



TUGAS AKHIR - KI141502

**PENENTUAN RUTE PERJALANAN
MENGGUNAKAN ALGORITMA A STAR DENGAN
MENGGUNAKAN DATA PETA
OPENSTREETMAP UNTUK DIGUNAKAN PADA
APLIKASI BERBASIS ANDROID CLEARROUTE**

**RIDHO PERDANA
NRP 5113100164**

Dosen Pembimbing I
Dr.tech.Ir. Raden Venantius Hari Ginardi, M.Sc.

Dosen Pembimbing II
Abdul Munif, S.Kom., M.Sc.

Departemen Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017



TUGAS AKHIR - KI141502

**PENENTUAN RUTE PERJALANAN
MENGGUNAKAN ALGORITMA A STAR
DENGAN MENGGUNAKAN DATA PETA
OPENSTREETMAP UNTUK DIGUNAKAN PADA
APLIKASI BERBASIS ANDROID CLEARROUTE**

**RIDHO PERDANA
NRP 5113100164**

**Dosen Pembimbing I
Dr.tech.Ir. Raden Venantius Hari Ginardi, M.Sc.**

**Dosen Pembimbing II
Abdul Munif, S.Kom., M.Sc.**

**Departemen Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017**

(Halaman ini sengaja dikosongkan)



UNDERGRADUATE THESES - KI141502

ROUTE DECISION WITH A STAR ALGORITHM WITH OPENSTREETMAP DATA TO BE IMPLEMENTED ON ANDROID APPLICATION CLEARROUTE

**RIDHO PERDANA
NRP 5113100164**

First Advisor
Dr.tech.Ir. Raden Venantius Hari Ginardi, M.Sc.

Second Advisor
Abdul Munif, S.Kom., M.Sc.

**Department of Informatics
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2017**

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

PENENTUAN RUTE PERJALANAN MENGGUNAKAN ALGORITMA A STAR DENGAN MENGGUNAKAN DATA PETA OPENSTREETMAP UNTUK DIGUNAKAN PADA APLIKASI BERBASIS ANDROID CLEARROUTE

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Manajemen Informasi
Program Studi S-1 Departemen Teknik Informatika
Fakultas Teknologi Informati
Institut Teknologi Sepuluh Nopember

Oleh:

RIDHO PERDANA
NRP: 5113100164

Disetujui oleh Pembimbing Tugas Akhir:

1. Dr.tech.Ir. Raden Venantius Hari Ginardi,
M.Sc.
NIP. 196505181992031003 (Pembimbing 1)
2. Abdul Munif, S.Kom., M.Sc.
NIP. 198608232015041004 (Pembimbing 2)

**SURABAYA,
Juni 2017**

(Halaman ini sengaja dikosongkan)

PENENTUAN RUTE PERJALANAN MENGGUNAKAN ALGORITMA A STAR DENGAN MENGGUNAKAN DATA PETA OPENSTREETMAP UNTUK DIGUNAKAN PADA APLIKASI BERBASIS ANDROID CLEARROUTE

Nama Mahasiswa : Ridho Perdana

NRP : 5113100164

Jurusan : Teknik Informatika FTIF-ITS

Dosen Pembimbing 1: Dr.tech. Ir. R.V.Hari Ginardi, M.Sc.

Dosen Pembimbing 2: Abdul Munif, S.Kom., M.Sc.

Abstrak

Aplikasi Clearroute adalah aplikasi berbasis Android yang dibangun untuk memudahkan pengguna untuk mengetahui informasi rute perjalanan sesuai dengan cuaca yang sedang / akan terjadi nantinya.

Fitur utama yang diberikan oleh aplikasi ini adalah diberikannya rute perjalanan yang disertakan dengan kondisi cuaca terkini pada rute tersebut. Rute yang diberikan oleh aplikasi ini tidak hanya 1, namun diberikan juga 2 rute alternatif agar mendapat kondisi cuaca yang terbaik untuk pengguna aplikasi.

Untuk menemukan rute perjalanan dengan parameter jumlah perempatan, jarak, serta waktu tempuh, digunakan algoritma A Star. Algoritma A Star yang digunakan adalah algoritma yang terdapat pada library pgRouting pada ekstensi basis data PostgreSQL. Isi dari basis data yang digunakan adalah data peta dari Openstreetmap. Agar PostgreSQL mampu memroses data spasial dari Openstreetmap, ekstensi PostGIS juga dipasangkan ke PostgreSQL yang terdapat dalam sistem. Laravel 5.4 digunakan untuk membungkus keseluruhan sistem menjadi REST API yang dapat diakses oleh klien aplikasi Clearroute.

Pengujian pada sistem ini dilakukan dengan cara melakukan permintaan rute perjalanan kepada API yang sudah dibuat, apakah setiap permintaan yang diminta mendapatkan rute

perjalanan yang dapat mengantarkan pengguna sampai ke tempat tujuan, sesuai dengan parameter yang ditetapkan.

Dari hasil pengujian, sistem yang telah dirancang dan dibuat telah memenuhi segala kebutuhan pengolahan data pada aplikasi Clearroute.

Kata kunci: Clearroute, pgRouting, PostgreSQL, PostGIS, Perempatan, Openstreetmap.

ROUTE MAKING WITH A STAR ALGORITHM WITH OPENSTREETMAP DATA TO BE IMPLEMENTED ON ANDROID APPLICATION CLEARROUTE

Student's Name	: Ridho Perdana
Student's ID	: 5113100164
Department	: Teknik Informatika FTIF-ITS
First Advisor	: Dr.tech. Ir. R.V. Hari Ginardi, M.Sc.
Second Advisor	: Abdul Munif, S.Kom., M.Sc.

Abstract

Clearroute is an application which is built to help user to gain information about travel route with current or later weather information. Clearroute is built for mobile smartphone with Android as the operation system.

The main feature of this application is it can give user the information about travel route with its current weather information. The given travel route from this application is not only 1, instead it also gives 2 alternate routes for the user get the best current weather information.

On the travel route decision, there are some parameters that is used to help the user to get the best route for them, those parameters are intersection, distance, and travel time. A Star algorithm is the algorithm to make the routing decision with those parameters work. PgRouting is the extension from the PostgreSQL database which has library to implement the A Star algorithm. The content of PostgreSQL is imported from Openstreetmap data. To make PostgreSQL can process spatial data from Openstreetmap, it need PostGIS extension. Laravel 5.4 is the framework to make the system become REST API and can be accessed from the client application.

The testing for this system is done by requesting travel route from the application client to the API, do all the travel route request getting route which fit with all the parameters from the

system. From the result of the testing, the system that has been designed and implemented has met all the needs of data processing for Clearroute application.

Keyword: Clearroute, pgRouting, PostgreSQL, PostGIS, Interseption, Openstreetmap.

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul:

“Penentuan Rute Perjalanan Menggunakan Algoritma A Star Dengan Menggunakan Data Peta Openstreetmap Untuk Digunakan Pada Aplikasi Berbasis Android Clearroute”

Harapan dari penulis semoga apa yang tertulis di dalam buku tugas akhir ini dapat bermanfaat bagi pengembangan ilmu pengetahuan saat ini, serta dapat memberikan kontribusi yang nyata.

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

1. Alm. Bapak Vence Alnasir, Ibu Esther Alnasir, Adik Eva Permatasari dan keluarga yang selalu memberikan dukungan penuh untuk menyelesaikan Tugas Akhir ini.
2. Bapak Hari Ginardi dan Bapak Abdul Munif selaku dosen pembimbing yang telah bersedia meluangkan waktu untuk memberikan petunjuk selama proses pengerjaan tugas akhir ini.
3. Bapak, Ibu dosen Jurusan Teknik Informatika ITS yang telah banyak memberikan ilmu dan bimbingan yang tak ternilai harganya bagi penulis.
4. Seluruh staf dan karyawan FTIf ITS yang banyak memberikan kelancaran administrasi akademik kepada penulis.

5. Gigih Suryaman, Arief Wahyu Megatama, Raden Fadil Rafii dan Wiwoho Aji Santoso yang selama ini memberikan semangat dalam hal studi maupun masalah lainnya.
6. Teman-teman BPH SCHEMATICS HMTC 2015, Departemen KMB HMTC Berkarya, yang telah memberikan pengalaman berharga kepada penulis selama masa studi penulis.
7. Teman-teman Administrator Lab Algoritma dan Pemrograman (Alpro), serta penghuni grup mantan KP di Jakarta.
8. Teman-teman angkatan 2013 jurusan Teknik Informatika ITS yang telah menemani perjuangan selama 4 tahun ini atas saran, masukan, dan dukungan terhadap penggeraan Tugas Akhir ini.
9. Serta pihak-pihak lain yang tidak dapat disebutkan disini yang telah banyak membantu penulis dalam penyusunan Tugas Akhir ini.

Penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini, namun penulis mohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya.

Surabaya, Juni 2017

Ridho Perdana

DAFTAR ISI

LEMBAR PENGESAHAN.....	v
Abstrak.....	vii
Abstract.....	ix
KATA PENGANTAR	xi
DAFTAR ISI.....	xiii
DAFTAR GAMBAR	xvii
DAFTAR TABEL.....	xix
DAFTAR KODE PROGRAM	xxi
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Permasalahan	2
1.4 Tujuan	2
1.5 Manfaat.....	3
1.6 Metodologi	3
1.6.1 Penyusunan Proposal Tugas Akhir	3
1.6.2 Studi Literatur	3
1.6.3 Analisis dan Desain Perangkat Lunak	4
1.6.4 Implementasi Perangkat Lunak.....	4
1.6.5 Pengujian dan Evaluasi.....	4
1.6.6 Penyusunan Buku	4
1.7 Sistematika Penulisan Laporan	5
BAB II TINJAUAN PUSTAKA	7
2.1 Algoritma A* (A Star).....	7
2.2 <i>Openstreetmap</i> (OSM)	12
2.3 <i>Badan Meteorologi Klimatologi dan Geofisika</i> (BMKG)	
14	
2.4 PostgreSQL	16
2.5 PostGIS	17
2.6 pgRouting	18
2.7 <i>Hypertext Preprocessor</i> (PHP).....	19
2.8 Mapbox	20
2.8.1 Mapbox Android SDK.....	20

2.9	<i>Application Programming Interface (API)</i>	21
2.10	<i>JavaScript Object Notation (JSON)</i>	21
2.11	<i>Geography JSON (GeoJSON)</i>	21
2.12	Osm2pgrouting.....	23
BAB III PERANCANGAN MODUL.....		25
3.1	Deskripsi Umum.....	25
3.1.1	Arsitektur Aplikasi Clearroute	25
3.1.1.1	<i>Applications</i>	27
3.1.1.2	<i>Backend-Services</i>	27
3.1.1.3	<i>Internal Data Resources</i>	28
3.1.1.4	<i>External Data Resources</i>	28
3.2	Desain Umum Sistem	29
3.2.1	Diagram Alur Desain Umum Sistem	30
3.2.1.1	Diagram Alur Persiapan Data Peta	31
3.2.1.2	Diagram Alur Penggunaan Data Peta	32
3.2.1.3	Diagram Alur Clearroute API.....	34
BAB IV IMPLEMENTASI.....		37
4.1	Lingkungan Implementasi	37
4.2	Implementasi	37
4.2.1	Implementasi Persiapan Data Peta	38
4.2.2	Implementasi Penggunaan Data Peta	44
4.2.3	Implementasi Clearroute <i>Application Programming Interface (API)</i>	53
BAB V UJI COBA DAN EVALUASI.....		69
5.1	Uji Coba	69
5.1.1	Mengubah Data Peta dari Openstreetmap ke Basis Data Spasial	69
5.1.2	Mengimplementasikan Algoritma A Star dengan pgRouting.....	77
5.1.3	Waktu Respon Server.....	82
5.1.4	Visualisasi Rute	88
5.2	Evaluasi	92
5.2.1	Mengubah Data Peta dari Openstreetmap ke Basis Data Spasial	92

5.2.2	Mengimplementasikan Algoritma A Star dengan pgRouting	93
5.2.3	Waktu Respon Server	93
5.2.4	Visualisasi Rute	95
BAB VI KESIMPULAN DAN SARAN	99
6.1	Kesimpulan.....	99
6.2	Saran.....	100
DAFTAR PUSTAKA	101
LAMPIRAN	103
BIODATA PENULIS	127

(Halaman ini sengaja dikosongkan)

DAFTAR GAMBAR

Gambar 2.1 Ilustrasi A Star 1	11
Gambar 2.2 Ilustrasi A Star 2.....	11
Gambar 2.3 Ilustrasi A Star 3	11
Gambar 2.4 Ilustrasi A Star 4.....	11
Gambar 2.5 Ilustrasi A Star 5	11
Gambar 2.6 Ilustrasi A Star 6.....	11
Gambar 2.7 Ilustrasi A Star 7	12
Gambar 2.8 Ilustrasi A Star 8.....	12
Gambar 2.9 Ilustrasi A Star 9	12
Gambar 2.10 Ilustrasi A Star 10	12
Gambar 2.11 Pengguna Openstreetmap Terdaftar	13
Gambar 2.12 Pembaharuan Basis Data Openstreetmap	14
Gambar 2.13 Contoh GeoJSON	22
Gambar 3.1 Diagram Arsitektur Aplikasi Clearroute.....	26
Gambar 3.2 Diagram Komponen Backend-Services.....	27
Gambar 3.3 External Data Resources.....	29
Gambar 3.4 Diagram Alur Modul Penentuan Rute Perjalanan ...	30
Gambar 3.5 Diagram Alur Persiapan Data	31
Gambar 3.6 Diagram Alur Penggunaan Data Peta	32
Gambar 3.7 Diagram Alur Clearroute API.....	34
Gambar 3.8 Pembuatan Rute Alternatif	35
Gambar 4.1 Situs mapzen.com.....	38
Gambar 4.2 Parameter fungsi pgr_analyzeGraph	40
Gambar 5.1 Proses Memasukkan Data jawa_timur 1.osm.....	74
Gambar 5.2 Proses Memasukkan Data jawa_timur 2.osm.....	74
Gambar 5.3 Proses Memasukkan Data jawa_timur 3.osm.....	75
Gambar 5.4 Proses Memasukkan Data jawa_timur 3.osm.....	75
Gambar 5.5 Isi Basis Data	76
Gambar 5.6 Isi Basis Data	76
Gambar 5.7 Isi Basis Data	76
Gambar 5.8 Hasil Implementasi Fungsi pgr_analyzegraph.....	77
Gambar 5.9 Isi Nilai Kolom cost_clearroute.....	78
Gambar 5.10 Contoh Nilai Pada Kolom cost_s.....	79

Gambar 5.11 Jalur yang Ditandai	80
Gambar 5.12 Uji Coba Jarak Dekat.....	81
Gambar 5.13 Uji Coba Jarak Sedang.....	81
Gambar 5.14 Uji Coba Jarak Jauh.....	82
Gambar 5.15 Hasil Uji Coba Waktu Respon 15 Pengguna Rute A	84
Gambar 5.16 Hasil Uji Coba Waktu Respon 15 Pengguna Rute B	84
Gambar 5.17 Hasil Uji Coba Waktu Respon 15 Pengguna Rute C	85
Gambar 5.18 Hasil Uji Coba Waktu Respon 25 Pengguna Rute A	85
Gambar 5.19 Hasil Uji Coba Waktu Respon 25 Pengguna Rute B	86
Gambar 5.20 Hasil Uji Coba Waktu Respon 25 Pengguna Rute C	86
Gambar 5.21 Hasil Uji Coba Waktu Respon 35 Pengguna Rute A	87
Gambar 5.22 Hasil Uji Coba Waktu Respon 35 Pengguna Rute B	87
Gambar 5.23 Hasil Uji Coba Waktu Respon 35 Pengguna Rute C	88
Gambar 5.24 Rute Dengan Tujuan Dekat Google Maps.....	89
Gambar 5.25 Rute Dengan Tujuan Dekat Clearroute.....	89
Gambar 5.26 Rute Dengan Tujuan Sedang Google Maps.....	90
Gambar 5.27 Rute Dengan Tujuan Sedang Clearroute	90
Gambar 5.28 Rute Dengan Tujuan Jauh Google Maps	91
Gambar 5.29 Rute Dengan Tujuan Jauh Clearroute	91
Gambar 5.30 Evaluasi Rute Google Maps 1	95
Gambar 5.31 Evaluasi Rute Google Maps 2	96
Gambar 5.32 Evaluasi Rute Perjalanan Dekat Clearroute	96
Gambar 5.33 Evaluasi Rute Perjalanan Sedang Clearroute.....	97
Gambar 5.34 Evaluasi Rute Perjalanan Jauh Clearroute	97

DAFTAR TABEL

Tabel 2.1 Nama Aplikasi Manajemen Basis Data Open Source	16
Tabel 2.2 Versi Java Beserta Tahun Rilis Error! Bookmark not defined.	
Tabel 4.1 Spesifikasi Perangkat Keras dan Perangkat Lunak	37
Tabel 4.2 Daftar parameter fungsi Algoritma Astar pgRouting	58
Tabel 4.3 Daftar Parameter fungsi TSP pgRouting	63
Tabel 5.1 Uji Coba Waktu Respon 15 Pengguna	83
Tabel 5.2 Uji Coba Waktu Respon 25 Pengguna	83
Tabel 5.3 Uji Coba Waktu Respon 35 Pengguna	83

(Halaman ini sengaja dikosongkan)

DAFTAR KODE PROGRAM

Kode Program 2.1 Pseudocode Algoritma A Star.....	9
Kode Program 2.2 Kode App Manifest Pada Android.....	20
Kode Program 2.3 Penggunaan Osm2pgrouting	23
Kode Program 4.1 Membuat basis data db_clearroute.....	38
Kode Program 4.2 Menambahkan ekstensi postgis dan pgrouting	39
Kode Program 4.3 Menjalankan tools Osm2pgrouting untuk memasukkan data peta ke basis data	39
Kode Program 4.4 Fungsi pgr_analyzeGraph	39
Kode Program 4.5 Penjelasan parameter fungsi pgr_analyzeGraph	40
Kode Program 4.6 Menambahkan kolom cost_clearroute	41
Kode Program 4.7 Membuat tabel duplikat	41
Kode Program 4.8 Fungsi UpdateCostClearroute	42
Kode Program 4.9 Memberi Penalti Pada Jalur yang Ingin Dihindari.....	43
Kode Program 4.10 Mempercepat Akses Basis Data.....	44
Kode Program 4.11 Mendapatkan koordinat awal	46
Kode Program 4.12 Penggambaran Peta Koordinat Awal	47
Kode Program 4.13 Penggunaan fungsi GeocoderAutoCompleteView pada Aplikasi Clearroute	48
Kode Program 4.14 Fungsi Penggambaran Rute Perjalanan.....	53
Kode Program 4.15 Fungsi Penentuan Rute Perjalanan Utama ..	56
Kode Program 4.16 Fungsi Algoritma Astar pgRouting	57
Kode Program 4.17 Fungsi penentuan rute alternatif.....	62
Kode Program 4.18 Fungsi TSP pgRouting	62
Kode Program 4.19 Fungsi penghasil rute perjalanan utama.....	63
Kode Program 4.20 Fungsi menghasilkan nilai random untuk koordinat.....	64
Kode Program 4.21 Fungsi menentukan batasan dan koordinat y baru.....	65
Kode Program 4.22 Fungsi menentukan batasan dan koordinat x baru.....	66

Kode Program 4.23 Fungsi penghasil rute perjalanan alternatif	67
Kode Program 5.1 Isi data jawa_timur.osm	73
Kode Program 5.2 Kode Program Pemanggil Algoritma A Star.....	78
Kode Program 5.3 Menambah memori swap	92
Kode Program 5.4 Pseudocode Penentuan Rute Perjalanan.....	94

BAB I

PENDAHULUAN

1.1 Latar Belakang

Setiap orang pasti pernah melakukan perjalanan. Dalam setiap perjalanan tentu banyak pertimbangan yang akan diperhitungkan untuk melaksanakan perjalanan tersebut, beberapa diantaranya adalah rute yang dilalui dan cuaca. Rute merupakan alur dan arah yang akan dilalui untuk mencapai tempat tujuan, dan cuaca adalah keadaan udara pada saat tertentu dan di wilayah tertentu yang relatif sempit (tidak luas) dan pada jangka waktu yang singkat.

Di Indonesia, terdapat Badan yang sudah secara resmi memiliki tugas untuk mengawasi dan memberikan berita mengenai cuaca yang sedang berlangsung dan akan terjadi, yaitu BMKG (Badan Meteorologi Klimatologi dan Geofisika). BMKG secara *realtime* memperbarui berita cuaca yang ada, agar dapat dilihat oleh masyarakat umum. BMKG Surabaya menggunakan acuan beberapa Pos Hujan yang biasanya merupakan kantor desa atau tempat umum besar lainnya untuk memberikan informasi mengenai cuaca di sekitar Pos Hujan tersebut.

Dewasa kini pengguna tidak perlu repot-repot memikirkan rute mana yang harus diambil, karena hal tersebut sudah dapat dibuat secara langsung oleh komputer. Komputer menentukan rute berdasarkan pilihan-pilihan yang sudah diatur oleh pengguna atau pengembangnya, dan semua pilihan itu diolah menggunakan suatu algoritma yang dirasa pengembang merupakan algoritma terbaik untuk menentukan rute yang diinginkan.

Pada tugas akhir ini, algoritma yang digunakan untuk menentukan rute perjalanan adalah A* (A Star). A* adalah algoritma pencarian jalan yang ditemukan oleh Peter Hart, Nils Nilsson dan Bertram Raphael dari *Stanford Research Institute*. A Star merupakan pengembangan dari algoritma pencarian jalan tercepat lainnya, yaitu Dijkstra. A Star adalah sebuah algoritma

best-first search, yang berarti menyelesaikan masalah dengan mencari dari semua kemungkinan jalan untuk mencapai tujuan dengan *cost* terendah.

1.2 Rumusan Masalah

Tugas Akhir ini mengangkat beberapa rumusan masalah sebagai berikut:

1. Bagaimana mengubah data peta dari Openstreetmap menjadi data *graph* di dalam Basis Data Spasial?
2. Bagaimana mengimplementasikan algoritma A* (A Star) pada penentuan rute?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini memiliki batasan sebagai berikut:

1. Data peta yang akan diolah hanya Kota Surabaya.
2. Parameter yang digunakan dalam penentuan rute adalah jarak, waktu tempuh, dan perempatan jalan.

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah sebagai berikut:

1. Membuat data *graph* Surabaya yang didapat dari Openstreetmap.
2. Menentukan rute perjalanan berdasarkan dengan jarak, waktu tempuh, dan perempatan jalan.
3. Mengimplementasikan algoritma A Star dalam penentuan rute.
4. Menghasilkan 3 rute perjalanan, 1 rute utama dan 2 rute alternatif.
5. Membuat dan menyiapkan titik dari rute yang didapat untuk digunakan pada modul lain.

1.5 Manfaat

Manfaat yang diperoleh dari pembuatan Tugas Akhir ini adalah dihasilkannya suatu API yang dapat digunakan pada aplikasi Android Clearroute dimana aplikasi tersebut dapat membantu pengguna untuk mengetahui kondisi cuaca pada jalur perjalanan yang akan dilaluinya, sehingga pengguna dapat mempersiapkan segala kebutuhan perjalanannya.

