *Activity Diagram* Menghapus Area



Gambar 3.18 *Activity Diagram* Menampilkan Visualisasi Area

f. *Activity Diagram* UC-6 (Melihat Dokumentasi)

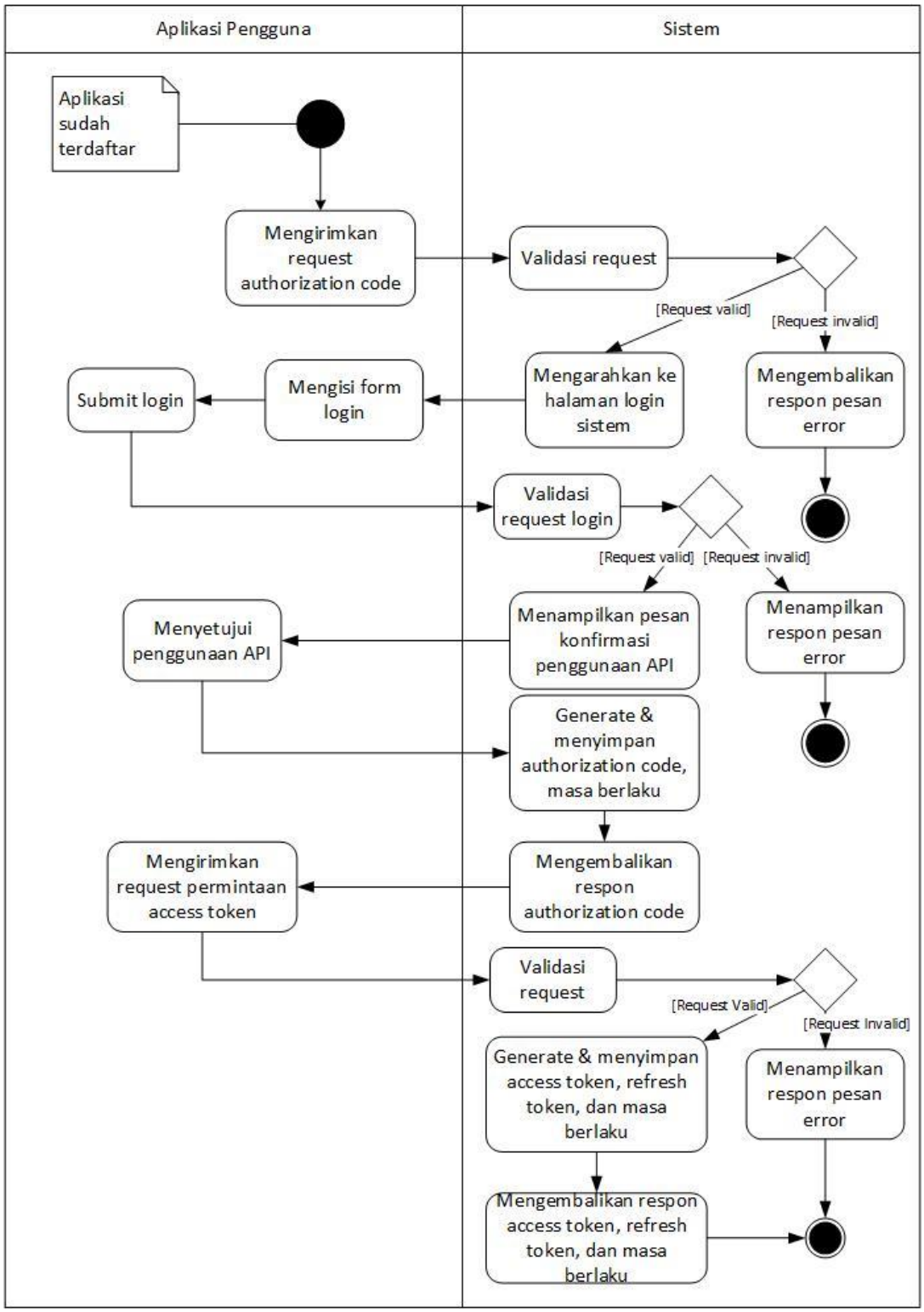
Dokumentasi merupakan suatu deskripsi penggunaan suatu layanan mulai dari tata cara pemakaian, pengaksesan sumber data, dan lain sebagainya. Pada diagram ini, menjelaskan alur kerja proses melihat dokumentasi. *Activity diagram* untuk melihat dokumentasi ditunjukkan pada Gambar 3.19.



Gambar 3.19 *Activity Diagram* Melihat Dokumentasi

g. *Activity Diagram* UC-7 (Mendapatkan Token Akses)

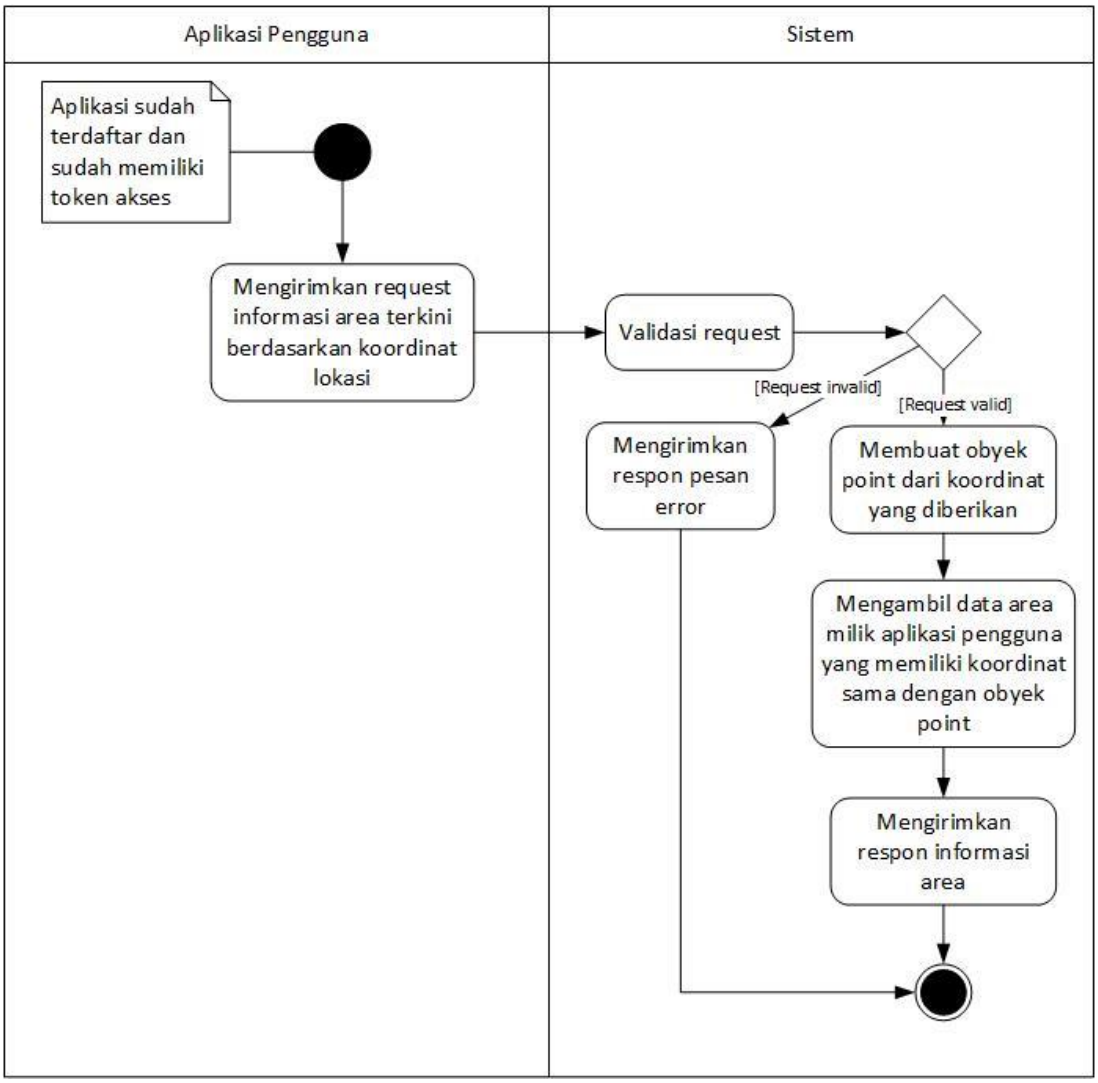
Token akses merupakan penanda izin akses aplikasi pengguna dalam mengakses sumber data. Pada diagram ini, menjelaskan proses aplikasi pengguna atau *client* dalam mendapatkan token akses. *Activity diagram* untuk mendapatkan token akses ditunjukkan pada Gambar 3.20.



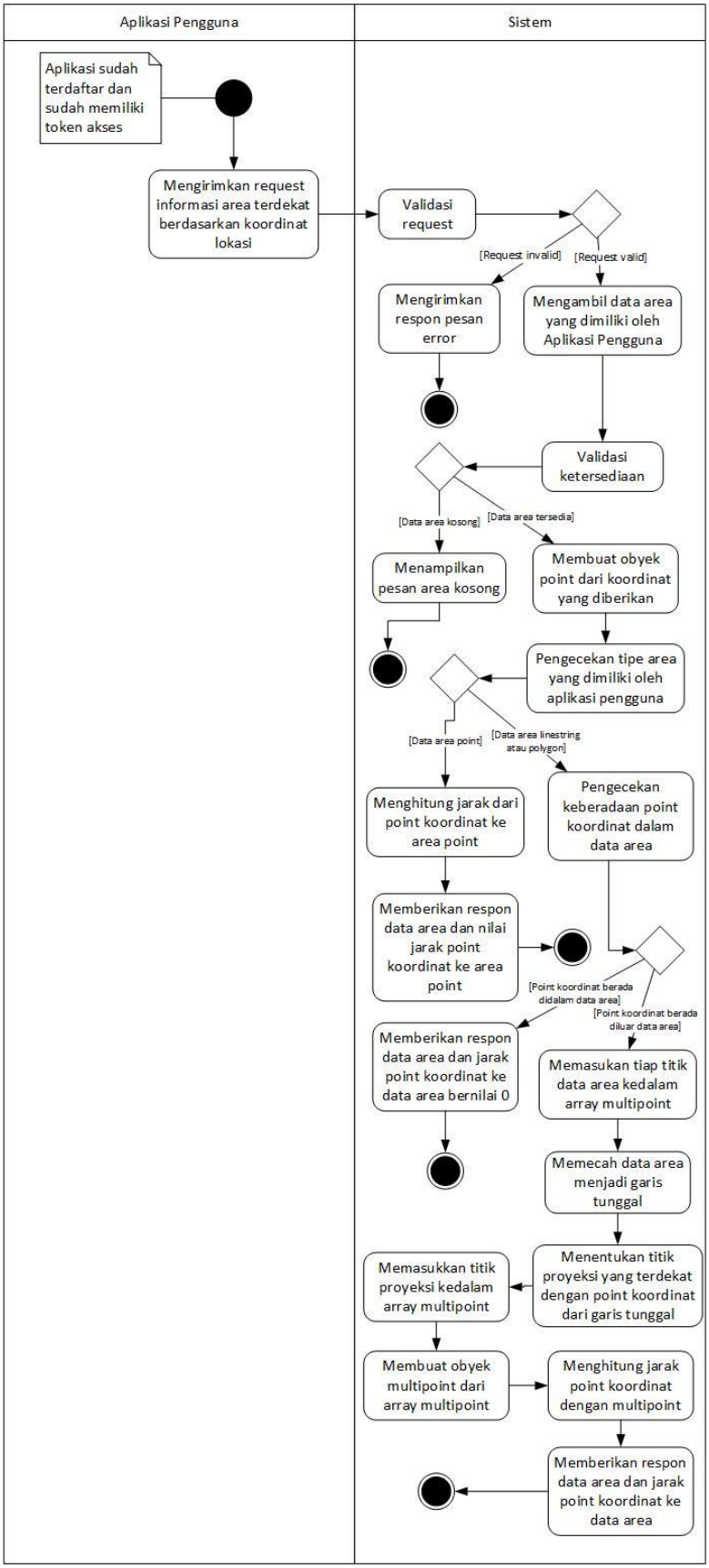
Gambar 3.20 *Activity Diagram* Mendapatkan Token Akses

h. *Activity Diagram* UC-8 (Mendapatkan Informasi Area)

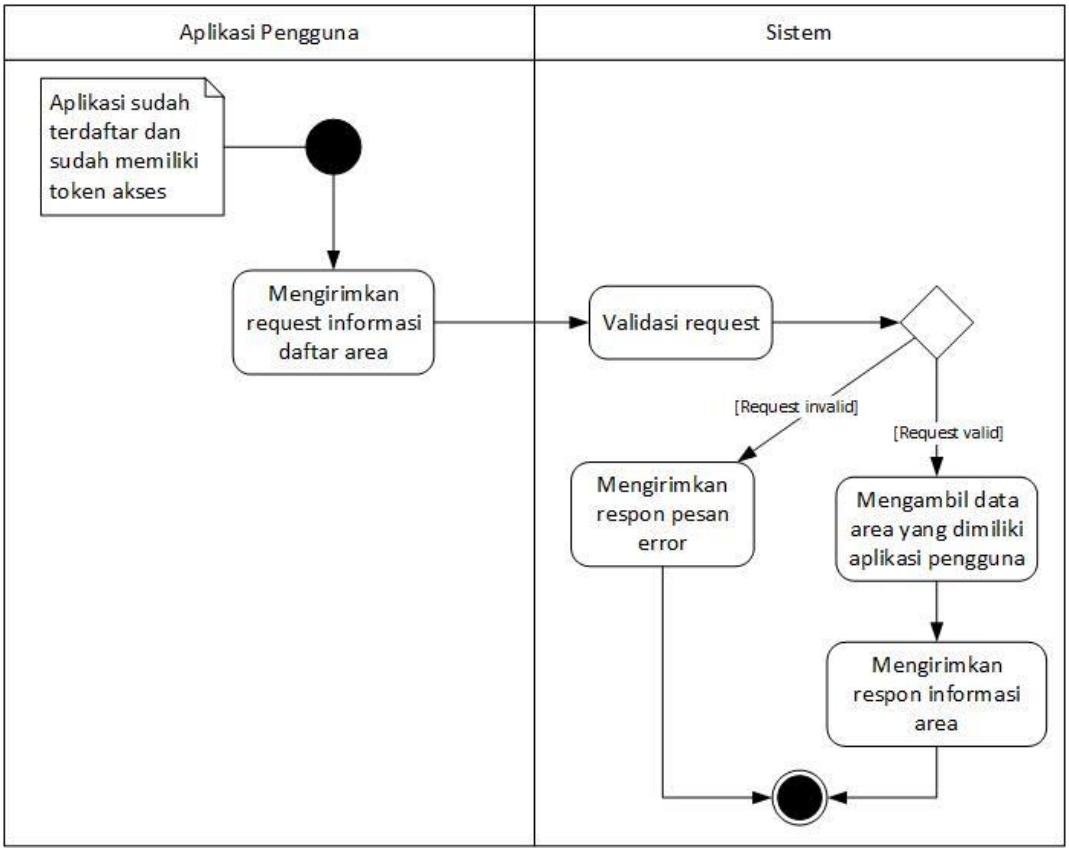
Proses mendapatkan informasi area terdiri dari proses mendapatkan informasi area terkini berdasarkan koordinat lokasi, proses mendapatkan informasi area terdekat berdasarkan koordinat lokasi, proses mendapatkan informasi daftar area, dan proses mendapatkan informasi area berdasarkan kata kunci. Pada diagram ini, menjelaskan alur kerja dari subproses mendapatkan informasi area. *Activity diagram* untuk mendapatkan informasi area ditunjukkan pada Gambar 3.21 (informasi area terkini), Gambar 3.22 (informasi area terdekat), Gambar 3.23 (informasi daftar area), dan Gambar 3.24 (informasi area berdasarkan kata kunci).



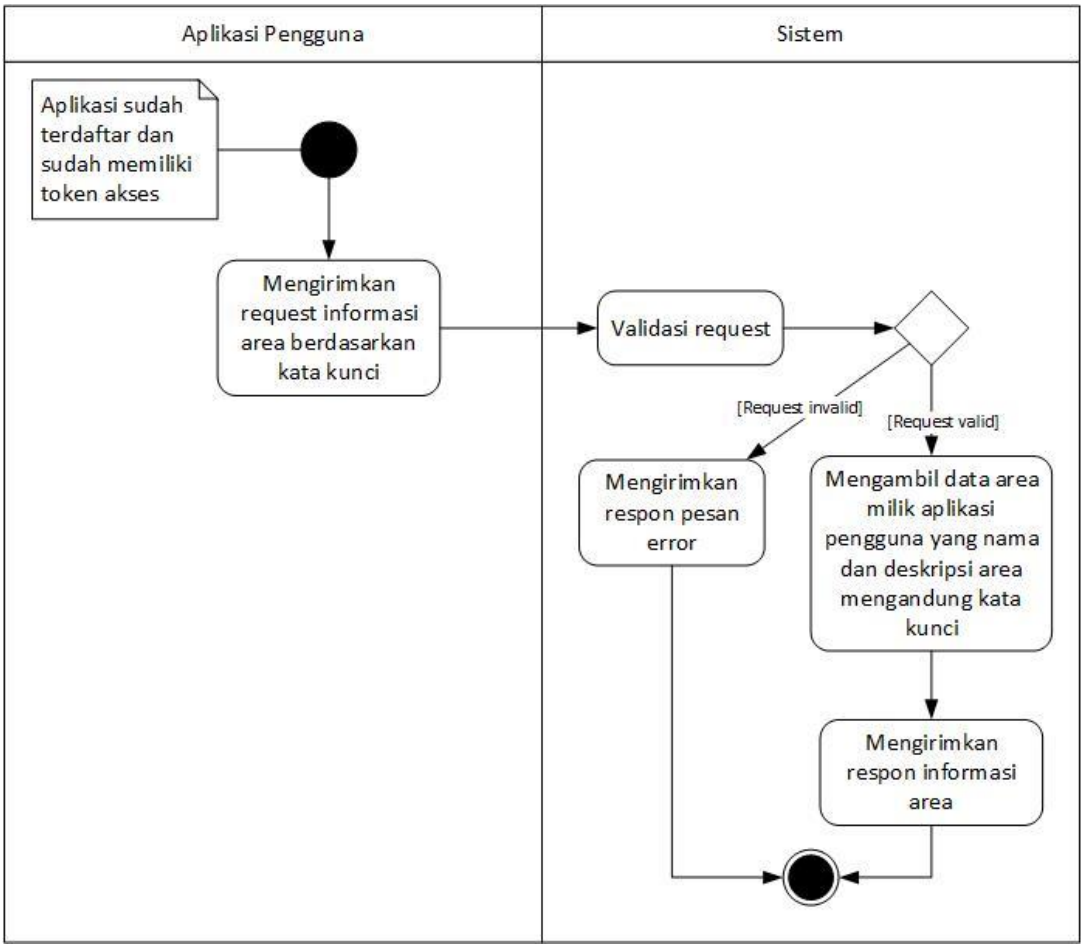
Gambar 3.21 *Activity Diagram* Mendapatkan Informasi Area Terkini



Gambar 3.22 *Activity Diagram* Mendapatkan Informasi Area Terdekat



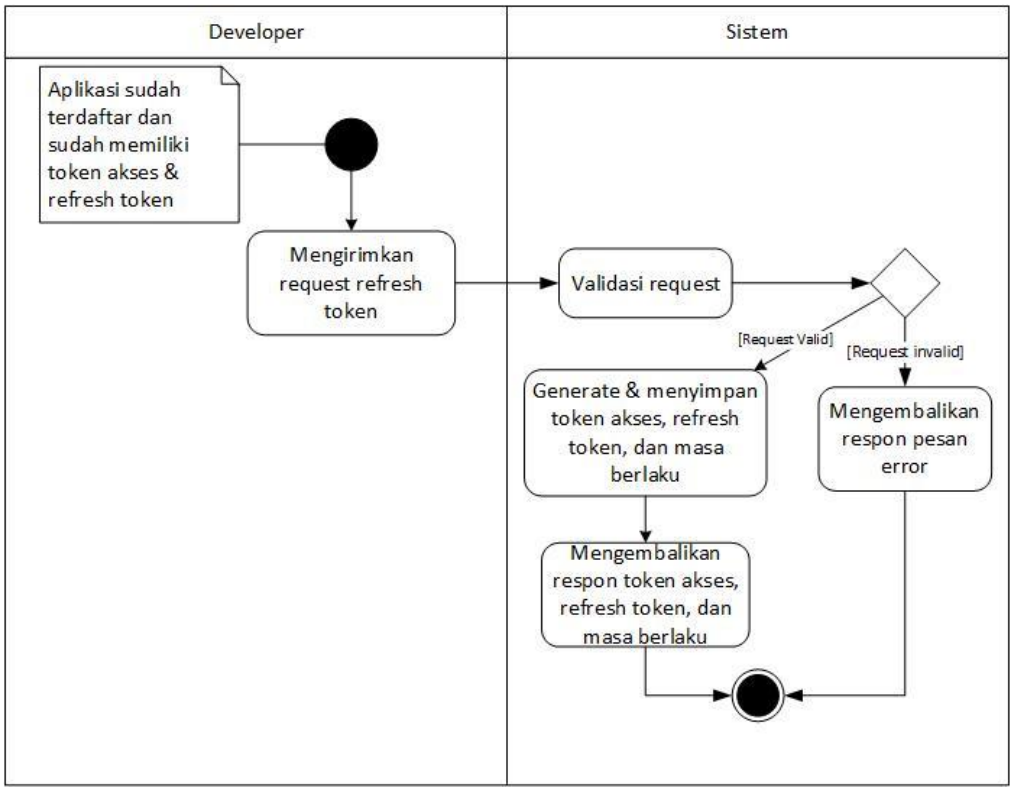
Gambar 3.23 *Activity Diagram* Mendapatkan Informasi Daftar Area



Gambar 3.24 *Activity Diagram* Mendapatkan Informasi Area Berdasarkan Kata Kunci

i. *Activity Diagram* UC-9 (Memperbarui Token Akses)

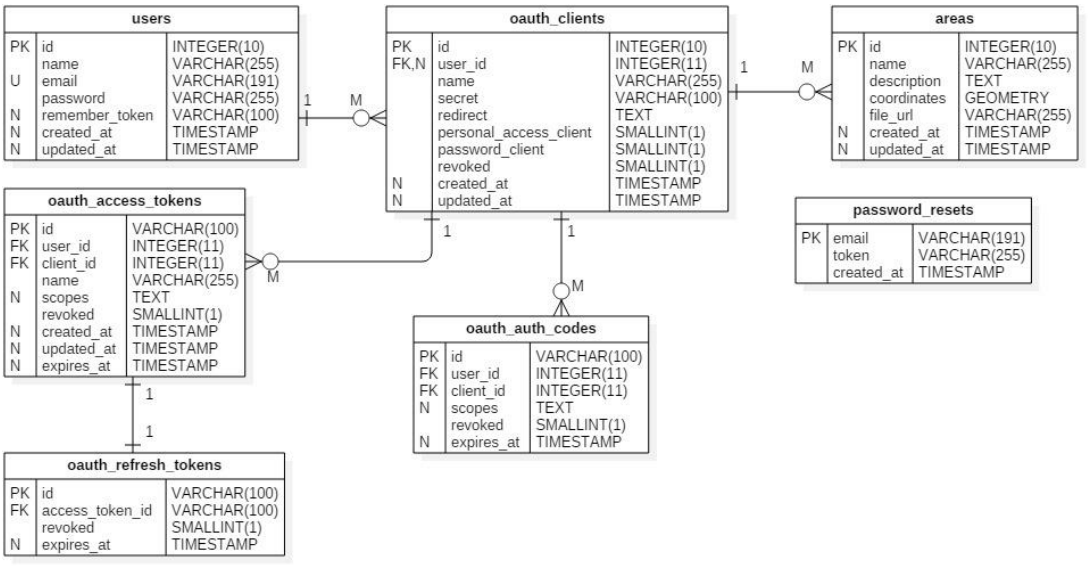
Pada diagram ini menjelaskan bagaimana alur kerja dalam memperbarui token akses. *Activity diagram* untuk memperbarui token akses ditunjukkan pada Gambar 3.25.