1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan dengan menggunakan metodologi sebagai berikut:

1.6.1 Penyusunan Proposal Tugas Akhir

Tahapan awal dari Tugas Akhir ini adalah penyusunan Proposal Tugas Akhir. Proposal Tugas Akhir berisi pendahuluan, deskripsi dan gagasan metode-metode yang dibuat dalam Tugas Akhir ini. Pendahuluan ini terdiri atas hal yang menjadi latar belakang diajukannya Tugas Akhir, rumusan masalah yang diangkat, batasan masalah untuk Tugas Akhir, dan manfaat dari hasil pembuatan Tugas Akhir ini. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan Tugas Akhir. Terdapat pula sub bab jadwal kegiatan yang menjelaskan jadwal penggerjaan Tugas Akhir.

1.6.2 Studi Literatur

Pada tahap ini, akan dicari studi literatur yang relevan untuk dijadikan referensi dalam penggerjaan Tugas Akhir. Studi literatur yang digunakan adalah berupa situs resmi dari Openstreetmap, *library pgRouting*, dan situs yang membahas algoritma A Star.

1.6.3 Analisis dan Desain Perangkat Lunak

Analisa dimulai dari dimasukkannya data peta sebagai *input* untuk membuat *graph* dan basis data spasial pada server. Lalu data *input* yang diberikan oleh pengguna (berupa titik awal dan tujuan) akan diterima dan diproses menggunakan algoritma A Star untuk mendapatkan rute perjalanan. Hasil dari langkah di atas adalah beberapa titik yang akan dibentuk menjadi *file* JSON. *File* tersebut akan diproses oleh modul Penentuan Titik Hujan sehingga dapat ditampilkan rute dengan kondisi cuaca pada aplikasi Clearroute.

1.6.4 Implementasi Perangkat Lunak

Implementasi merupakan tahap untuk membangun metode-metode yang sudah diajukan pada proposal Tugas Akhir. Untuk menjalankan metode yang sudah diajukan, akan digunakan beberapa bahasa pemrograman PHP, SQL, dan Java.

1.6.5 Pengujian dan Evaluasi

Pada tahap ini dilakukan pengujian terhadap sistem yang dibangun. Pengujian dan evaluasi sistem dilakukan untuk mencari masalah yang mungkin timbul dan melakukan perbaikan jika ditemukan kesalahan pada sistem.

1.6.6 Penyusunan Buku

Pada tahap ini dilakukan penyusunan buku yang menjelaskan seluruh konsep, teori dasar dari metode yang digunakan, implementasi, serta hasil yang telah dikerjakan sebagai dokumentasi dari pelaksanaan Tugas Akhir.

1.7 Sistematika Penulisan Laporan

Sistematika penulisan laporan Tugas Akhir adalah sebagai berikut:

1. Bab I. Pendahuluan

Bab ini berisikan penjelasan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan dari pembuatan Tugas Akhir.

2. Bab II. Tinjauan Pustaka

Bab ini berisi kajian teori atau penjelasan dari metode, algoritma, *library*, dan *tools* yang digunakan dalam penyusunan Tugas Akhir ini. Bab ini berisi tentang algoritma A Star, Openstreetmap, BMKG, PostgreSQL, PostGIS, pgRouting, bahasa pemrograman PHP, bahasa pemrograman Mapbox, *Application Programming Interface* (API), tipe data JSON, tipe data GeoJSON, Osm2pgsqlouting.

3. Bab III. Perancangan Perangkat Lunak

Bab ini berisi pembahasan mengenai perancangan dari pembuatan modul penentuan rute untuk aplikasi Clearroute. Pembahasan dimulai dari mengolah data peta hingga menjadi graph untuk digunakan dengan algoritma A Star sesuai dengan parameter yang diinginkan, sampai dengan penggambarannya pada aplikasi Clearroute.

4. Bab IV. Implementasi

Bab ini menjelaskan implementasi yang berbentuk kode sumber dari proses memasukkan data peta ke dalam basis data PostgreSQL, mengubah nilai kolom sesuai dengan parameter yang akan digunakan, *syntax SQL* untuk memanggil fungsi algoritma A Star pada pgRouting, pembuatan rute alternatif.

5. Bab V. Hasil Uji Coba dan Evaluasi

Bab ini berisikan hasil uji coba dan evaluasi dari implementasi yang telah dilakukan untuk menyelesaikan masalah yang dibahas pada Tugas Akhir.

6. Bab VI. Kesimpulan dan Saran

Bab ini merupakan bab yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan, masalah-masalah yang dialami pada proses penggerjaan Tugas Akhir, dan saran untuk pengembangan solusi ke depannya.

7. Daftar Pustaka

Bab ini berisi daftar pustaka yang dijadikan literatur dalam Tugas Akhir.

8. Lampiran

Dalam lampiran terdapat tabel-tabel data hasil uji coba dan kode sumber program secara keseluruhan.

BAB II

TINJAUAN PUSTAKA

Bab ini berisi pembahasan mengenai teori-teori dasar atau penjelasan dari metode, algoritma, *library* dan *tools* yang digunakan dalam Tugas Akhir.

2.1 Algoritma A* (A Star)

A* (diucapkan A Star) adalah salah satu metode yang terkenal untuk menemukan jalur terpendek antara 2 lokasi dalam suatu area. A Star dikembangkan pada tahun 1968 untuk menggabungkan pendekatan heuristic seperti Best-First-Search (BFS) dan pendekatan formal seperti Algoritma Djikstra.

A Star merupakan sebuah *informed search algorithm* atau sebuah *Best-first search*, yang berarti menyelesaikan masalah dengan mencari berdasarkan semua kemungkinan jalur untuk ke tujuan yang melibatkan nilai terkecil (jarak tempuh, waktu, dsb), dan diantara semua jalur A Star memilih yang terlihat paling mungkin untuk mencapai tujuan. Diformulasikan sesuai dengan *weighted graphs*, dimulai dari node spesifik, A Star membuat sebuah *tree* yang dimulai dari node tersebut, dan menyebar satu persatu sampai menemukan *node tujuan*.

Dalam setiap iterasi, A Star butuh menentukan jalur mana yang akan dikembangkan ke jalur lainnya. Hal tersebut dilakukan berdasarkan perhitungan nilai (total *weight*). Secara spesifik, A Star memilih jalur yang meminimalisir

$$f(n) = g(n) + h(n) \quad (2.1)$$

Dimana n adalah node terakhir dari jalur, g(n) adalah nilai/cost jalur dari node awal sampai ke n, dan h(n) adalah nilai heuristik yang menghitung nilai yang terkecil jalur dari n ke tujuan.

Untuk memperjelas mengenai algoritma A Star, Kode Program 2.1 akan menampilkan *pseudocode* dari algoritma tersebut [1] [2].

```
1  function A*(start, goal)
2      // The set of nodes already evaluated.
3      closedSet := {}
4      // The set of currently discovered nodes that
5      // are not evaluated yet.
6      // Initially, only the start node is known.
7      openSet := {start}
8      // For each node, which node it can most
9      // efficiently be reached from.
10     // If a node can be reached from many nodes,
11     cameFrom will eventually contain the
12     // most efficient previous step.
13     cameFrom := the empty map
14
15     // For each node, the cost of getting from the
16     start node to that node.
17     gScore := map with default value of Infinity
18     // The cost of going from start to start is
19     zero.
20     gScore[start] := 0
21     // For each node, the total cost of getting
22     from the start node to the goal
23     // by passing by that node. That value is
24     partly known, partly heuristic.
25     fScore := map with default value of Infinity
26     // For the first node, that value is
27     completely heuristic.
28     fScore[start] :=
29     heuristic_cost_estimate(start, goal)
30
31     while openSet is not empty
32         current := the node in openSet having the
33         lowest fScore[] value
34         if current = goal
35             return reconstruct_path(cameFrom,
36             current)
37
38         openSet.Remove(current)
39         closedSet.Add(current)
```

```

40         for each neighbor of current
41             if neighbor in closedSet
42                 continue          // Ignore the
43             neighbor which is already evaluated.
44             // The distance from start to a
45             neighbor
46             tentative_gScore := gScore[current] +
47             dist_between(current, neighbor)
48             if neighbor not in openSet      //
49             Discover a new node
50                 openSet.Add(neighbor)
51             else if tentative_gScore >=
52             gScore[neighbor]
53                 continue          // This is not
54             a better path.
55
56             // This path is the best until now.
57             Record it!
58             cameFrom[neighbor] := current
59             gScore[neighbor] := tentative_gScore
60             fScore[neighbor] := gScore[neighbor] +
61             heuristic_cost_estimate(neighbo, goal)
62
63             return failure
64
65             function reconstruct_path(cameFrom, current)
66                 total_path := [current]
67                 while current in cameFrom.Keys:
68                     current := cameFrom[current]
69                     total_path.append(current)
70                 return total_path

```

Kode Program 2.1 Pseudocode Algoritma A Star

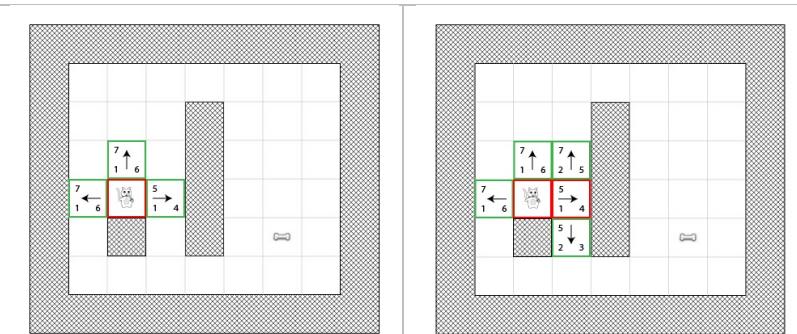
Algoritma A Star diawali dengan menyiapkan variabel penampung untuk set node yang sudah dievaluasi dan yang belum dievaluasi. Dibuat juga variabel untuk menyimpan langkah urutan

node dengan langkah terefisien. Setelah dipersiapkan variabel penampung, perlu ditentukan fungsi heuristik yang akan digunakan pada algoritma A Star.

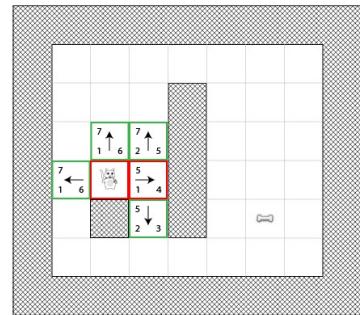
Dalam setiap perulangan *node*, dimasukkan nilai terkecil dari *cost node* awal ke tujuan, selain itu juga ditandai setiap *node* yang sudah dikunjungi/dievaluasi. Pada akhirnya, algoritma akan mengembalikan jalur yang dilalui untuk menuju tujuan. Untuk lebih memperjelas mengenai algoritma A Star, Gambar 2.1-Gambar 2.10 akan memperlihatkan contoh kasus dan penjelasan dari algoritma A Star.

Pada Gambar 2.1 diperlihatkan posisi awal yang berupa gambar kucing menentukan langkah yang dapat diambil dan dimasukkan ke openset (set yang belum dievaluasi) beserta dengan nilai *cost* nya. Gambar 2.2 memperlihatkan kucing tersebut mengambil nilai *cost* terkecil dan dimasukkan ke closedset (set yang sudah dievaluasi) dan mengambil nilai openset baru. Pada Gambar 2.3 hanya terdapat 1 jalur yang dapat dimasukkan ke closedset karena yang lainnya merupakan dinding. Langkah memilih *cost* terkecil dan dimasukkan ke dalam closedset terus diulang hingga ditemukan 2 langkah yang dapat diambil yang sama-sama dekat dengan tujuan, hal tersebut dapat dilihat pada Gambar 2.7. Lalu dilakukan sekali lagi iterasi, sehingga lokasi tujuan sudah masuk ke closedset, hal tersebut dapat dilihat pada Gambar 2.9. Dan pada akhirnya algoritma hanya tinggal melakukan *backward* untuk mendapatkan semua langkah dari closedset yang sudah disimpan [3].

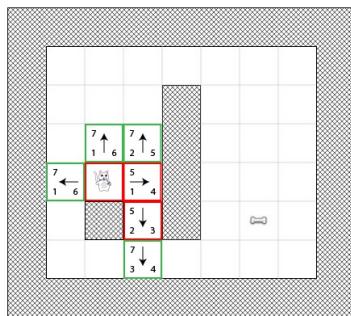
Sesuai dengan rumus persamaan matematika (2.1), algoritma A Star tidak akan menemukan jalur yang paling optimal apabila tidak digunakan fungsi heuristik yang *admissible*, dimana suatu fungsi dapat dikatakan sebagai *Admissible Heuristic* apabila tidak pernah menentukan cost ke tujuan secara berlebihan [4].



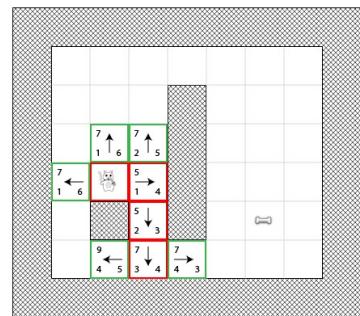
Gambar 2.1 Ilustrasi A Star 1



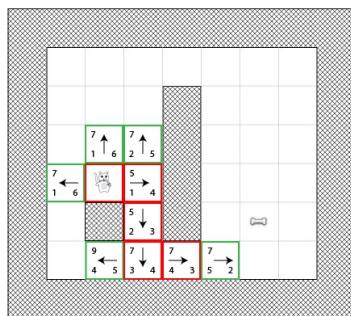
Gambar 2.2 Ilustrasi A Star 2



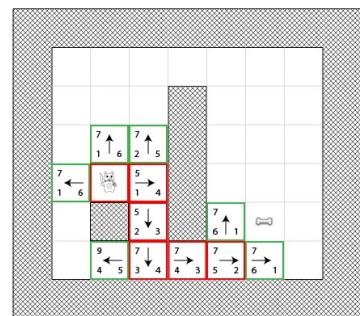
Gambar 2.3 Ilustrasi A Star 3



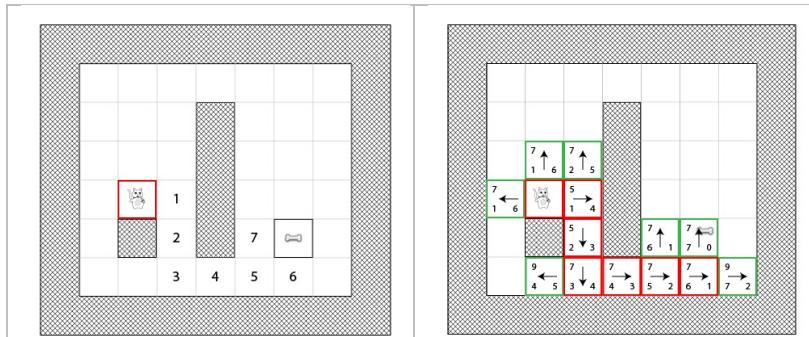
Gambar 2.4 Ilustrasi A Star 4



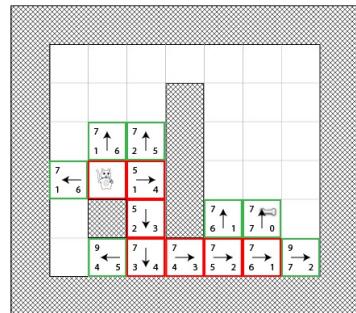
Gambar 2.5 Ilustrasi A Star 5



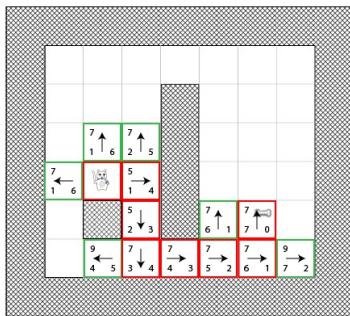
Gambar 2.6 Ilustrasi A Star 6



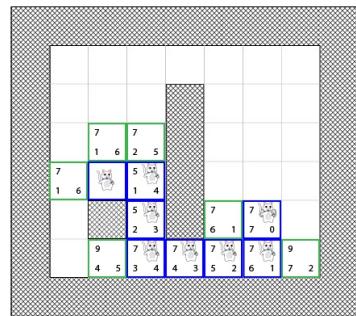
Gambar 2.7 Ilustrasi A Star 7



Gambar 2.8 Ilustrasi A Star 8



Gambar 2.9 Ilustrasi A Star 9



Gambar 2.10 Ilustrasi A Star 10

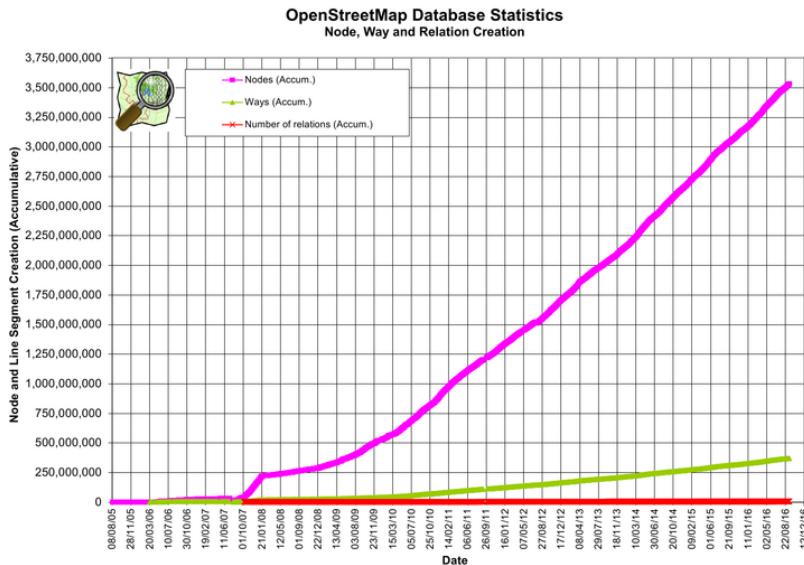
2.2 Openstreetmap (OSM)

Openstreetmap dibuat oleh komunitas peta yang saling berkontribusi dan merawat data mengenai jalan, jejak, kafe, stasiun, dan banyak hal lainnya di seluruh dunia. Openstreetmap menekankan pengetahuan lokal. Penyumbang menggunakan foto udara, perangkat GPS, dan peta lapangan untuk memverifikasi tingkat akurasi OSM dan selalu terbaharui [5]. Pengguna Openstreetmap terus berkembang setiap tahunnya. Gambar 2.11 akan memperlihatkan grafik pengguna yang terdaftar di situs Openstreetmap.



Gambar 2.11 Pengguna Openstreetmap Terdaftar

Dengan banyaknya pengguna yang terdaftar, menyebabkan basis data spasial Openstreetmap terus terbaharui, Gambar 2.12 akan memperlihatkan grafik pembaharuan basis data spasial Openstreetmap. Pembaharuan yang dilakukan adalah penambahan / pengubahan *nodes*, *ways*, dan *relations* yang terdapat di basis data spasial Openstreetmap.



Gambar 2.12 Pembaharuan Basis Data Openstreetmap

Openstreetmap juga sudah banyak digunakan oleh 3rd party application dalam hal penentuan rute perjalanan [6], diantaranya adalah:

1. Mapquest Open
2. OSRM, Skobbler
3. YOURS
4. GraphHopper Maps
5. CartoType
6. OpenRouteService
7. CycleStreets
8. Valhalla

2.3 Badan Meteorologi Klimatologi dan Geofisika (BMKG)

BMKG merupakan suatu lembaga pemerintah non departemen yang memiliki tugas untuk melaksanakan tugas pemerintahan di

bidang meteorologi, klimatologi dan geofisika. Dalam melaksanakan tugasnya BMKG menyelenggarakan fungsi berupa [7]:

- Perumusan kebijakan nasional dan kebijakan umum di bidang meteorologi, klimatologi, dan geofisika.
- Perumusan kebijakan teknis di bidang meteorologi, klimatologi, dan geofisika.
- Koordinasi kebijakan, perencanaan dan program di bidang meteorologi, klimatologi, dan geofisika.
- Pelaksanaan, pembinaan dan pengendalian observasi, dan pengolahan data dan informasi di bidang meteorologi, klimatologi, dan geofisika.
- Pelayanan data dan informasi di bidang meteorologi, klimatologi, dan geofisika.
- Penyampaian informasi kepada instansi dan pihak terkait serta masyarakat berkenaan dengan perubahan iklim.
- Penyampaian informasi dan peringatan dini kepada instansi dan pihak terkait serta masyarakat berkenaan dengan bencana karena faktor meteorologi, klimatologi, dan geofisika.
- Pelaksanaan kerja sama internasional di bidang meteorologi, klimatologi, dan geofisika.
- Pelaksanaan penelitian, pengkajian, dan pengembangan di bidang meteorologi, klimatologi, dan geofisika.
- Pelaksanaan, pembinaan, dan pengendalian instrumentasi, kalibrasi, dan jaringan komunikasi di bidang meteorologi, klimatologi, dan geofisika.
- Koordinasi dan kerja sama instrumentasi, kalibrasi, dan jaringan komunikasi di bidang meteorologi, klimatologi, dan geofisika.
- Pelaksanaan pendidikan dan pelatihan keahlian dan manajemen pemerintahan di bidang meteorologi, klimatologi, dan geofisika.
- Pelaksanaan pendidikan profesional di bidang meteorologi, klimatologi, dan geofisika.
- Pelaksanaan manajemen data di bidang meteorologi, klimatologi, dan geofisika.
- Pembinaan dan koordinasi pelaksanaan tugas administrasi di lingkungan BMKG.

- Pengelolaan barang milik/kekayaan negara yang menjadi tanggung jawab BMKG.
- Pengawasan atas pelaksanaan tugas di lingkungan BMKG.
- Penyampaian laporan, saran, dan pertimbangan di bidang meteorologi, klimatologi, dan geofisika.

2.4 PostgreSQL

PostgreSQL merupakan salah satu sistem basis data relasional *open-source* yang ada. Terdapat beberapa basis data *open-source* yang sering digunakan oleh pengembang aplikasi, Tabel 2.1 akan memperlihatkan nama aplikasi manajemen basis data *open-source* dengan peringkatnya per bulan Mei 2017 [8].

Tabel 2.1 Nama Aplikasi Manajemen Basis Data *Open Source*

Peringkat	Nama
1	MySQL
2	PostgreSQL
3	MongoDB
4	Cassandra
5	Redis

PostgreSQL sudah berjalan lebih dari 15 tahun dan sudah terbukti memiliki arsitektur yang dapat diandalkan, mempunyai integritas data yang tinggi, dan tingkat kesalahan yang rendah [9].

Basis data ini dapat bekerja pada semua sistem operasi yang tersedia seperti Linux, Unix, dan Windows. PostgreSQL memiliki kemampuan penuh untuk mendukung perintah-perintah SQL seperti:

1. *Foreign Key*
2. *Joins*
3. *Views*
4. *Triggers*
5. *Stored Procedures*

dan juga memiliki dukungan terhadap tipe data SQL:2008. Basis data PostgreSQL juga memiliki kemampuan untuk menyimpan *object* data biner yang besar seperti video, gambar, dan suara.

Beberapa perusahaan besar sudah menggunakan PostgreSQL sebagai sistem basis datanya [9], diantaranya adalah:

1. Apple
2. Fujitsu
3. Cisco
4. Sun Microsystems
5. Dan lain-lain

2.5 PostGIS

PostGIS adalah ekstensi basis data spasial untuk PostgreSQL. PostGIS menambahkan fitur untuk menjalankan *query* berbasis lokasi pada SQL [10]. PostGIS menyediakan beberapa fitur [11], yaitu:

1. Fungsi yang dapat memroses dan menganalisis untuk vektor dan data *raster* untuk memperkecil, membentuk, mengklasifikasikan dan mengumpulkan/menggabungkan dengan kemampuan SQL.
2. Peta *raster* aljabar untuk proses secara mendetail.
3. Fungsi SQL spasial untuk data vektor dan *raster*.
4. Mendukung memasukkan/menghasilkan ESRI *shapefile* data vektor melalui *commandline* dan *GUI* dan mendukung lebih banyak format dengan aplikasi pihak ketiga yang *open source*.
5. *Command-line* untuk memasukkan data *raster* dari banyak format standar: GeoTiff, NetCDF, PNG, JPG, dan lain-lain.
6. Fungsi me-*render* dan memasukkan data vektor untuk format standar seperti KML, GML, GeoJSON, GeoHash dan WKT menggunakan SQL.
7. Me-*render* data raster dalam berbagai macam format standar seperti GeoTIFF, PNG, JPG, NetCDF, dan lain-lain dengan menggunakan SQL.

8. Fungsi SQL untuk *raster*/vektor ekstrusi yang mulus dari nilai *pixel* dengan area geometri, menjalankan status berdasarkan geometri, memotong *raster* berdasarkan geometri, dan memvektorkan banyak *raster*.
9. Mendukung objek 3D, indeks spasial, dan fungsinya.
10. Mendukung Topologi Jaringan.
11. Paket Tiger Loader / Geocoder / Reverse Geocoder / mengutilisasi US Census Tiger Data.

2.6 pgRouting

pgRouting merupakan ekstensi dari PostGIS / PostgreSQL. Dengan adanya pgRouting, PostgreSQL mampu menyimpan basis data spasial yang mana dapat diolah untuk membuat rute perjalanan [12].

Kelebihan dari basis data rute perjalanan adalah:

1. Data dan atribut dapat dimodifikasi oleh banyak aplikasi klien, seperti QGIS, dan uDig melalui JDBC, ODBC, atau langsung melalui pl/pgSQL. Aplikasi klien dapat berupa aplikasi PC ataupun aplikasi perangkat bergerak.
2. Perubahan data dapat direfleksikan secara langsung melalui mesin/skema penentuan rute. Tidak diperlukan perhitungan diawal.
3. Parameter *Cost* dapat secara dinamis dihitung melalui SQL dan nilainya dapat berasal dari banyak tabel dan kolom.

Selain kelebihan diatas, pgRouting memiliki *library* yang mempunyai fitur:

1. All Pairs Shortest Path, Johnson's Algorithm
2. All Pairs Shortest Path, Floyd-Warshal Algorithm
3. Shortest Path A*
4. Bi-directional Dijkstra Shortest Path

5. Bi-directional A* Shortest Path
6. Shortest Path Dijkstra
7. Driving Distance
8. K-Shortest Path, Multiple Alternatives Paths
9. K-Dijkstra, One to Many Shortest Path
10. Travelling Sales Person
11. Turn Restriction Shortest Path (TRSP)

2.7 *Hypertext Preprocessor (PHP)*

PHP: Hypertext Preprocessor adalah bahasa skrip yang dapat ditanamkan atau disisipkan ke dalam HTML. PHP banyak digunakan untuk memrogram web dinamis. PHP dapat digunakan untuk membangun sebuah CMS.

Kelebihan PHP dari Bahasa pemrograman web yang lain:

1. Bahasa pemrograman PHP adalah sebuah bahasa skrip yang tidak melakukan sebuah kompilasi dalam penggunaannya.
2. Web Server yang mendukung PHP mudah didapatkan seperti apache, IIS, Lighttpd, dan Xitami dengan konfigurasi yang relative murah.
3. Dalam sisi pemahaman, PHP adalah bahasa skrip yang paling mudah karena memiliki referensi yang banyak.
4. PHP adalah bahasa *open source* yang dapat digunakan di berbagai mesin (Linux, Unix, Macintosh, Windows) dan dapat dijalankan perintah-perintah sistem [13].

2.8 Mapbox

Mapbox didirikan oleh sebuah tim kecil di Washington DC pada tahun 2010. semenjak didirikan, Mapbox telah banyak digunakan oleh perusahaan contohnya adalah Foursquare pada tahun 2012, dan USA Today ketika dilaksanakannya pemilihan umum pada tahun 2012. Pada tahun 2014 Mapbox merilis Android SDK yang dapat diimplementasikan pada perangkat Android [14].

2.8.1 Mapbox Android SDK

Mapbox Android SDK sudah dirilis sejak bulan Mei tahun 2014. Untuk menggunakan Android SDK dari Mapbox, diperlukan minimal API 15, oleh karena itu dalam App Manifest di kode program perlu ditambahkan

```
<uses-sdk android:minSdkVersion="15"
          android:targetSdkVersion="integer"
          android:maxSdkVersion="integer" />
```

Kode Program 2.2 Kode App Manifest Pada Android

Mapbox Android SDK menyediakan beberapa fitur yang dapat digunakan pada aplikasi Android yang ingin diciptakan [15], yaitu

1. *Geocoding API & autocompletion*
2. *Direction API & route display*
3. *Navigation SDK*
4. *Static Maps API & Android integration*
5. *Geospatial analysis functionality*, diadaptasi dari projek Turf
6. *Map Matching API*
7. *Line Simplification*

2.9 Application Programming Interface (API)

Application Program Interface (API) adalah kumpulan dari rutinitas, protocol dan alat untuk membangun aplikasi perangkat lunak. Sebuah API menspesifikasikan cara komponen aplikasi saling berinteraksi. API yang bagus mempermudah proses pengembangan aplikasi dengan menyediakan semua blok-blok pembuatan. Seorang programmer akan menggabungkan blok-blok tersebut [16].

2.10 JavaScript Object Notation (JSON)

JSON adalah format pertukaran data yang sangat ringan. Data mudah dibaca dan ditulis oleh manusia. Data mudah dipecah dan dibuat oleh mesin. JSON dibuat berdasarkan subset dari Bahasa Pemrograman JavaScript. JSON adalah format tulisan yang independen tetapi masih menggunakan konvensi yang familiar untuk programmer Bahasa C beserta keluarganya, termasuk C, C++, C#, Java, JavaScript, Perl, Python, dan banyak lagi. Sifat tersebut menyebabkan JSON sebagai jenis pertukaran data yang ideal antar bahasa [17].

JSON mendukung 2 tipe struktur data yang paling sering digunakan [18], yaitu:

1. Koleksi nama/nilai yang berpasangan
Berbagai macam bahasa pemrograman mendukung tipe struktur data ini dalam berbagai nama, seperti objek, baris, *struct*, *dictionary*, tabel *hash*, *keyed list*, atau *associative array*.
2. Daftar nilai yang terurut
Dalam berbagai bahasa pemrograman, sering dipanggil sebagai *array*, *vector*, *list*, atau *sequence*.

2.11 Geography JSON (GeoJSON)

GeoJSON adalah format untuk berbagai jenis struktur data geografi. Sebuah objek GeoJSON dapat merepresentasikan geometri,

fitur, dan koleksi dari fitur-fitur. GeoJSON mendukung tipe geometri *Point*, *LineString*, *Polygon*, *MultiPoint*, *MultiLineString*, *MultiPolygon*, dan *GeometryCollection*. Fitur di dalam GeoJSON berisi sebuah objek geometri dan properti tambahan, serta sebuah koleksi fitur yang merepresentasikan sebuah daftar dari banyak fitur.

Sebuah GeoJSON yang utuh selalu dalam bentuk objek. Di dalam GeoJSON, sebuah objek terdiri atas sebuah koleksi dari nama-nilai yang sepasang, yang juga disebut sebagai anggota. Dalam setiap anggota, nama akan selalu dalam tipe *string*. Nilai dari anggota dapat berupa *string*, *number*, *object*, *array*, dan salah satu dari: *true*, *false*, dan *null* [19]. Gambar 2.3 akan menampilkan contoh GeoJSON.

```
{  
  "type": "Feature",  
  "geometry": {  
    "type": "Point",  
    "coordinates": [125.6, 10.1]  
  },  
  "properties": {  
    "name": "Dinagat Islands"  
  }  
}
```

Gambar 2.13 Contoh GeoJSON

Dilihat dari Gambar 2.3, GeoJSON yang dijadikan contoh adalah GeoJSON dengan tipe Feature, dan tipe Geometry berupa Point, dan terdapat nilai koordinat 125.6 , 10.1. Selain itu pada GeoJSON di atas disertakan juga properti name berupa Dinagat Islands.

2.12 Osm2pgrouting

Osm2pgrouting adalah perangkat yang ditulis oleh Daniel Wendt. Perangkat ini berfungsi untuk memasukkan data peta Openstreetmap ke dalam basis data spasial [20, hal. 2]. Fitur dari Osm2pgRouting adalah [20]:

1. Menggunakan konfigurasi XML untuk memilih tipe jalan dan kelas yang akan dimasukkan.
2. Membuat tipe dan tabel kelas, yang mana membantu menciptakan nilai fungsi *cost* yang mutakhir.

Untuk menggunakan Osm2pgRouting terdapat beberapa persyaratan, yaitu:

1. PostgreSQL
2. PostGIS
3. pgRouting

Cara penggunaan dari *tools* Osm2pgrouting dapat dilihat pada Kode Program 2.3.

```
./osm2pgrouting -file your-OSM-XML-File.osm \
                 -conf mapconfig.xml \
                 -dbname routing \
                 -user postgres \
                 -clean
```

Kode Program 2.3 Penggunaan Osm2pgrouting

(Halaman ini sengaja dikosongkan)

BAB III

PERANCANGAN MODUL

Bab ini menjelaskan tentang perancangan dan pembuatan modul penentuan rute pada aplikasi Clearroute. Modul yang dibuat pada Tugas Akhir ini adalah diawali dari persiapan data peta yang akan digunakan (mengunduh peta, dan mengolah data peta), menggunakan data peta tersebut pada Clearroute API yang akan memroses data tersebut agar dapat dihasilkan rute perjalanan, dan yang terakhir adalah pembuatan Clearroute API, dimana semua implementasi kode program untuk dapat menghasilkan rute perjalanan, dibungkus menjadi satu kesatuan REST API agar dapat digunakan oleh klien aplikasi Clearroute yang merupakan aplikasi Android. Untuk memperjelas setiap perancangan di atas, akan disertakan gambar dan diagram alur.

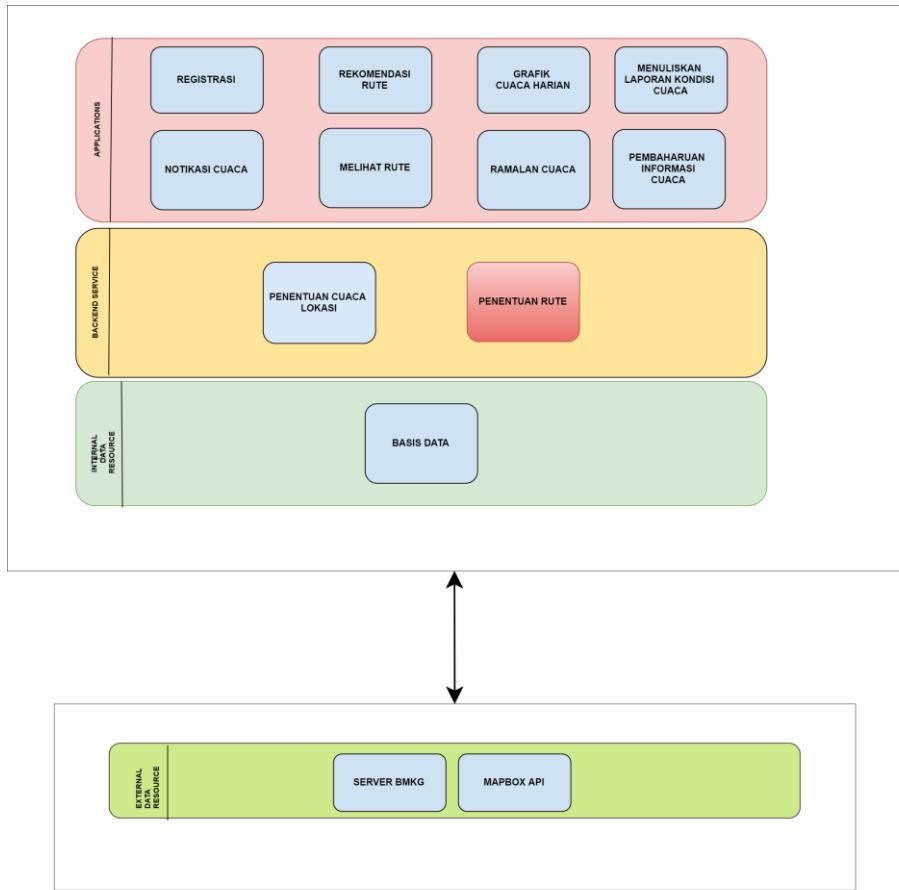
3.1 Deskripsi Umum

Pada sub bab ini akan ditampilkan dan dijelaskan diagram dasar dari aplikasi Clearroute, serta keterkaitannya dengan modul yang dibuat pada Tugas Akhir ini. Jenis arsitektur yang digunakan pada aplikasi Clearroute adalah *N-Tiers architecture*, dimana aplikasi dibagi menjadi beberapa komponen tergantung dengan sumber dan fungsionalnya pada aplikasi Clearroute.

3.1.1 Arsitektur Aplikasi Clearroute

Gambar 3.1 menampilkan gambar diagram keseluruhan dari aplikasi Clearroute. Terdapat 3 komponen utama yaitu *Application*, *Backend-Service*, *Internal Data Resource*, dan *External Data*

Resource. Posisi modul ini dapat dilihat pada kotak yang berwarna merah di bagian *Backend-Services*.



Gambar 3.1 Diagram Arsitektur Aplikasi Clearroute

3.1.1.1 *Applications*

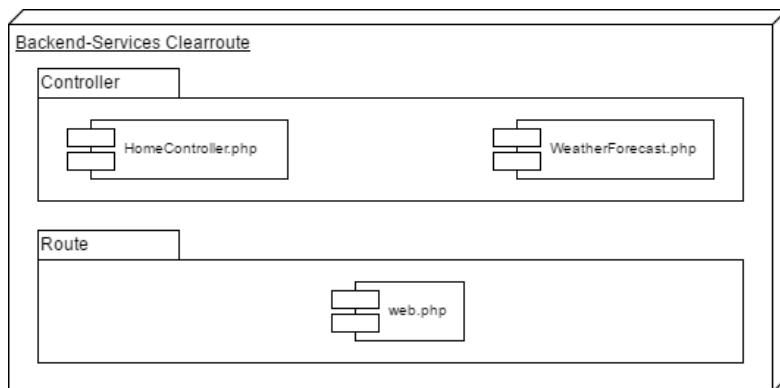
Bagian ini merupakan fitur-fitur yang terdapat pada aplikasi Clearroute itu sendiri. Terdapat fitur

1. Registrasi,
2. Rekomendasi rute,
3. Grafik cuaca harian,
4. Menuliskan laporan kondisi cuaca,
5. notifikasi cuaca,
6. pembaharuan informasi cuaca.

Pada Tugas Akhir akan dibangun sebuah sistem pengolahan data agar fitur rekomendasi rute dapat berjalan dengan baik. Fokus Tugas Akhir ini adalah pada bagian pengembangan *backend services* dan pengolahan data pada *internal data resources* untuk menunjang jalannya aplikasi.

3.1.1.2 *Backend-Services*

Bagian ini merupakan sistem yang akan mengurus 2 hal utama yang akan dikerjakan oleh server Clearroute, yaitu penentuan cuaca lokasi, dan penentuan rute perjalanan.



Gambar 3.2 Diagram Komponen *Backend-Services*

Gambar 3.2 menjelaskan isi dari bagian *Backend-Services* pada aplikasi Clearroute. Pada modul penentuan rute, semua fungsi yang berperan untuk memperoleh rute yang diinginkan terdapat pada file HomeController.php dalam komponen *Controller*, dan alamat untuk mendapatkan hasil dari rute tersebut diatur dalam file web.php dalam komponen *Route*.

3.1.1.3 *Internal Data Resources*

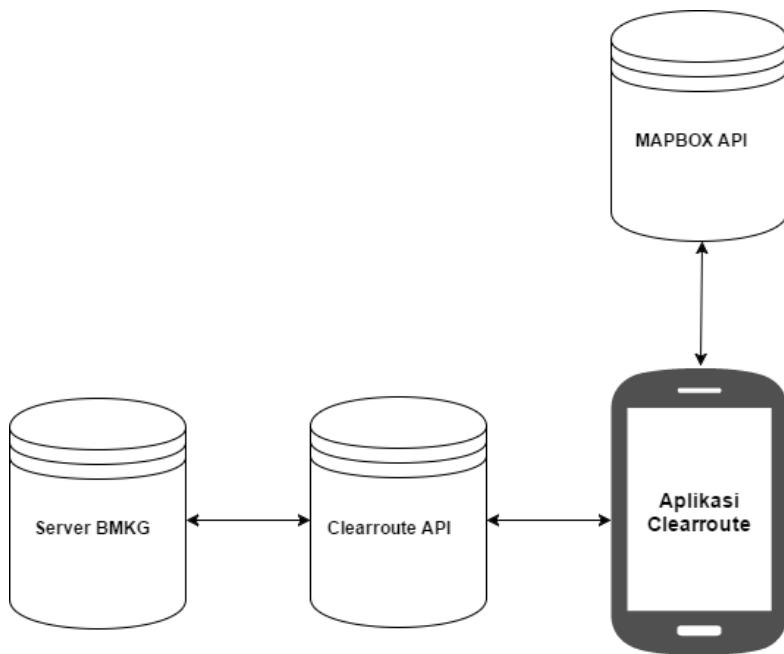
Bagian ini adalah bagian yang mengatur basis data yang akan digunakan oleh aplikasi Clearroute. Jenis basis data yang digunakan adalah PostgreSQL ditambahkan dengan ekstensi PostGIS dan pgRouting agar dapat mengolah basis data spasial dan menghasilkan rute perjalanan sesuai dengan tujuan Tugas Akhir.

Lampiran D memperlihat seluruh isi basis data yang akan dibuat agar dapat menyelesaikan tujuan dari Tugas Akhir ini. Semua tabel di atas (kecuali backup_ways, datapos, dan rekaman) adalah tabel yang otomatis dibuat ketika *import* data Openstreetmap ke dalam basis data PostgreSQL.

3.1.1.4 *External Data Resources*

Bagian ini merupakan sumber data luar yang digunakan oleh aplikasi Clearroute. Terdapat 2 sumber data luar yang digunakan, yaitu Data dari server BMKG dan data dari Mapbox API.

Gambar 3.3 menjelaskan agar aplikasi Clearroute dapat berjalan, diperlukan 3 sumber data, sumber data internal yaitu Clearroute API, dan 2 sumber data eksternal yaitu dari Server BMKG untuk ramalan dan kondisi cuaca, serta Mapbox API untuk menggambarkan peta dan rute yang sudah dibuat oleh tugas akhir ini.

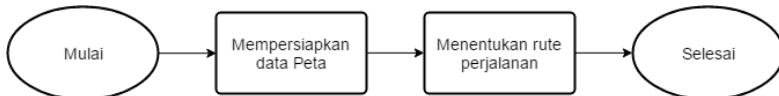


Gambar 3.3 External Data Resources

3.2 Desain Umum Sistem

Pada Sub bab ini, akan dijelaskan mengenai proses perancangan modul penentuan rute perjalanan yang nantinya diimplementasikan pada aplikasi Clearroute.

3.2.1 Diagram Alur Desain Umum Sistem



Gambar 3.4 Diagram Alur Modul Penentuan Rute Perjalanan

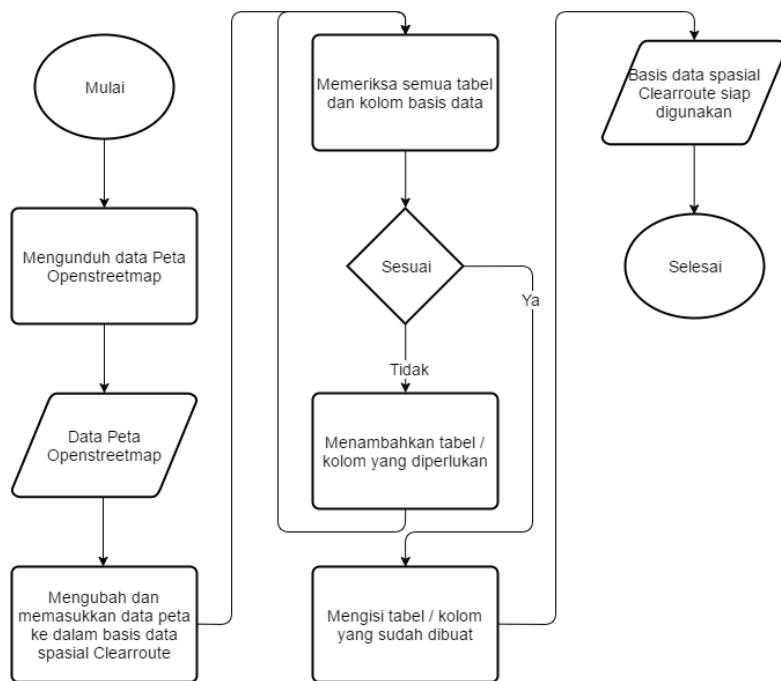
Gambar 3.4 menjelaskan mekanisme dari Modul penentuan rute perjalanan yang akan dibuat. Modul ini dibagi menjadi 2 tahapan, yang pertama adalah mempersiapkan data peta dan yang kedua adalah menggunakan data peta yang sudah siap untuk menentukan rute yang akan ditampilkan pada aplikasi Clearroute.

Dalam mempersiapkan data peta dilakukan beberapa tahapan dari mengunduh data peta yang akan digunakan, mengubah dan memasukkan data peta ke basis data, mengubah/mengolah data peta yang sudah dimasukkan, hingga pada akhirnya basis data spasial Clearroute siap digunakan.

Setelah tahapan mempersiapkan data peta selesai, akan dilakukan tahapan selanjutnya yaitu menentukan rute perjalanan yang akan ditampilkan ke pengguna aplikasi Clearroute. Tahapan menentukan rute perjalanan ini akan dilakukan pada API yang akan dibuat pada *server-side* aplikasi Clearroute.

Ketika tahapan mempersiapkan data peta dan menentukan rute perjalanan telah selesai, tahapan selanjutnya adalah menggunakan rute tersebut pada aplikasi Android Clearroute. Untuk proses secara mendetail dari ketiga tahapan yang disebutkan di atas, dapat dilihat pada penjelasan sub bab selanjutnya.

3.2.1.1 Diagram Alur Persiapan Data Peta



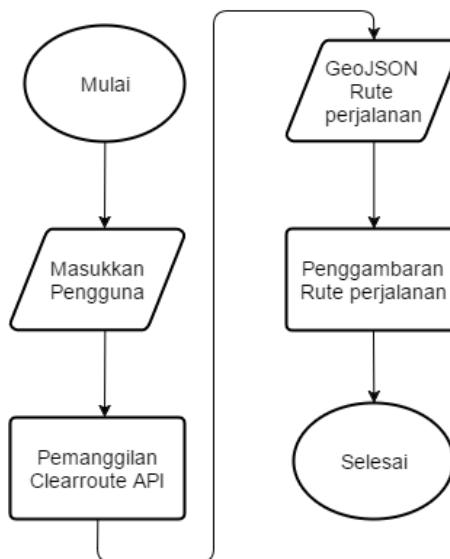
Gambar 3.5 Diagram Alur Persiapan Data

Pada gambar 3.5, terdapat diagram alur yang menjelaskan proses persiapan dari data peta yang akan digunakan untuk menentukan rute perjalanan. Terdapat beberapa langkah yang dilakukan pada tahap ini, yaitu:

1. Tahap paling awal adalah mengunduh data peta OpenStreetMap yang akan digunakan. Pada Tugas Akhir ini, penulis mengunduh data dari situs <http://mapzen.com> dan mengunduh peta Provinsi Jawa Timur.