Gambar 3.25 *Activity Diagram* Memperbarui Token Akses

3.11 *Entity Relationship Diagram*

ERD (*Entity Relationship Diagram*) merupakan sebuah diagram yang menjelaskan relasi antar entitas dalam perancangan suatu basis data. Tiap entitas memiliki atribut yang merepresentasikan suatu tabel yang berisi berbagai kolom. Pada penelitian kali ini, penulis merancang sebuah ERD yang menggambarkan rancangan basis data pada sistem. ERD tersebut ditunjukkan pada Gambar 3.26.



Gambar 3.26 *Entity Relationship Diagram* MyHilvabs

Pada Gambar 3.23 terdapat tujuh tabel yang memiliki relasi dengan tabel lainnya. Setiap tabel memiliki struktur tabel masing – masing. Adapun penjelasan dari struktur setiap tabel adalah sebagai berikut:

a. Struktur Tabel users

Tabel 3.1 Tabel users

Nama Kolom	Tipe Data	Panjang	Keterangan
id	Unsigned Integer	10	Primary key
name	Varchar	255	
email	Varchar	191	Unique
password	Varchar	255	
remember_token	Varchar	100	Nullable
created_at	Timestamp		Nullable
updated_at	Timestamp		Nullable

Tabel 3.1 merupakan tabel users yang digunakan untuk menyimpan data *user*. *User* disini merupakan *developer*. Kolom id merupakan *primary key* pada tabel users. Kolom name menyimpan data nama *user*. Kolom email menyimpan data alamat *email user*. Kolom password menyimpan data kata sandi *user*. Kolom remember_token menyimpan data *token session* untuk fitur *remember me*. Kolom created_at menyimpan data tanggal saat pertama kali *user* dibuat. Kolom update_at menyimpan data tanggal saat terakhir kali

user melakukan perubahan data. Tabel ini memiliki relasi 1:M (*One to Many*) terhadap tabel oauth_clients.

b. Struktur Tabel oauth_clients

Tabel 3.2 Tabel oauth_clients

Nama Kolom	Tipe Data	Panjang	Keterangan
id	Unsigned Integer	10	Primary key
user_id	Unsigned Integer	11	Foreign Key, Nullable
name	Varchar	255	
secret	Varchar	100	
redirect	Text		
personal_access_client	Smallint	1	
password_client	Smallint	1	
revoked	Smallint	1	Nullable
created_at	Timestamp		Nullable
updated_at	Timestamp		Nullable

Tabel 3.2 merupakan tabel oauth_clients yang digunakan untuk menyimpan data *client* atau aplikasi pengguna. Tabel oauth_clients merupakan tabel yang telah disediakan oleh Laravel untuk pengembangan API. *Client* disini merupakan aplikasi pengguna. Kolom id merupakan *primary key* pada tabel oauth_clients. Kolom user_id merupakan *foreign key* yang mengarah pada tabel users. Kolom name menyimpan data nama *client*. Kolom redirect menyimpan data URL *client* yang didaftarkan. Kolom name dan redirect merupakan data yang wajib diisi pada saat membuat *client* atau aplikasi pengguna. Kolom secret menyimpan data unik yang disebut sebagai *password client*. Kolom secret secara langsung dibuat oleh sistem ketika proses pembuatan *client*. Kolom revoked menyimpan data status keaktifan *client*. Kolom created_at menyimpan data tanggal saat pertama kali *client* dibuat. Kolom update_at menyimpan data tanggal saat terakhir kali *client* melakukan perubahan data. Tabel ini memiliki berbagai relasi dengan tabel lainnya, diantarnya ialah relasi 1:M (*One to Many*) dengan tabel areas, relasi 1:M (*One to Many*) dengan tabel oauth_auth_codes, relasi 1:M (*One to Many*) dengan tabel oauth_access_tokens, dan relasi M:1 (*Many to One*) dengan tabel users.

c.

d. Struktur Tabel areas

Tabel 3.3 Tabel areas

Nama Kolom	Tipe Data	Panjang	Keterangan
id	Unsigned Integer	10	Primary key
name	Varchar	255	
description	Text		
coordinates	Geometry		
client_id	Unsigned Integer	10	Foreign Key
*file*_url	Varchar	255	
created_at	Timestamp		Nullable
updated_at	Timestamp		Nullable

Tabel 3.3 merupakan tabel areas yang digunakan untuk menyimpan data area. Kolom id merupakan *primary key* pada tabel areas. Kolom client_id merupakan *foreign key* yang mengarah pada tabel oauth_clients. Kolom client_id menyimpan data kepemilikan suatu area oleh *client*. Kolom name menyimpan data nama area. Kolom description menyimpan data deskripsi area. Kolom coordinates menyimpan data koordinat – koordinat area. Kolom *file*_url menyimpan data URL dari *file* KML yang disimpan. Kolom created_at menyimpan data tanggal saat pertama kali *client* dibuat. Kolom update_at menyimpan data tanggal saat terakhir kali *client* melakukan perubahan data. Tabel ini memiliki relasi M:1 (*Many to One*) dengan tabel oauth_clients.

e. Struktur Tabel oauth_auth_codes

Tabel 3.4 Tabel oauth_auth_codes

Nama Kolom	Tipe Data	Panjang	Keterangan
id	Varchar	100	Primary key
user_id	Unsigned Integer	11	
client_id	Unsigned Integer	11	
scopes	Text		
revoked	Smallint	1	
expires_at	Datetime		

Tabel 3.4 merupakan tabel oauth_auth_codes yang digunakan untuk menyimpan data wewenang otorisasi dari *resource owner* yang akan ditukar menjadi token akses. Tabel oauth_auth_codes merupakan tabel yang telah disediakan oleh Laravel untuk pengembangan API. Kolom id merupakan primary key pada tabel oauth_auth_codes.

Kolom id sebagai *authorization code* yang akan diberikan kepada *client* untuk ditukar menjadi *access token*. Kolom user_id menyimpan data *user* yang memberikan *authorization code* kepada suatu *client*. Kolom client_id menyimpan data *client* yang menerima *authorization code* dari *user* dan client_id sebagai *foreign key* yang mengarah pada tabel oauth_clients. Kolom scopes menyimpan data jangkauan akses terhadap informasi. Kolom revoked menyimpan status keaktifan *authorization code*. Kolom expires_at menyimpan data tanggal untuk durasi masa berlaku *authorization code*. Tabel ini memiliki relasi M:1 (*Many to One*) dengan tabel oauth_clients.

f. Struktur Tabel oauth_access_tokens

Tabel 3.5 Tabel oauth_access_tokens

Nama Kolom	Tipe Data	Panjang	Keterangan
id	Varchar	100	Primary key
user_id	Unsigned Integer	11	
client_id	Unsigned Integer	11	
name	Varchar	255	
scopes	Text		
revoked	Smallint	1	
created_at	Timestamp		Nullable
updated_at	Timestamp		Nullable
expires_at	Datetime		

Tabel 3.5 merupakan tabel oauth_access_tokens yang digunakan untuk menyimpan data token akses. Data token akses digunakan *client* dalam mengakses sumber data. Tabel oauth_access_tokens merupakan tabel yang telah disediakan oleh Laravel untuk pengembangan API. Kolom user_id menyimpan data *user* yang mengarah pada tabel users. Kolom client_id menyimpan data *client* yang mengarah pada tabel clients. Kolom nama menyimpan data nama *client*. Kolom scope menyimpan data jangkauan akses *client*. Kolom revoked menyimpan status keaktifan token akses. Kolom created_at menyimpan data tanggal saat pertama kali token akses dibuat. Kolom update_at menyimpan data tanggal saat terakhir kali token akses melakukan perubahan data. Kolom expires_at menyimpan data tanggal untuk durasi masa berlaku token akses. Tabel ini memiliki relasi M:1 (*Many to One*) dengan tabel oauth_clients dan relasi 1:1 (*One to One*) dengan tabel oauth_refresh_tokens.

g. Struktur Tabel oauth_refresh_tokens

Tabel 3.6 Tabel oauth_refresh_tokens

Nama Kolom	Tipe Data	Panjang	Keterangan
id	Varchar	100	Primary key
access_token_id	Varchar	100	
revoked	Smallint	1	
expires_at	Datetime		

Tabel 3.6 merupakan tabel oauth_refresh_tokens yang digunakan untuk menyimpan data *refresh token*. Data *refresh token* digunakan dalam proses memperbarui token akses. Tabel oauth_refresh_tokens merupakan tabel yang telah disediakan oleh Laravel untuk pengembangan API. Kolom id merupakan *primary key* pada tabel oauth_refresh_tokens. Kolom access_token_id menyimpan data token akses. Kolom revoked menyimpan status keaktifan *refresh token*. Kolom expires_at menyimpan data tanggal untuk durasi masa berlaku *refresh token*. Tabel ini memiliki relasi 1:1 (*One to One*) dengan tabel oauth_access_client.

h. Struktur Tabel password_resets

Tabel 3.7 Tabel password_resets

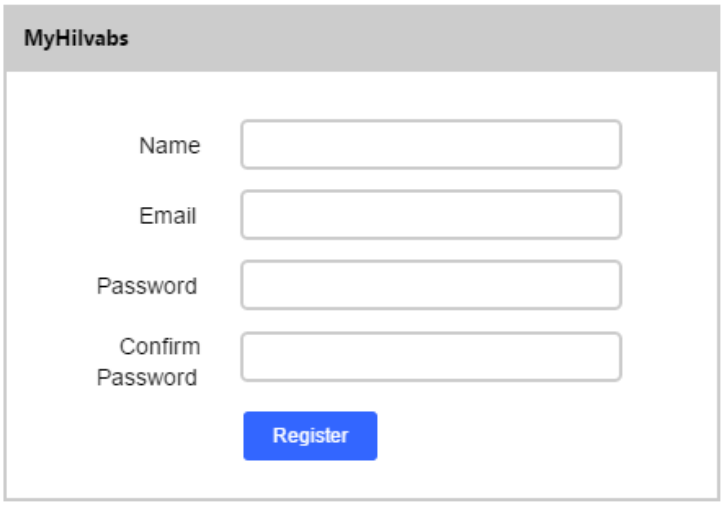
Nama Kolom	Tipe Data	Panjang	Keterangan
email	Varchar	191	Primary key
token	Varchar	255	
created_at	Timestamp		

Tabel 3.7 merupakan tabel yang digunakan untuk menyimpan token dalam proses *reset password*. Tabel password_resets merupakan tabel yang telah disediakan oleh Laravel untuk menangani penyimpanan token *reset password*. Kolom email merupakan *primary key* dalam tabel ini. Kolom email menyimpan data alamat *email user* yang melakukan *reset password*. Kolom token menyimpan data token untuk proses *reset password*. Kolom created_at menyimpan data tanggal pembuatan token. Data token tersebut akan dikirimkan ke alamat *email user* yang melakukan *reset password*.

3.12 Rancangan Antarmuka

Setelah melakukan analisis kebutuhan antarmuka, penulis membuat rancangan antarmuka yang akan menjadi dasar pembuatan aplikasi berbasis *web*. Adapun rancangan antarmuka yang dibuat adalah sebagai berikut:

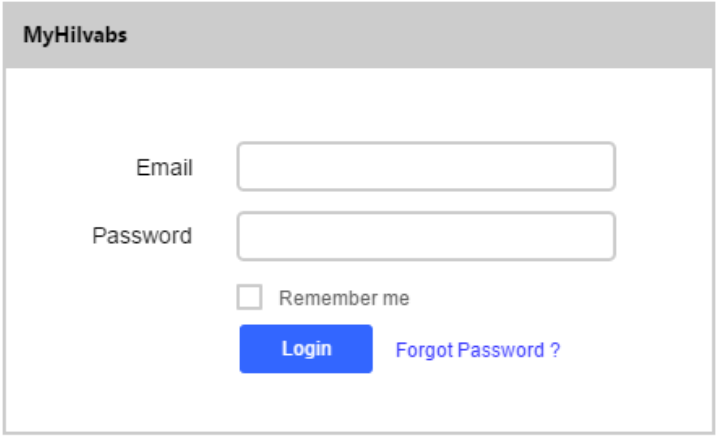
a. Halaman Register



Gambar 3.27 Rancangan Halaman Register

Gambar 3.27 merupakan rancangan halaman register. Proses *register* berfungsi untuk mendapatkan akun sistem. Terdapat *form* berupa *name*, *email*, *password* dan *confirm password* yang dibutuhkan untuk proses *register*.

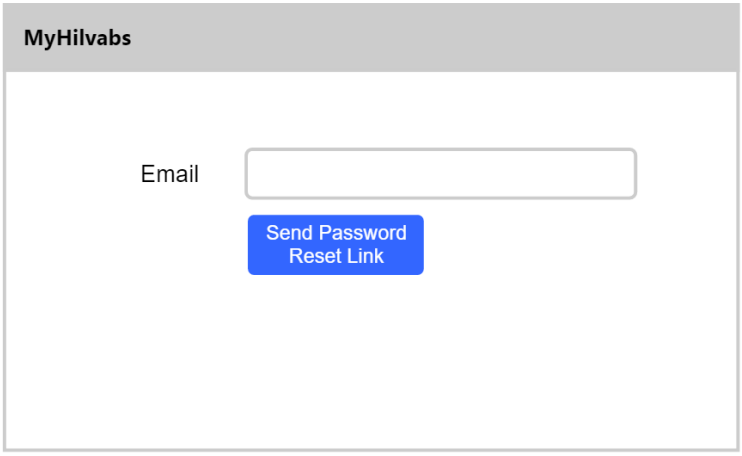
b. Halaman Login



Gambar 3.28 Rancangan Halaman Login

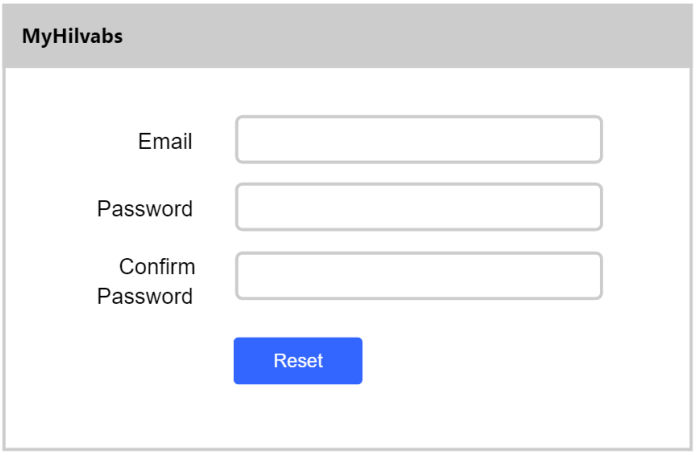
Gambar 3.28 merupakan rancangan halaman login. Sebelum mengakses sistem, developer terlebih dahulu harus melakukan *login*. Terdapat *form* berupa *email* dan *password* yang dibutuhkan untuk proses *login*. *Checkbox* remember me digunakan untuk menyimpan isian *form* login ke dalam *session* sehingga ketika *developer* ingin kembali masuk ke dalam sistem, maka secara otomatis *form* login akan terisi. Teks "forgot password ?" berfungsi untuk mengarahkan *developer* ke halaman reset password.

c. Halaman Reset Password



Gambar 3.29 Rancangan Halaman Reset Password – Pengiriman Tautan Reset Password

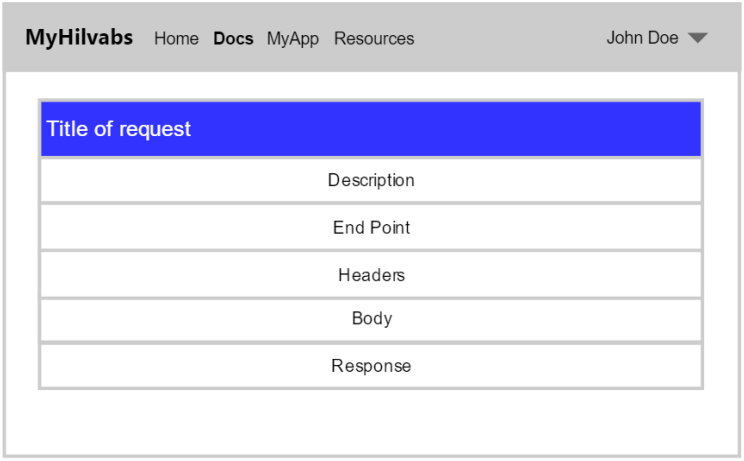
Gambar 3.29 merupakan rancangan halaman reset password untuk pengiriman tautan reset password. Terdapat *form* berupa *email* yang dibutuhkan dalam proses pengiriman token *reset password*.



Gambar 3.30 Rancangan Halaman Reset Password – Pengaturan Ulang Password Baru

Gambar 3.30 merupakan rancangan halaman reset password untuk pengaturan ulang *password* baru. Terdapat *form* berupa *email*, *password*, dan *confirm password* yang dibutuhkan dalam mengatur ulang *password* akun.

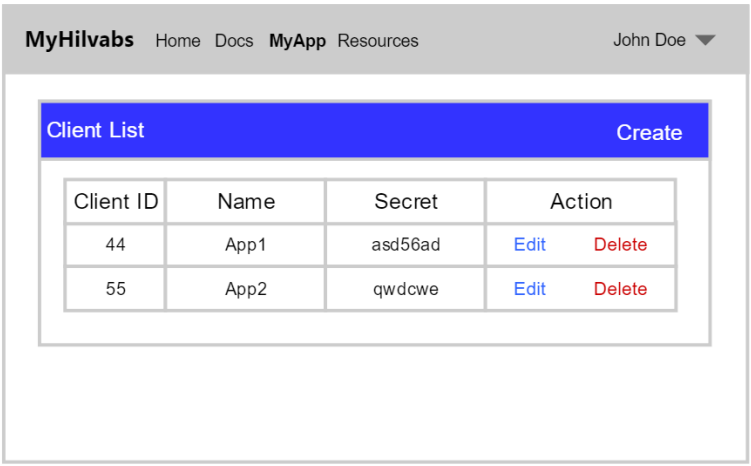
d. Halaman Dokumentasi



Gambar 3.31 Rancangan Halaman Dokumentasi

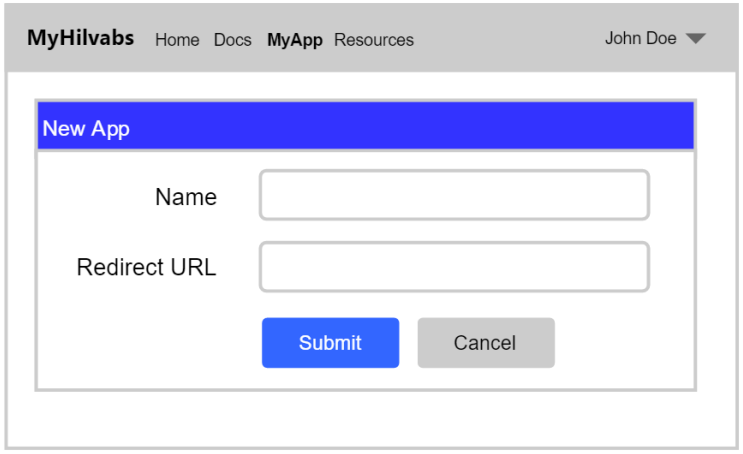
Gambar 3.31 merupakan rancangan halaman dokumentasi. Halaman dokumentasi berisi penjelasan penggunaan permintaan akses sumber data. Setiap permintaan dilengkapi dengan deskripsi, *endpoint*, *headers*, *body* (opsional), dan respons yang akan diterima *client*.

e. Halaman My App



Gambar 3.32 Rancangan Halaman My App

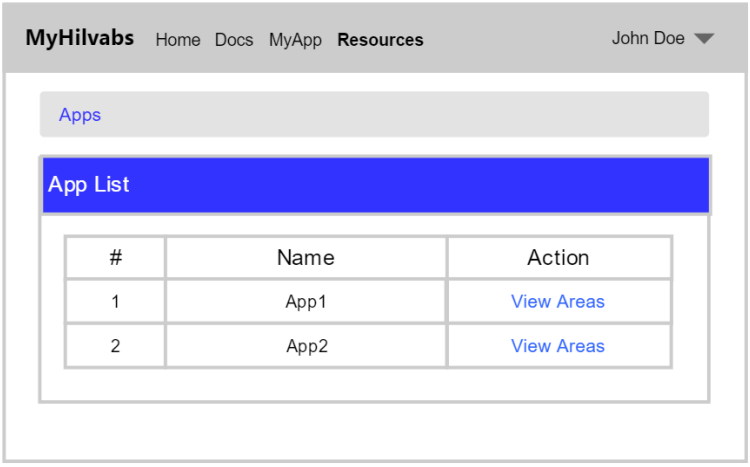
Gambar 3.32 merupakan rancangan halaman my app. Halaman ini menampilkan daftar aplikasi pengguna atau *client* (dalam istilah OAuth) milik *developer*. Terdapat sebuah tabel yang menampilkan data aplikasi pengguna atau *client*. Selain itu terdapat fitur untuk melakukan pengelolaan pada aplikasi pengguna atau *client* yaitu fitur mendaftar, mengubah, dan menghapus data aplikasi pengguna atau *client*. Rancangan antarmuka fitur mendaftar dan mengubah data aplikasi pengguna atau *client* dapat dilihat pada Gambar 3.33.