2. Tahap kedua adalah memasukkan data tersebut ke dalam basis data spasial yang sudah dipersiapkan. *Tools* yang digunakan untuk melakukan tahap ini adalah osm2pgrouting. Osm2pgrouting secara otomatis mengolah data peta yang diunduh agar dapat dimasukkan ke basis data spasial.
3. Tahap ketiga adalah memeriksa tabel yang sudah dibuat oleh Osm2pgrouting. Apabila tabel dan kolom yang dibuat sudah sesuai, maka tidak perlu dilakukan perubahan basis data.
4. Tahap keempat adalah memperbarui nilai tabel / kolom agar sesuai dengan fungsi penentuan rute yang akan dibuat.
5. Apabila semua tahap sudah dilakukan, maka basis data sudah siap untuk digunakan pada fungsi penentuan rute perjalanan.

3.2.1.2 Diagram Alur Penggunaan Data Peta

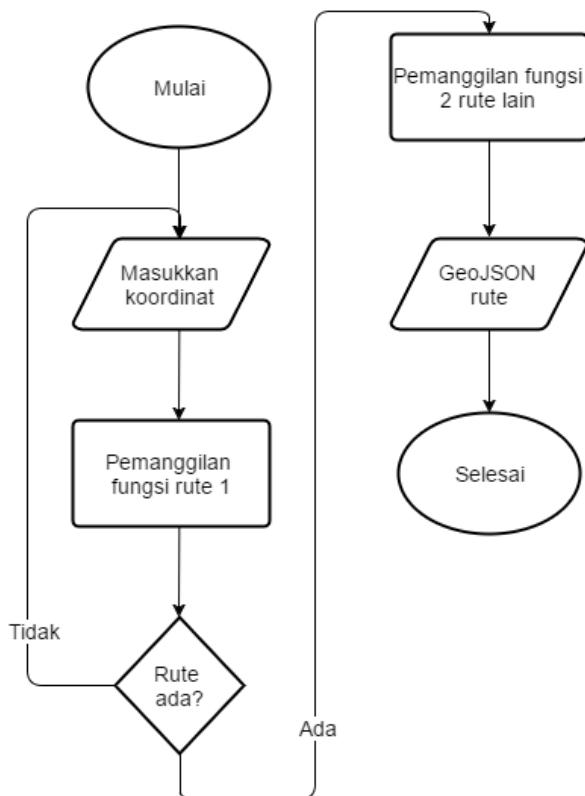


Gambar 3.6 Diagram Alur Penggunaan Data Peta

Gambar 3.6 menjelaskan alur yang dilakukan untuk menggunakan data peta yang sudah dipersiapkan pada tahap persiapan data peta. Terdapat beberapa langkah yang dilakukan pada tahap ini, yaitu:

1. Tahap pertama adalah menerima masukkan dari pengguna aplikasi Clearroute. Data masukkan ini berupa titik koordinat asal dan titik koordinat tujuan.
2. Setelah menerima data masukkan pengguna, data tersebut akan digunakan pada Clearroute API yang sudah dibuat.
3. Clearroute API akan mengembalikan tipe data GeoJSON rute perjalanan yang sudah dibuat.
4. Dengan adanya GeoJSON dari Clearroute API, rute perjalanan dapat digambarkan pada aplikasi Clearroute.

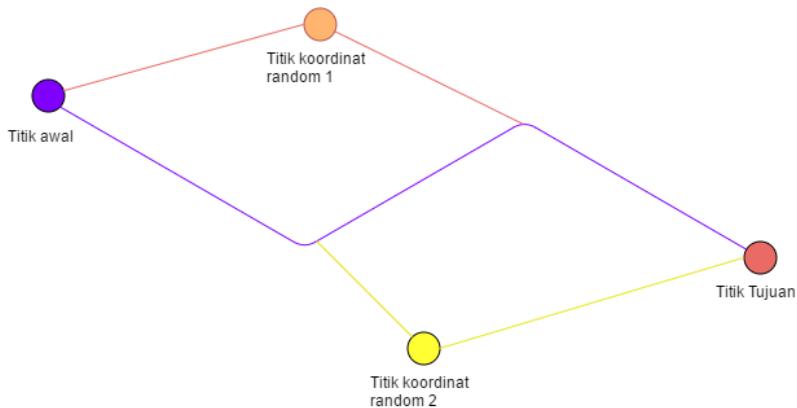
3.2.1.3 Diagram Alur Clearroute API



Gambar 3.7 Diagram Alur Clearroute API

Gambar 3.7 memperlihatkan rancangan API yang akan dibuat agar dapat menghasilkan rute yang diinginkan dengan data peta yang sudah dipersiapkan sebelumnya. Terdapat beberapa langkah pada tahapan ini, yaitu:

1. Tahap pertama adalah menerima masukkan dari pengguna. Masukkan yang diterima adalah berupa titik koordinat asal dan tujuan.
2. Tahap kedua adalah memanggil fungsi yang akan menghasilkan rute utama.
3. Selanjutnya akan diperiksa, apakah rute dapat dibuat atau tidak. Apabila tidak maka pengguna diperlukan untuk mengganti data masukkan.
4. Tahap keempat adalah memanggil fungsi yang akan menghasilkan 2 rute alternatif. 2 rute alternatif ini dihasilkan dengan cara membuat titik baru secara acak, yang mana titik tersebut harus dilalui oleh rute utama. Gambar 3.8 akan menggambarkan titik – titik tersebut.



Gambar 3.8 Pembuatan Rute Alternatif

5. Apabila tidak terdapat kesalahan, maka akan dihasilkan tipe data GeoJSON yang dapat digunakan oleh aplikasi Clearroute untuk menggambarkan rute perjalanan.

(Halaman ini sengaja dikosongkan)

BAB IV

IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi dari perancangan yang sudah dilakukan pada bab sebelumnya. Implementasi berupa kode sumber untuk membangun program. Sebelum masuk ke penjelasan implementasi, akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

4.1 Lingkungan Implementasi

Spesifikasi perangkat keras dan perangkat lunak yang digunakan untuk melakukan implementasi dapat dilihat pada Tabel 4.1.

Tabel 4.1 Spesifikasi Perangkat Keras dan Perangkat Lunak

Perangkat	Jenis Perangkat	Spesifikasi
Perangkat Keras	Prosesor	Intel(R) Core(TM) i3-2120 CPU @ 3.30GHz
	Memori	4 GB 1600
Perangkat Lunak	Sistem Operasi	Ubuntu Xenial 16.04 LTS
	Perangkat Pengembang	Apache Server 2.0

4.2 Implementasi

Pada sub bab implementasi akan menjelaskan bagaimana pembangunan perangkat lunak secara detail dan menampilkan kode sumber serta langkah-langkah yang dilakukan pada semua tahapan pembuatan Modul penentuan rute perjalanan untuk Aplikasi Clearroute.

4.2.1 Implementasi Persiapan Data Peta

Sesuai dengan perancangan yang sudah dilakukan, tahapan awal pada pembuatan modul ini adalah mempersiapkan data peta yang nantinya akan digunakan untuk menentukan rute perjalanan.

Dalam pencarian data peta, penulis mengunduh data Openstreetmap dari situs <http://mapzen.com>. Peta yang diunduh adalah peta Provinsi Jawa Timur. Penulis mengunduh peta dalam format .osm agar dapat diolah pada tahap selanjutnya.



Gambar 4.1 Situs mapzen.com

Setelah proses unduh selesai, penulis memasukkan data peta Openstreetmap dengan menggunakan *tools* Osm2pgsql. Langkah – langkah untuk memasukkan data Openstreetmap ke basis data adalah sebagai berikut:

1. Membuat basis data PostgreSQL pada *server*.

```
createdb db_clearroute
```

**Kode Program 4.1 Membuat basis data
db_clearroute**

2. Menambahkan *extension* postgis dan pgRouting pada basis data yang sudah dibuat.

```
psql --dbname db_clearroute -c 'CREATE EXTENSION postgis'
psql --dbname db_clearroute -c 'CREATE EXTENSION pgRouting'
```

Kode Program 4.2 Menambahkan ekstensi postgis dan pgRouting

3. Menjalankan *program* Osm2pgRouting untuk memasukkan data peta ke basis data yang sudah dibuat.

```
osm2pgrouting --f jawa_timur.osm --conf mapconfig.xml --dbname db_clearroute --username postgres --clean
```

Kode Program 4.3 Menjalankan tools Osm2pgRouting untuk memasukkan data peta ke basis data

4. Setelah sukses, maka basis data yang dibuat akan berisi data peta yang sudah diunduh sebelumnya. Beberapa tabel (nama dan tipe dapat dilihat pada bagian perancangan) akan otomatis dibuat.

Data peta yang sudah ada di dalam basis data masih perlu dianalisa untuk meminimalisir *error*. Langkah untuk menganalisa data tersebut adalah sebagai berikut.

1. Menjalankan fungsi *pgr_analyzeGraph*.

```
select pgr_analyzeGraph('ways', 0.001,
id:='gid');
```

Kode Program 4.4 Fungsi pgr_analyzeGraph

Fungsi ini memiliki beberapa parameter yang dapat/harus diisi. Berikut keterangannya.

```
varchar pgr_analyzeGraph(text edge_table,
double precision tolerance, text
the_geom:='the_geom', text id:='id', text
source:='source',text
target:='target',text rows_where:='true')
```

Kode Program 4.5 Penjelasan parameter fungsi pgr_analyzeGraph

Parameters

The analyze graph function accepts the following parameters:

edge_table:	<code>text</code> Network table name. (may contain the schema name as well)
tolerance:	<code>float8</code> Snapping tolerance of disconnected edges. (in projection unit)
the_geom:	<code>text</code> Geometry column name of the network table. Default value is <code>the_geom</code> .
id:	<code>text</code> Primary key column name of the network table. Default value is <code>id</code> .
source:	<code>text</code> Source column name of the network table. Default value is <code>source</code> .
target:	<code>text</code> Target column name of the network table. Default value is <code>target</code> .
rows_where:	<code>text</code> Condition to select a subset or rows. Default value is <code>true</code> to indicate all rows.

Gambar 4.2 Parameter fungsi pgr_analyzeGraph

Setelah selesai menganalisa data yang ada di dalam basis data, penulis kembali memeriksa tabel dan kolom yang sudah dibuat oleh *tools* Osm2pgRouting, dan ternyata masih diperlukan 1 kolom baru untuk menyimpan nilai *cost* yang dapat menampung nilai *cost* gabungan antara perempatan, dan waktu tercepat sampai sesuai dengan batasan modul penentuan rute perjalanan pada aplikasi Clearroute. Demi keamanan dan kelancaran, penulis menggandakan tabel *vertice* yang sudah dibuat oleh *tools* Osm2pgRouting.

```
CREATE TABLE backup_ways AS  
TABLE ways;
```

Kode Program 4.7 Membuat tabel duplikat

Pada tabel hasil duplikat, akan dibuat kolom baru, berikut kode program yang akan dijalankan:

```
ALTER TABLE backup_ways ADD  
cost_clearroute DOUBLE PRECISION;
```

Kode Program 4.6 Menambahkan kolom cost_clearroute

Cost_clearroute adalah kolom cost yang nantinya akan digunakan sebagai parameter untuk menjalankan fungsi algoritma A Star. Pada Tugas Akhir ini, cost_clearroute adalah modifikasi dari kolom cost_s bawaan Osm2pgRouting yang dibuat berdasarkan kecepatan dan jarak ke lokasi tujuan.

Untuk mengisi kolom cost_clearroute, dibuatlah sebuah fungsi yang diberi nama updateCostClearroute, kode program dari fungsi tersebut dapat dilihat pada Kode Program 4.8:

```

CREATE OR REPLACE FUNCTION
updateCostFix(i INTEGER) RETURNS INTEGER
AS $$

DECLARE
    sql_gid text;
    rec record;

BEGIN
    sql_gid := 'SELECT * FROM
backup_ways';
    FOR rec IN EXECUTE sql_gid
        LOOP
            EXECUTE 'UPDATE
backup_ways SET cost_clearroute = (SELECT
cost_s FROM backup_ways WHERE gid =
'||rec.gid||') WHERE gid =
'||rec.gid||';
        END LOOP;

    RETURN i+1;
END;
$$
language plpgsql volatile STRICT;

```

Kode Program 4.8 Fungsi UpdateCostClearroute

Fungsi UpdateCostClearroute hanya mengisi kolom cost_clearroute dengan menduplikasi isi kolom cost_s yang sudah dibuat oleh Osm2pgrouting. Kolom cost_s merupakan cost tiap jalur dengan berpatokan dengan kecepatan dan jarak tempuh yang harus dilalui, oleh karena itu penulis menggunakan kolom ini acuan dalam pembuatan cost baru.

Selain dari jarak dan waktu tempuh, penulis juga membutuhkan cost untuk setiap koordinat perempatan yang ada di Surabaya. Untuk mengetahui titik-titik perempatan yang harus

dihindari oleh rute perjalanan, penulis melihat data perempatan dari situs <http://dishub.surabaya.go.id/index.php/post/id/1551>. Untuk detil koordinat lokasi perempatan yang dihindari, dapat dilihat pada bagian lampiran. Daftar koordinat tersebut dapat dilihat pada Lampiran C dan Lampiran E.

Setelah mengetahui koordinat perempatan yang harus dihindari, akan dijalankan kode program pada Lampiran 1 sesuai dengan koordinat yang ada.

Langkah terakhir dalam persiapan data peta, adalah membuat rute perjalanan agar menghindari jalan-jalan kecil di daerah perumahan, dan lebih memprioritaskan jalan besar. Hal tersebut dilakukan dengan cara menandai tipe jalur yang ingin diprioritaskan dan yang ingin diabaikan. Kode program 4.9 akan memperlihatkan kode yang melakukan hal tersebut.

1	UPDATE osm_way_classes SET penalty=2.0 WHERE
2	class_id = 112;
3	UPDATE osm_way_classes SET penalty=1.5 WHERE
4	class_id = 110;
5	UPDATE osm_way_classes SET penalty=0.8 WHERE
6	class_id = 109;
7	UPDATE osm_way_classes SET penalty=0.5 WHERE
8	class_id = 124;
9	UPDATE osm_way_classes SET penalty=0.5 WHERE
10	class_id = 108;

Kode Program 4.9 Memberi Penalti Pada Jalur yang Ingin Dihindari

Sebagai penutup dari persiapan data peta, dilakukan beberapa langkah seperti *indexing*, *vacuuming*, *clustering*, serta *analyzing* pada tabel backup_ways, agar pemrosesan basis data dapat menjadi lebih cepat. Langkah ini dilakukan sesuai dengan saran dari halaman web <http://revenant.ca/www/postgis/workshop/indexing.html>. Kode

program untuk menjalankan hal tersebut dapat dilihat pada Kode Program 4.10.

```

1  CREATE INDEX percepat_akses ON backup_ways (gid,
2    source, target);
3  CREATE INDEX percepat_akses_geom ON backup_ways
4    USING GIST (the_geom);
5
6  VACUUM ANALYZE backup_ways;
7
8  CLUSTER backup_ways USING percepat_akses_geom;
9
10 ANALYZE backup_ways;
```

Kode Program 4.10 Mempercepat Akses Basis Data

4.2.2 Implementasi Penggunaan Data Peta

Setelah data peta berhasil dimasukan ke basis data, maka data tersebut sudah dapat diakses melalui *client* (aplikasi Android Clearroute). Sesuai dengan rancangan yang sudah di jelaskan pada bab sebelumnya, aplikasi Clearroute memerlukan masukkan berupa koordinat awal dan koordinat tujuan. Untuk dapat memperoleh koordinat tujuan, dapat dilihat pada kode program 4.11.

```

1  private void getLocation()
2  {
3      //memanggil servis lokasi android
4      manager = (LocationManager)
5      getSystemService(LOCATION_SERVICE);
6
7      //pengecekan apakah gps sudah diaktifkan
8      if
9      (!manager.isProviderEnabled(LocationManager.GPS_PROVIDER)
10     )) {
11         //pemanggilan fungsi aktifasi gps
```

```
13         buildAlertMessageNoGps();
14     }
15
16     //pengecekan perijinan akses lokasi
17     if
18     (ActivityCompat.checkSelfPermission(getApplicationContext(),
19      Manifest.permission.ACCESS_FINE_LOCATION)
20 != PackageManager.PERMISSION_GRANTED &&
21     ActivityCompat.checkSelfPermission(getApplicationContext(),
22      Manifest.permission.ACCESS_COARSE_LOCATION)
23 != PackageManager.PERMISSION_GRANTED) {
24         return;
25     }
26     try{
27         //mengambil lokasi dari gps
28         mLastLocation =
29     manager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
30
31         if (mLastLocation == null){
32             //mengambil lokasi dari jaringan layanan
33             telepon
34             mLastLocation =
35     manager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
36
37             if(mLastLocation==null)
38             {
39                 //mengambil lokasi terbaru lewat gps
40
41             manager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
42             0, 0, (LocationListener) this);
43             mLastLocation =
44             manager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
45
46             }
47             }
48             //debug latitude longitude
49             else if (mLastLocation != null)
50             Log.d("Location : ","Lat = "+
51             mLastLocation.getLatitude() + " Lng");
52         }catch (Exception e)
```

```

53     {
54         //apabila gagal mendapat lokasi
55         Log.d("Gagal lokasi terbaru", "fail");
56     }
57 }
```

Kode Program 4.11 Mendapatkan koordinat awal

Langkah selanjutnya setelah mendapatkan koordinat awal adalah, memperlihatkan lokasi koordinat awal yang sudah didapat, pada peta. Untuk penggambaran peta pada aplikasi Clearroute, digunakan Mapbox API sebagai layanan yang dapat memperlihatkan peta secara *online*. Kode program 4.12 akan memperlihatkan cara penggambaran peta dengan menggunakan Mapbox API.

```

1 mapbox.getMapAsync(new OnMapReadyCallback() {
2     @Override
3     public void onMapReady(MapboxMap mapboxMap) {
4         map = mapboxMap;
5
6         map.moveCamera(CameraUpdateFactory.newCameraPosition(
7             new CameraPosition.Builder()
8                 .target(new
9                     LatLng(mLastLocation.getLatitude(),
10                    mLastLocation.getLongitude())))
11                 .zoom(15)
12                 .tilt(15)
13                 .build()
14             ));
15
16
17         MarkerViewOptions markerViewOptions = new
18 MarkerViewOptions()
19             .position(new
20             LatLng(mLastLocation.getLatitude(),
21             mLastLocation.getLongitude()));
22
23         map.addMarker(markerViewOptions);
```

```

24     asal_lokasi = new
25     LatLng(mLastLocation.getLatitude(),
26     mLastLocation.getLongitude());
27   }
28 });

```

Kode Program 4.12 Penggambaran Peta Koordinat Awal

Langkah selanjutnya setelah mendapatkan koordinat awal adalah mendapatkan koordinat tujuan dari pengguna aplikasi. Untuk mendapatkan koordinat tujuan, penulis menggunakan fungsi GeocoderAutoCompleteView yang ada pada Mapbox API agar dapat mendapatkan nama – nama tempat/jalan yang akan dituju oleh pengguna aplikasi Clearroute. Kode program 4.13 akan memperlihatkan cara menggunakan fungsi tersebut.

```

1 // Deklarasi fungsi Geocoder MapboxAPI
2 GeocoderAutoCompleteView autoComplete =
3 (GeocoderAutoCompleteView)
4 findViewById(R.id.autocomplete);
5
6 // Memanggil akses token mapbox api
7 autoComplete.setAccessToken(MapboxAccountManager.get
8 Instance().getAccessToken());
9
10 autoComplete.setType(GeocodingCriteria.TYPE_POI);
11 autoComplete.setOnFeatureListener(new
12 GeocoderAutoCompleteView.OnFeatureListener() {
13     @Override
14     public void OnFeatureClick(CarmenFeature
15     feature) {
16         Position position = feature.asPosition();
17         //mendapatkan koordinat tujuan
18         target_lokasi = new
19         LatLng(position.getLatitude(),
20         position.getLongitude());
21

```

```

22
23     //memperbaharui peta agar menampilkan lokasi
24     tujuan
25         updateMap(position.getLatitude(),
26 position.getLongitude());
27
28     //memanggil fungsi pencarian rute
29     getRoute(asal_lokasi, target_lokasi);
30 }
31 });

```

Kode Program 4.13 Penggunaan fungsi GeocoderAutoCompleteView pada Aplikasi Clearroute

Apabila sudah mendapatkan koordinat awal dan koordinat tujuan, aplikasi Clearroute sudah dapat memanggil fungsi untuk penentuan dan penggambaran rute perjalanan. Sesuai dengan rancangan penggunaan data peta, fungsi tersebut memanggil Clearroute API yang sudah dibuat dan menerima *file* GeoJSON sebagai nilai kembalian. Isi dari *file* GeoJSON dapat dilihat pada Lampiran F. Pemanggilan Clearroute API melalui metode REST dibantu dengan *library* Retrofit. Kode program 4.14 akan memperlihatkan cara kerja penggambaran rute yang sudah didapatkan dari Clearroute API.

```

1  private void getRoute(LatLng latLngAsal, LatLng
2  latLngTujuan)
3  {
4      //pemanggilan url dari Clearroute API
5      StringBuilder urlbaru = new
6      StringBuilder("http://riset.alpro.if.its.ac.id/cle
7      arroute/public/index.php/getroutecuaca1/");
8
9      //memanggil longitude asal
10     urlbaru.append(latLngAsal.getLongitude()+"\"");

```

```
12 //memanggil latitude asal
13 urlbaru.append(latLngAsal.getLatitude() + "/");
14
15 //memanggil longitude tujuan
16 urlbaru.append(latLngTujuan.getLongitude() + "/");
17
18 //memanggil latitude asal
19 urlbaru.append(latLngTujuan.getLatitude() + "/");
20
21 //deklarasi fungsi retrofit
22 Retrofit retrofit = new Retrofit.Builder()
23     .baseUrl("http://clearroute.net")
24
25     .addConverterFactory(GsonConverterFactory.create())
26 )
27         .build();
28
29
30     GetRoute service =
31 retrofit.create(GetRoute.class);
32     Call<JsonElement> call =
33 service.getRoute(urlbaru.toString());
34     Log.d("url= ", urlbaru.toString());
35
36     call.enqueue(new Callback<JsonElement>() {
37         @Override
38             public void onResponse(Call<JsonElement>
39 call, Response<JsonElement> response) {
40                 //menerima response ke dalam Json
41                 Element
42                     JsonElement jsonElement =
43 response.body();
44                     JsonObject jsonArray =
45 jsonElement.getAsJsonObject();
46
47                     //mendapatkan rute utama
48                     jsonArray jsonArray1 =
49 jsonArray.get("0").getAsJSONArray();
50
51 
```

```
52          //mendapatkan rute alternatif 1
53          JSONArray jsonArray2 =
54      jsonArray.get("1").getAsJSONArray();
55
56          //mendapatkan rute alternatif 2
57          JSONArray jsonArray3 =
58      jsonArray.get("2").getAsJSONArray();
59
60          //perulangan penggambaran rute utama
61          for(int a = 0; a < jsonArray1.size();
62      a++)
63          {
64              try {
65                  JSONObject jsonObject = new
66      JSONObject(jsonArray1.get(a).getAsString()).get
67      ("json").getAsString();
68                  JSONObject jsonObjectgeometry
69      = new
70      JSONObject(jsonObject.get("geometry").toString());
71                  String string_geocoordinates =
72      jsonObjectgeometry.get("coordinates").toString();
73
74                  JSONArray array_geocoordinate
75      = new JSONArray(string_geocoordinates);
76                  for(int i=0;
77      i<array_geocoordinate.length(); i++)
78                  {
79                      JSONArray coord =
80      array_geocoordinate.getJSONArray(i);
81                      LatLng latLng = new
82      LatLng(coord.getDouble(1), coord.getDouble(0));
83                      latLngs.add(latLng);
84                  }
85                  map.addPolyline(new
86      PolylineOptions()
87          .addAll(latLngs)
88          .color(Color.parseColor("#3bb2d0"))
89          .width(4));
90
91      
```

```
92             latLngs.clear();
93         } catch (JSONException e) {
94             e.printStackTrace();
95         }
96     }
97
98     //perulangan penggambaran rute
99 alternatif 1
100    for(int a = 0; a < jsonArray2.size();
101        a++)
102    {
103        try {
104            JSONObject jsonobject = new
105            JSONObject(jsonArray2.get(a).getAsString());
106            JSONObject jsonobjectgeometry
107            = new
108            JSONObject(jsonobject.get("geometry").toString());
109            String string_geocoordinates =
110            jsonobjectgeometry.get("coordinates").toString();
111
112            JSONArray array_geocoordinate
113            = new JSONArray(string_geocoordinates);
114            for(int i=0;
115                i<array_geocoordinate.length(); i++)
116            {
117                JSONArray coord =
118                array_geocoordinate.getJSONArray(i);
119                LatLng latLng = new
120                LatLng(coord.getDouble(1), coord.getDouble(0));
121                latLngs.add(latLng);
122            }
123            map.addPolyline(new
124            PolylineOptions()
125            .addAll(latLngs)
126            .color(Color.parseColor("#757575"))
127            .width(4));
128            latLngs.clear();
129        } catch (JSONException e) {
```

```
132             e.printStackTrace();
133         }
134     }
135
136     //perulangan penggambaran rute
137 alternatif 2
138     for(int a = 0; a < jsonArray3.size();
139 a++)
140     {
141         try {
142             JSONObject jsonobject = new
143 JSONArray(jsonArray3.get(a).getAsString()).get
144 ("json").getAsString();
145             JSONObject jsonobjectgeometry
146 = new
147 JSONArray(jsonobject.get("geometry").toString());
148             String string_geocoordinates =
149 jsonobjectgeometry.get("coordinates").toString();
150
151             JSONArray array_geocoordinate
152 = new JSONArray(string_geocoordinates);
153             for(int i=0;
154 i<array_geocoordinate.length(); i++)
155             {
156                 JSONArray coord =
157 array_geocoordinate.getJSONArray(i);
158                 LatLng latLng = new
159 LatLng(coord.getDouble(1), coord.getDouble(0));
160                 latLngs.add(latLng);
161             }
162             map.addPolyline(new
163 PolylineOptions()
164                     .addAll(latLngs)
165
166 .color(Color.parseColor("#757575"))
167                     .width(4));
168             latLngs.clear();
169         } catch (JSONException e) {
170             e.printStackTrace();
171         }
```

```

172         }
173     }
174
175     @Override
176     public void onFailure(Call<JsonElement>
177     call, Throwable t) {
178         Log.d("gagal response", t.toString());
179     });
}

```

Kode Program 4.14 Fungsi Penggambaran Rute Perjalanan

4.2.3 Implementasi Clearroute *Application Programming Interface* (API)

Clearroute API adalah REST API yang dibuat oleh penulis agar fungsi penentuan rute perjalanan dapat diakses oleh *client*, yang mana *client* dari API ini adalah aplikasi Android Clearoute. Sesuai dengan perancangan yang sudah dibuat, pada Implementasi Clearroute API terdapat beberapa langkah yang perlu dilakukan.

Langkah pertama adalah mempersiapkan fungsi pada PostgreSQL yang nantinya fungsi tersebut akan digunakan pada Clearroute API. Diperlukan 2 fungsi dalam modul penentuan perjalanan, yang pertama adalah fungsi untuk menentukan rute utama, dan fungsi untuk menentukan rute alternatif.

Fungsi untuk menentukan rute utama dibuat berdasarkan kode program yang didapat dari github dengan alamat https://github.com/Zia-/pgr_aStarFromAtoBviaC dan untuk penjelasan kodennya dapat dilihat pada kode program 4.15

```
1  create or replace function pgr_normalroute(IN tbl
2    character varying,
3    variadic double precision[],
4    OUT seq integer,
5    OUT gid integer,
6    OUT name text,
7    OUT cost double precision,
8    OUT geom geometry,
9    OUT x double precision,
10   OUT y double precision)
11  RETURNS SETOF record AS
12  $body$
13  declare
14  arrayLengthHalf integer;
15  a integer;
16  x1 double precision;
17  b integer;
18  y1 double precision;
19  sql_node text;
20  REC_ROUTE record;
21  source_var integer;
22  target_var integer;
23  node record;
24  sql_astar text;
25  rec_astar record;
26 begin
27 -- menghapus tabel sementara apabila sudah ada
28 drop table if exists tmp;
29 -- membuat tabel sementara
30 create temporary table tmp(id integer, node_id
31 integer, x double precision, y double precision);
32 -- mendefinisikan ukuran array
33 -- ($2, 1) berarti parameter kedua dan array nya
34 merupakan array 1 dimensi
35 arrayLengthHalf = (array_length($2,1))/2;
36 -- Untuk perulangan sesuai dengan tabel yg dibuat,
37 index 0 diabaikan dan dimulai dari 2[1]
38 For i in 1..arrayLengthHalf Loop
39   a := i*2-1;
40   x1 := $2[a];
```

```

42 b := a+1;
43 y1 := $2[b];
44 -- Memasukkan node id yang didapat dari query di
45 bawah, ke dalam tabel sementara
46 execute 'insert into tmp (id, node_id, x, y)
47 select'||i||', id, st_x(the_geom)::double
48 precision, st_y(the_geom)::double precision
49 from ways_vertices_pgr ORDER BY the_geom <->
50 ST_GeometryFromText(''Point(''||x1||' ||y1|| '')'', 
51 4326) limit 1;';
52 End Loop;
53
54 sql_node := 'SELECT * FROM tmp';
55 -- Mengambil kolom geom dari tabel sementara
56 seq := 0;
57 source_var := -1;
58 FOR REC_ROUTE IN EXECUTE sql_node
59 LOOP
60 -- Mengecek apakah parameter merupakan koordinat
61 awal
62 If (source_var = -1) Then
63 execute 'select node_id from tmp where node_id =
64 '||REC_ROUTE.node_id||' into node';
65 source_var := node.node_id;
66 -- Apabila parameter merupakan koordinat tujuan
67 Else
68 execute 'select node_id from tmp where node_id =
69 '||REC_ROUTE.node_id||' into node';
70 target_var := node.node_id;
71 sql_astar := 'SELECT b.gid, a.cost, b.the_geom,
72 b.name, b.source, b.target, b.x1 AS x, b.y1 AS y
73 FROM ||
74 pgr_astar('''SELECT gid::integer AS id,
75 source::integer, target::integer, ' ||
76 'cost_clearroute * penalty::double precision AS
77 cost, reverse_cost_s * penalty::double precision
78 AS reverse_cost, x1, y1, x2, y2 FROM '
79 || quote_ident(tbl) || ' AS r JOIN
80 osm_way_classes USING (class_id), (SELECT
81 ST_Expand(ST_Extent(the_geom),0.1) as box FROM

```

```

82 backup_ways as l1 WHERE l1.source = ' || 
83 source_var || ' OR l1.target = ' || target_var || 
84 ') as box
85 WHERE r.the_geom && box.box'', '
86 || source_var || ',' || target_var || ', true,
87 true) AS a LEFT JOIN ' || quote_ident(tbl) || ' AS
88 b ON (a.id2 = b.gid) ORDER BY a.seq
89 ';
90 -- Menjalankan fungsi algoritma A star pada
91 pgrouting, dan mengembalikan hasilnya
92 For rec_astar in execute sql_astar
93 Loop
94 seq := seq +1 ;
95 gid := rec_astar.gid;
96 name := rec_astar.name;
97 cost := rec_astar.cost;
98 geom := rec_astar.the_geom;
99 x := rec_astar.x;
100 y := rec_astar.y;
101 RETURN NEXT;
102 End Loop;
103 END IF;
104 END LOOP;
105 return;
106
107 --EXCEPTION
108 --WHEN internal_error THEN
109 --seq := seq +1 ;
110 --gid := rec_astar.gid;
111 --name := rec_astar.name;
112 --cost := 9999.9999;
113 --geom := rec_astar.the_geom;
114
115 end;
116 $body$
117 language plpgsql volatile STRICT;

```

Kode Program 4.15 Fungsi Penentuan Rute Perjalanan Utama

Pada kode program 4.15 diperlihatkan keseluruhan fungsi yang akan menghasilkan rute perjalanan dengan menggunakan algoritma A Star yang disediakan oleh pgRouting. Pada baris 15 – 26 adalah deklarasi variabel yang akan digunakan pada fungsi ini. Baris 29 – 32 adalah pengecekan dan pembuatan tabel sementara yang akan menyimpan id untuk digunakan pada perulangan selanjutnya. Pada baris 39 – 52 adalah langkah dimana memasukkan hasil pencarian id node ke dalam tabel sementara. Baris 54 adalah menampilkan semua hasil yang sudah dimasukkan ke dalam tabel sementara. Untuk lebih jelas mengenai fungsi algoritma A Star itu sendiri, kode program 4.16 dan tabel 3 akan memperlihatkan secara lebih terperinci.

```

1  SELECT a.seq AS seq, b.gid AS gid, b.name AS name,
2    a.cost AS cost, b.the_geom AS geom, b.source,
3    b.target, b.x1 AS x, b.y1 AS y FROM pgr_astar('
4      SELECT gid::integer AS id,
5        source::integer,
6        target::integer,
7        cost_clearroute * penalty::double precision AS
8        cost,
9        reverse_cost_s * penalty::double precision AS
10       reverse_cost,
11       x1, y1, x2, y2
12     FROM backup_ways JOIN osm_way_classes USING
13       (class_id), (SELECT
14         ST_Expand(ST_Extent(the_geom),0.1) as box FROM
15         backup_ways as l1 WHERE l1.source = Node_id_awal
16         OR l1.target = node_id_tujuan) as box
17       WHERE r.the_geom && box.box'', Node_id_awal,
18         node_id_tujuan, true, true) AS a LEFT JOIN
19         backup_ways AS b ON (a.id2 = b.gid) ORDER BY
20         a.seq;
21

```

Kode Program 4.16 Fungsi Algoritma Astar pgRouting

Tabel 4.2 Daftar parameter fungsi Algoritma Astar pgRouting

Kolom	Tipe	Deskripsi
Id	ANY-INTEGER	Identitas edge
Source	ANY-INTEGER	Identitas vertex awal
Target	ANY-INTEGER	Identitas vertex tujuan
Cost	ANY-NUMERICAL	Nilai dari edge (source dan target). Apabila negatif maka tidak termasuk dalam graph
Reverse_cost	ANY-NUMERICAL	Nilai dari edge (target dan source). Apabila negatif maka tidak termasuk dalam graph
X1	ANY-NUMERICAL	Koordinat X dari vertex awal
Y1	ANY-NUMERICAL	Koordinat Y dari vertex awal
X2	ANY-NUMERICAL	Koordinat X dari vertex tujuan
Y2	ANY-NUMERICAL	Koordinat Y dari vertex tujuan

Dalam penggunaan *library* algoritma A Star pgRouting diperlukan beberapa parameter yaitu id *node* asal (source), id *node* tujuan (target), nilai *cost* (*cost_clearroute*), nilai *reverse_cost* (*reverse_cost_s*) yang dijadikan patokan penentuan rute, keterangan satu arah atau tidak, dan nilai *reverse_cost*. Pada implementasi algoritma A Star untuk pencarian rute perjalanan, dilakukan

pembatasan area pencarian rute, agar algoritma tidak perlu mencari rute keseluruhan data peta yang ada di basis data. Hal tersebut dilakukan dengan cara membatasi area pencarian dengan menggunakan acuan titik awal dan titik tujuan, dan dilebarkan sebanyak 0.1 derajat. Fungsi tersebut dapat dilihat pada baris 15-18.

Fungsi kedua setelah penentuan rute utama perjalanan, adalah penentuan rute alternatif yang dapat dilalui oleh pengguna aplikasi Clearroute. Penentuan rute alternatif ini dibuat dengan cara mencari titik koordinat random yang harus dilalui oleh rute perjalanan selain rute utama yang sudah dibuat. Fungsi ini juga dibuat berdasarkan github yang sama seperti fungsi penentuan rute utama perjalanan. Kode program 4.17 akan memperlihatkan kode fungsi secara keseluruhan.

```

1  create or replace function
2  pgr_aStarFromAtoBviaC_line(IN tbl character
3  varying,
4  variadic double precision[],,
5  OUT seq integer,
6  OUT gid integer,
7  OUT name text,
8  OUT cost double precision,
9  OUT geom geometry,
10 OUT x double precision,
12 OUT y double precision)
13 RETURNS SETOF record AS
14 $body$
15 declare
16 arrayLengthHalf integer;
17 a integer;
18 x1 double precision;
19 b integer;
20 y1 double precision;
21 sql_tsp text;
22 rec_tsp record;
23 source_var integer;
24 target_var integer;
25 node record;
```

```

26 sql_astar text;
27 rec_astar record;
28 begin
29 -- menghapus tabel sementara apabila sudah ada
30 drop table if exists matrix;
31
32 -- membuat tabel sementara
33 create temporary table matrix(id integer, node_id
34 integer, x double precision, y double precision);
35
36 -- mendefinisikan ukuran array
37 -- ($2, 1) berarti parameter kedua dan array nya
38 merupakan array 1 dimensi
39 arrayLengthHalf = (array_length($2,1))/2;
40
41 -- Untuk perulangan sesuai dengan tabel yg dibuat,
42 index 0 diabaikan dan dimulai dari 2[1]
43 For i in 1..arrayLengthHalf Loop
44 a := i*2-1;
45 x1 := $2[a];
46 b := a+1;
47 y1 := $2[b];
48 -- Memasukkan node id yang didapat dari query di
49 bawah, ke dalam tabel sementara
50 execute 'insert into matrix (id, node_id, x, y)
51 select'||i||', id, st_x(the_geom)::double
52 precision, st_y(the_geom)::double precision from
53 ways_vertices_pgr ORDER BY the_geom <->
54 ST_GeometryFromText(''Point('||x1|| '||y1||' ''),
55 4326) limit 1;';
56 End Loop;
57
58 -- mengkalkulasikan node yang ada dengan algoritma
59 TSP, agar sesuai dengan urutan jaraknya
60 sql_tsp := 'select seq, id1, id2,
61 round(cost::numeric, 5) AS cost from
62 pgr_tsp(''select id, x, y from matrix order by
63 id'', 1, '||arrayLengthHalf||')';
64
65 -- mengambil kolom the geom

```

```

66 seq := 0;
67 source_var := -1;
68 FOR rec_tsp IN EXECUTE sql_tsp
69 LOOP
70 -- Mengecek apakah parameter merupakan koordinat awal
71 If (source_var = -1) Then
72 execute 'select node_id from matrix where id =
73'||rec_tsp.id2||' into node;
74 source_var := node.node_id;
75 -- Apabila parameter merupakan koordinat tujuan
76 Else
77 execute 'select node_id from matrix where id =
78'||rec_tsp.id2||' into node;
79 target_var := node.node_id;
80 sql_astar := 'SELECT b.gid, a.cost, b.the_geom,
81 b.name, b.source, b.target, b.x1 AS x, b.y1 AS y
82 FROM ' ||
83 'pgr_astar(''SELECT gid::integer AS id,
84 source::integer, target::integer, ' ||
85 'cost_clearroute * penalty::double precision AS
86 cost, reverse_cost_s * penalty::double precision
87 AS reverse_cost, x1, y1, x2, y2 FROM '
88 || quote_ident(tbl) || ' AS r JOIN
89 osm_way_classes USING (class_id), (SELECT
90 ST_Expand(ST_Extent(the_geom),0.1) as box FROM
91 backup_ways as l1 WHERE l1.source = ' ||
92 source_var || ' OR l1.target = ' || target_var ||
93 ') as box
94 WHERE r.the_geom && box.box'','
95 || source_var || ',' || target_var || ', true,
96 true) AS a LEFT JOIN ' || quote_ident(tbl) || ' AS
97 b ON (a.id2 = b.gid) ORDER BY a.seq';
98 For rec_astar in execute sql_astar
99 Loop
100 seq := seq +1 ;
101 gid := rec_astar.gid;
102 name := rec_astar.name;
103 cost := rec_astar.cost;
104 geom := rec_astar.the_geom;
105

```

```

106 x := rec_astar.x;
107 y := rec_astar.y;
108 RETURN NEXT;
109 End Loop;
110 source_var := target_var;
111 END IF;
112 END LOOP;
113 return;
114
115 --EXCEPTION
116 --WHEN internal_error THEN
117 --seq := seq +1 ;
118 --gid := rec_astar.gid;
119 --name := rec_astar.name;
120 --cost := 9999.9999;
121 --geom := rec_astar.the_geom;
122
123 end;
124 $body$
125 language plpgsql volatile STRICT;

```

Kode Program 4.17 Fungsi penentuan rute alternatif

Pada fungsi tersebut terdapat perbedaan dengan fungsi penentuan rute utama, yaitu dipanggilnya fungsi TSP pgRouting. Fungsi ini digunakan agar titik koordinat random yang digunakan sebagai acuan rute alternatif, dapat diurutkan sesuai dengan jarak terdekatnya dengan titik awal dan hanya dikunjungi sekali. Kode program 4.18 akan memperlihatkan isi dari fungsi TSP pgRouting.

```

1 sql_tsp := 'select seq, id1, id2,
2 round(cost::numeric, 5) AS cost from
3 pgr_tsp('''select id, x, y from
4 matrix order by id''', 1,'||arrayLengthHalf||')';

```

Kode Program 4.18 Fungsi TSP pgRouting

Tabel 4.3 Daftar Parameter fungsi TSP pgRouting

Kolom	Tipe	Deskripsi
Matrix_sql	Query	Hasil query fungsi sebelumnya
Start_id	BIGINT	Identitas id titik awal
End_id	BIGINT	Identitas id titik akhir

Langkah selanjutnya setelah membuat fungsi sql pada basis data PostgreSQL adalah membuat fungsi di dalam Clearroute API yang dapat memanggil fungsi dari basis data, dan pada akhirnya mampu memberikan nilai kembali ke *client* berupa *file* GeoJSON. Fungsi dibuat pada bahasa pemrograman PHP, dengan menggunakan kerangka kerja Laravel.

Penulis membuat 5 fungsi pada Clearroute API yaitu, penghasil rute perjalanan utama, menghasilkan nilai angka acak untuk koordinat, menghasilkan batas koordinat x baru, menghasilkan batas koordinat y baru, dan yang terakhir fungsi penghasil rute perjalanan alternatif. Untuk memanggil rute utama kode program 4.19 akan memperlihatkan kode yang digunakan.

1	\$array[0] = DB::SELECT("SELECT
2	jsonb_build_object('type', 'Feature',
3	'properties', '{}', 'geometry',
4	ST_AsGeoJSON(geom)::jsonb) AS json FROM (SELECT *
5	FROM pgr_normalroute('backup_ways',
6	". \$x1.", ". \$y1.", ". \$x3.", ". \$y3.")) AS row WHERE
7	row.gid IS NOT NULL");
8	

Kode Program 4.19 Fungsi penghasil rute perjalanan utama

Pada kode program 4.19, diperlihatkan hasil dari *query* yang dijalankan akan membuat suatu objek dalam format GeoJSON. Hasil dari objek tersebut disimpan dalam suatu variabel array. Parameter yang digunakan untuk memanggil fungsi ini adalah koordinat awal (x1 dan y1) dan koordinat tujuan (x3 dan y3).

Fungsi kedua yang dibuat oleh penulis adalah fungsi untuk menghasilkan titik koordinat acak. Fungsi tersebut dapat dilihat pada kode program 4.20.

```

1  function randomFloat($min = 0, $max = 1) {
2      return $min + mt_rand() / mt_getrandmax()
3      * ($max - $min);
4  }

```

Kode Program 4.20 Fungsi menghasilkan nilai random untuk koordinat

Penulis membuat fungsi untuk menentukan nilai acak ini. Karena apabila hanya dengan menggunakan fungsi bawaan *mt_rand()* yang ada pada PHP, tidak bisa didapatkan nilai acak yang tingkat ketelitiannya sesuai dengan titik koordinat, yang mana titik koordinat memiliki ketelitian hingga 5 digit di belakang koma. Fungsi ini mengambil parameter nilai koordinat awal, dan koordinat akhir.

Fungsi ketiga dan keempat yang dibuat adalah fungsi untuk menghasilkan batasan titik koordinat baru (*latitude* dan *longitude*) yang selanjutnya nilai batasan ini akan digunakan oleh fungsi sebelumnya (*randomFloat*).

```

1  function randomLatitude($latitude)
2  {
3      $number_asli = $latitude;
4      //mengalikan latitude asli dengan -1,
5      Karena posisi geografis indonesia
6      $number = $number_asli*-1;
7

```

```

8      //mengambil angka di depan koma
9      $int_number = floor($number);
10
11     //mengambil angka di belakang koma
12     $decimal = $number - $int_number;
13
14     //mengacak kemungkinan, apakah koordinat
15     digeser ke atas, atau ke bawah
16     if(mt_rand(1,10)>5)
17     {
18         $decimal_batas_baru = $decimal*1.1;
19         $batas_baru =
20         ($int_number+$decimal_batas_baru)*-1;
21         }
22         else
23         {
24             $decimal_batas_baru = $decimal*1.1;
25             $decimal_batas_baru =
26             $decimal_batas_baru - $decimal;
27             $batas_baru = ($int_number+($decimal-
28             $decimal_batas_baru))*-1;
29             }
30
31         return $batas_baru;
32     }
33
34     function finalLatitude($latitude_awal,
35     $latitude_akhir)
36     {
37         //memanggil fungsi koordinat acak
38         return $this->randomFloat($latitude_awal,
39         $this->randomLatitude($latitude_akhir));
40     }

```

Kode Program 4.21 Fungsi menentukan batasan dan koordinat y baru

1	function randomLongitude(\$longitude)
2	{

```
3      $number_asli = $longitude;
4      $number = $number_asli;
5      $int_number = floor($number);
6
7      //mengambil angka di belakang koma
8      $decimal = $number - $int_number;
9
10     //mengacak kemungkinan koordinat digeser
11     ke kanan atau ke kiri
12     if(mt_rand(1,10)>5)
13     {
14         //digeser ke kanan
15         $decimal_batas_baru = $decimal*1.005;
16         $batas_baru =
17         ($int_number+$decimal_batas_baru);
18     }
19     else
20     {
21         //digeser ke kiri
22         $decimal_batas_baru = $decimal*1.005;
23         $decimal_batas_baru =
24         $decimal_batas_baru - $decimal;
25         $batas_baru = ($int_number+($decimal-
26         $decimal_batas_baru));
27     }
28
29     return $batas_baru;
30 }
31
32 function finalLongitude($longitude_awal,
33 $longitude_akhir)
34 {
35     //memanggil fungsi koordinat acak
36     return $this->randomFloat($longitude_awal,
37     $this->randomLongitude($longitude_akhir));
38 }
```

Kode Program 4.22 Fungsi menentukan batasan dan koordinat x baru

Kode program 4.21 dan 4.22 menampilkan fungsi yang dapat menghasilkan batasan koordinat baru, yang hasilnya akan digunakan oleh fungsi sebelumnya (randomFloat) yang dapat menghasilkan koordinat acak (dapat dilihat pada baris 38 dan 36). Sesuai dengan perancangan yang dibuat, batasan koordinat didapatkan dengan cara menggeser koordinat asli (0.1 untuk *latitude*, dan 0.005 untuk *longitude*). Nilai tersebut didapatkan dengan cara *trial and error* agar tidak terjadi batasan yang didapatkan terlalu jauh dari koordinat aslinya.