Gambar 3.33 Rancangan Halaman My App – Pendaftaran dan Pengubahan Aplikasi Pengguna

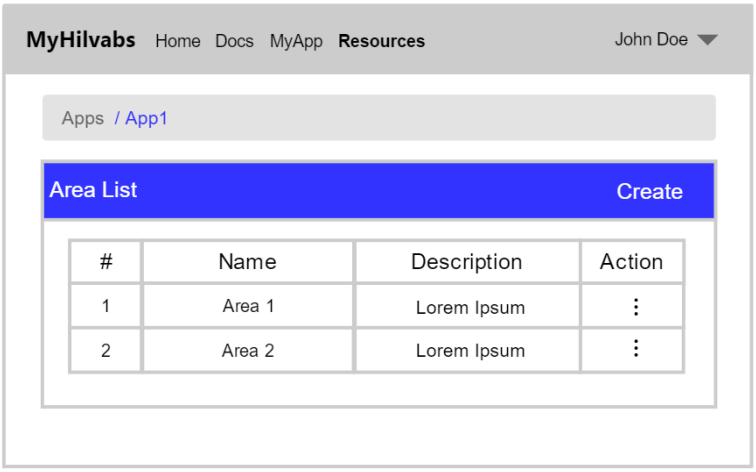
Gambar 3.33 merupakan rancangan halaman aplikasi untuk fitur pendaftaran dan pengubahan aplikasi pengguna. Pada rancangan ini, terdapat *form* berupa nama dan *redirect* URL yang dibutuhkan dalam proses mendaftar maupun mengubah data aplikasi pengguna.

f. Halaman Resources



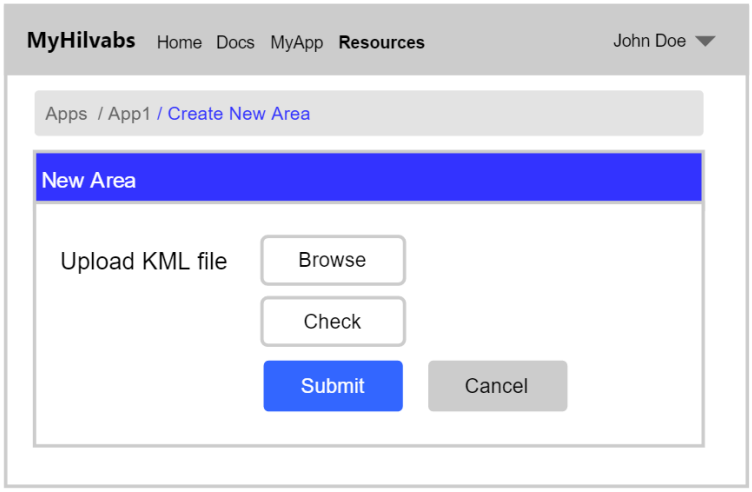
Gambar 3.34 Rancangan Halaman Resources – Daftar Aplikasi Pengguna

Gambar 3.34 merupakan rancangan halaman resources daftar aplikasi pengguna. Pada saat *developer* menuju halaman resources, maka sistem akan menampilkan terlebih dahulu daftar aplikasi. *Developer* dapat memilih sumber data milik aplikasi mana yang ingin ditelusuri berdasarkan daftar aplikasi yang telah ditampilkan. Setelah memilih aplikasi pengguna, maka sistem akan menampilkan daftar area yang ditunjukkan pada Gambar 3.35.



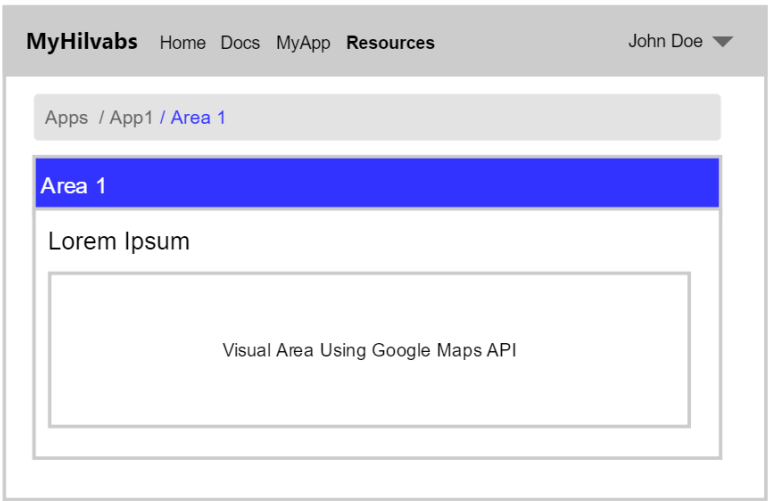
Gambar 3.35 Rancangan Halaman Resources – Daftar Area

Gambar 3.35 merupakan rancangan halaman resources daftar area. Halaman ini memuat daftar area yang dimiliki oleh aplikasi pengguna yang dipilih sebelumnya oleh *developer*. Selain itu terdapat fitur untuk melakukan pengelolaan data area yaitu fitur membuat, mengubah, visualisasi, dan menghapus data area. Rancangan antarmuka fitur membuat dan mengubah data area dapat dilihat pada Gambar 3.36 dan fitur visualisasi data area dapat dilihat pada Gambar 3.37.



Gambar 3.36 Rancangan Halaman Resources – Pembuatan dan Pengubahan Area

Gambar 3.36 merupakan rancangan halaman resources untuk fitur membuat dan mengubah data area. Terdapat *form* yang digunakan untuk mengunggah *file* KML yang dibutuhkan dalam proses mengelola area. Selain itu terdapat tombol check untuk melihat visualisasi *file* KML secara langsung.



Gambar 3.37 Rancangan Halaman Resources – Visualisasi Area

Gambar 3.37 merupakan rancangan halaman resources untuk fitur visualisasi area. Terdapat nama dan deskripsi dari area yang dipilih sebelumnya. Visualisasi area menggunakan Google Maps API dengan mengirimkan *file* KML. Sehingga, *developer* dapat melihat lokasi area dengan tampilan yang menarik.

3.13 Rancangan *Request* dan *Response* RESTful API

Pada tahap ini penulis membuat rancangan *request* dan *response* yang memuat aturan permintaan sumber data dan hasil kembalian dari layanan. Adapun rancangan *request* dan *response* RESTful API yang dibuat adalah sebagai berikut:

a. *Request* dan *Reponse* Kode Otorisasi

Request kode otorisasi digunakan untuk meminta penerbitan kode otorisasi yang nantinya digunakan dalam *request* token akses. Kode otorisasi merupakan kode hasil persetujuan *developer* kepada aplikasi LBS untuk menggunakan sumber data. Adapun rancangan *request* dan *response* kode otorisasi ditunjukkan pada Tabel 3.8.

Tabel 3.8 Rancangan *Request* dan *Response* Kode Otorisasi

Method	GET
URL	http://localhost:8000/oauth/authorize?client_id=CLIENT_ID&redirect_uri=REDIRECT_URL&response_type=code
Headers	-
Body	-
Keterangan	Pada url terdapat beberapa parameter yang wajib diisikan seperti, CLIENT_ID dan REDIRECT_URL. Kedua data tersebut diperoleh setelah developer mendaftarkan aplikasi LBS miliknya.

Response	{
	"code" : "ashdajskhd7ffhgfx5uykjuqwe899HYt786876…"
	}

b. *Request* dan *Response* Token Akses

Request token akses dilakukan untuk meminta penerbitan token akses dengan menukarkan kode otorisasi yang didapatkan dari hasil *request* kode otorisasi. Token akses tersebut digunakan sebagai penanda otentikasi aplikasi LBS dalam mengakses sumber data. Adapun rancangan *request* dan *response* token akses ditunjukkan pada Tabel 3.9.

Tabel 3.9 Rancangan *Request* dan *Response* Token Akses

Method	POST
URL	http://localhost:8000/oauth/token
Headers	-
Body	grant_type, client_id, client_secret, redirect_uri, code
Keterangan	Pada kolom body, terdapat beberapa isian yang dikirimkan saat melakukan request token akses yaitu grant_type berisi nilai statis yang berupa ‘authorization_code’, client_id berisi nilai id aplikasi LBS, client_secret berisi nilai karakter unik dari aplikasi LBS atau klien yang didapatkan setelah developer mendaftarkan aplikasi LBS atau klien. redirect_uri berisi nilai url dari aplikasi LBS yang telah didaftarkan. code berisi nilai kode otorisasi yang didapatkan dari hasil *request* kode otorisasi sebelumnya.

Response	<pre>{
	“token_type” : “Bearer”,
	“expires_in”: 3686500,
	“access_token” : “34s7sdasdsda…”,
	“refresh_token” : “asd67576ads…”
	}</pre>

c. *Request* dan *Response* Refresh Token

Request refresh token bertujuan untuk meminta pembaruan token akses dengan menyerahkan refresh token. Dengan mengirimkan *request* refresh token, maka aplikasi LBS tidak perlu melakukan *request* kode otorisasi. Hasil dari *request* refresh token sama dengan *request* token akses. Adapun rancangan *request* dan *response* refresh token ditunjukkan pada Tabel 3.10.

Tabel 3.10 Rancangan *Request* dan *Response* Refresh Token

Method	POST
URL	http://localhost:8000/oauth/token
Headers	-
Body	grant_type, refresh_token, client_id, client_secret
Keterangan	Pada kolom body, terdapat beberapa isian yang dikirimkan saat melakukan *request refresh token* yaitu grant_type berisi nilai statis yang berupa ‘refresh_token’, refresh_token berisi nilai token refresh yang didapatkan pada saat token akses diterbitkan. client_id berisi nilai id aplikasi LBS, client_secret berisi nilai karakter unik dari aplikasi LBS atau klien yang didapatkan setelah developer mendaftarkan aplikasi LBS atau klien.

Response	<pre>{
	“token_type” : “Bearer”,
	“expires_in”: 3686500,
	“access_token” : “34s7sdasdsda…”,
	“refresh_token” : “asd67576ads…”
	}</pre>

d. *Request* dan *Response* Informasi Daftar Area

Request informasi daftar area bertujuan untuk memperoleh informasi daftar area yang dimiliki oleh *developer*. Aplikasi LBS dapat melakukan *request* ini setelah mendapatkan token akses. Adapun rancangan *request* dan *response* informasi daftar area ditunjukkan pada Tabel 3.11.

Tabel 3.11 Rancangan *Request* dan *Response* Informasi Daftar Area

Method	GET
URL	http://localhost:8000/api/areas
Headers	Authorization
Body	-
Keterangan	Pada kolom headers, terdapat satu *headers* yang wajib disertakan yaitu Authorization. Authorization berisi tipe token dan token akses dalam satu string. Seperti contoh 'Bearer s6guyusdf87…'. *Headers* authorization tersebut sebagai bukti otentikasi aplikasi LBS dalam mengakses sumber data.

Response	{
	"status": "successed",
	"data": [
	{
	"id": 20,
	"name": "Sardonoharjo",
	"description": "Sardonoharjo merupakan …",
	"created_at": "2018-07-31 14:13:40",
	"created_at_for_humans": "12 hours ago",
	"updated_at": "2018-07-31 14:13:40",
	"updated_at_for_humans": "12 hours ago",
	"file_url": "http://localhost:8000/.../file.kml ",
	"type": "Polygon"
	},
	{ … },
	],
	"message": ""
	}

e. *Request* dan *Response* Informasi Area Terkini

Request informasi area terkini bertujuan untuk memperoleh informasi area terkini yang sesuai dengan koordinat yang diberikan oleh aplikasi LBS pada parameter long dan lat. Aplikasi LBS dapat melakukan *request* ini setelah mendapatkan token akses. Adapun rancangan *request* dan *response* informasi area terkini ditunjukkan pada Tabel 3.12.

Tabel 3.12 Rancangan *Request* dan *Response* Informasi Area Terkini

Method	GET
URL	http://localhost:8000/api/areas/current?long=LONG&lat=LAT
Headers	Authorization
Body	-
Keterangan	• Pada kolom url, terdapat parameter long berisi nilai *longitude* dan parameter lat berisi nilai *latitude*. Batas nilai long yang diizinkan antara -180 dan 180. Batas nilai lat yang diizinkan antara -90 dan 90. • Pada kolom headers, terdapat satu *headers* yang wajib disertakan yaitu Authorization. Authorization berisi tipe token dan token akses dalam satu string. Seperti contoh 'Bearer s6guyusdf87…'. *Headers* authorization tersebut sebagai bukti otentikasi aplikasi LBS dalam mengakses sumber data.

Response	{
	"status": "successed",
	"data": [
	{
	"id": 20,
	"name": "Sardonoharjo",
	"description": "Sardonoharjo merupakan …",
	"created_at": "2018-07-31 14:13:40",
	"created_at_for_humans": "12 hours ago",
	"updated_at": "2018-07-31 14:13:40",
	"updated_at_for_humans": "12 hours ago",
	"file_url": "http://localhost:8000/.../file.kml ",
	"type": "Polygon"
	},
	{ … },
	],
	"message": ""
	}

f. *Request* dan *Response* Informasi Area Terdekat

Request informasi area terdekat bertujuan untuk memperoleh informasi area terdekat yang sesuai dengan koordinat yang diberikan oleh aplikasi LBS pada parameter long dan lat. Aplikasi LBS dapat melakukan *request* ini setelah mendapatkan token akses. Adapun rancangan *request* dan *response* informasi area terdekat ditunjukkan pada Tabel 3.13.

Tabel 3.13 Rancangan *Request* dan *Response* Informasi Area Terdekat

Method	GET
URL	http://localhost:8000/api/areas/nearest?long=LONG&lat=LAT
Headers	Authorization
Body	-
Keterangan	• Pada kolom url, terdapat parameter long berisi nilai *longitude* dan parameter lat berisi nilai *latitude*. Batas nilai long yang diizinkan

	antara -180 dan 180. Batas nilai lat yang diizinkan antara -90 dan 90. • Pada kolom headers terdapat satu *headers* yang wajib disertakan yaitu Authorization. Authorization berisi tipe token dan token akses dalam satu string. Seperti contoh 'Bearer s6guyusdf87…'. *Headers* authorization tersebut sebagai bukti otentikasi aplikasi LBS dalam mengakses sumber data.

Response	{
	"status": "successed",
	"data": [
	{
	"id": 20,
	"name": "Sardonoharjo",
	"description": "Sardonoharjo merupakan …",
	"created_at": "2018-07-31 14:13:40",
	"created_at_for_humans": "12 hours ago",
	"updated_at": "2018-07-31 14:13:40",
	"updated_at_for_humans": "12 hours ago",
	"file_url": "http://localhost:8000/.../file.kml ",
	"type": "Polygon"
	},
	{ … },
	],
	"message": ""
	}

g. *Request* dan *Response* Informasi Area Berdasarkan Kata Kunci

Request informasi area berdasarkan kata kunci bertujuan untuk memperoleh informasi area yang sesuai dengan kata kunci yang diberikan oleh aplikasi LBS. Aplikasi LBS dapat melakukan *request* ini setelah mendapatkan token akses. Adapun rancangan *request* dan *response* informasi area berdasarkan kata kunci ditunjukkan pada Tabel 3.14.

Tabel 3.14 Rancangan *Request* dan *Response* Informasi Area Berdasarkan Kata Kunci

Method	GET
URL	http://localhost:8000/api/areas/search?q=**KEYWORD**
Headers	Authorization
Body	-
Keterangan	• Pada kolom url, terdapat parameter yang berisi kata kunci pencarian suatu area. Aplikasi LBS dapat mengirimkan kata kunci apapun. • Pada kolom headers, terdapat satu *headers* yang wajib disertakan yaitu Authorization. Authorization berisi tipe token dan token akses dalam satu string. Seperti contoh 'Bearer s6guyusdf87…'. *Headers* authorization tersebut sebagai bukti otentikasi aplikasi LBS dalam mengakses sumber data.
Response	{ "status": "successed", "data": [{

	<pre>"id": 20,
	"name": "Sardonoharjo",
	"description": "Sardonoharjo merupakan …",
	"created_at": "2018-07-31 14:13:40",
	"created_at_for_humans": "12 hours ago",
	"updated_at": "2018-07-31 14:13:40",
	"updated_at_for_humans": "12 hours ago",
	"file_url": "http://localhost:8000/.../file.kml ",
	"type": "Polygon"
	},
	{ … },
	],
	"message": ""
	}</pre>

3.14 Rancangan Fungsi Pembuatan Dan Pengolahan Data Spasial

Layanan lokasi ini dirancang untuk pembuatan dan pengolahan data spasial. Adapun beberapa fungsi mysql spatial yang digunakan dalam penelitian ini ditunjukkan pada Tabel 3.15.

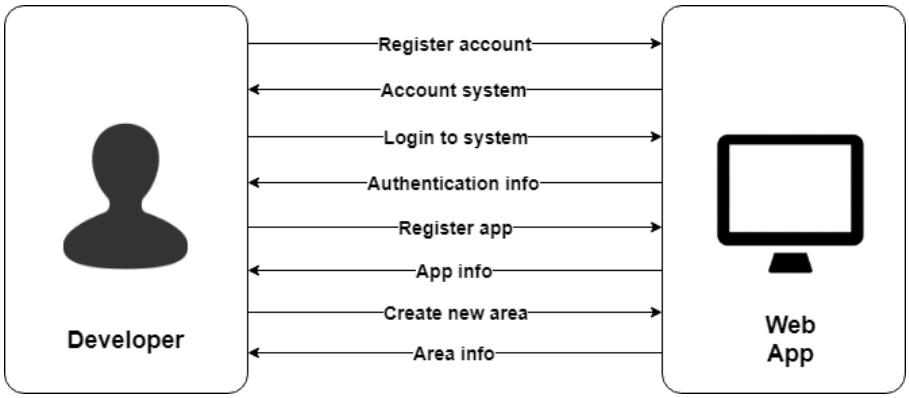
Tabel 3.15 Rancangan Fungsi Pembuatan Dan Pengolahan Data Spasial

No	Nama Fungsi	Contoh	Keterangan
1	ST_GeomFromText()	• `ST_GeomFromText('POINT(2 5)')` • `ST_GeomFromText('LINESTRING(0 0, 0 5, 5 5)')` • `ST_GeomFromText('POLYGON((0 0, 0 5, 5 5, 5 0, 0 0))')` • `ST_GeomFromText('MULTIPOINT(0 0, 2 2, 3 5)')`	Fungsi ST_GeomFromText() merupakan fungsi yang digunakan untuk membuat obyek geometri tipe data spasial apapun dengan menggunakan format WKT(*Well-Known Text*).
2	MBRContains()	`MBRContains(ST_GeomFromText('POLYGON((0 0, 0 5, 5 5, 5 0, 0 0))'), ST_GeomFromText('POINT(2 5)'))`	Fungsi MBRContains() digunakan untuk memeriksa apakah suatu koordinat lokasi yang diberikan berada dalam suatu obyek geometri.
3	ST_Distance_Sphere()	• `ST_Distance_Sphere(ST_GeomFromText('POINT(0 0)')), ST_GeomFromText('POINT(2 5)'))` • `ST_Distance_Sphere(ST_GeomFromText('POINT(0 0)')), ST_GeomFromText('MULTIPOINT(2 5, 10 10, 5 5,`	Fungsi ST_Distance_Sphere() digunakan untuk menghitung jarak minimum antara dua titik pada permukaan bumi.

		10 5, 5 10)'))	

3.15 Rancangan Skenario Penggunaan RESTful API

Pada tahapan ini penulis membuat rancangan skenario penggunaan RESTful API. Rancangan tersebut menggambarkan alur penggunaan mulai dari *developer* membuat akun sistem sampai dengan aplikasi pengguna dapat mengakses sumber data. Adapun rancangan skenario penggunaan RESTful API yang ditunjukkan pada Gambar 3.38 dan Gambar 3.39.