Fungsi terakhir yang dibuat adalah fungsi yang dapat memanggil rute perjalanan alternatif yang didapat dari fungsi di dalam basis data.

```

1 //dilakukan perulangan 2x Karena dibutuhkan 2 rute
2 alternatif
3 for($i=1; $i<=2; $i++)
4     {
5         //pencarian koordinat acak x
6         $x2 = $this->finalLongitude($x3, $x1);
7
8         //pencarian koordinat acak y
9         $y2 = $this->finalLatitude($y3, $y1);
10        $array[$i] = DB::SELECT("SELECT
11 jsonb_build_object(
12             'type',          'Feature',
13             'properties',   '{}',
14             'geometry',
15             ST_AsGeoJSON(geom)::jsonb
16             ) AS json FROM (SELECT * FROM
17             pgr_aStarFromAtoBviaC_line('backup_ways',
18             ".$x1.", ".$y1.", ".$x2.", ".$y2.", ".$x3.", ".$y3."))"
19             AS row WHERE row.gid IS NOT NULL");
20
21 }
```

Kode Program 4.23 Fungsi penghasil rute perjalanan alternatif

Pada kode program 4.23 diperlihatkan fungsi yang dapat menghasilkan rute perjalanan alternatif. Fungsi tersebut dijalankan sebanyak 2 kali, Karena diperlukan 2 rute alternatif dalam Tugas Akhir ini. Dapat dilihat pada baris 6 dan 9, fungsi ini memanggil fungsi penghasil koordinat acak yang sudah dibuat sebelumnya.

BAB V

UJI COBA DAN EVALUASI

Pada Bab ini akan dilakukan tahap ujicoba dan evaluasi sesuai dengan rancangan dan implementasi modul penentuan rute perjalanan aplikasi Clearroute. Dari hasil yang didapatkan setelah melakukan uji coba, akan dilakukan evaluasi sehingga dapat ditarik kesimpulan pada bab selanjutnya.

5.1 Uji Coba

Pada subbab ini akan dilakukan rangkaian ujicoba sesuai dengan rumusan masalah dan implementasi yang sudah dilakukan pada bab sebelumnya. Hasil dari ujicoba akan dilakukan evaluasi pada subbab selanjutnya.

5.1.1 Mengubah Data Peta dari Openstreetmap ke Basis Data Spasial

Sesuai dengan implementasi dan permasalahan yang ada, uji coba yang pertama dilakukan adalah memasukkan data .osm yang sudah diunduh ke dalam basis data spasial. Kode Program 5.1 akan memperlihatkan sebagian isi data jawa_timur.osm yang sudah diunduh dibuka dengan menggunakan *text editor*.

```
1  <?xml version='1.0' encoding='UTF-8'?>
2  <osm version="0.6" generator="osmconvert 0.8.5"
3  timestamp="2017-02-14T15:02:02Z">
4      <bounds minlat="-8.90678"
5      minlon="110.8630371" maxlat="-6.6536953"
6      maxlon="114.7521972"/>
7          <node id="59073041" lat="-7.9420691"
8          lon="112.9529769" version="9" timestamp="2017-02-
9          13T23:14:09Z" changeset="46064699" uid="99999"
10         user="snodnipper">
11             <tag k="condition" v="active"/>
```

```
12      <tag k="description" v="Active  
13 stratovolcano. Last erupted 2004."/>  
14      <tag k="ele" v="2329"/>  
15      <tag k="is_in" v="Indonesia, Java,  
16 East Java"/>  
17      <tag k="is_in:country_code" v="ID"/>  
18      <tag k="last_eruption" v="2016"/>  
19      <tag k="name" v="Gunung Bromo  
20 (volcano)"/>  
21      <tag k="natural" v="volcano"/>  
22      <tag k="tourism" v="attraction"/>  
23      <tag k="type" v="stratovolcano"/>  
24      <tag k="wikidata" v="Q679590"/>  
25      <tag k="wikipedia" v="en:Mount  
26 Bromo"/>  
27      </node>  
28      <node id="59074119" lat="-7.9479599"  
29 lon="112.9721443" version="1" timestamp="2007-09-  
30 23T16:49:26Z" changeset="505142" uid="308"  
31 user="MichaelCollinson">  
32      <tag k="name" v="Tengger Caldera"/>  
33      <tag k="is_in" v="Indonesia, Java,  
34 East Java"/>  
35      <tag k="natural" v="caldera"/>  
36      <tag k="created_by" v="Potlatch  
37 alpha"/>  
38      </node>  
39      <node id="59075621" lat="-7.6756777"  
40 lon="112.6077362" version="2" timestamp="2009-07-  
41 28T11:34:54Z" changeset="1963803" uid="308"  
42 user="MichaelCollinson">  
43      <tag k="name" v="Tretes"/>  
44      <tag k="is_in" v="Indonesia"/>  
45      <tag k="place" v="town"/>  
46      <tag k="created_by" v="Potlatch  
47 alpha"/>  
48      <tag k="description" v="Old Dutch  
49 hill station"/>  
50      </node>  
51
```

```
52      <node id="59076926" lat="-7.3573387"
53      lon="111.2685893" version="1" timestamp="2007-09-
54      23T16:51:19Z" changeset="505142" uid="308"
55      user="MichaelCollinson">
56          <tag k="name" v="Tretes"/>
57          <tag k="is_in" v="Indonesia"/>
58          <tag k="place" v="town"/>
59          <tag k="created_by" v="Potlatch
60 alpha"/>
61      </node>
62      <node id="116249267" lat="-8.1619165"
63      lon="114.4364434" version="7" timestamp="2013-01-
64      13T10:48:18Z" changeset="14632407" uid="326704"
65      user="Bernhard Hiller"/>
66          <node id="257690830" lat="-6.8054927"
67      lon="110.9108387" version="5" timestamp="2014-10-
68      12T12:32:05Z" changeset="26024039" uid="1705820"
69      user="Bayu Adi Styawan"/>
70          <node id="257690831" lat="-6.8062198"
71      lon="110.9178611" version="4" timestamp="2013-02-
72      04T18:52:18Z" changeset="14913499" uid="338759"
73      user="dawnbreak"/>
74          <node id="257690832" lat="-6.8063815"
75      lon="110.9225603" version="4" timestamp="2014-10-
76      12T12:32:05Z" changeset="26024039" uid="1705820"
77      user="Bayu Adi Styawan"/>
78      ...
79      ...
80      ...
81      <node id="1070165873" lat="-8.022" lon="113.5964"
82      version="1" timestamp="2010-12-29T04:28:49Z"
83      changeset="6794724" uid="151670" user="BlueArrow">
84          <tag k="name" v="Gunung
85          Cemorokandang"/>
86          <tag k="note" v="Modify date: 2010-
87          05-27"/>
88          <tag k="source" v="NGA-GNS"/>
89          <tag k="gns:dsg" v="MT"/>
90          <tag k="natural" v="peak"/>
91          <tag k="is_in:state" v="Jawa Timur"/>
```

```
91           <tag k="is_in:country"
92   v="Indonesia"/>
93           <tag k="is_in:country_code" v="ID"/>
94       </node>
95       <node id="1070165881" lat="-7.6928"
96   lon="112.4432" version="1" timestamp="2010-12-
97 29T04:28:49Z" changeset="6794724" uid="151670"
98 user="BlueArrow">
99           <tag k="name" v="Gunung Orooroombo"/>
100          <tag k="note" v="Modify date: 2008-
101 09-11"/>
102          <tag k="source" v="NGA-GNS"/>
103          <tag k="gns:dsg" v="MT"/>
104          <tag k="natural" v="peak"/>
105          <tag k="is_in:state" v="Jawa Timur"/>
106          <tag k="is_in:country"
107 v="Indonesia"/>
108          <tag k="is_in:country_code" v="ID"/>
109       </node>
110       <node id="1070165978" lat="-8.3376"
111   lon="113.7265" version="2" timestamp="2016-05-
112 03T14:44:47Z" changeset="39067996" uid="3332493"
113 user="nyentrikdotcom">
114           <tag k="name" v="Gunung Sanen"/>
115           <tag k="note" v="Modify date: 2010-
116 03-18"/>
117           <tag k="source" v="NGA-GNS"/>
118           <tag k="gns:dsg" v="HLL"/>
119           <tag k="natural" v="peak"/>
120           <tag k="is_in:state" v="Jawa Timur"/>
121           <tag k="is_in:country"
122 v="Indonesia"/>
123           <tag k="is_in:country_code" v="ID"/>
124       </node>
125       <node id="1070166033" lat="-7.681111"
126   lon="111.090278" version="1" timestamp="2010-12-
127 29T04:28:53Z" changeset="6794724" uid="151670"
128 user="BlueArrow">
129           <tag k="name" v="Bukit Jemowo"/>
130
```

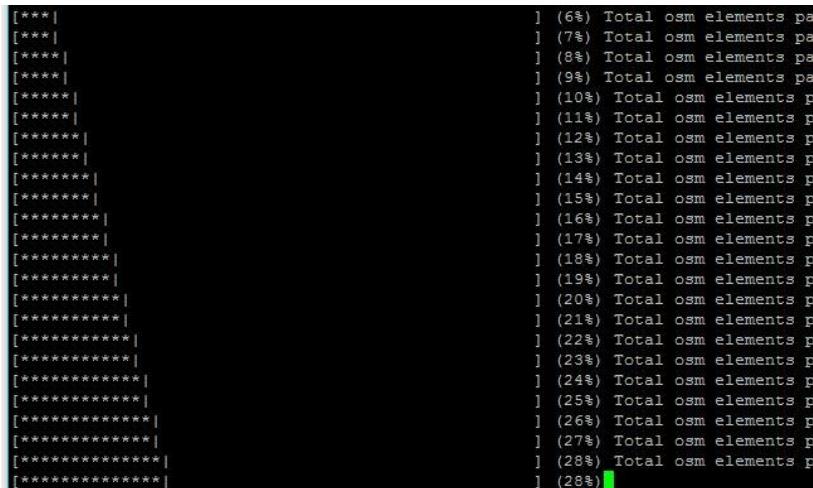
```

131 <tag k="note" v="Modify date: 2007-
132 03-30"/>
133 <tag k="source" v="NGA-GNS"/>
134 <tag k="gns:dsg" v="MT"/>
135 <tag k="natural" v="peak"/>
136 <tag k="is_in:state" v="Jawa
137 Tengah"/>
138 <tag k="is_in:country"
139 v="Indonesia"/>
140 <tag k="is_in:country_code" v="ID"/>
141 </node>
142 <node id="1070166037" lat="-7.9808"
143 lon="113.2549" version="1" timestamp="2010-12-
144 29T04:28:53Z" changeset="6794724" uid="151670"
145 user="BlueArrow">
146 <tag k="name" v="Gunung Melawang"/>
147 <tag k="note" v="Modify date: 2008-
148 09-12"/>
149 <tag k="source" v="NGA-GNS"/>
150 <tag k="gns:dsg" v="MT"/>
151 <tag k="natural" v="peak"/>
152 <tag k="is_in:state" v="Jawa Timur"/>
153 <tag k="is_in:country"
154 v="Indonesia"/>
155 <tag k="is_in:country_code" v="ID"/>
156 </node>
157 ...
...

```

Kode Program 5.1 Isi data jawa_timur.osm

Dengan menggunakan tools Osm2pgrouting, data jawa_timur.osm akan dimasukkan ke basis data spasial yang sudah dibuat dan dipasangkan ekstensi postgis serta pgRouting. Gambar 5.1 – 5.4 akan memperlihatkan proses memasukkan data tersebut ke dalam basis data.



Gambar 5.1 Proses Memasukkan Data jawa_timur 1.osm

```
Creating 'osm_way_classes': OK
Adding auxiliary tables to database...

Export Types ...
    Processing 4 way types:      Inserted: 4 in osm_way_types

Export Classes ...
    Processing way's classes:   Inserted: 36 in osm_way_classes

Export Relations ...
    Processing 1 relations:     Inserted: 1 in osm_relations

Export RelationsWays ...
    Processing way's relations: Inserted: 0 in relations_ways

Export Ways ...
    Processing 1579520 ways:
[|] (1%) Ways Processed: 20
000      Split Ways generated: 42775 Vertices inserted 40277 Inserted 42775 s
plit ways
[*|] (5%) Ways Processed: 80
000      Split Ways generated: 62638 Vertices inserted 52404 Inserted 62637 s
plit ways
[*****|] (11%)
```

Gambar 5.2 Proses Memasukkan Data jawa_timur 2.osm

```

Export Types ...
    Processing 4 way types:      Inserted: 4 in osm_way_types

Export Classes ...
    Processing way's classes:   Inserted: 36 in osm_way_classes

Export Relations ...
    Processing 1 relations:     Inserted: 1 in osm_relations

Export RelationsWays ...
    Processing way's relations: Inserted: 0 in relations_ways

Export Ways ...
    Processing 1579520 ways:
[|] (1%)      Ways Processed: 20
000      Split Ways generated: 42775 Vertices inserted 40277 Inserted 42775 s
split ways
[**|] (5%)      Ways Processed: 80
000      Split Ways generated: 62638 Vertices inserted 52404 Inserted 62637 s
split ways
[*****|] (12%)     Ways Processed: 2
00000      Split Ways generated: 47904 Vertices inserted 36784 Inserted 47900 s
split ways
[*****|] (27%) 

```

Gambar 5.3 Proses Memasukkan Data jawa_timur 3.osm

```

Creating Foreign Keys ...
Foreign keys for osm_way_classes table created
Foreign keys for relations_ways table created
Foreign keys for Ways table created
#####
size of streets: 1579520
#####

```

Gambar 5.4 Proses Memasukkan Data jawa_timur 3.osm

Setelah proses di atas selesai, basis data yang sudah dibuat sebelumnya akan terisi dengan tabel dan fungsi – fungsi yang sudah dibuat secara otomatis oleh Osm2pgsql, Gambar 5.5 – 5.7 dan Lampiran C akan memperlihatkan isi dari basis data tersebut.

	osm_nodes	8.0 KiB
	osm_relations	24.0 KiB
	osm_way_classes	24.0 KiB
	osm_way_types	24.0 KiB

Gambar 5.5 Isi Basis Data

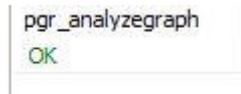
	relations_ways	0 B
	spatial_ref_sys	8.5 MiB
	ways	162.3 MiB
	ways_vertices_pgr	40.8 MiB

Gambar 5.6 Isi Basis Data

	_pgr_checkverttab
	_pgr_createindex
	_pgr_createindex
	_pgr_dijkstra
	_pgr_dijkstra
	_pgr_dijkstra
	_pgr_dijkstra
	_pgr_drivingdistance
	_pgr_drivingdistance
	_pgr_endpoint
	_pgr_getcolumnname
	_pgr_getcolumnname
	_pgr_getcolumntype
	_pgr_getcolumntype
	_pgr_gettablename

Gambar 5.7 Isi Basis Data

Sesuai dengan dokumentasi dari *tools Osm2pgrouting*, isi dari basis data spasial yang sudah dibuat harus diproses lagi dengan menggunakan fungsi `pgr_analyzegraph`, Gambar 5.8 akan menampilkan hasil dari implementasi fungsi tersebut.



Gambar 5.8 Hasil Implementasi Fungsi `pgr_analyzegraph`

5.1.2 Mengimplementasikan Algoritma A Star dengan pgRouting

Dalam penentuan rute perjalanan, terdapat 3 parameter yang dijadikan acuan yaitu jarak perjalanan, waktu tempuh, serta perempatan yang sering terjadi kemacetan. Ketiga parameter ini digabungkan menjadi 1 nilai (`cost_clearroute`) yang mana nilai tersebut akan digunakan pada kode program yang sudah dibuat. Kode Program 5.2 akan memperlihatkan Kode Program yang akan mengimplementasikan algoritma A Star dan Gambar 5.9 akan memperlihatkan beberapa nilai yang ada di kolom `cost_clearroute`

1	SELECT a.seq AS seq, b.gid AS gid, b.name AS name,
2	a.cost AS cost, b.the_geom AS geom, b.source,
3	b.target, b.x1 AS x, b.y1 AS y FROM pgr_dijkstra('
4	SELECT gid::integer AS id,
5	source::integer,
6	target::integer,
7	cost_clearroute::double precision AS
8	cost,
9	reverse_cost::double precision AS
10	reverse_cost,
11	x1, y1, x2, y2
12	FROM backup_ways',
13	116452, 60644, true, true) AS a LEFT JOIN ways
	AS b ON (a.id2 = b.gid) ORDER BY a.seq;

--	--

Kode Program 5.2 Kode Program Pemanggil Algoritma A Star

cost_clearroute
10.6288137477538
6.33017860343852
2.51898752815221
4.76266614809253
4.67647212411752
4.57923999705678
9.70884673947021
13.5470801927378
14.4309359001633
12.5279371486341

Gambar 5.9 Isi Nilai Kolom cost_clearroute

Parameter jarak dan waktu tempuh didapat dari kolom *cost_s* yang dibuat otomatis oleh *tools Osm2pgrouting*. Menurut jawaban dari *link https://gis.stackexchange.com/questions/198200/how-are-cost-and-reverse-cost-computed-in-pgrouting* nilai *cost_s* didapat dari perhitungan kolom *maxspeed* dan kolom *length_m*, oleh karena itu nilai *cost_s* dapat dikatakan mewakilkan parameter jarak dan waktu tempuh. Gambar 5.10 akan memperlihatkan contoh nilai pada kolom *cost_s*.

cost_s
46.5271666055136
5.641871364647
2.62601556072263
6.48473609242768
70.9535348197253
3.9014954128041
14.7193726195668
10.4955666465768
7.43759665247346
21.560624354875
16.6491243981857
37.3958866564003
8.42953137460007
10.2057218090886
92.0808647588126

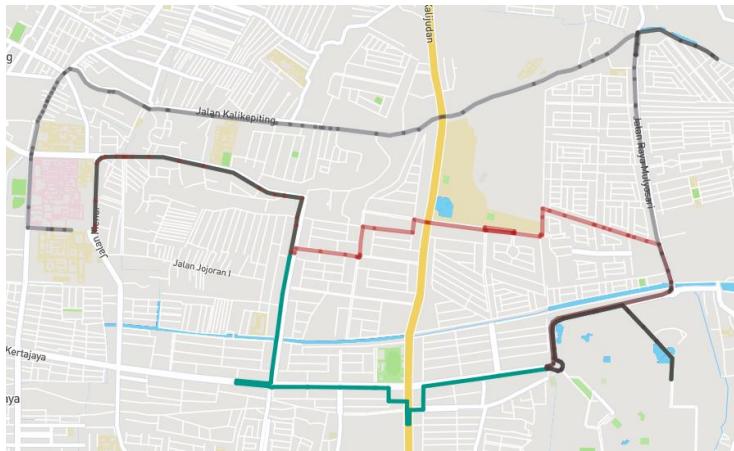
Gambar 5.10 Contoh Nilai Pada Kolom cost_s

Parameter perempatan jalan didapat dengan cara menandai jalur yang memiliki perempatan dengan potensi kemacetan tinggi. Data tersebut didapat dengan melihat data tempat *traffic light* Kota Surabaya dan data kemacetan Kota Surabaya yang di dapat dari Google Maps. Jalur ditandai dengan cara meninggikan nilai *cost* pada jalur tersebut, pada Tugas Akhir ini nilai *cost* diubah menjadi 150. Gambar 5.11 akan memperlihatkan beberapa data jalur yang sudah ditandai.

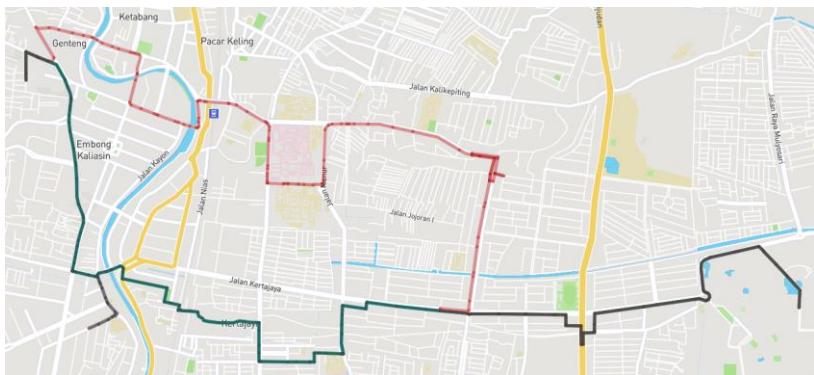
gid	name	x1	y1
160,323	Jalan R.A. Kartini	112.735011	-7.2788032
186,080	Jalan Raya Dukuh Kupang	112.7134233	-7.2902967
234,143	Jalan Karang Tembok	112.7463489	-7.2264229
733	Jalan Sutorejo	112.7829861	-7.2634184
2,697	Jalan Mulyorejo	112.7829861	-7.2634184
110,342	Jalan Blauran	112.7331438	-7.2584095
125,569	Jalan Tidar	112.7275378	-7.2570339
158,616	Jalan Manyar Kertoajo	112.762282	-7.2794478
158,648	Jalan Menur	112.762282	-7.2794478
158,657	Jalan Manyar Kertoarjo	112.7623171	-7.2793127

Gambar 5.11 Jalur yang Ditandai

Untuk menguji rute perjalanan, dilakukan 3x penentuan titik tujuan. Titik tujuan dibagi 3 menjadi dekat, sedang dan jauh. Semua pengujian dilakukan dengan Gedung Teknik Informatika sebagai titik awal perjalanan. Gambar 5.12 akan menampilkan rute perjalanan dengan tujuan Restauran Wapo (112.75994, -7.270502) sebagai tujuan dekat, Gambar 5.13 akan menampilkan rute perjalanan dengan tujuan Tunjungan Plaza (112.7381256, -7.2630531) sebagai tujuan sedang, dan Gambar 5.14 akan menampilkan rute perjalanan dengan tujuan UNESA (112.6691293, -7.301802) sebagai tujuan jauh. Terdapat 3 warna pada rute, warna hijau toska adalah rute utama, merah adalah rute alternatif 1, dan abu-abu adalah rute alternatif 2.



Gambar 5.12 Uji Coba Jarak Dekat



Gambar 5.13 Uji Coba Jarak Sedang



Gambar 5.14 Uji Coba Jarak Jauh

5.1.3 Waktu Respon Server

Pada uji coba perhitungan *response time* kali ini dibuat kriteria pengujian mulai dari jarak perjalanan dan jumlah pengguna yang mengakses. Kriteria pengujian skenario uji coba satu dibuat sebagai berikut:

1. Rute Perjalanan
 - a. Departemen Informatika - Carls'jr Kertajaya (3.4 KM)
 - b. Departemen Informatika - Tunjungan Plaza (8.5 KM)
 - c. Departemen Informatika - Kampus Unesa (17.7 KM)
2. Jumlah Pengguna
 - a. 15 pengguna
 - b. 25 pengguna
 - c. 35 pengguna

Uji coba ini dijalankan selama 1 menit dimana dalam waktu tersebut terdapat jumlah pengguna yang telah ditentukan pada kriteria

melakukan *request* data secara bersamaan. Hasil uji coba ini dapat dilihat pada Tabel 5.1, Tabel 5.2, Tabel 5.3 dan Gambar 5.15 – 5.23.

Tabel 5.1 Uji Coba Waktu Respon 15 Pengguna

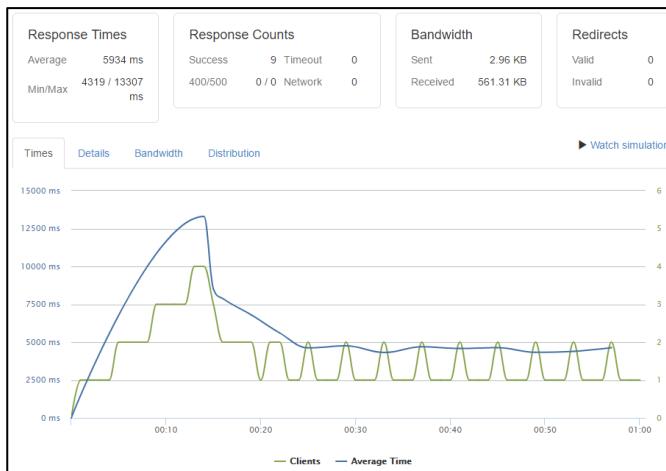
Rute	Waktu minimum	Waktu Maksimum	Rata-Rata
A	4319 ms	13307 ms	5934 ms
B	5304 ms	9354 ms	5790 ms
C	5609 ms	7632 ms	6305 ms

Tabel 5.2 Uji Coba Waktu Respon 25 Pengguna

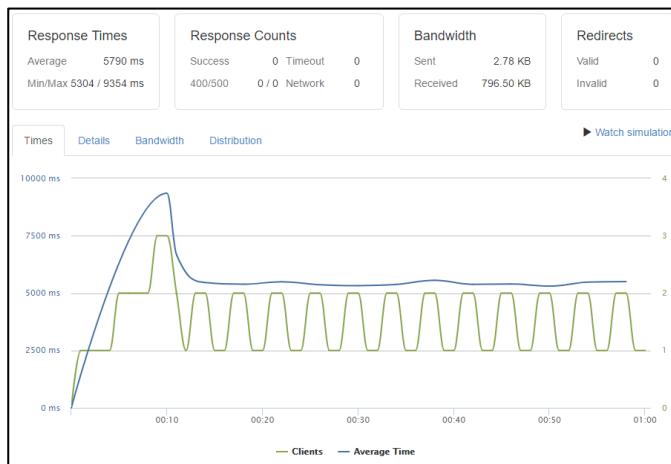
Rute	Waktu minimum	Waktu Maksimum	Rata-Rata
A	5967 ms	13062 ms	8186 ms
B	6509 ms	18989 ms	11179 ms
C	6524 ms	15256 ms	10724 ms

Tabel 5.3 Uji Coba Waktu Respon 35 Pengguna

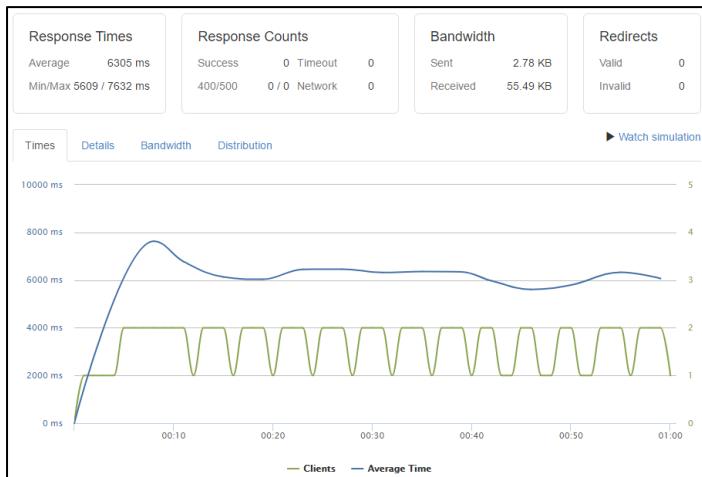
Rute	Waktu minimum	Waktu Maksimum	Rata-Rata
A	6308 ms	29994 ms	13930 ms
B	8299 ms	31284 ms	16835 ms
C	9288 ms	39123 ms	19355 ms



Gambar 5.15 Hasil Uji Coba Waktu Respon 15 Pengguna Rute A



Gambar 5.16 Hasil Uji Coba Waktu Respon 15 Pengguna Rute B



Gambar 5.17 Hasil Uji Coba Waktu Respon 15 Pengguna Rute C



Gambar 5.18 Hasil Uji Coba Waktu Respon 25 Pengguna Rute A



Gambar 5.19 Hasil Uji Coba Waktu Respon 25 Pengguna Rute B



Gambar 5.20 Hasil Uji Coba Waktu Respon 25 Pengguna Rute C



Gambar 5.21 Hasil Uji Coba Waktu Respon 35 Pengguna Rute A



Gambar 5.22 Hasil Uji Coba Waktu Respon 35 Pengguna Rute B

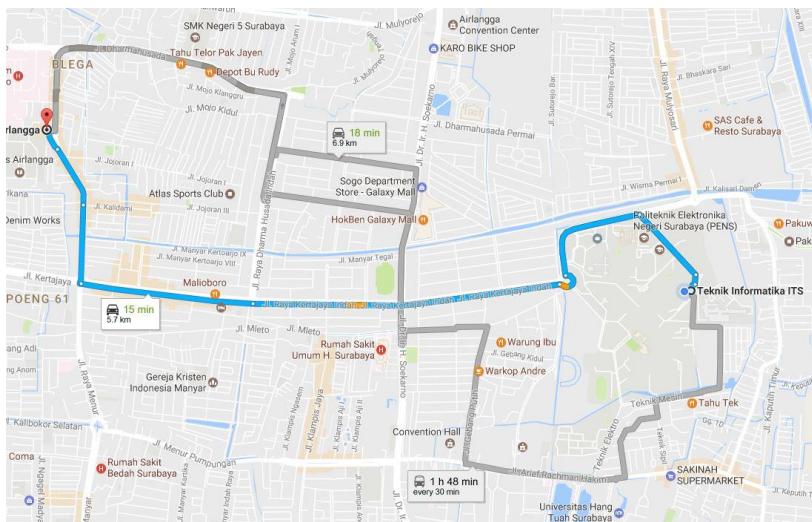


Gambar 5.23 Hasil Uji Coba Waktu Respon 35 Pengguna Rute C

5.1.4 Visualisasi Rute

Pada ujicoba ini akan dibandingkan hasil penentuan rute yang di dapat dari Tugas Akhir ini, dengan rute yang ada pada aplikasi Google Maps.

1. Diawali dengan percobaan pada titik tujuan terdekat. Titik yang digunakan sebagai tujuan adalah Restauran Wapo (112.75994, -7.270502). Gambar 5.24 akan menampilkan rute yang dibuat Google Maps, dan Gambar 5.25 akan menampilkan rute yang dibuat oleh Tugas Akhir ini.

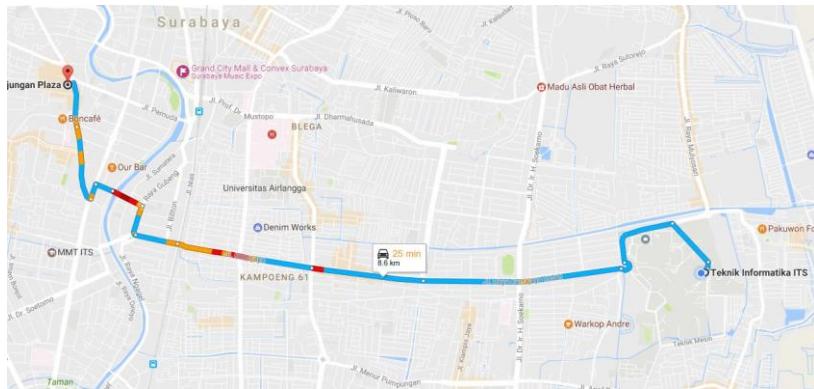


Gambar 5.24 Rute Dengan Tujuan Dekat Google Maps

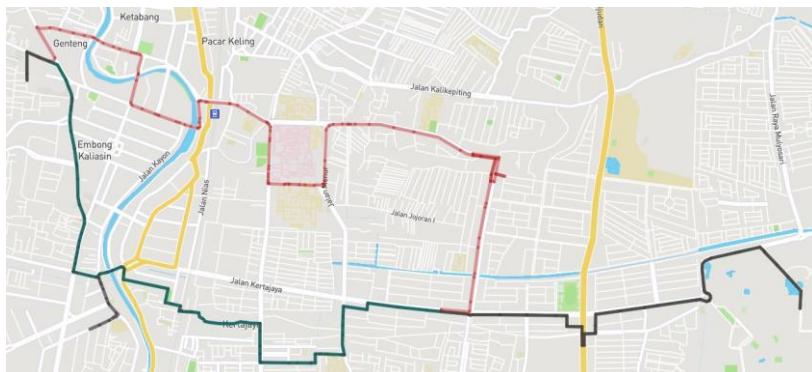


Gambar 5.25 Rute Dengan Tujuan Dekat Clearroute

2. Percobaan kedua adalah pencarian rute dengan tujuan rute dengan jarak sedang. Titik tujuan yang digunakan adalah menuju Tunjungan Plaza (112.7381256, -7.2630531). Gambar 5.26 akan menampilkan rute yang dibuat Google Maps, dan Gambar 5.27 akan menampilkan rute yang dibuat oleh Tugas Akhir ini.

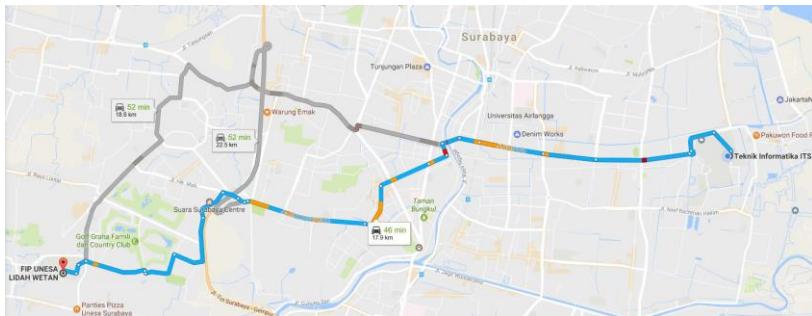


Gambar 5.26 Rute Dengan Tujuan Sedang Google Maps



Gambar 5.27 Rute Dengan Tujuan Sedang Clearroute

3. Percobaan terakhir adalah pencarian rute dengan tujuan rute dengan jarak jauh. Titik tujuan yang digunakan adalah menuju UNESA (112.6691293, -7.301802). Gambar 5.28 akan menampilkan rute yang dibuat Google Maps, dan Gambar 5.29 akan menampilkan rute yang dibuat oleh Tugas Akhir ini.



Gambar 5.28 Rute Dengan Tujuan Jauh Google Maps



Gambar 5.29 Rute Dengan Tujuan Jauh Clearroute

5.2 Evaluasi

Pada subbab evaluasi akan dibahas mengenai hasil ujicoba yang dilakukan pada subbab sebelumnya. Hasil dari evaluasi yang dibahas pada subbab ini akan dimasukkan ke bagian kesimpulan dan saran pada bab selanjutnya.

5.2.1 Mengubah Data Peta dari Openstreetmap ke Basis Data Spasial

Ketika dilakukan ujicoba, proses tersebut harus dilakukan 2 kali karena pada percobaan pertama terjadi kegagalan. Kegagalan yang terjadi adalah adanya kesalahan yang menampilkan tulisan “Process Killed”. Kegagalan tersebut terjadi ketika memori *swap* yang ada pada linux kurang, jawaban tersebut didapat dari link github <https://github.com/pgRouting/osm2pgsql/issues/20>. Kurangnya memori *swap* terjadi karena ukuran data yang ingin dimasukkan ke basis data terlalu besar, yaitu 1.7gb. Oleh karena itu penulis menambahkan jumlah memori *swap* yang ada menjadi 6gb dengan menuliskan kode program 5.3 pada terminal linux.

```
1 sudo swapon -s
2 sudo swapoff /swapfile
3 sudo fallocate -l 6G /swapfile
4 sudo mkswap /swapfile
5 sudo swapon /swapfile
6 sudo swapon -s
```

Kode Program 5.3 Menambah memori swap

5.2.2 Mengimplementasikan Algoritma A Star dengan pgRouting

Terdapat 4 evaluasi terkait dengan implementasi algoritma A Star dengan pgRouting, dan hal tersebut adalah sebagai berikut:

1. Nilai *cost* untuk penempatan yang harus dihindari masih belum pasti. Penulis hanya mengira-ngira pemberian nilai *cost* yang dianggap cukup tinggi agar *library* algoritma A Star pada pada ekstensi pgRouting mengabaikan jalur tersebut.
2. Penggunaan ekstensi pgRouting dalam penentuan rute perjalanan dapat dikatakan mudah. Penulis hanya perlu menggunakan 1 fungsi yang sesuai dengan algoritma yang ingin digunakan (dalam kasus ini adalah pgr_astar). Namun kekurangannya adalah tidak bisanya memodifikasi algoritma apabila dirasa kurang sesuai dengan studi kasus yang dihadapi.
3. Titik acak yang dibuat dan digunakan oleh penulis masih terlalu acak. Terdapat suatu kasus dimana titik acak yang didapatkan terlalu jauh dari titik awal dan titik tujuan, sehingga menyebabkan rute perjalanan menjadi lebih jauh.

5.2.3 Waktu Respon Server

Pada pengujian waktu respon server, hasil yang didapatkan sudah melalui beberapa tahapan untuk meminimalisir waktu respon dari server, tahapan tersebut adalah:

1. Dilakukannya *indexing*, *vacuuming*, *clustering*, serta *analyzing* ketika mempersiapkan basis data spasial yang akan digunakan oleh Clearroute API.
2. Pembatasan area pencarian rute oleh algoritma A Star dengan menggunakan fungsi *ST_Expand* dan *ST_Extent* dari PostGIS agar area pencarian rute hanya terbatas agregasi dari

geometry titik awal dan titik tujuan, dan diperlebar sebanyak 0.1 derajat.

Namun ternyata, didapatkan waktu rata-rata yang cukup tinggi, terutama pada waktu respon penentuan rute perjalanan jauh (19355 ms). Hal tersebut dikarenakan beberapa hal, yaitu:

1. Tidak dilakukannya optimisasi *server* yang digunakan sebagai tempat Clearroute API, seperti menggunakan *load balancer* untuk mengatur jumlah *traffic* pengguna dalam mengakses API.
2. Kode program penentuan rute perjalanan (terutama rute perjalanan alternatif) masih kurang optimal, dikarenakan adanya perulangan pemanggilan fungsi *library* A Star, sebanyak 5 kali dalam 1 kali proses penentuan rute perjalanan. Hal tersebut dapat dilihat pada *pseudocode* yang ada pada Kode Program 5.4.

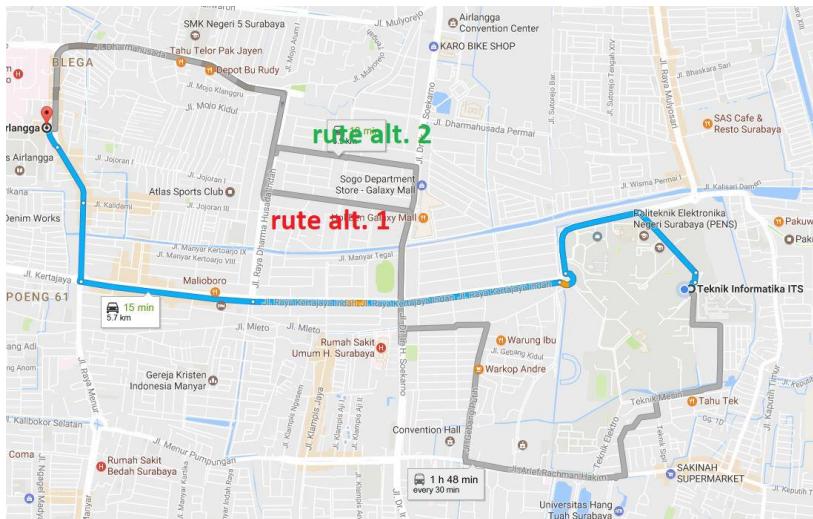
1	Ambil titik asal
2	Cari rute ke titik tujuan (1)
3	Simpan ke array 1 sebagai rute utama
4	
5	Ambil titik asal
6	Tentukan titik random untuk rute alternatif 1
7	Cari rute dari titik asal ke titik random (2)
8	Cari rute dari titik random ke titik tujuan (3)
9	Simpan ke array 2 sebagai rute alternatif 1
10	
11	Ambil titik asal
12	Tentukan titik random untuk rute alternatif 2
13	Cari rute dari titik asal ke titik random (4)
14	Cari rute dari titik random ke titik tujuan (5)
15	Simpan ke array 3 sebagai rute alternatif 2

Kode Program 5.4 Pseudocode Penentuan Rute Perjalanan

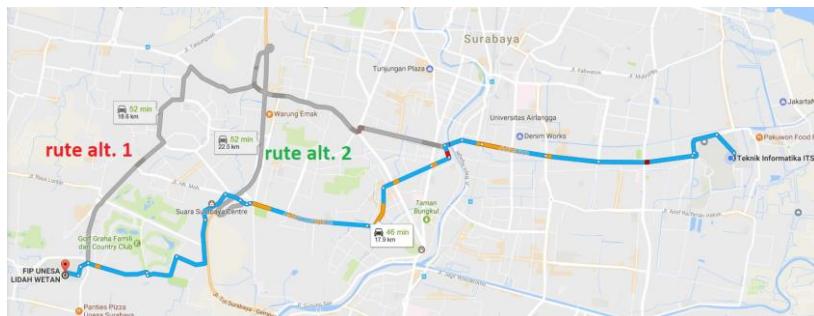
5.2.4 Visualisasi Rute

Pada visualisasi rute yang didapat dari Tugas Akhir ini dan rute yang didapat dari Google Maps, ditemukan beberapa perbedaan, yaitu:

1. Dilihat dari rute yang didapat dari Google Maps, rute alternatif yang diberikan tidak terlalu jauh dari rute utama. Hal tersebut dapat dilihat pada Gambar 5.30 – 5.31.

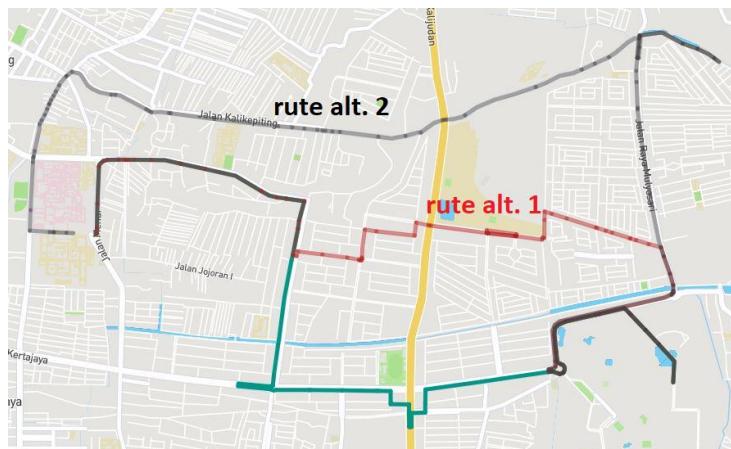


Gambar 5.30 Evaluasi Rute Google Maps 1



Gambar 5.31 Evaluasi Rute Google Maps 2

2. Rute alternatif yang dihasilkan oleh tugas akhir terkadang terlalu jauh dari rute utama. Hal tersebut dikarenakan penentuan titik acak yang diambil untuk penentuan rute alternatif masih kurang batasan. Hal tersebut dapat dilihat pada Gambar 5.32 – 5.34.



Gambar 5.32 Evaluasi Rute Perjalanan Dekat Clearroute



Gambar 5.33 Evaluasi Rute Perjalanan Sedang Clearroute



Gambar 5.34 Evaluasi Rute Perjalanan Jauh Clearroute

(Halaman ini sengaja dikosongkan)

BAB VI

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Setelah melakukan implementasi, uji coba, dan mengevaluasi hasilnya. Penulis dapat menarik beberapa kesimpulan, yaitu:

1. Data peta dari Openstreetmap dapat dengan mudah dimasukkan ke basis data spasial dengan menggunakan *tools* Osm2pgrouting. Pengguna hanya tinggal mengikuti panduan yang ada pada alamat web dari Osm2pgrouting.
2. Ketika proses memasukkan data peta Openstreetmap ke basis data spasial, Osm2pgrouting masih memiliki kemungkinan untuk terjadinya *error*, terutama apabila data peta yang dimasukkan terlalu besar, dan RAM dari komputer pengguna yang kurang cukup untuk menjalankan proses *import* data.
3. Algoritma A Star dapat digunakan untuk menentukan rute perjalanan dengan menggunakan ekstensi pgRouting PostgreSQL. Cara pemanggilan fungsi dan parameter yang digunakan pada ekstensi ini sudah jelas dituliskan pada dokumentasi dan *workshop* yang ada di situs web pgRouting.
4. Data peta provinsi Jawa Timur yang digunakan pada Tugas Akhir ini, masih dapat digunakan lagi pada studi kasus lainnya. Pengguna hanya tinggal menyesuaikan nilai atribut yang akan digunakan nantinya.
5. Dalam penentuan rute perjalanan, parameter waktu tempuh yang didapat dari data peta Openstreetmap masih kurang bagus, dikarenakan data kecepatan minimal dan maksimal yang diberikan masih belum sesuai dengan realita yang ada di Negara Indonesia.
6. Dalam menentukan rute alternatif, diperlukan beberapa parameter agar dapat menjadi patokan sehingga rute alternatif tidak benar-benar acak, yang dapat menyebabkan rute alternative menjadi lebih jauh daripada rute utamanya.

7. Waktu respon server yang terhitung lama dikarenakan paduan kode program penentuan rute alternatif, dan Algoritma A Star yang masih belum salin mendukung. Masih diperlukan kompleksitas yang tinggi untuk menjalankan fungsi penentuan rute alternatif.