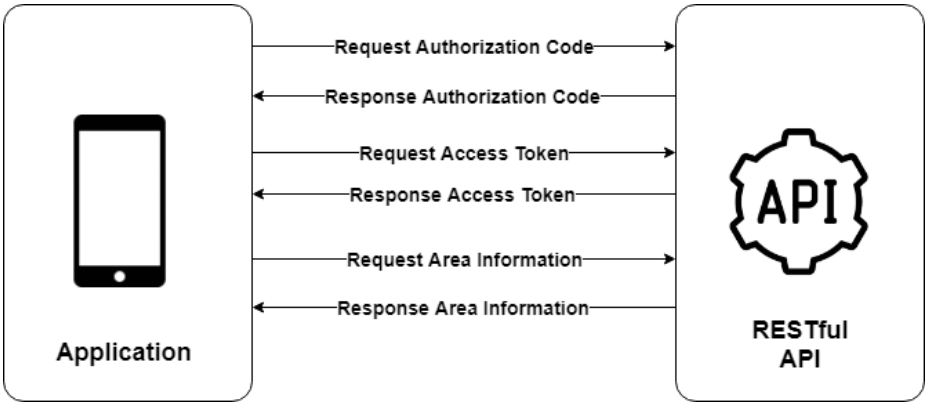


Gambar 3.38 Rancangan Skenario Pengguna RESTful API (1)

a. Langkah pertama yang dilakukan *developer* adalah mendaftarkan akun sistem. *Developer* dapat menuju halaman register dengan mengisi *form* register.
b. Setelah melakukan persetujuan pembuatan akun, maka sistem memberikan data akun kepada *developer*.
c. Selanjutnya, *developer* dapat melakukan *login* untuk memulai menggunakan layanan. *Developer* dapat menuju halaman login dengan mengisi *form* login.
d. Sistem akan melakukan validasi terhadap data login yang dikirimkan. Selanjutnya, sistem akan mengarahkan *developer* ke halaman home apabila data login bernilai valid.
e. *Developer* dapat mendaftarkan aplikasi pengguna miliknya dengan menuju halaman my app. Pada halaman tersebut memuat *form* pendaftaran aplikasi pengguna dengan mengisi data nama dan URL aplikasi.
f. Sistem akan melakukan validasi terhadap data aplikasi yang dikirimkan. Selanjutnya, sistem akan menampilkan aplikasi pengguna yang telah didaftarkan apabila data aplikasi yang dikirimkan sebelumnya bernilai valid. Informasi aplikasi yang diberikan oleh sistem berupa nama aplikasi, client_id, client_secret, dan url aplikasi.
g. Untuk membuat data area, *developer* harus memiliki aplikasi yang sudah terdaftar dalam sistem. *Developer* dapat menuju halaman resources untuk memilih aplikasi mana yang

ingin ditambahkan data area. Pada halaman tersebut memuat *form* untuk mengunggah peta area dengan menggunakan *file* KML.

h. Sistem melakukan validasi terhadap *file* KML yang dikirimkan. Sistem akan menampilkan respons berhasil dan informasi area apabila *file* KML bernilai valid.



Gambar 3.39 Rancangan Skenario Pengguna RESTful API (2)

i. Setelah *developer* mendaftarkan aplikasi pengguna miliknya, maka aplikasi pengguna harus meminta persetujuan pengaksesan sumber data dengan mengirimkan *request* kode otorisasi. *Request* tersebut bernilai valid apabila sudah memenuhi aturan *request* kode otorisasi.

j. Selanjutnya, sistem akan mengarahkan *developer* ke halaman login untuk menyetujui penerbitan kode otorisasi aplikasi pengguna. Sistem akan mengembalikan respons data kode otorisasi dalam format JSON.

k. Kemudian, aplikasi pengguna harus menukarkan kode otorisasi tersebut menjadi token akses agar dapat mengakses sumber data. Penukaran kode otorisasi dapat dilakukan dengan mengirimkan *request* token akses. *Request* tersebut bernilai valid apabila sudah memenuhi aturan *request* token akses.

l. Sistem akan mengembalikan respons berupa data token akses dalam format JSON apabila request bernilai valid.

m. Untuk mengakses informasi area, aplikasi pengguna dapat melakukan *request* area dengan menyertakan token akses. Token akses tersebut merupakan bukti otentikasi aplikasi pengguna.

n. Sistem akan mengembalikan respons informasi area dalam format JSON apabila *request* bernilai valid.

3.16 Rancangan Pengujian Sistem

Analisis pengujian sistem merupakan analisa metode pengujian terhadap sistem yang akan dikembangkan. Metode pengujian yang akan dipakai pada penelitian ini adalah metode *black box*. *Black box* merupakan metode pengujian sistem yang berfokus pada fungsionalitas sistem. Terdapat banyak teknik pada metode *black box*. Salah satunya ialah *equivalence partitioning*. *Equivalence partitioning* merupakan teknik pengujian perangkat lunak yang membagi domain masukan menjadi partisi nilai valid dan invalid. *Equivalence partitioning* membutuhkan data sampel yang bersifat representatif atau dapat mewakili nilai dari tiap partisi untuk melakukan uji coba terhadap masukan. Pengujian sistem dilakukan dengan memasukan data sesuai rancangan kasus uji. Penilaian terhadap pengujian sistem dilihat dari respons yang ditampilkan oleh sistem atas masukan yang diberikan dengan membandingkan rancangan respons dibuat apakah sesuai rancangan atau tidak. Hasil pengujian akan dibahas pada BAB selanjutnya. Rancangan kasus uji dengan menggunakan teknik *equivalence partitioning* dapat dilihat pada Tabel 3.16.

Tabel 3.16 Rancangan Kasus Uji

Kode	Spesifikasi	Contoh Kasus Uji	Jenis	Respons
TC-1	Masukan nama pengguna yang diizinkan dengan panjang 1 sampai 255 karakter	INPUT : “Muhammad Akbar”	Valid	-
TC-2	Masukan nama pengguna yang diizinkan dengan panjang 1 sampai 255 karakter	INPUT : “”	Invalid	Pesan error “The name may not be greater than 255 characters”
TC-3	Masukan *email* sesuai dengan format *email*	INPUT : “akbar@example.com”	Valid	-
TC-4	Masukan *email* sesuai dengan format *email*	INPUT : “akbar@example	Invalid	Pesan error “The email must be a valid email address.”
TC-5	Masukan *password* yang diizinkan dengan panjang 6 sampai 255 karakter	INPUT : “********”	Valid	-
TC-6	Masukan *password* yang diizinkan dengan panjang 6 sampai 255 karakter	INPUT : “****”	Invalid	Pesan error “The password must be at least 6 characters.”
TC-7	Masukan *password* konfirmasi sesuai dengan	Password : “rewq10ty” INPUT : “rewq10ty”	Valid	-

	isian *password* sebelumnya			
TC-8	Masukan *password* konfirmasi sesuai dengan isian *password* sebelumnya	Password : “rewq10ty” INPUT : “qwerty99”	Invalid	Pesan error “The password confirmation does not match.”
TC-9	Masukan *email* yang diizinkan hanya yang sudah terdaftar	Terdaftar : myadmin@example.com INPUT : myadmin@example.com	Valid	-
TC-10	Masukan *email* yang diizinkan hanya yang sudah terdaftar	Terdaftar : myadmin@example.com INPUT : Admin101@example.com	Invalid	Pesan error “These credentials do not match our records.”
TC-11	Masukan *password* yang diizinkan hanya yang sudah terdaftar	Terdaftar : ****** INPUT : ******	Valid	-
TC-12	Masukan *password* yang diizinkan hanya yang sudah terdaftar	Terdaftar : ****** INPUT : **********	Invalid	Pesan error “These credentials do not match our records.”
TC-13	Masukan nama aplikasi yang diizinkan dengan panjang 1 sampai 255 karakter	INPUT : “LBS Mapping App”	Valid	-
TC-14	Masukan nama aplikasi yang diizinkan dengan panjang 1 sampai 255 karakter	INPUT : “”	Invalid	Pesan error “The name may not be greater than 255 characters.”
TC-15	Masukan *redirect* URL yang diizinkan sesuai format URL	INPUT : “http://www.pemetaan.org”	Valid	-
TC-16	Masukan *redirect* URL yang diizinkan sesuai format URL	INPUT : “pemetaan”	Invalid	Pesan error “The redirect format is invalid.”
TC-17	Masukan *file* KML yang diizinkan dengan maksimal ukuran 100 KB	INPUT : *file* KML dengan ukuran 10 KB	Valid	-
TC-18	Masukan *file* KML yang diizinkan dengan maksimal ukuran 100 KB	INPUT : *file* KML dengan ukuran 120 KB	Invalid	Pesan *error* “Maximum file 100 KB ”
TC-19	Masukan *file* KML yang diizinkan dengan terdapat *tag Polygon*/*Linestring*/*Point*	INPUT : <Polygon>..</Polygon>	Valid	-

TC-20	Masukan *file* KML yang diizinkan dengan terdapat *tag Polygon*/*Linestring*/*Point*	INPUT : -	Invalid	Pesan *error* “Map doesn't contain any object”
TC-21	Masukan *file* KML yang diizinkan dengan terdapat hanya satu obyek (*Polygon*/*Linestring*/*Point*)	INPUT : <Polygon> … </Polygon>	Valid	-
TC-22	Masukan *file* KML yang diizinkan dengan terdapat hanya satu obyek (*Polygon*/*Linestring*/*Point*)	INPUT : <Polygon>..</Polygon> <Point>…</Point>	Invalid	Pesan *error* “Map contains more than one object”
TC-23	Masukan *file* KML yang diizinkan dengan *tag* name, description, atau coordinates tidak kosong	INPUT : <name>Area A</name> <description>Desc Area A</description> <coordinates>24, 34,0 …</coordinates>	Valid	-
TC-24	Masukan *file* KML yang diizinkan dengan *tag* name, description, atau coordinates tidak kosong	INPUT : <name>Area A</name> <description></description> <coordinates>24, 34,0 …</coordinates>	Invalid	Pesan *error* “Name / Description / Coordinates in your KML file is empty”
TC-25	Masukan *file* KML yang diizinkan dengan *tag* name memiliki isi panjang karakter dari 1 sampai 255	INPUT : <name>Area</name>	Valid	-
TC-26	Masukan *file* KML yang diizinkan dengan *tag* name memiliki isi panjang karakter dari 1 sampai 255	INPUT : Lebih dari 255 karakter	Invalid	Pesan *error* “The name may not be greater than 255 characters.”
TC-27	Masukan *file* KML yang diizinkan dengan *tag* Polygon memiliki minimal 4 koordinat	INPUT : <coordinates>1,1,0 2,2,0 2,1,0 1,1,0 </coordinates>	Valid	-
TC-28	Masukan *file* KML yang diizinkan dengan *tag* Polygon memiliki minimal 4 koordinat	INPUT : <coordinates>1,1,0 2,2,0</coordinates>	Invalid	Pesan *error* “Polygon must has minimal 4 points”
TC-29	Masukan *file* KML yang	INPUT :	Valid	-

	diizinkan dengan *tag* Polygon memiliki koordinat awal sama dengan koordinat akhir	Koordinat awal : 10,10,0 Koordinat akhir : 10,10,0		
TC-30	Masukan *file* KML yang diizinkan dengan *tag* Polygon memiliki koordinat awal sama dengan koordinat akhir	INPUT : Koordinat awal : 10,10,0 Koordinat akhir : 5,10,0	Invalid	Pesan *error* "First and last coordinate of Polygon doesn't match"
TC-31	Masukan *file* KML yang diizinkan dengan *tag* Linestring memiliki lebih dari satu titik koordinat	INPUT : <coordinates>11,20 15,20 30,10</coordinates>	Valid	-
TC-32	Masukan *file* KML yang diizinkan dengan *tag* Linestring memiliki lebih dari satu titik koordinat	INPUT : <coordinates>11,20</coordinates>	Invalid	Pesan *error* "Linestring must has minimal 2 points"
TC-33	Masukan *client* ID yang diizinkan hanya angka bilangan bulat positif	INPUT : 4	Valid	-
TC-34	Masukan *client* ID yang diizinkan hanya angka bilangan bulat positif	INPUT : -5	Invalid	Pesan *error* "Client authentication failed"
TC-35	Masukan *client secret* yang diizinkan dengan panjang 1 sampai 100 karakter	INPUT : "gts557s….."	Valid	-
TC-36	Masukan *client secret* yang diizinkan dengan panjang 1 sampai 100 karakter	INPUT : lebih dari 100 karakter	Invalid	Pesan *error* "Client authentication failed"
TC-37	Masukan area ID yang diizinkan hanya angka bilangan bulat positif	INPUT : 1	Valid	-
TC-38	Masukan area ID yang diizinkan hanya angka bilangan bulat positif	INPUT : -6	Invalid	Halaman not found
TC-39	Masukan *access token* sesuai format	INPUT : "Bearer s7sd…"	Valid	-
TC-40	Masukan *access token* sesuai format	INPUT : "s7sd…"	Invalid	Pesan *error* "Unauthenticated"
TC-41	Masukan *refresh token* sesuai format	INPUT : "a2s7sd…"	Valid	-
TC-42	Masukan *refresh token* sesuai format	INPUT : ""	Invalid	Pesan *error* "The refresh token is invalid"
TC-43	Masukan *authorization code* sesuai format	INPUT : "c70977a2fa9…"	Valid	-
TC-44	Masukan *authorization*	INPUT : ""	Invalid	Pesan *error*

	code sesuai format			"Invalid parameter value"
TC-45	Masukan *longitude* yang diizinkan angka dari rentang -180 sampai 180	INPUT : Longitude : 150.123	Valid	-
TC-46	Masukan *longitude* yang diizinkan angka dari rentang -180 sampai 180	INPUT : Longitude : 210	Invalid	Pesan *error* "The long must be between -180 and 180"
TC-47	Masukan *latitude* yang diizinkan angka dari rentang -90 sampai 90	INPUT : Latitude : 85	Valid	-
TC-48	Masukan *latitude* yang diizinkan angka dari rentang -90 sampai 90	INPUT : Latitude : -105	Invalid	Pesan *error* "The lat must be between -90 and 90"
TC-49	Masukan kata kunci yang diizinkan dengan minimal panjang 1 karakter	INPUT : "Gedung A"	Valid	-
TC-50	Masukan kata kunci yang diizinkan dengan minimal panjang 1 karakter	INPUT : -	Invalid	Pesan error "Parameter q is required"

Setelah membuat rancangan kasus uji, penulis mendefinisikan rancangan pengujian *use case* dengan menggunakan rancangan kasus uji yang telah dibuat. Rancangan pengujian *use case* dapat dilihat pada Tabel 3.17.

Tabel 3.17 Rancangan Pengujian *Use Case*

Use Case	**Kasus Uji Yang Dipakai**
UC-1 (*Register*)	TC-1, TC-2, TC-3, TC-4, TC-5, TC-6, TC-7, TC-8
UC-2 (*Login*)	TC-9, TC-10, TC-11, TC-12
UC-3 (*Reset Password*)	TC-9, TC-10, TC-5, TC-6, TC-7, TC-8
UC-4 (Mengelola Aplikasi Pengguna)	Mendaftar Aplikasi Pengguna: TC-13, TC-14, TC-15, TC-16 Mengubah Aplikasi Pengguna: TC-13, TC-14, TC-15, TC-16
UC-5 (Mengelola Area)	Menampilkan Daftar Area: TC-33, TC-34 Membuat Area: TC-17, TC-18, TC-19, TC-20, TC-21, TC-22, TC-23, TC-24, TC-25, TC-26, TC-27, TC-28, TC-29, TC-30, TC-31, TC-32, TC-45, TC-46, TC-47, TC-48 Mengubah Area: TC-17, TC-18, TC-19, TC-20, TC-21, TC-22, TC-23, TC-24, TC-25, TC-26, TC-27, TC-28, TC-29, TC-30, TC-31, TC-32, TC-45, TC-46, TC-47, TC-48, TC-37, TC-38 Visualisasi Area: TC-37, TC-38

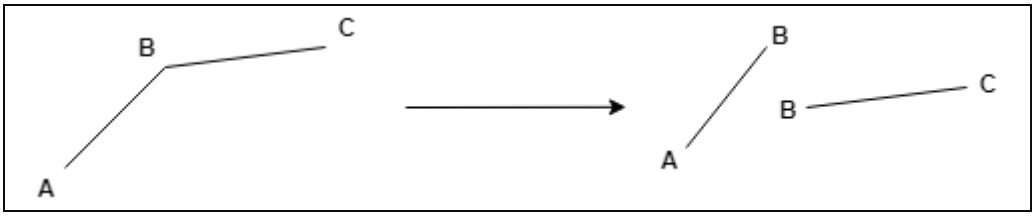
UC-6 (Melihat Dokumentasi)	-
UC-7 (Mendapatkan Token Akses)	TC-15, TC-16, TC-33, TC-34, TC-35, TC-36, TC-43, TC-44
UC-8 (Mendapatkan Informasi Area)	Informasi Area Terkini : TC-39, TC-40, TC-45, TC-46, TC-47, TC-48 Informasi Area Terdekat : TC-39, TC-40, TC-45, TC-46, TC-47, TC-48 Informasi Daftar Area : TC-39, TC-40 Informasi Area Berdasarkan Kata Kunci : TC-39, TC-40, TC-49, TC-50
UC-9 (Memperbarui Token Akses)	TC-33, TC-34, TC-35, TC-36, TC-41, TC-42

3.17 Rumus Perhitungan Yang Digunakan

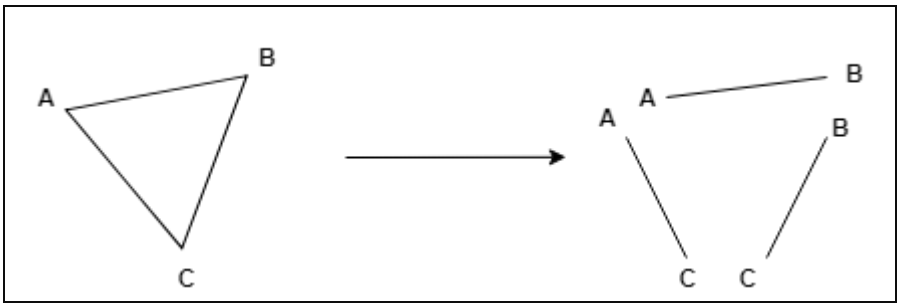
Pada penelitian ini, penulis mengembangkan satu fungsi yang digunakan untuk mencari area terdekat. Pencarian area terdekat menggunakan fungsi analisis dari mysql spatial yaitu ST_Distance_Sphere(Geometri, Geometri). ST_Distance_Sphere(Geometri, Geometri) merupakan fungsi yang digunakan untuk menghitung jarak minimum antara dua titik pada bola. Parameter yang diterima oleh fungsi tersebut hanya obyek geometri *point* dan *multipoint*. Dengan keterbatasan itu, maka penulis menggunakan rumus *point on line* untuk mencari titik proyeksi dalam garis (Sunshine, 2013). Tujuannya agar obyek geometri *linestring* dan *polygon* dapat digunakan dalam fungsi ST_Distance_Sphere. Adapun langkah – langkah perhitungan area terdekat adalah sebagai berikut:

a. Memecah obyek geometri *linestring* atau *polygon*

   Tahapan ini merupakan proses untuk memecah obyek geometri *linestring* atau *polygon* menjadi garis – garis tersendiri. Tujuannya ialah agar dapat mencari titik proyeksi dari setiap garis yang telah dipecah. Pemecahan garis dari obyek geometri *linestring* ditunjukkan pada Gambar 3.40 dan obyek geometri *polygon* ditunjukkan pada Gambar 3.41.



Gambar 3.40 Obyek Geometri *Linestring* Yang Dipecah



Gambar 3.41 Obyek Geometri *Polygon* Yang Dipecah

b. Mendapatkan titik proyeksi pada garis

```
public function getPointProjection(Point $p)
{
  $v1 = $this->point_1;
  $v2 = $this->point_2;

  // get dot product of $e1, $e2
  $e1 = new Point($v2->x() - $v1->x(), $v2->y() - $v1->y());
  $e2 = new Point($p->x() - $v1->x(), $p->y() - $v1->y());
  $valDp = ($e1->x() * $e2->x()) + ($e1->y() * $e2->y());

  // get squared length of $e1
  $len2 = $e1->x() * $e1->x() + $e1->y() * $e1->y();

  $pp = new Point((double)($v1->x() + ($valDp * $e1->x()) / $len2),
                  (double)($v1->y() + ($valDp * $e1->y()) / $len2));
  return $pp;
}
```

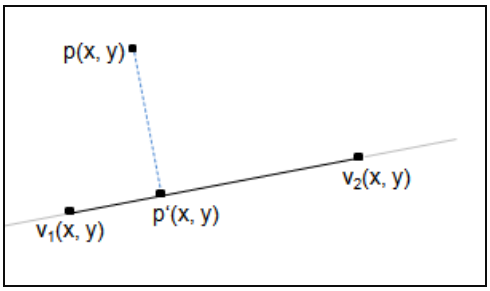
Gambar 3.42 Kode Sumber *Method* getPointProjection

Gambar 3.42 memuat suatu *method* getPointProjection yang digunakan untuk mencari titik proyeksi dari garis. Terdapat parameter p untuk memasukkan titik koordinat. Variabel v1 dan v2 merupakan variabel yang menyimpan nilai titik awal dan titik akhir dari garis. Variabel e1 merupakan variabel yang menyimpan nilai vektor dari v1 ke v2. Variabel e2 merupakan variabel yang menyimpan nilai vektor dari v1 ke p. Selanjutnya, kita membuat variabel valDp untuk menyimpan nilai *dot product* dari e1 dan e2. Selanjutnya, kita membuat variabel len2 untuk menyimpan nilai hasil kuadrat dari e1. Kemudian, kita membuat variabel pp yang menyimpan titik proyeksi dengan perhitungan menggunakan variabel yang telah dibuat selanjutnya. Hasil yang diberikan oleh *method* getPointProjection merupakan titik proyeksi.

c. Memeriksa posisi titik proyeksi

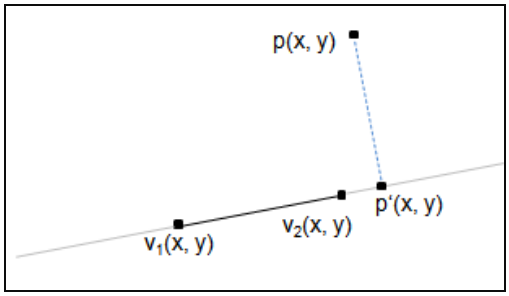
Hasil yang diberikan oleh *method* getPointProjection hanya berupa titik proyeksi. Namun, titik proyeksi tersebut belum tentu berada di dalam garis yang ditunjukkan pada

Gambar 3.43 atau di luar garis yang ditunjukkan pada Gambar 3.44. Sehingga kita memerlukan pemeriksaan titik proyeksi apakah di dalam atau di luar garis.



Gambar 3.43 Titik Proyeksi Di Dalam Garis

Sumber : Sunshine (2013)



Gambar 3.44 Titik Proyeksi Di Luar Garis

Sumber : Sunshine (2013)

```
public function contains(Point $point)
{
  return ($this->point_1->x() <= $point->x() &&
          $point->x() <= $this->point_2->x()) &&
         ($this->point_1->y() <= $point->y() &&
          $point->y() <= $this->point_2->y());
}
```

Gambar 3.45 Kode Sumber *Method* contains

Gambar 3.45 memuat *method* contains yang digunakan untuk memeriksa apakah titik proyeksi berada dalam garis. Parameter point merupakan nilai dari titik proyeksi yang akan diperiksa. Variabel point_1 merupakan variabel yang menyimpan nilai titik awal dari garis. Variabel point_2 merupakan variabel yang menyimpan nilai titik akhir dari garis. Hasil keluaran dari *method* contains adalah nilai *boolean*. Jika hasil kembalian bernilai *true* atau berada di dalam garis, maka titik proyeksi akan disimpan ke dalam *array* untuk dimasukkan dalam pembuatan obyek *multipoint*. Jika hasil kembalian

bernilai *false* atau berada di luar garis, maka kita dapat memilih titik awal atau titik akhir yang dekat dengan titik koordinat. Untuk mencari jarak dari dua titik dapat dilihat pada Gambar 3.46.

```
public function getDistanceTwoPoints(Point $point_1, Point $point_2)
{
 $x1 = $point_1->x();
 $y1 = $point_1->y();
 $x2 = $point_2->x();
 $y2 = $point_2->y();
 return sqrt(pow(($x2-$x1),2) + pow(($y2-$y1),2));
}
```

Gambar 3.46 Kode Sumber *Method* getDistanceTwoPoints

Gambar 3.46 memuat *method* getDistanceTwoPoints yang digunakan untuk mencari jarak dari dua titik. Terdapat parameter point_1 dan point_2 yang merupakan dua titik yang akan dicari nilai jaraknya. Pada kasus ini, point_1 diisi dengan titik koordinat yang diberikan dan point_2 diisi dengan titik awal atau akhir dari garis. Jika nilai jarak titik awal garis dengan titik koordinat lebih kecil daripada nilai jarak titik akhir garis dengan titik koordinat, maka titik awal garis akan disimpan ke dalam *array* untuk dimasukkan dalam pembuatan obyek *multipoint*. Hal tersebut berlaku apabila nilai jarak titik akhir garis dengan titik koordinat lebih kecil. Setelah mendapatkan titik – titik koordinat dari setiap garis yang telah dipecah untuk pembuatan obyek *multipoint*, maka kita dapat melakukan pemanggilan terhadap fungsi ST_Distance_Sphere(Geometri, Geometri).

BAB IV
IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Implementasi

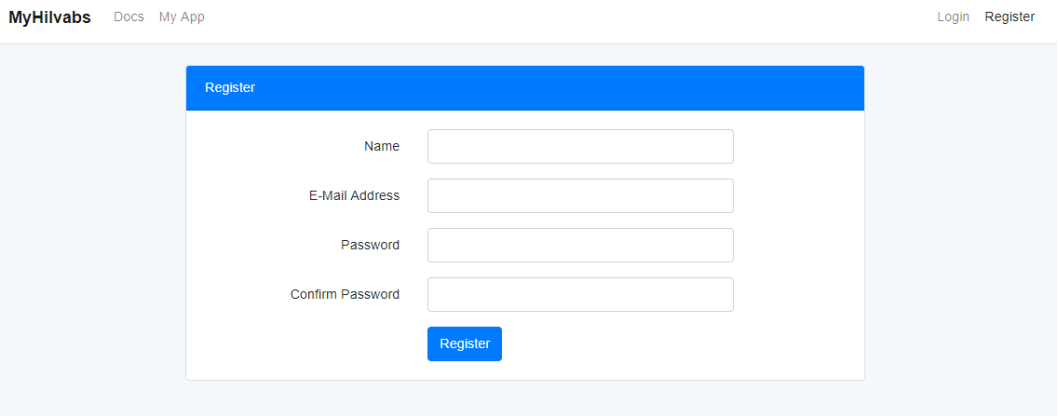
Proses implementasi sistem menerapkan hasil analisis dan rancangan yang telah dibahas pada BAB sebelumnya. Penerapan tersebut meliputi arsitektur sistem, *use case diagram*, *activity diagram*, *entity relationship diagram*, rancangan antarmuka khusus untuk aplikasi berbasis *web*, rancangan *request* dan *response* RESTful API, rancangan fungsi pembuatan dan pengolahan data spasial, rancangan skenario penggunaan RESTful API, dan rumus perhitungan yang digunakan. Proses implementasi sistem dipisah berdasarkan produk yang dihasilkan yaitu RESTful API dan aplikasi berbasis *web*.

4.1.1 Aplikasi berbasis *web*

Implementasi aplikasi berbasis *web* ditunjukkan dengan hasil *screenshot* pada rancangan antarmuka yang telah dirancang sebelumnya. Adapun detail implementasi dari aplikasi berbasis *web* adalah sebagai berikut:

a. Halaman Register

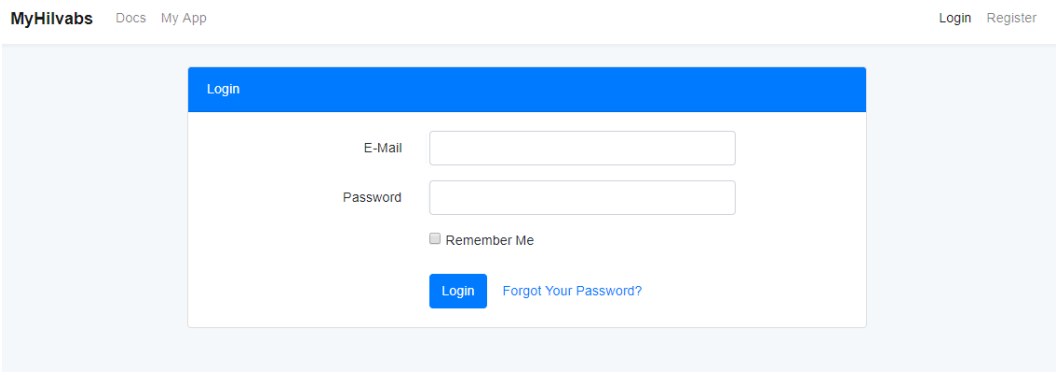
Halaman register memuat fitur pembuatan akun sistem bagi *developer*. Terdapat *form* register yang digunakan untuk mengisi data akun seperti nama, *email*, dan *password*. Setelah melakukan *register*, maka *developer* mendapatkan akun sistem. Halaman *register* ditunjukkan pada Gambar 4.1.



Gambar 4.1 Implementasi Halaman Register

b. Halaman Login

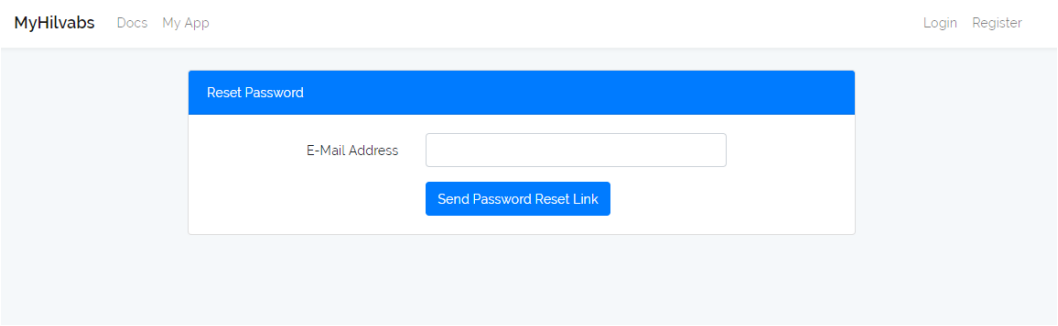
Halaman login memuat fitur untuk masuk ke dalam sistem. Terdapat *form* login yang digunakan untuk mengisi data akun seperti *email* dan *password*. Setelah melakukan proses *login*, maka sistem akan mengarahkan ke halaman home. Halaman login ditunjukkan pada Gambar 4.2.



Gambar 4.2 Implementasi Halaman *Login*

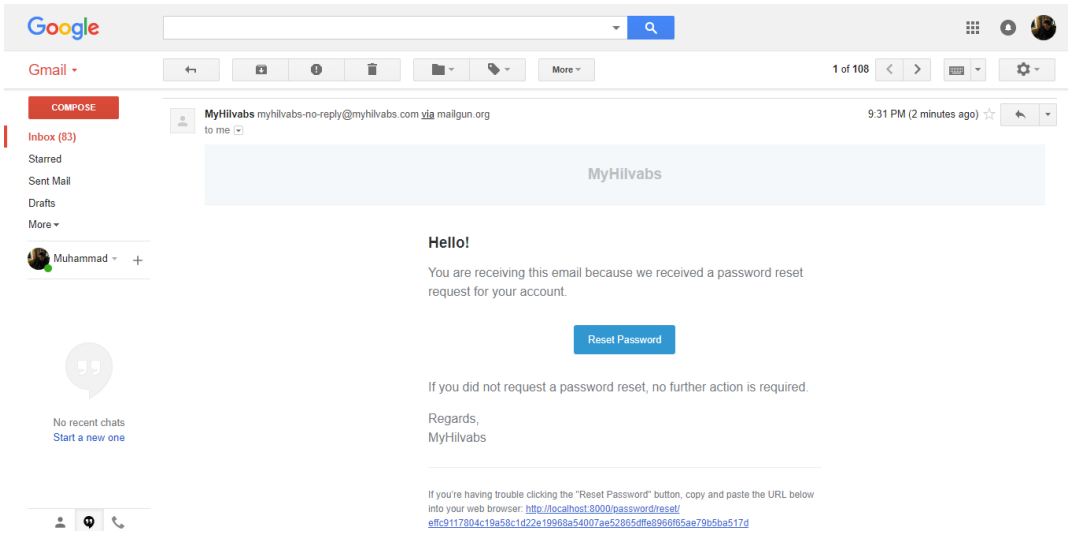
c. Halaman Reset Password

Halaman reset password memuat fitur untuk mengatur ulang *password* akun sistem. Ketika *developer* melakukan klik pada teks “forgot your password ?”, maka sistem akan mengarahkan ke halaman pengiriman tautan reset password. Terdapat *form* yang memuat isian *email* yang menjadi alamat tujuan pengiriman token *reset password*. Halaman pengiriman tautan reset password ditunjukkan pada Gambar 4.3.



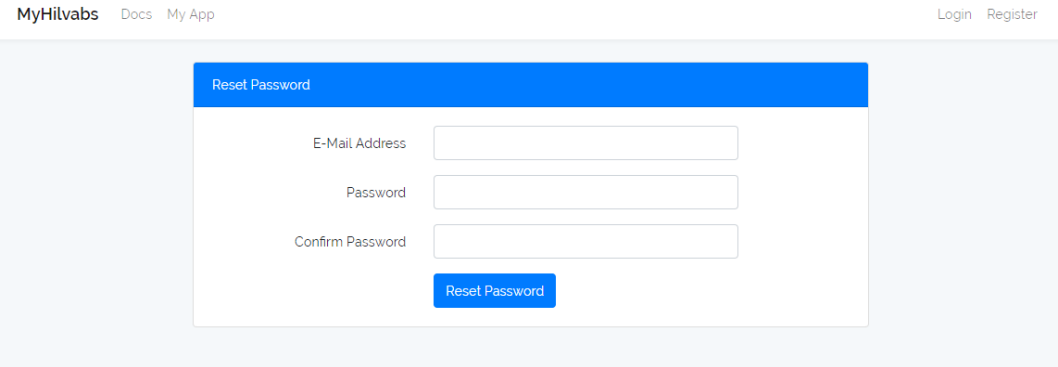
Gambar 4.3 Implementasi Halaman Reset Password – Pengiriman Tautan *Reset Password*

Sistem akan mengirimkan token *reset password* ke alamat *email* yang telah diisi. Isi pesan token *reset password* ditunjukkan pada Gambar 4.4.



Gambar 4.4 Token *Reset Password*

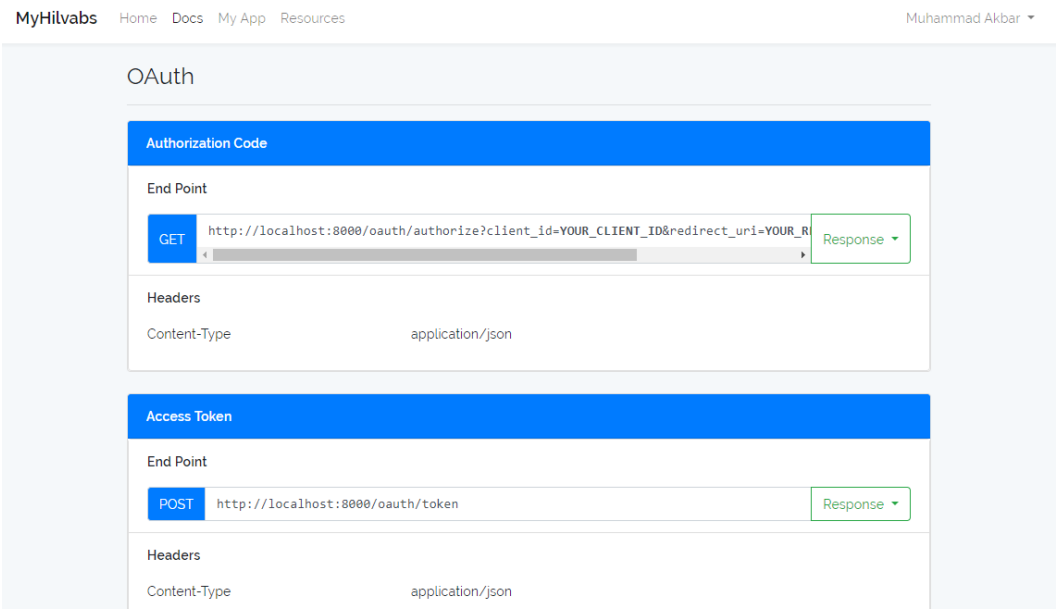
Setelah menerima token *reset password*, *developer* menuju tautan *reset password* yang diberikan oleh sistem pada Gambar 4.4. Tautan tersebut mengarahkan *developer* menuju halaman pengaturan ulang password baru. Halaman pengaturan ulang password baru ditunjukkan pada Gambar 4.5.



Gambar 4.5 Implementasi Halaman Reset Password – Pengaturan Ulang Password Baru

d. Halaman Dokumentasi

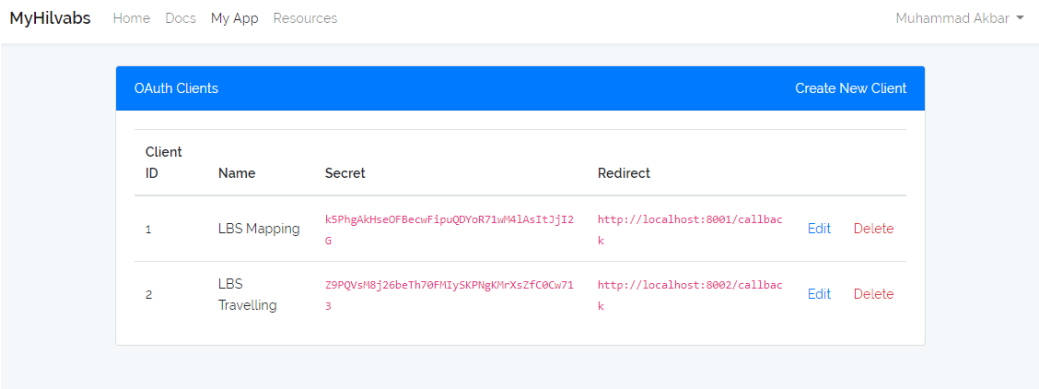
Halaman dokumentasi memuat penjelasan mengenai penggunaan layanan lokasi untuk aplikasi pengguna. Halaman dokumentasi dapat diakses tanpa perlu melakukan *login*. Halaman dokumentasi ditunjukkan pada Gambar 4.6.



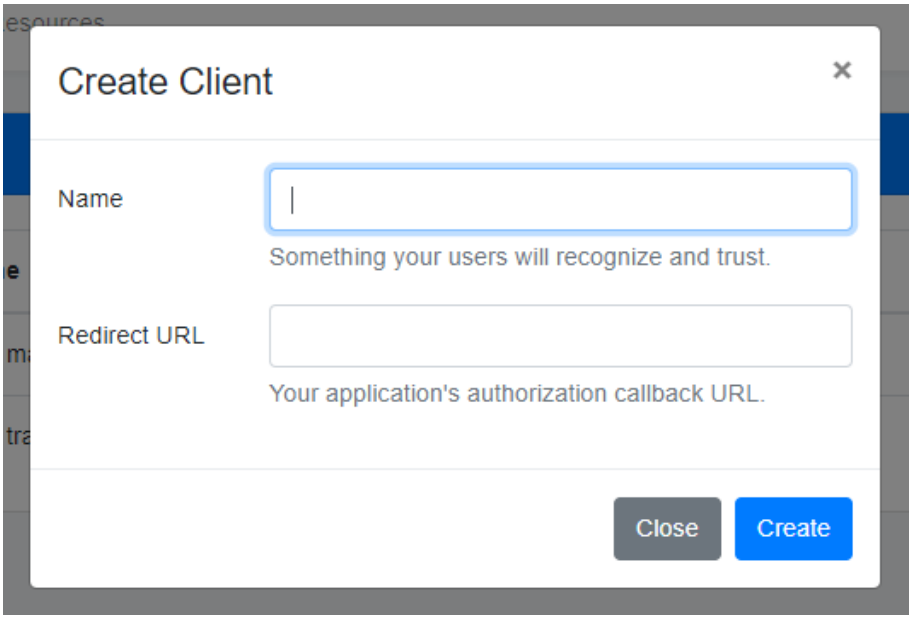
Gambar 4.6 Implementasi Halaman Dokumentasi

e. Halaman My App

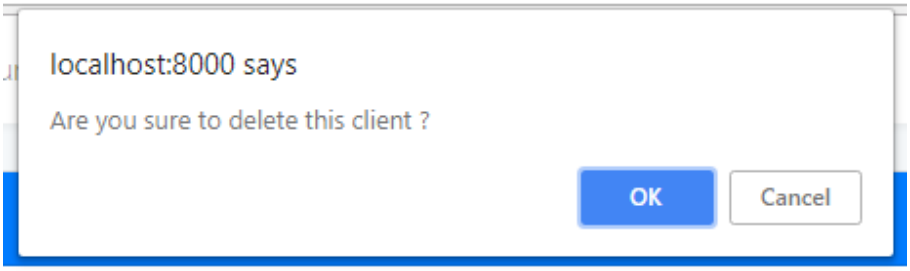
Halaman my app memuat fitur untuk mengelola aplikasi pengguna mulai dari menampilkan daftar, mendaftarkan, mengubah, dan menghapus aplikasi pengguna. Untuk masuk ke dalam halaman my app, maka *developer* harus melakukan *login* terlebih dahulu. Terdapat daftar aplikasi pengguna yang dimuat ke dalam sebuah tabel yang dilengkapi dengan menu edit dan delete. Selain itu, terdapat menu "create new client" yang digunakan untuk menampilkan *form* pendaftaran aplikasi pengguna. *Form* pengubahan aplikasi pengguna sama seperti *form* pendaftaran aplikasi pengguna. Saat *developer* memilih menu delete pada salah satu aplikasi pengguna maka sistem akan menampilkan pesan konfirmasi penghapusan. Halaman my app ditunjukkan pada Gambar 4.7. *Form* pendaftaran aplikasi pengguna ditunjukkan pada Gambar 4.8. Pesan konfirmasi ditunjukkan pada Gambar 4.9.



Gambar 4.7 Implementasi Halaman My App – Daftar Aplikasi Pengguna



Gambar 4.8 Implementasi Halaman My App – Form Pendaftaraan Aplikasi Pengguna

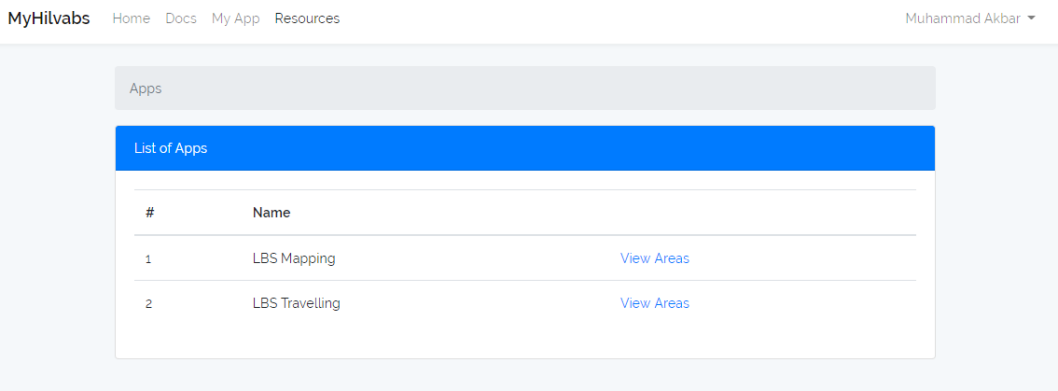


Gambar 4.9 Implementasi Halaman My App – Pesan Konfirmasi Hapus Aplikasi Pengguna

f. Halaman Resources

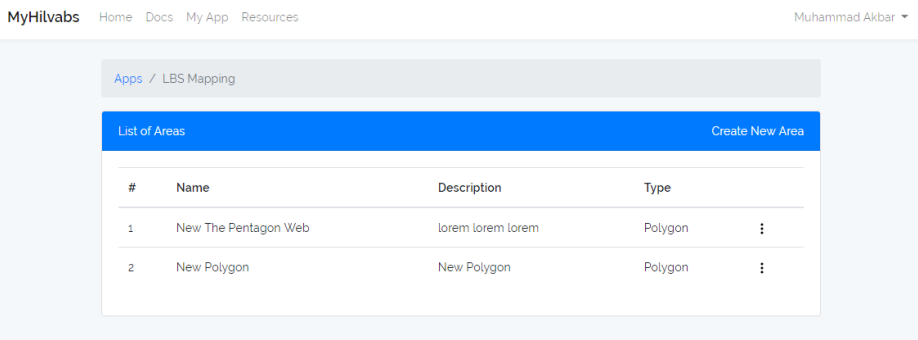
Halaman *resources* memuat fitur untuk mengelola area mulai dari menampilkan daftar, membuat, mengubah, menghapus, dan menampilkan visualisasi area. Halaman *resources*

akan tersedia, apabila *developer* telah melakukan *login*. Sebelum melakukan pengelolaan area, *developer* memilih terlebih dahulu aplikasi pengguna yang data areanya akan dikelola. Terdapat daftar aplikasi pengguna yang dimuat ke dalam sebuah tabel. Daftar aplikasi pengguna ditunjukkan pada Gambar 4.10.

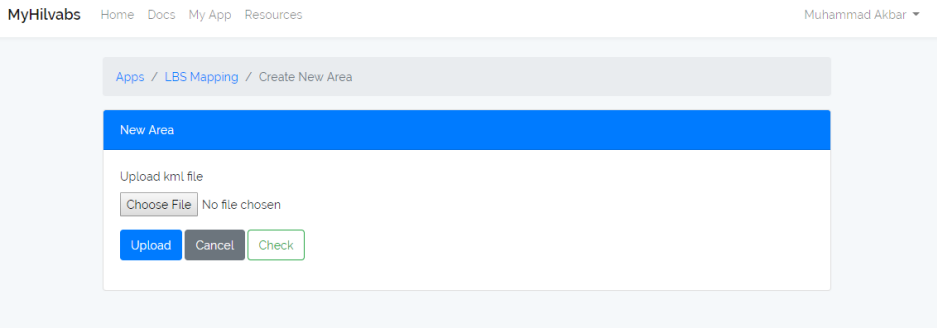


Gambar 4.10 Implementasi Halaman *Resources* – Daftar Aplikasi Pengguna

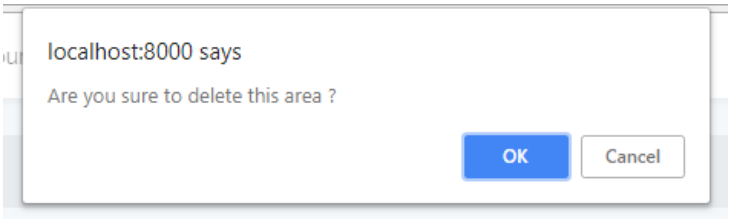
Setelah memilih salah satu aplikasi pengguna, maka sistem akan menampilkan daftar area yang dimiliki oleh aplikasi tersebut. Terdapat daftar area yang dimuat ke dalam sebuah tabel yang dilengkapi dengan menu edit, delete, dan view. Selain itu, terdapat menu "create new area" untuk mengarahkan pada halaman pembuatan area. Pada halaman tersebut, terdapat *form* pengunggahan *file* KML. Halaman pengubahan area sama seperti halaman pembuatan area. Saat *developer* memilih menu delete pada salah satu area maka sistem akan menampilkan pesan konfirmasi penghapusan. Saat *developer* memilih menu view maka sistem akan mengarahkan pada halaman visualisasi area yang memuat detail area dan visual area. Visualisasi area menggunakan layanan Google Maps API yang berisi obyek area pada *file* KML. Daftar area ditunjukkan pada Gambar 4.11. Halaman pembuatan area ditunjukkan pada Gambar 4.12. Pesan konfirmasi penghapusan area ditunjukkan pada Gambar 4.13. Halaman visualisasi area ditunjukkan pada Gambar 4.14.



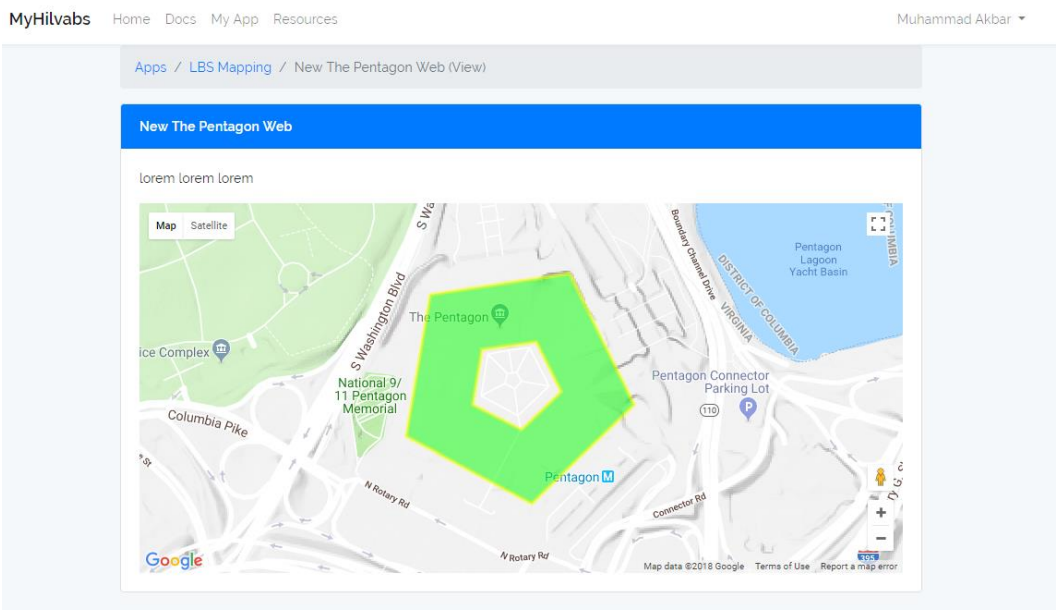
Gambar 4.11 Implementasi Halaman Resources – Daftar Area



Gambar 4.12 Implementasi Halaman Resources – Pembuatan Area



Gambar 4.13 Implementasi Halaman Resources – Pesan Konfirmasi Hapus Area



Gambar 4.14 Implementasi Halaman Resources – Visualisasi Area

4.1.2 RESTful API

Implementasi RESTful API ditunjukkan dengan hasil *screenshot* pada setiap permintaan dan respons yang diberikan oleh sistem. Adapun detail implementasi dari RESTful API adalah sebagai berikut:

a. Token Akses

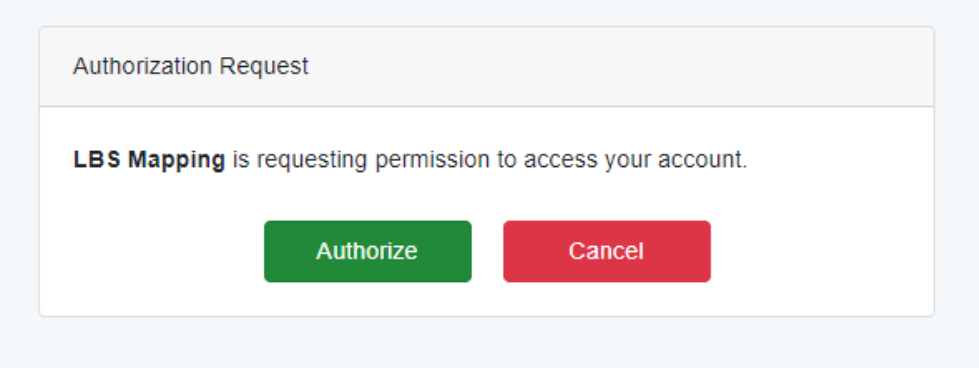
Token akses dapat diperoleh melalui permintaan yang dikirimkan oleh aplikasi pengguna kepada layanan lokasi. Hal pertama yang dilakukan oleh aplikasi pengguna untuk mendapatkan token akses ialah meminta kode otorisasi dari *developer*.

```
<?php
use Illuminate\Http\Request;
Route::get('/access_token', function () {
  $query = http_build_query([
   'client_id' => 1,
   'redirect_uri' => 'http://localhost:8001/callback',
   'response_type' => 'code',
   'scope' => ''
   ]);
 return redirect('http://localhost:8000/oauth/authorize?'.$query);
});
```

Gambar 4.15 Kode Sumber HTTP *Request* Kode Otorisasi

Gambar 4.15 merupakan contoh kode sumber HTTP *request* kode otorisasi yang dilakukan oleh aplikasi pengguna. Penjelasan kode sumber *admin* ialah saat aplikasi pengguna menangkap permintaan akses ke /access_token maka aplikasi pengguna

membuat variabel untuk mengisi parameter dari permintaan kode otorisasi. Client_id merupakan penanda data aplikasi pengguna yang melakukan permintaan. Redirect_uri merupakan alamat yang dituju oleh sistem dalam mengirimkan respons kode otorisasi ke aplikasi pengguna. Selanjutnya, aplikasi pengguna mengirimkan HTTP *request* kode otorisasi ke sistem beserta isi parameter yang telah dibuat. Setelah mengirimkan permintaan, maka sistem akan mengarahkan ke halaman persetujuan otoritas pengaksesan sumber data oleh aplikasi pengguna. Namun, terlebih dahulu *developer* harus *login* ke dalam sistem. Persetujuan otoritas pengaksesan sumber data ditunjukkan pada Gambar 4.16.



Gambar 4.16 Persetujuan Otoritas Pengaksesan Sumber Data

Setelah *developer* menyetujui otoritas pengaksesan sumber data oleh aplikasi pengguna, maka sistem melakukan *redirect* ke alamat tujuan penerimaan yang diisi sebelumnya pada parameter redirect_uri beserta nilai kode otorisasi.

b.

```
<?php
use Illuminate\Http\Request;
Route::get('/callback', function (Request $request) {
 $response = (new GuzzleHttp\Client)->post('http://localhost:8000/oauth/token', [
   'form_params' => [
    'grant_type' => 'authorization_code',
    'client_id' => 1,
    'client_secret' => 'jwobK7KxGgN3Mxi0VjX7bPsB8aqtOHCrIKyxjYSd',
    'redirect_uri' => 'http://localhost:8001/callback',
    'code' => $request->code,
   ]
  ]);

  session()->put('token', json_decode((string) $response->getBody(), true));
  return $response->getBody();
});
```

Gambar 4.17 Kode Sumber HTTP *Request* Token Akses

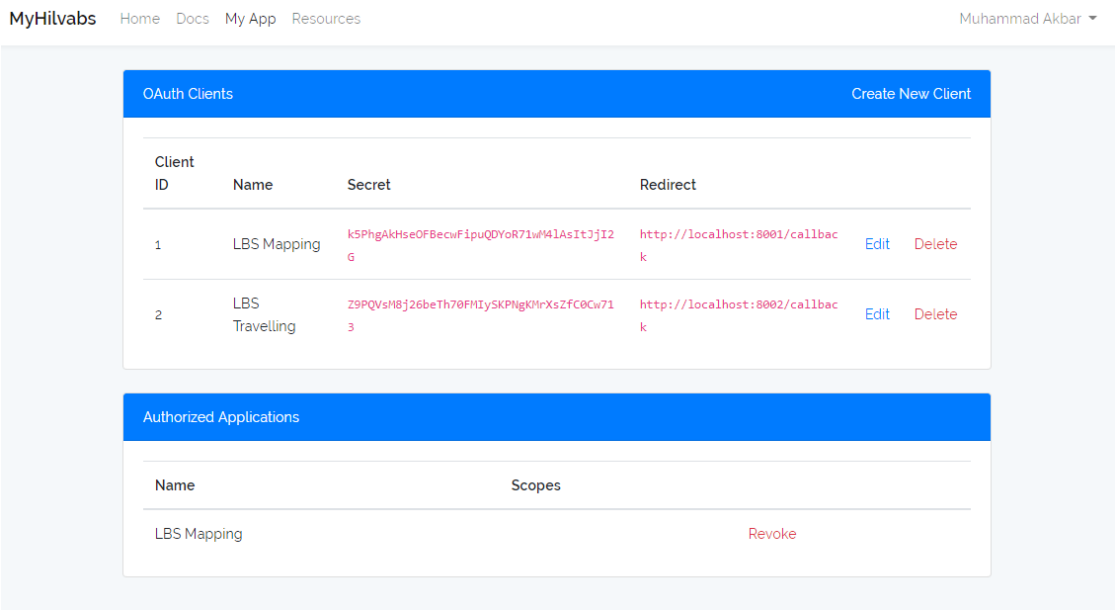
Gambar 4.17 merupakan kode sumber HTTP *request* token akses. Pada gambar tersebut, aplikasi pengguna menangkap respons yang diberikan oleh sistem setelah sistem tersebut melakukan *redirect*. Selanjutnya aplikasi pengguna melakukan HTTP *request* token akses. Terdapat beberapa *body* atau isi permintaan yang harus diisi sebelum melakukan HTTP *request* seperti grant_type, client_id, client_secret, redirect_uri, code. Code berisi kode otorisasi yang didapatkan pada tahapan HTTP *request* kode otorisasi. Setelah mengirimkan permintaan ke sistem, maka sistem mengirimkan respons berupa token_type, expires_in, access_token, dan refresh_token. Respons dari sistem ditunjukkan pada Gambar 4.18.


```
{
  "token_type": "Bearer",
  "expires_in": 31535999,
  "access_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0aSI6ImYxYzAzYjlmM2I2YzliNmMxYjEwZmI1NGFkZjU0MDliMzg5MjAzMGMyZjI0Mjg4ZTMwMzZlNWU5NmE5ZDFkZWU2Mzc4Y
TYxMDgwZjM2NTg3In0.eyJhdWQiOiIxIiwianRpIjoiZjFjMDNiOWYzYjZjOWI2YzFiMTBmYjU0YWRmNTQwOWIzODkyMDMwYzJmMjQyODhlMzAzNmU1ZTk2YTlkMWRlZTYzNzhhNjE
wODBmMzY1ODciLCJpYXQiOjE1MjY2MjgxMzgsIm5iZiI6MTUyNjYyODEzOCwiZXhwIjoxNTU4MTY0MTM4LCJzdWIiOiIxIiwic2NvcGVzIjpbXX0.f6D-JF2-
VBW5hMqPQcw9oOs113g9m93ho18La6qBkcBX8JYceZ3u0jYAhki8POmLUkhwS2tkZIizd7XVfxiLD9Wo6ePR8n-
3tKIzp4CB2jmo3Cis0w0e2VyYLWcYDLetqioZQW0qOaCUiSbFeDX9G2s4E7L8t_lfRle_vZmNB4mn9AAUbNfk_9Of3uHlAcGwytF9Z4XkLPurVXALH8FNdsYCF9kCbWnV5SW0vdoZF
S_pV2_wALzXoA1SCwtBf0jvZagHGQSFrmYvqy6iW6u-yE8tpabulhRVVSrpYl7LmYOWOn4W9StEU0ZqBG8Nifc-wJwDZvZaPIbtecNSJgk7CERyYNp6E8uL4eaTYmhksMOH-
OCTnifvhzojHnTE1ggTyAjAvJJLOZ2Onne6btqwd8aL_qxrEMnpcqmTUz0ewWpwowhSZIcpJGvaE2kpUA27HOOAOYZ3dWmfZcFW_osclYdUGKteeGbidT8JcqhrejBHcMIIlNDlRXm
wtJiEKZKRpM6I0cboqprZL6ehQJmhs-GyVUJxgiE7sZL6Xy_fZlS30RleXKYW2eD2tqtgkJ7P0ZxlDx1GO5w_EnGRrWymEAdmbzQdBAhrS8xP9uZrwptIHNfNeOaC9ArZG_Rcx-
dDaKxLjkfQZhD-Rg6jpsmWPtsCTLW9Mgd6Lq-RtAWfZQg",
  "refresh_token":
"def5020046ac33d325eda3192b9846afe21c93d230cb8078ce847aff8750b7cbf821a66e664ddb191f5150c817582723822ad47d23e6d68592e40053ea6d2a86b48dea3cb
2f1ee064bf7874e3f8cb04ab0df4d987425b09d8a316f2f7f6edefbbac04f5504e69e45e6ae116584422912f127242e5c683f1e66e5249c01f909dc1ae884e13955cf92244
2bdd13bb16e25da74f2a4c7551113db57976d21735d52b1a738e2783c75dc3f86a5ba7df8f6b6755f71c761adb14a8c949203e7ddc760554a8ee2bf67252ab0248bce11917
a90cf90d5d647c95f16cd300dc42f9c8840fcaaafe41407baf23304787330e73f3bc006ebf00e5ab2f709dd0873561a7d3f91378860d47e7c9efb9e2c2bb6691f4231998e5
a8d3568c49c050f9c7516d385f2ffc40d1ad13e0524a6a4e0ed6a529d5b5ff7b784d26ea2e41b9c143962fcef58459b79ef8e4a42b9fc5ee4439616e1a6d6db706502ba0d3
6cc6414e967d50180"
}
```


Gambar 4.18 Respons Permintaan Token Akses

Setelah mendapatkan token akses, maka *developer* dapat melakukan pencabutan otoritas aplikasi pengguna dalam mengakses sumber data. Untuk melakukan pencabutan otoritas

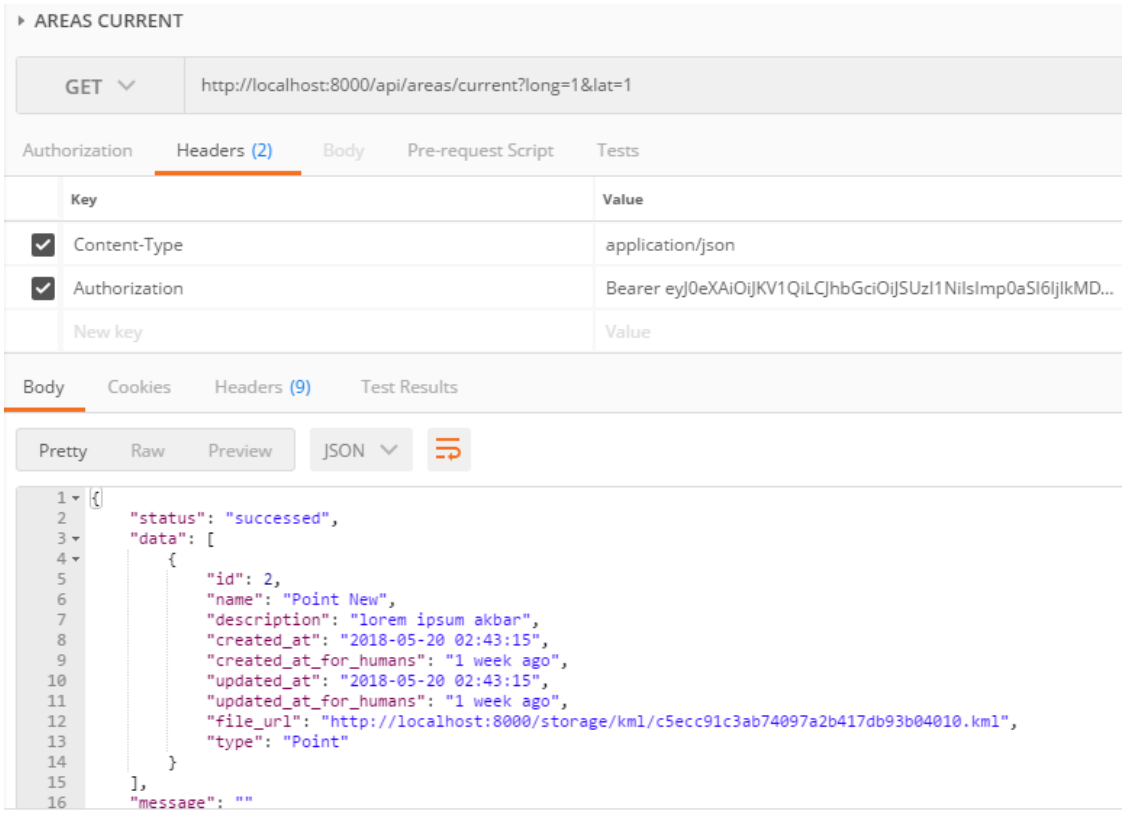
dapat ditemukan pada halaman my app. Pada halaman tersebut, terdapat daftar aplikasi pengguna yang sudah mendapatkan otoritas dan tombol revoke untuk melakukan pencabutan otoritas. Halaman my app dapat dilihat pada Gambar 4.19.



Gambar 4.19 Halaman My App – Daftar Aplikasi Yang Sudah Mendapatkan Otoritas

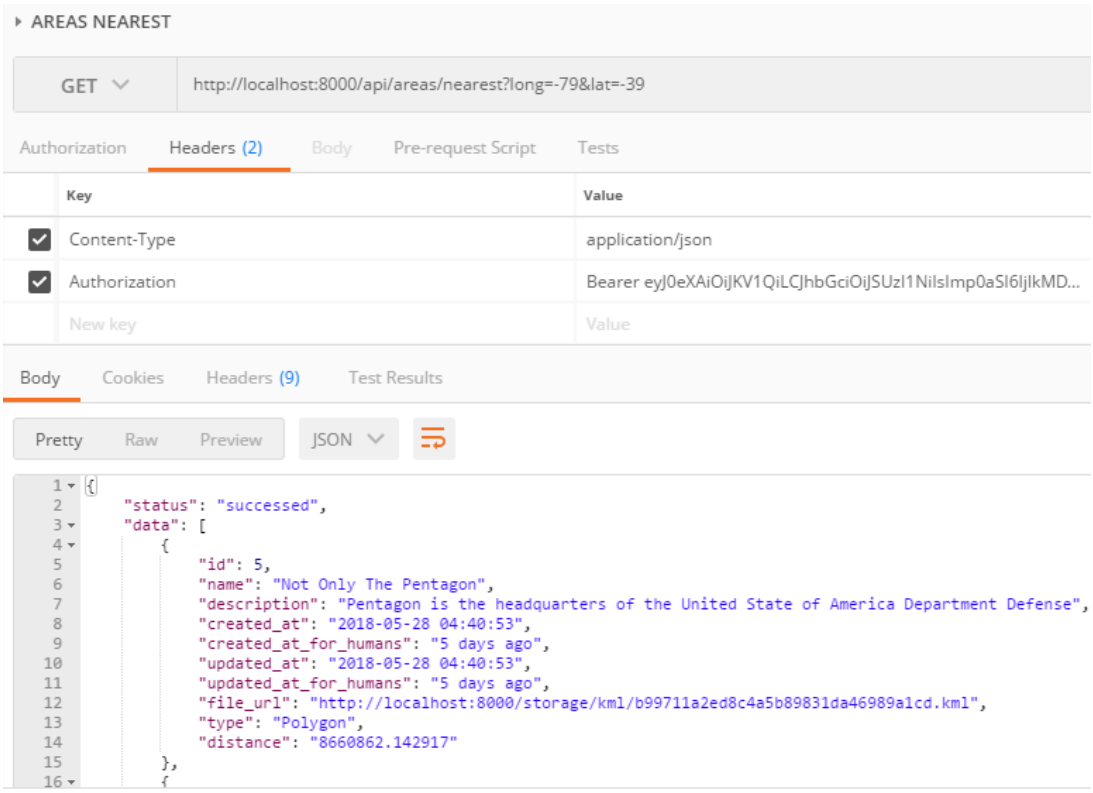
c. Informasi Area

Dalam memperoleh informasi area, aplikasi pengguna dapat mengirimkan HTTP *request* ke sistem. Permintaan tersebut memuat *endpoint* sumber data, HTTP *method*, dan *headers*. *Headers* memuat informasi tambahan yang akan dikirimkan ke *server*. Pada permintaan informasi area, terdapat dua *headers* yang dimuat yaitu *content-type* dan *authorization*. *Content-type* mendeskripsikan konteks isi pesan. *Authorization* mendeskripsikan token akses sebagai bukti sah dalam mengakses sumber data. Terdapat beberapa informasi area yang bisa didapatkan yaitu informasi area terkini, area terdekat, daftar area, dan area berdasarkan kata kunci. Informasi area terkini ditunjukkan pada Gambar 4.20. Informasi area terdekat ditunjukkan pada Gambar 4.21. Informasi daftar area ditunjukkan pada Gambar 4.22. Informasi area berdasarkan kata kunci ditunjukkan pada Gambar 4.23.



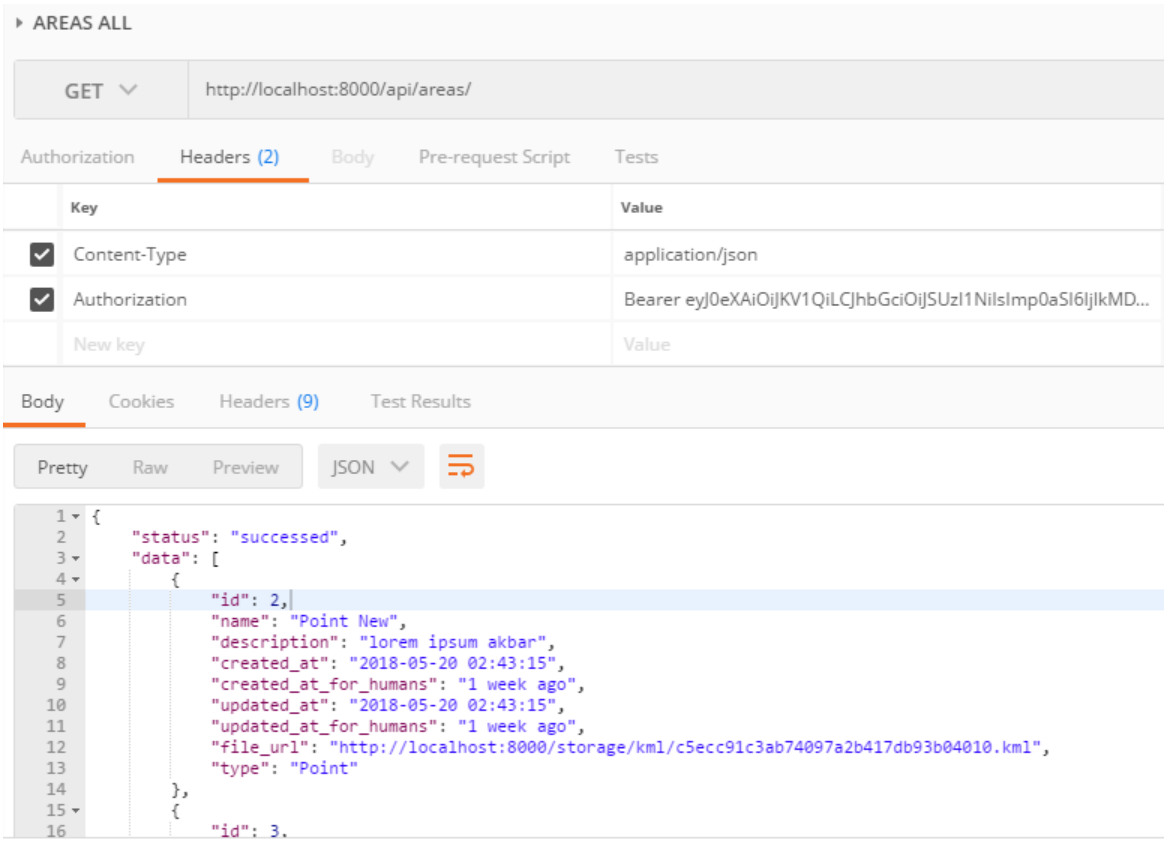
Gambar 4.20 *Request* dan *Response* Informasi Area Terkini

Gambar 4.20 merupakan *request* dan response infomasi area terkini berdasarkan koordinat lokasi. Pada gambar tersebut memuat *endpoint* dan *method* GET yang digunakan dalam mengirimkan permintaan. Terdapat parameter long dan lat yang digunakan untuk memasukan nilai koordinat. *Headers authorization* berisi token akses yang didapatkan pada tahapan sebelumnya dan wajib disertakan saat melakukan permintaan. Respons yang diberikan berisi informasi area terkini yang mengandung nilai koordinat (*longitude* dan *latitude*) dalam format JSON.



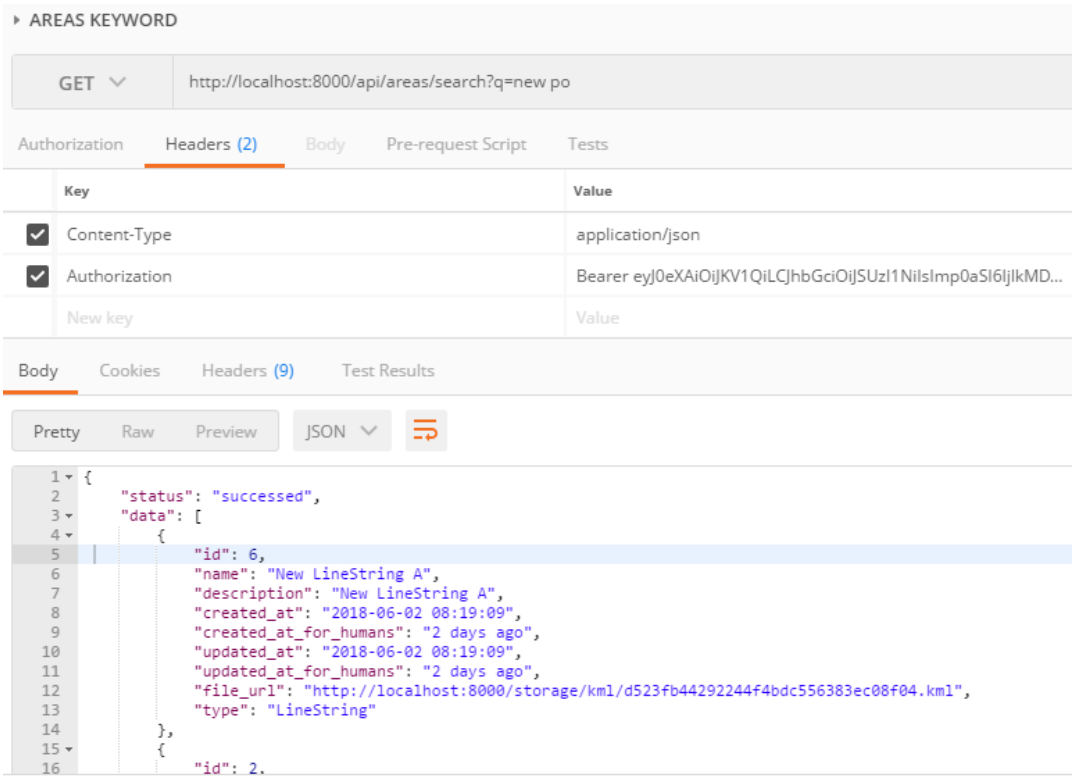
Gambar 4.21 *Request* dan *Response* Informasi Area Terdekat

Gambar 4.21 merupakan *request* dan *response* infomasi area terdekat berdasarkan koordinat lokasi. Pada gambar tersebut memuat *endpoint* dan *method* GET yang digunakan dalam mengirimkan permintaan. Terdapat parameter long dan lat yang digunakan untuk memasukan nilai koordinat. *Headers authorization* berisi token akses yang didapatkan pada tahapan sebelumnya dan wajib disertakan saat melakukan permintaan. Respons yang diberikan berisi informasi area terdekat dari nilai koordinat (*longitude* dan *latitude*) yang diberikan dalam format JSON. Nilai distance memuat jarak antara koordinat yang diberikan dengan obyek – obyek peta yang dimiliki oleh aplikasi pengguna. Satuan yang dipakai nilai distance yaitu meter.



Gambar 4.22 *Request* dan *Response* Informasi Daftar Area

Gambar 4.22 merupakan *request* dan *response* infomasi daftar area. Pada gambar tersebut memuat *endpoint* dan *method* GET yang digunakan dalam mengirimkan permintaan. *Headers authorization* berisi token akses yang didapatkan pada tahapan sebelumnya dan wajib disertakan saat melakukan permintaan. Respons yang diberikan berisi informasi daftar area yang dimiliki oleh aplikasi pengguna dalam format JSON.



Gambar 4.23 *Request* dan *Response* Informasi Area Berdasarkan Kata Kunci

Gambar 4.23 merupakan *request* dan *response* infomasi area berdasarkan kata kunci. Pada gambar tersebut memuat *endpoint* dan *method* GET yang digunakan dalam mengirimkan permintaan. Terdapat parameter q yang memuat kata kunci area yang ingin dicari. *Headers authorization* berisi token akses yang didapatkan pada tahapan sebelumnya dan wajib disertakan saat melakukan permintaan. Respons yang diberikan berisi informasi area berdasarkan kata kunci yang dimiliki oleh aplikasi pengguna dalam format JSON.

d. Memperbarui Token Akses

Token akses memiliki masa berlaku pengguna. Untuk itu diperlukan fitur yang dapat memperbarui token akses tanpa perlu melakukan pemberian otoritas. Dalam memperbarui token akses, aplikasi pengguna dapat mengirimkan HTTP *request* ke sistem.

```
<?php
use Illuminate\Http\Request;
Route::get('/refresh', function (Request $request) {
 $response = (new GuzzleHttp\Client)->post('http://localhost:8000/oauth/token', [
  'form_params' => [
    'grant_type' => 'refresh_token',
    'refresh_token' =>
'asd78dsf897shjahsasd98098sdf908aklqwewq098wqeqwe0sdfsdfdsf98sd09fsdcxvns89s7dfasda
sd889as7d897asd987asjkhsajkdnvjxcv87d89as7d0sad87as89d7s8a97d98sa7d89sa7d98hjkashdk
jashdkjashs
asydasuidyuisayd8as7d89721opqwiepoqimpoqipeoiqe98qw09e8qw908e098xzcoxuiousdskfjhsjk
dbmnxcvbmncxbvs
9sa8d9as89908s09f8sd90f8dsfsdufioquwioeuqwkdhfjkhdsjkfhsdvmcxnbvmncxbjhsksahjkdhasj
kdhsakyiqwueyuq
dsa9d890as8d90as8dfkjdsflkjdslkfjmxcnvmdsfusdfidsfopidsfdsjfkldsjflkdsjkflcxvncxmas
doiuasioduaoias
    ',
    'client_id' => 1,
    'client_secret' => 'NaqgpBFmB6rwxlHdouVDCeIRBYOgMRvombNdbthK',
    'scope' => '',
    ]
  ]);

  session()->put('token', json_decode((string) $response->getBody(), true));
  return $response->getBody();
});
```

Gambar 4.24 Kode Sumber HTTP *Request* Permintaan Perbaruan Token Akses

Gambar 4.24 merupakan contoh kode sumber HTTP *request* untuk memperbarui token akses. Permintaan memperbarui token akses hanya dapat dikirimkan oleh aplikasi pengguna. Penjelasan kode sumber *admin* ialah saat aplikasi pengguna menangkap permintaan akses ke /refresh maka sistem dari aplikasi pengguna mengirimkan HTTP *request* ke sistem layanan lokasi. Terdapat beberapa isian *body* yang harus disertai seperti grant_type, refresh_token yang didapatkan pada proses mendapatkan akses token, client_id, client_secret, dan scope. Setelah mengirimkan permintaan, maka sistem layanan lokasi memberikan respons yang berisi tipe token, token akses baru, refresh token baru, dan masa berlaku. Respons yang dikirimkan oleh sistem layanan lokasi ditunjukkan pada Gambar 4.25.

Gambar 4.25 Respons Perbaruan Token Akses

4.2 Pengujian

Tahapan berikut merupakan tahapan pengujian sistem dengan menggunakan teknik *equivalence partitioning*. Adapun pengujian sistem berdasarkan *use case* adalah sebagai berikut:

a. UC-1 (*Register*)

Pengujian sistem terhadap UC-1 (*Register*) dilakukan pada sisi *developer* untuk menguji fungsi registrasi akun. Pengujian fungsi tersebut dilakukan dengan mengisi *form* register sesuai dengan rancangan kasus uji yang telah dibuat sebelumnya. Hasil pengujian dilihat berdasarkan respons yang ditampilkan oleh sistem dalam menangani dua jenis masukan yang diberikan yaitu valid maupun invalid. Adapun hasil pengujian UC-1 (*Register*) dapat dilihat pada Tabel 4.1.

Tabel 4.1 Hasil Pengujian UC-1 (*Register*)

No	Kode Kasus Uji	Aktivitas	Hasil	Keterangan
1	TC-1, TC-2	Memberi masukan kolom name pada *form* register di halaman register	Berhasil	Saat sistem menerima masukan TC-2, maka sistem menampilkan pesan *error* “The name may not be greater than 255 characters.” pada kolom name

2	TC-3, TC-4	Memberi masukan kolom email pada *form* register di halaman register	Berhasil	Saat sistem menerima masukan TC-4, maka sistem menampilkan pesan *error* "The email must be a valid email address." pada kolom email
3	TC-5, TC-6	Memberi masukan kolom password pada *form* register di halaman register	Berhasil	Saat sistem menerima masukan TC-6, maka sistem menampilkan pesan *error* "The password must be at least 6 characters." pada kolom password
4	TC-7, TC-8	Memberi masukan kolom confirmation password pada *form* register di halaman register	Berhasil	Saat sistem menerima masukan TC-8, maka sistem menampilkan pesan *error* "The password confirmation does not match." pada kolom confirmation password

b. UC-2 (*Login*)

Pengujian sistem terhadap UC-2 (*Login*) dilakukan pada sisi *developer* untuk menguji fungsi *login*. Pengujian fungsi tersebut dilakukan dengan mengisi *form* login sesuai dengan rancangan kasus uji yang telah dibuat sebelumnya. Hasil pengujian dilihat berdasarkan respons yang ditampilkan oleh sistem dalam menangani dua jenis masukan yang diberikan yaitu valid maupun invalid. Adapun hasil pengujian UC-2 (*Login*) dapat dilihat pada Tabel 4.2.

Tabel 4.2 Hasil Pengujian UC-2 (*Login*)

No	**Kode Kasus Uji**	**Aktivitas**	**Hasil**	**Keterangan**
1	TC-9, TC-10	Memberi masukan kolom email pada *form* login di halaman login	Berhasil	Saat sistem menerima masukan TC-10, maka sistem menampilkan pesan *error* "These credentials do not match our records." pada kolom email
2	TC-11, TC-12	Memberi masukan kolom password pada *form* login di halaman login	Berhasil	Saat sistem menerima masukan TC-12, maka sistem menampilkan pesan *error* "These credentials do not match our records." pada kolom password

c. UC-3 (Reset Password)

Pengujian sistem terhadap UC-3 (*Reset Password*) dilakukan pada sisi *developer* untuk menguji fungsi *reset password*. Pengujian fungsi tersebut dilakukan dengan mengisi *form* pengiriman tautan reset password dan *form* pengaturan ulang password baru. Pengisian *form* tersebut sesuai dengan rancangan kasus uji yang telah dibuat sebelumnya. Hasil pengujian dilihat berdasarkan respons yang ditampilkan oleh sistem dalam menangani dua jenis masukan yang diberikan yaitu valid maupun invalid. Adapun hasil pengujian UC-3 (*Reset Password*) dapat dilihat pada Tabel 4.3.

Tabel 4.3 Hasil Pengujian UC-3 (*Reset Password*)

No	Kode Kasus Uji	Aktivitas	Hasil	Keterangan
1	TC-9, TC-10	Memberi masukan kolom email pada *form* pengiriman tautan reset password di halaman reset password – pengiriman tautan reset password	Berhasil	Saat sistem menerima masukan TC-10, maka sistem menampilkan pesan *error* "We can't find a user with that e-mail address." pada kolom email
2	TC-9, TC-10	Memberi masukan kolom email pada *form* pengaturan ulang password baru di halaman reset password – pengaturan ulang password baru	Berhasil	Saat sistem menerima masukan TC-10, maka sistem menampilkan pesan *error* "This password reset token is invalid." pada kolom email
3	TC-5, TC-6	Memberi masukan kolom password pada *form* pengaturan ulang password baru di halaman reset password – pengaturan ulang password baru	Berhasil	Saat sistem menerima masukan TC-6, maka sistem menampilkan pesan *error* "The password must be at least 6 characters." pada kolom password
4	TC-7, TC-8	Memberi masukan kolom confirmation password pada *form* pengaturan ulang password baru di halaman reset password – pengaturan ulang password baru	Berhasil	Saat sistem menerima masukan TC-8, sistem menampilkan pesan *error* "The password confirmation does not match." pada kolom confirmation password

d. UC-4 (Mengelola Aplikasi Pengguna)

Pengujian sistem terhadap UC-4 (Mengelola Aplikasi Pengguna) dilakukan pada sisi *developer* untuk menguji fungsi mengelola aplikasi pengguna. Pengujian fungsi tersebut dengan mengisi *form* pendaftaran aplikasi pengguna dan *form* pengubahan aplikasi

pengguna. Pengisian *form* tersebut sesuai dengan rancangan kasus uji yang telah dibuat sebelumnya. Hasil pengujian dilihat berdasarkan respons yang ditampilkan oleh sistem dalam menangani dua jenis masukan yang diberikan yaitu valid maupun invalid. Adapun hasil pengujian UC-4 (Mengelola Aplikasi Pengguna) dapat dilihat pada Tabel 4.4.

Tabel 4.4 Hasil Pengujian UC-4 (Mengelola Aplikasi Pengguna)

No	Kode Kasus Uji	Aktivitas	Hasil	Keterangan
1	TC-13, TC-14	Mendaftarkan Aplikasi Pengguna: memberi masukan kolom name pada *form* pendaftaran aplikasi pengguna di halaman my app	Berhasil	Saat sistem menerima masukan TC-14, maka sistem menampilkan pesan *error* "The name may not be greater than 255 characters."
2	TC-15, TC-16	Mendaftarkan Aplikasi Pengguna: memberi masukan kolom redirect URL pada *form* pendaftaran aplikasi pengguna di halaman my app	Berhasil	Saat sistem menerima masukan TC-16, maka sistem menampilkan pesan *error* "The redirect format is invalid."
3	TC-13, TC-14	Mengubah Aplikasi Pengguna: memberi masukan kolom name pada *form* pengubahan aplikasi pengguna di halaman my app	Berhasil	Saat sistem menerima masukan TC-14, maka sistem menampilkan pesan *error* "The name may not be greater than 255 characters."
4	TC-15, TC-16	Mengubah Aplikasi Pengguna: memberi masukan kolom redirect URL pada *form* pengubahan aplikasi pengguna di halaman my app	Berhasil	Saat sistem menerima masukan TC-16, sistem menampilkan pesan *error* "The redirect format is invalid."

e. UC-5 (Mengelola Area)

Pengujian sistem terhadap UC-5 (Mengelola Area) dilakukan pada sisi *developer* untuk menguji fungsi mengelola area. Pengujian fungsi tersebut dilakukan dengan mengisi parameter url area, mengisi *form* pendaftaran aplikasi pengguna dan mengisi *form* pengubahan aplikasi pengguna. Pengisian *form* tersebut sesuai dengan rancangan kasus uji yang telah dibuat sebelumnya. Hasil pengujian dilihat berdasarkan respons yang ditampilkan oleh sistem dalam menangani dua jenis masukan yang diberikan yaitu valid maupun invalid. Adapun hasil pengujian UC-5 (Mengelola Area) dapat dilihat pada Tabel 4.5.

Tabel 4.5 Hasil Pengujian UC-5 (Mengelola Area)

No	Kode Kasus Uji	Aktivitas	Hasil	Keterangan
1	TC-33, TC-34	Menampilkan Daftar Area: memberi masukan parameter client_id pada URL untuk menampilkan daftar area yang dimiliki oleh *client* atau aplikasi pengguna	Berhasil	Saat sistem menerima masukan TC-33, maka sistem menampilkan halaman not found apabila *developer* tidak memiliki data *client* dengan client_id tersebut. Saat sistem menerima masukan TC-34, maka sistem menampilkan halaman not found
2	TC-17, TC-18	Membuat Area: memberi masukan kolom *file* KML pada *form* pembuatan area di halaman resources - pembuatan area	Berhasil	Saat sistem menerima masukan TC-18, maka sistem menampikan pesan *error* “Maximum file 100 KB ” pada kolom *file* KML
3	TC-19, TC-20	Membuat Area: memberi masukan kolom *file* KML pada *form* pembuatan area di halaman resources - pembuatan area	Berhasil	Saat sistem menerima masukan TC-20, maka sistem menampilkan pesan *error* “Map doesn't contain any object” pada kolom *file* KML
4	TC-21, TC-22	Membuat Area: memberi masukan kolom *file* KML pada *form* pembuatan area di halaman resources - pembuatan area	Berhasil	Saat sistem menerima TC-22, maka sistem menampilkan pesan *error* “Map contains more than one object”
5	TC-23, TC-24	Membuat Area: memberi masukan kolom *file* KML pada *form* pembuatan area di halaman resources - pembuatan area	Berhasil	Saat sistem menerima TC-24, maka sistem menampilkan pesan *error* “Name / Description / Coordinates in your KML file is empty”
6	TC-25, TC-26	Membuat Area: memberi masukan kolom *file* KML pada *form* pembuatan area di halaman resources - pembuatan area	Berhasil	Saat sistem menerima TC-26, sistem menampilkan pesan *error* “The name may not be greater than 255 characters.”
7	TC-27, TC-28	Membuat Area: memberi masukan kolom *file* KML pada *form* pembuatan area di halaman resources - pembuatan area	Berhasil	Saat sistem menerima masukan TC-28, maka sistem menampilkan pesan *error* “Polygon must has minimal 4 points”
8	TC-29, TC-30	Membuat Area: memberi masukan kolom *file* KML pada *form* pembuatan area di halaman resources -	Berhasil	Saat sistem menerima masukan TC-30, sistem menampilkan pesan *error* “First and last coordinate of Polygon doesn't

		pembuatan area		match”
9	TC-31, TC-32	Membuat Area: memberi masukan kolom *file* KML pada *form* pembuatan area di halaman resources - pembuatan area	Berhasil	Saat sistem menerima masukan TC-32, sistem menampilkan pesan *error* “Linestring must has minimal 2 points”
10	TC-45, TC-46	Membuat Area: memberi masukan kolom *file* KML pada *form* pembuatan area di halaman resources - pembuatan area	Berhasil	Saat sistem menerima masukan TC-46, maka sistem menampilkan pesan *error* “The long must be between -180 and 180”
11	TC-47, TC-48	Membuat Area: memberi masukan kolom *file* KML pada *form* pembuatan area di halaman resources - pembuatan area	Berhasil	Saat sistem menerima masukan TC-48, maka sistem menampilkan pesan *error* “The lat must be between -90 and 90”
12	TC-37, TC-38	Mengubah Area: memberi masukan parameter area_id pada URL untuk menampilkan halaman resources pengubahan area	Berhasil	Saat sistem menerima masukan TC-37, maka sistem menampilkan halaman not found, jika *client* tidak memiliki data area dengan area_id tersebut TC-38: sistem menampilkan halaman not found
13	TC-17, TC-18	Mengubah Area: memberi masukan kolom *file* KML pada *form* pengubahan area di halaman resources - pengubahan area	Berhasil	Saat sistem menerima masukan TC-18, maka sistem menampikan pesan *error* “Maximum file 100 KB ”pada kolom *file* KML
14	TC-19, TC-20	Mengubah Area: memberi masukan kolom *file* KML pada *form* pengubahan area di halaman resources - pengubahan area	Berhasil	Saat sistem menerima masukan TC-20, maka sistem menampilkan pesan *error* “Map doesn't contain any object” pada kolom *file* KML
15	TC-21, TC-22	Mengubah Area: memberi masukan kolom *file* KML pada *form* pengubahan area di halaman resources - pengubahan area	Berhasil	Saat sistem menerima masukan TC-22, maka sistem menampilkan pesan *error* “Map contains more than one object”
16	TC-23, TC-24	Mengubah Area: memberi masukan kolom *file* KML pada *form* pengubahan area di halaman resources - pengubahan area	Berhasil	Saat sistem menerima masukan TC-24, maka sistem menampilkan pesan *error* “Name / Description / Coordinates in your KML *file* is empty”

17	TC-25, TC-26	Mengubah Area: memberi masukan kolom *file* KML pada *form* pengubahan area di halaman resources - pengubahan area	Berhasil	Saat sistem menerima masukan TC-26, maka sistem menampilkan pesan *error* "The name may not be greater than 255 characters."
18	TC-27, TC-28	Mengubah Area: memberi masukan kolom *file* KML pada *form* pengubahan area di halaman resources - pengubahan area	Berhasil	Saat sistem menerima masukan TC-28, maka sistem menampilkan pesan *error* "Polygon must has minimal 4 points"
19	TC-29, TC-30	Mengubah Area: memberi masukan kolom *file* KML pada *form* pengubahan area di halaman resources - pengubahan area	Berhasil	Saat sistem menerima masukan TC-30, maka sistem menampilkan pesan *error* "First and last coordinate of Polygon doesn't match"
20	TC-31, TC-32	Mengubah Area: memberi masukan kolom *file* KML pada *form* pengubahan area di halaman resources - pengubahan area	Berhasil	Saat sistem menerima masukan TC-32, maka sistem menampilkan pesan *error* "Linestring must has minimal 2 points"
21	TC-45, TC-46	Mengubah Area: memberi masukan kolom *file* KML pada *form* pengubahan area di halaman resources - pengubahan area	Berhasil	Saat sistem menerima masukan TC-46, maka sistem menampilkan pesan *error* "The long must be between -180 and 180"
22	TC-47, TC-48	Mengubah Area: memberi masukan kolom *file* KML pada *form* pengubahan area di halaman resources - pengubahan area	Berhasil	Saat sistem menerima masukan TC-48, maka sistem menampilkan pesan *error* "The lat must be between -90 and 90"
23	TC-37, TC-38	Visualisasi Area: memberi masukan parameter area_id pada URL untuk menampilkan halaman resources visualisasi area	Berhasil	Saat sistem menerima masukan TC-37, sistem menampilkan halaman not found apabila *client* tidak memiliki data area dengan area_id tersebut Saat sistem menerima masukan TC-38, maka sistem menampilkan halaman not found

f. UC-6 (Melihat Dokumentasi)

Pengujian sistem terhadap UC-6 (Melihat Dokumentasi) dilakukan pada sisi *developer* untuk menguji fungsi melihat dokumentasi. Pengujian fungsi tersebut dilakukan dengan melakukan klik menu dokumentasi pada *navigation bar*. Hasil pengujian dilihat berdasarkan respons yang ditampilkan oleh sistem. Adapun hasil pengujian UC-6 (Melihat Dokumentasi) dapat dilihat pada Tabel 4.6.

Tabel 4.6 Hasil Pengujian UC-6 (Melihat Dokumentasi)

No	Kode Kasus Uji	Aktivitas	Hasil	Keterangan
1	-	Membuka URL ke halaman dokumentasi	Berhasil	Sistem menampilkan halaman dokumentasi

g. UC-7 (Mendapatkan Token Akses)

Pengujian sistem terhadap UC-7 (Mendapatkan Token Akses) dilakukan pada sisi aplikasi pengguna untuk menguji fungsi mendapatkan token akses. Pengujian fungsi tersebut dilakukan dengan mengisi parameter url pada *request* kode otorisasi dan *body* pada *request* token akses. Pengisian tersebut sesuai dengan rancangan kasus uji yang telah dibuat sebelumnya. Hasil pengujian dilihat berdasarkan respons yang ditampilkan oleh sistem dalam menangani dua jenis masukan yang diberikan yaitu valid maupun invalid. Adapun hasil pengujian UC-7 (Mendapatkan Token Akses) dapat dilihat pada Tabel 4.7.

Tabel 4.7 Hasil Pengujian UC-7 (Mendapatkan Token Akses)

No	Kode Kasus Uji	Aktivitas	Hasil	Keterangan
1	TC-33, TC-34	Memberi masukan parameter client_id pada permintaan kode otorisasi	Berhasil	Saat sistem menerima masukan TC-34, maka sistem mengembalikan respons “Client authentication failed”
2	TC-15, TC-16	Memberi masukan parameter redirect_uri pada permintaan kode otorisasi	Berhasil	Saat sistem menerima masukan TC-16, maka sistem mengembalikan respons “Client authentication failed”
3	TC-33, TC-34	Memberi masukan *body* client_id pada permintaan token akses	Berhasil	Saat sistem menerima masukan TC-34, maka sistem mengembalikan respons “Client authentication failed”
4	TC-35, TC-36	Memberi masukan *body* client_secret pada permintaan token akses	Berhasil	Saat sistem menerima masukan TC-36, maka sistem mengembalikan respons “Client authentication failed”
5	TC-15, TC-16	Memberi masukan *body* redirect_url pada permintaan token akses	Berhasil	Saat sistem menerima masukan TC-16, maka sistem mengembalikan respons “Client authentication failed”

6	TC-43, TC-44	Memberi masukan *body* code pada permintaan token akses	Berhasil	Saat sistem menerima masukan TC-44, maka sistem mengembalikan respons "Invalid parameter value"

h. UC-8 (Mendapatkan Informasi Area)

Pengujian sistem terhadap UC-8 (Mendapatkan Informasi Area) dilakukan pada sisi aplikasi pengguna untuk menguji fungsi mendapatkan informasi area. Pengujian fungsi tersebut dilakukan dengan mengisi parameter url dan *headers* pada setiap *request* informasi area. Pengisian tersebut sesuai dengan rancangan kasus uji yang telah dibuat sebelumnya. Hasil pengujian dilihat berdasarkan respons yang ditampilkan oleh sistem dalam menangani dua jenis masukan yang diberikan yaitu valid maupun invalid. Adapun hasil pengujian UC-8 (Mendapatkan Informasi Area) dapat dilihat pada Tabel 4.8.

Tabel 4.8 Hasil Pengujian UC-8 (Mendapatkan Informasi Area)

No	Kode Kasus Uji	Aktivitas	Hasil	Keterangan
1	TC-39, TC-40	Mendapatkan informasi area terkini: memberi masukan token akses pada *headers* authorization	Berhasil	Saat sistem menerima masukan TC-40, maka sistem mengembalikan respons "Unauthenticated"
2	TC-45, TC-46	Mendapatkan informasi area terkini: memberi masukan parameter long pada URL	Berhasil	Saat sistem menerima masukan TC-45, maka sistem mengambalikan respons "Coordinate is not in any serviced area" apabila tidak ada area yang memiliki koordinat tersebut. Saat sistem menerima masukan TC-46, maka sistem mengambalikan respons "The long must be between -180 and 180"
3	TC-47, TC-48	Mendapatkan informasi area terkini: memberi masukan parameter lat pada URL	Berhasil	Saat sistem menerima masukan TC-47, maka sistem mengambalikan respons "Coordinate is not in any serviced area" apabila tidak ada area yang memiliki koordinat tersebut. Saat sistem menerima masukan TC-48, maka sistem mengambalikan respons "The lat

				must be between -90 and 90”
4	TC-39, TC-40	Mendapatkan informasi area terdekat: memberi masukan token akses pada *headers* authorization	Berhasil	Saat sistem menerima masukan TC-40, maka sistem mengembalikan respons “Unauthenticated”
5	TC-45, TC-46	Mendapatkan informasi area terdekat: memberi masukan parameter long pada URL	Berhasil	Saat sistem menerima masukan TC-46, maka sistem mengambalikan respons “The long must be between -180 and 180”
6	TC-47, TC-48	Mendapatkan informasi area terkini: memberi masukan parameter lat pada URL	Berhasil	Saat sistem menerima masukan TC-48, maka sistem mengambalikan respons “The lat must be between -90 and 90”
7	TC-39, TC-40	Mendapatkan informasi daftar area: memberi masukan token akses pada *headers* authorization	Berhasil	Saat sistem menerima masukan TC-40, maka sistem mengembalikan respons “Unauthenticated”
8	TC-39, TC-40	Mendapatkan informasi area berdasarkan kata kunci: memberi masukan token akses pada *headers* authorization	Berhasil	Saat sistem menerima masukan TC-40, maka sistem mengembalikan respons “Unauthenticated”
9	TC-49, TC-50	Mendapatkan informasi area berdasarkan kata kunci: memberi masukan pada parameter q	Berhasil	Saat sistem menerima masukan TC-50, maka sistem mengembalikan respons “Parameter q is required”

i. UC-9 (Memperbarui Token Akses)

Pengujian sistem terhadap UC-9 (Memperbarui Token Akses) dilakukan pada sisi aplikasi pengguna untuk menguji fungsi memperbarui token akses. Pengujian fungsi tersebut dilakukan dengan mengisi *body* pada *request* refresh token. Pengisian tersebut sesuai dengan rancangan kasus uji yang telah dibuat sebelumnya. Hasil pengujian dilihat berdasarkan respons yang ditampilkan oleh sistem dalam menangani dua jenis masukan yang diberikan yaitu valid maupun invalid. Adapun hasil pengujian UC-9 (Memperbarui Token Akses) dapat dilihat pada Tabel 4.9.

Tabel 4.9 Hasil Pengujian UC-9 (Memperbarui Token Akses)

No	Kode Kasus Uji	Aktivitas	Hasil	Keterangan
1	TC-33, TC-34	Memberi masukan *body* client_id pada *request* perbaruan token akses	Berhasil	Saat sistem menerima masukan TC-34, maka sistem mengembalikan respons "Client authentication failed"
2	TC-35, TC-36	Memberi masukan *body* client_secret pada *request* perbaruan token akses	Berhasil	Saat sistem menerima masukan TC-36, maka sistem mengembalikan respons "Client authentication failed"
3	TC-41, TC-42	Memberi masukan *body* refresh_token pada *request* perbaruan token akses	Berhasil	Saat sistem menerima masukan TC-42, maka sistem mengembalikan respons "The refresh token is invalid"

Berdasarkan hasil pengujian yang telah dilakukan menggunakan teknik *equivalence partitioning*, semua pengujian terhadap *use case* berhasil dilakukan. Sistem mengembalikan respons yang sesuai dengan rancangan pengujian berdasarkan masukan yang diberikan baik valid maupun invalid.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan penulis, maka dapat diambil beberapa kesimpulan, yaitu:

a. Penelitian ini menghasilkan sebuah RESTful API yang dapat digunakan sebagai layanan lokasi yang menyediakan informasi area secara abstraksi level tinggi sehingga dapat memenuhi kebutuhan spesifik aplikasi LBS.
b. Penelitian yang dikembangkan juga menyediakan aplikasi berbasis *web* yang memudahkan *developer* dalam mengelola aplikasi miliknya dan area. Selain itu, *developer* dapat secara bebas mendefinisikan peta area yang disesuaikan dengan kebutuhan aplikasi miliknya.
c. Berdasarkan hasil pengujian menggunakan teknik *equivalence partitioning*, layanan ini telah memenuhi rancangan pengujian dengan berbagai kasus uji.

5.2 Saran

Adapun beberapa saran yang dapat diberikan untuk pengembangan lanjutan dari layanan ini, yaitu:

a. Penambahan fitur untuk menyimpan data area secara kolektif seperti *multipoint*, *multilinestring*, atau *multipolygon*.
b. Penambahan fitur untuk mengelola data *altitude* dari koordinat sehingga memungkinkan dapat mendukung obyek 3D.

DAFTAR PUSTAKA

Adamczyk, P., Smith, P. H., Johnson, R. E., & Hafiz, M. (2011). REST: From Research to Practice. https://doi.org/10.1007/978-1-4419-8303-9

Boyd, R. (2012). *Getting Started with OAuth 2.0*. (M. Loukides & S. Wallace, Eds.). O'Reilly Media, Inc.

Dhingra, S. (2013). REST vs. SOAP: Choosing the best web service. Retrieved February 25, 2018, from http://searchmicroservices.techtarget.com/tip/REST-vs-SOAP-Choosing-the-best-web-service

Dirgahayu, T. (2016). Application-Specific High-Level Location Service.

Google. (2017). KML Tutorial | Keyhole Markup Language | Google Developers. Retrieved April 27, 2018, from https://developers.google.com/kml/documentation/kml_tut

Hardt, D. (2012). The OAuth 2.0 Authorization Framework, 1–76.

Kusuma R., W., Yapie, A. K., & Mulyani, E. S. (2013). Aplikasi Location Based Service (LBS) Taman Mini Indonesia Indah (TMII) Berbasis Android. *Seminar Nasional Aplikasi Teknologi Informasi 2013*, 13–18.

MuleSoft Videos. (2015). What is an API? - YouTube. Retrieved February 25, 2018, from https://www.youtube.com/watch?v=s7wmiS2mSXY

Open Geospatial Consortium (OGC). (2005). Open Location Services.

Oracle. (n.d.). MySQL :: MySQL 5.7 Reference Manual :: 11.5 Spatial Data Types. Retrieved April 22, 2018, from https://dev.mysql.com/doc/refman/5.7/en/spatial-types.html

Parecki, A. (2012). OAuth 2 Simplified • Aaron Parecki. Retrieved April 26, 2018, from https://aaronparecki.com/oauth-2-simplified/

Prasetyo, E., Hamzah, A., & Sutanta, E. (2016). PENYEDIAAN APLIKASI LAYANAN LOKASI BERBASIS LOCATION BASED SERVICE (LBS), *4*(1), 29–37.

RestApiTutorial. (n.d.). What is REST? Retrieved April 29, 2018, from http://www.restapitutorial.com/lessons/whatisrest.html#

Steiniger, S., Neun, M., Edwardes, A., & Lenz, B. (2012). Foundations of LBS. Retrieved from http://www.e-cartouche.ch/content_reg/cartouche/LBSbasics/en/text/LBSbasics.pdf

Sumadikarta, I., & Nugroho, A. A. (2017). Perancangan Aplikasi Location Based Service BPJS Kesehatan Berbasis Android, *2*(2), 33–43.

Sunshine. (2013). (Projected) Point on Line (2D) Algorithm. Retrieved August 5, 2018, from http://www.sunshine2k.de/coding/java/PointOnLine/PointOnLine.html

Virrantaus, K., Markkula, J., Garmash, A., Terziyan, V., Veijalainen, J., Katanosov, A., & Tirri, H. (2001). Developing GIS-supported location-based services. *Proceedings of the 2nd International Conference on Web Information Systems Engineering, WISE 2001*, *2*, 66–75. https://doi.org/10.1109/WISE.2001.996708

Wagh, K., & Thool, R. (2012). A Comparative study of SOAP vs REST web services provisioning techniques for mobile host, (April 2015).

LAMPIRAN

Lampiran A

FORM-TA/TF-A3

UNIVERSITAS ISLAM INDONESIA
Jurusan Teknik Informatika FTI

SARAN/USULAN PRESENTASI KEMAJUAN TUGAS AKHIR

Nama Mhs. : MUHAMMAD AKBAR
No. Mhs. : 14523277
Judul TA : Pengembangan RESTful API untuk Application Specific High Level Location Service

- Bagaimana skenario uji layanan ?
- Gambarkan alur menjadi flow diagram.
- Uji validitas bagaimana metodenya ?
- Tools-tools dijelaskan secara komprehensif.

Nilai kemajuan Tugas Akhir: _____ (0 – 100)
(studi pustaka, perancangan, penguasaan materi, ketepatan)

Yogyakarta, ...17-04-2018

Dosen,

..HARI...SETIAJI.
(nama terang)

Dilampirkan pada Laporan TA yang diajukan untuk pendadaran