6.2 Saran

Dari kesimpulan yang sudah diambil pada sub-bab sebelumnya, penulis dapat memberikan saran kepada pembaca buku Tugas Akhir ini, yaitu:

1. Penggunaan *tools* yang dapat memasukkan data peta berskala besar ke dalam basis data spasial, dengan tingkat *error* yang rendah.
2. Penggunaan jenis algoritma lain dalam hal menentukan rute perjalanan. Masih terdapat beberapa algoritma lain yang sudah disediakan ekstensi pgRouting di dalam daftar pustakanya.
3. Lebih banyak dilakukan survei di Kota Surabaya, atau kota lainnya sesuai dengan studi kasus, agar dapat memastikan kebenaran data peta Openstreetmap. Dan apabila ada atribut yang kurang benar, pengguna dapat membenahi isi datanya serta memberikan *feedback* ke pihak Openstreetmap agar memperbarui data petanya.
4. Optimasi basis data dan server penyedia API lebih ditekankan lagi, agar proses *transfer* data yang diminta *client* lebih cepat diterima.

DAFTAR PUSTAKA

- [1] “A* search algorithm,” *Wikipedia*. 03-Jan-2017.
- [2] “Introduction to A*.” [Daring]. Tersedia pada: <http://www.redblobgames.com/pathfinding/a-star/introduction.html>. [Diakses: 04-Jan-2017].
- [3] “Introduction to A* Pathfinding,” *Ray Wenderlich* .
- [4] “Admissible heuristic,” *Wikipedia*. 20-Apr-2017.
- [5] “OpenStreetMap,” *OpenStreetMap*. [Daring]. Tersedia pada: <http://www.openstreetmap.org/about>. [Diakses: 04-Jan-2017].
- [6] “Routing - OpenStreetMap Wiki.” [Daring]. Tersedia pada: <http://wiki.openstreetmap.org/wiki/Routing>. [Diakses: 12-Mei-2017].
- [7] B. 2017 dan B. M. Geofisika Klimatologi dan, “Tugas dan Fungsi | BMKG,” *BMKG / Badan Meteorologi, Klimatologi, dan Geofisika*. [Daring]. Tersedia pada: ?p=tugas-funghi&lang=ID. [Diakses: 04-Jan-2017].
- [8] “DB-Engines Ranking Open Source vs. Commercial DBMS.” [Daring]. Tersedia pada: https://db-engines.com/en/ranking_osvsc. [Diakses: 29-Mei-2017].
- [9] “PostgreSQL: About.” [Daring]. Tersedia pada: <https://www.postgresql.org/about/>. [Diakses: 04-Jan-2017].
- [10] “PostGIS — Spatial and Geographic Objects for PostgreSQL.” [Daring]. Tersedia pada: <http://postgis.net/>. [Diakses: 11-Apr-2017].
- [11] “PostGIS — PostGIS Feature List.” [Daring]. Tersedia pada: <http://postgis.net/features/>. [Diakses: 15-Mei-2017].
- [12] “pgRouting Project — Open Source Routing Library.” [Daring]. Tersedia pada: <http://pgrouting.org/>. [Diakses: 04-Jan-2017].
- [13] “PHP: What is PHP? - Manual.” [Daring]. Tersedia pada: <http://php.net/manual/en/intro-whatis.php>. [Diakses: 04-Jan-2017].

- [14] “History,” *Mapbox*. [Daring]. Tersedia pada:
<https://www.mapbox.com/about/history/>. [Diakses: 06-Jun-2017].
- [15] “Mapbox Android SDK,” *Mapbox*. [Daring]. Tersedia pada:
<https://www.mapbox.com/android-sdk/>. [Diakses: 06-Jun-2017].
- [16] “Application programming interface,” *Wikipedia*. 05-Jan-2017.
- [17] “JSON.” [Daring]. Tersedia pada: <http://www.json.org/>.
[Diakses: 04-Jan-2017].
- [18] “JSON Structures | JSON tutorial,” *w3resource*. [Daring].
Tersedia pada:
<http://www.w3resource.com/JSON/structures.php>. [Diakses: 01-Jun-2017].
- [19] “GeoJSON Specification.” [Daring]. Tersedia pada:
<http://geojson.org/geojson-spec.html#introduction>. [Diakses: 11-Apr-2017].
- [20] “osm2pgRouting - Import OSM data into pgRouting Database — Open Source Routing Library.” [Daring]. Tersedia pada:
<http://pgrouting.org/docs/tools/osm2pgrouting.html>. [Diakses: 12-Apr-2017].

LAMPIRAN

```
1 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
2 gid = 158311;  
3 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
4 gid = 145607;  
5 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
6 gid = 146074;  
7 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
8 gid = 146075;  
9 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
10 gid = 147378;  
12 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
13 gid = 147395;  
14 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
15 gid = 155236;  
16 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
17 gid = 157855;  
18 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
19 gid = 158822;  
20 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
21 gid = 185658;  
22 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
23 gid = 229627;  
24 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
25 gid = 239984;  
26 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
27 gid = 2416;  
28 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
29 gid = 108457;  
30 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
31 gid = 117667;  
32 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
33 gid = 160323;  
34 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
35 gid = 186080;  
36 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
37 gid = 234143;  
38 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
39 gid = 733;
```

```
40 UPDATE backup_ways SET cost_clearroute = 150 WHERE
41 gid = 2697;
42 UPDATE backup_ways SET cost_clearroute = 150 WHERE
43 gid = 110342;
44 UPDATE backup_ways SET cost_clearroute = 150 WHERE
45 gid = 125569;
46 UPDATE backup_ways SET cost_clearroute = 150 WHERE
47 gid = 158616;
48 UPDATE backup_ways SET cost_clearroute = 150 WHERE
49 gid = 158648;
50 UPDATE backup_ways SET cost_clearroute = 150 WHERE
51 gid = 158657;
52 UPDATE backup_ways SET cost_clearroute = 150 WHERE
53 gid = 158677;
54 UPDATE backup_ways SET cost_clearroute = 150 WHERE
55 gid = 158681;
56 UPDATE backup_ways SET cost_clearroute = 150 WHERE
57 gid = 158842;
58 UPDATE backup_ways SET cost_clearroute = 150 WHERE
59 gid = 159868;
60 UPDATE backup_ways SET cost_clearroute = 150 WHERE
61 gid = 161555;
62 UPDATE backup_ways SET cost_clearroute = 150 WHERE
63 gid = 161556;
64 UPDATE backup_ways SET cost_clearroute = 150 WHERE
65 gid = 161622;
66 UPDATE backup_ways SET cost_clearroute = 150 WHERE
67 gid = 162223;
68 UPDATE backup_ways SET cost_clearroute = 150 WHERE
69 gid = 162224;
70 UPDATE backup_ways SET cost_clearroute = 150 WHERE
71 gid = 163013;
72 UPDATE backup_ways SET cost_clearroute = 150 WHERE
73 gid = 177887;
74 UPDATE backup_ways SET cost_clearroute = 150 WHERE
75 gid = 183541;
76 UPDATE backup_ways SET cost_clearroute = 150 WHERE
77 gid = 183542;
78 UPDATE backup_ways SET cost_clearroute = 150 WHERE
79 gid = 183544;
```

```
80 UPDATE backup_ways SET cost_clearroute = 150 WHERE
81 gid = 1607;
82 UPDATE backup_ways SET cost_clearroute = 150 WHERE
83 gid = 183726;
84 UPDATE backup_ways SET cost_clearroute = 150 WHERE
85 gid = 1844150;
86 UPDATE backup_ways SET cost_clearroute = 150 WHERE
87 gid = 1841502;
88 UPDATE backup_ways SET cost_clearroute = 150 WHERE
89 gid = 184543;
90 UPDATE backup_ways SET cost_clearroute = 150 WHERE
91 gid = 184559;
92 UPDATE backup_ways SET cost_clearroute = 150 WHERE
93 gid = 1687;
94 UPDATE backup_ways SET cost_clearroute = 150 WHERE
95 gid = 184562;
96 UPDATE backup_ways SET cost_clearroute = 150 WHERE
97 gid = 185308;
98 UPDATE backup_ways SET cost_clearroute = 150 WHERE
99 gid = 185358;
100 UPDATE backup_ways SET cost_clearroute = 150 WHERE
101 gid = 185437;
102 UPDATE backup_ways SET cost_clearroute = 150 WHERE
103 gid = 185418;
104 UPDATE backup_ways SET cost_clearroute = 150 WHERE
105 gid = 185422;
106 UPDATE backup_ways SET cost_clearroute = 150 WHERE
107 gid = 185444;
108 UPDATE backup_ways SET cost_clearroute = 150 WHERE
109 gid = 160645;
110 UPDATE backup_ways SET cost_clearroute = 150 WHERE
111 gid = 185675;
112 UPDATE backup_ways SET cost_clearroute = 150 WHERE
113 gid = 185947;
114 UPDATE backup_ways SET cost_clearroute = 150 WHERE
115 gid = 186059;
116 UPDATE backup_ways SET cost_clearroute = 150 WHERE
117 gid = 186078;
118 UPDATE backup_ways SET cost_clearroute = 150 WHERE
119 gid = 186079;
```

```
120 UPDATE backup_ways SET cost_clearroute = 150 WHERE
121 gid = 186226;
122 UPDATE backup_ways SET cost_clearroute = 150 WHERE
123 gid = 186238;
124 UPDATE backup_ways SET cost_clearroute = 150 WHERE
125 gid = 186239;
126 UPDATE backup_ways SET cost_clearroute = 150 WHERE
127 gid = 186268;
128 UPDATE backup_ways SET cost_clearroute = 150 WHERE
129 gid = 186274;
130 UPDATE backup_ways SET cost_clearroute = 150 WHERE
131 gid = 187043;
132 UPDATE backup_ways SET cost_clearroute = 150 WHERE
133 gid = 187098;
134 UPDATE backup_ways SET cost_clearroute = 150 WHERE
135 gid = 187148;
136 UPDATE backup_ways SET cost_clearroute = 150 WHERE
137 gid = 187476;
138 UPDATE backup_ways SET cost_clearroute = 150 WHERE
139 gid = 187552;
140 UPDATE backup_ways SET cost_clearroute = 150 WHERE
141 gid = 187554;
142 UPDATE backup_ways SET cost_clearroute = 150 WHERE
143 gid = 187555;
144 UPDATE backup_ways SET cost_clearroute = 150 WHERE
145 gid = 1938;
146 UPDATE backup_ways SET cost_clearroute = 150 WHERE
147 gid = 2027;
148 UPDATE backup_ways SET cost_clearroute = 150 WHERE
149 gid = 187562;
150 UPDATE backup_ways SET cost_clearroute = 150 WHERE
151 gid = 194342;
152 UPDATE backup_ways SET cost_clearroute = 150 WHERE
153 gid = 195329;
154 UPDATE backup_ways SET cost_clearroute = 150 WHERE
155 gid = 223427;
156 UPDATE backup_ways SET cost_clearroute = 150 WHERE
157 gid = 227796;
158 UPDATE backup_ways SET cost_clearroute = 150 WHERE
159 gid = 228143;
```

```
160 UPDATE backup_ways SET cost_clearroute = 150 WHERE
161 gid = 228147;
162 UPDATE backup_ways SET cost_clearroute = 150 WHERE
164 gid = 229541;
165 UPDATE backup_ways SET cost_clearroute = 150 WHERE
166 gid = 186121;
167 UPDATE backup_ways SET cost_clearroute = 150 WHERE
168 gid = 195676;
169 UPDATE backup_ways SET cost_clearroute = 150 WHERE
170 gid = 229624;
171 UPDATE backup_ways SET cost_clearroute = 150 WHERE
172 gid = 231193;
173 UPDATE backup_ways SET cost_clearroute = 150 WHERE
174 gid = 231424;
175 UPDATE backup_ways SET cost_clearroute = 150 WHERE
176 gid = 231579;
177 UPDATE backup_ways SET cost_clearroute = 150 WHERE
178 gid = 231705;
179 UPDATE backup_ways SET cost_clearroute = 150 WHERE
180 gid = 231706;
181 UPDATE backup_ways SET cost_clearroute = 150 WHERE
182 gid = 231822;
183 UPDATE backup_ways SET cost_clearroute = 150 WHERE
184 gid = 232065;
185 UPDATE backup_ways SET cost_clearroute = 150 WHERE
186 gid = 232182;
187 UPDATE backup_ways SET cost_clearroute = 150 WHERE
188 gid = 232709;
189 UPDATE backup_ways SET cost_clearroute = 150 WHERE
190 gid = 232816;
191 UPDATE backup_ways SET cost_clearroute = 150 WHERE
192 gid = 232834;
193 UPDATE backup_ways SET cost_clearroute = 150 WHERE
194 gid = 232835;
195 UPDATE backup_ways SET cost_clearroute = 150 WHERE
196 gid = 232836;
197 UPDATE backup_ways SET cost_clearroute = 150 WHERE
198 gid = 1262;
199 UPDATE backup_ways SET cost_clearroute = 150 WHERE
200 gid = 233811;
```

```
201 UPDATE backup_ways SET cost_clearroute = 150 WHERE
202 gid = 233814;
203 UPDATE backup_ways SET cost_clearroute = 150 WHERE
204 gid = 234792;
205 UPDATE backup_ways SET cost_clearroute = 150 WHERE
206 gid = 235680;
207 UPDATE backup_ways SET cost_clearroute = 150 WHERE
208 gid = 235723;
209 UPDATE backup_ways SET cost_clearroute = 150 WHERE
210 gid = 235724;
211 UPDATE backup_ways SET cost_clearroute = 150 WHERE
212 gid = 235794;
213 UPDATE backup_ways SET cost_clearroute = 150 WHERE
214 gid = 235799;
215 UPDATE backup_ways SET cost_clearroute = 150 WHERE
216 gid = 236258;
217 UPDATE backup_ways SET cost_clearroute = 150 WHERE
218 gid = 236268;
219 UPDATE backup_ways SET cost_clearroute = 150 WHERE
220 gid = 236269;
221 UPDATE backup_ways SET cost_clearroute = 150 WHERE
222 gid = 239985;
223 UPDATE backup_ways SET cost_clearroute = 150 WHERE
224 gid = 242018;
225 UPDATE backup_ways SET cost_clearroute = 150 WHERE
226 gid = 244480;
227 UPDATE backup_ways SET cost_clearroute = 150 WHERE
228 gid = 2415001;
229 UPDATE backup_ways SET cost_clearroute = 150 WHERE
230 gid = 2415002;
231 UPDATE backup_ways SET cost_clearroute = 150 WHERE
232 gid = 240;
234 UPDATE backup_ways SET cost_clearroute = 150 WHERE
235 gid = 245;
236 UPDATE backup_ways SET cost_clearroute = 150 WHERE
237 gid = 418;
238 UPDATE backup_ways SET cost_clearroute = 150 WHERE
239 gid = 571;
240 UPDATE backup_ways SET cost_clearroute = 150 WHERE
241 gid = 583;
```

```
242 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
243 gid = 584;  
244 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
245 gid = 585;  
246 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
247 gid = 724;  
248 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
249 gid = 740;  
250 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
251 gid = 741;  
252 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
253 gid = 1467;  
254 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
255 gid = 2028;  
256 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
257 gid = 2070;  
258 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
259 gid = 2092;  
260 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
261 gid = 2210;  
262 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
263 gid = 2230;  
264 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
265 gid = 1404;  
266 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
267 gid = 21503;  
268 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
269 gid = 21504;  
270 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
271 gid = 2546;  
272 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
273 gid = 113084;  
274 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
275 gid = 2670;  
276 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
277 gid = 4461;  
278 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
279 gid = 4472;  
280 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
281 gid = 4730;
```

```
282 UPDATE backup_ways SET cost_clearroute = 150 WHERE
283 gid = 4779;
284 UPDATE backup_ways SET cost_clearroute = 150 WHERE
285 gid = 4801;
286 UPDATE backup_ways SET cost_clearroute = 150 WHERE
287 gid = 9492;
288 UPDATE backup_ways SET cost_clearroute = 150 WHERE
289 gid = 9493;
290 UPDATE backup_ways SET cost_clearroute = 150 WHERE
291 gid = 10119;
292 UPDATE backup_ways SET cost_clearroute = 150 WHERE
293 gid = 10120;
294 UPDATE backup_ways SET cost_clearroute = 150 WHERE
295 gid = 10160;
296 UPDATE backup_ways SET cost_clearroute = 150 WHERE
297 gid = 46714;
298 UPDATE backup_ways SET cost_clearroute = 150 WHERE
299 gid = 102623;
300 UPDATE backup_ways SET cost_clearroute = 150 WHERE
301 gid = 113086;
302 UPDATE backup_ways SET cost_clearroute = 150 WHERE
303 gid = 113105;
304 UPDATE backup_ways SET cost_clearroute = 150 WHERE
305 gid = 113106;
306 UPDATE backup_ways SET cost_clearroute = 150 WHERE
307 gid = 107347;
308 UPDATE backup_ways SET cost_clearroute = 150 WHERE
309 gid = 107351;
310 UPDATE backup_ways SET cost_clearroute = 150 WHERE
311 gid = 5101;
312 UPDATE backup_ways SET cost_clearroute = 150 WHERE
313 gid = 10154;
314 UPDATE backup_ways SET cost_clearroute = 150 WHERE
315 gid = 49017;
316 UPDATE backup_ways SET cost_clearroute = 150 WHERE
317 gid = 538150;
318 UPDATE backup_ways SET cost_clearroute = 150 WHERE
319 gid = 54965;
320 UPDATE backup_ways SET cost_clearroute = 150 WHERE
321 gid = 107169;
```

```
322 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
323 gid = 108459;  
324 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
325 gid = 108677;  
326 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
327 gid = 108774;  
328 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
329 gid = 109767;  
330 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
331 gid = 109771;  
332 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
333 gid = 110400;  
334 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
335 gid = 110626;  
336 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
337 gid = 110747;  
338 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
339 gid = 111345;  
340 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
341 gid = 1114150;  
342 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
343 gid = 112892;  
344 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
345 gid = 114191;  
346 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
347 gid = 114194;  
348 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
349 gid = 114839;  
350 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
351 gid = 115511;  
352 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
353 gid = 115534;  
354 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
355 gid = 115535;  
356 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
357 gid = 115764;  
358 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
359 gid = 116415;  
360 UPDATE backup_ways SET cost_clearroute = 150 WHERE  
361 gid = 117154;
```

```

362 UPDATE backup_ways SET cost_clearroute = 150 WHERE
363 gid = 117158;
364 UPDATE backup_ways SET cost_clearroute = 150 WHERE
365 gid = 116908;
366 UPDATE backup_ways SET cost_clearroute = 150 WHERE
367 gid = 119964;
368 UPDATE backup_ways SET cost_clearroute = 150 WHERE
369 gid = 119995;
370 UPDATE backup_ways SET cost_clearroute = 150 WHERE
371 gid = 120877;
372 UPDATE backup_ways SET cost_clearroute = 150 WHERE
373 gid = 125608;
374 UPDATE backup_ways SET cost_clearroute = 150 WHERE
375 gid = 125609;
376 UPDATE backup_ways SET cost_clearroute = 150 WHERE
377 gid = 125610;
378 UPDATE backup_ways SET cost_clearroute = 150 WHERE
379 gid = 125628;
380 UPDATE backup_ways SET cost_clearroute = 150 WHERE
381 gid = 127008;
382 UPDATE backup_ways SET cost_clearroute = 150 WHERE
383 gid = 127022;
384 UPDATE backup_ways SET cost_clearroute = 150 WHERE
385 gid = 185256;
386 UPDATE backup_ways SET cost_clearroute = 150 WHERE
387 gid = 158299;
388 UPDATE backup_ways SET cost_clearroute = 150 WHERE
389 gid = 2426
390 UPDATE backup_ways SET cost_clearroute = 150 WHERE
391 gid = 113094;

```

Lampiran A Mengubah nilai cost node perempatan

1	UPDATE backup_ways SET reverse_cost_s = 150 WHERE gid = 158311;
2	UPDATE backup_ways SET reverse_cost_s = 150 WHERE gid = 145607;
3	
4	
5	

```
6 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
7 gid = 146074;
8 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
9 gid = 146075;
10 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
12 gid = 147378;
13 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
14 gid = 147395;
15 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
16 gid = 155236;
17 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
18 gid = 157855;
19 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
20 gid = 158822;
21 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
22 gid = 185658;
23 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
24 gid = 229627;
25 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
26 gid = 239984;
27 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
28 gid = 2416;
29 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
30 gid = 108457;
31 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
32 gid = 117667;
33 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
34 gid = 160323;
35 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
36 gid = 186080;
37 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
38 gid = 234143;
39 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
40 gid = 733;
41 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
42 gid = 2697;
43 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
44 gid = 110342;
45 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
46 gid = 125569;
```

```
47 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
48 gid = 158616;
49 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
50 gid = 158648;
51 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
52 gid = 158657;
53 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
54 gid = 158677;
55 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
56 gid = 158681;
57 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
58 gid = 158842;
59 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
60 gid = 159868;
61 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
62 gid = 161555;
63 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
64 gid = 161556;
65 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
66 gid = 161622;
67 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
68 gid = 162223;
69 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
70 gid = 162224;
71 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
72 gid = 163013;
73 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
74 gid = 177887;
75 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
76 gid = 183541;
77 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
78 gid = 183542;
79 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
80 gid = 183544;
81 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
82 gid = 1607;
83 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
84 gid = 183726;
85 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
86 gid = 1844150;
```

```
87 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
88 gid = 1841502;  
89 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
90 gid = 184543;  
91 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
92 gid = 184559;  
93 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
94 gid = 1687;  
95 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
96 gid = 184562;  
97 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
98 gid = 185308;  
99 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
100 gid = 185358;  
101 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
102 gid = 185437;  
103 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
104 gid = 185418;  
105 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
106 gid = 185422;  
107 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
108 gid = 185444;  
109 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
110 gid = 160645;  
111 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
112 gid = 185675;  
113 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
114 gid = 185947;  
115 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
116 gid = 186059;  
117 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
118 gid = 186078;  
119 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
120 gid = 186079;  
121 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
122 gid = 186226;  
123 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
124 gid = 186238;  
125 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
126 gid = 186239;
```

```
127 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
128 gid = 186268;
129 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
130 gid = 186274;
131 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
132 gid = 187043;
133 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
134 gid = 187098;
135 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
136 gid = 187148;
137 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
138 gid = 187476;
139 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
140 gid = 187552;
141 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
142 gid = 187554;
143 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
144 gid = 187555;
145 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
146 gid = 1938;
147 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
148 gid = 2027;
149 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
150 gid = 187562;
151 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
152 gid = 194342;
153 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
154 gid = 195329;
155 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
156 gid = 223427;
157 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
158 gid = 227796;
159 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
160 gid = 228143;
161 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
162 gid = 228147;
164 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
165 gid = 229541;
166 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
167 gid = 186121;
```

```
168 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
169 gid = 195676;  
170 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
171 gid = 229624;  
172 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
173 gid = 231193;  
174 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
175 gid = 231424;  
176 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
177 gid = 231579;  
178 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
179 gid = 231705;  
180 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
181 gid = 231706;  
182 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
183 gid = 231822;  
184 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
185 gid = 232065;  
186 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
187 gid = 232182;  
188 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
189 gid = 232709;  
190 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
191 gid = 232816;  
192 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
193 gid = 232834;  
194 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
195 gid = 232835;  
196 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
197 gid = 232836;  
198 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
199 gid = 1262;  
200 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
201 gid = 233811;  
202 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
203 gid = 233814;  
204 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
205 gid = 234792;  
206 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
207 gid = 235680;
```

```
208 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
209 gid = 235723;
210 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
211 gid = 235724;
212 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
213 gid = 235794;
214 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
215 gid = 235799;
216 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
217 gid = 236258;
218 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
219 gid = 236268;
220 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
221 gid = 236269;
222 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
223 gid = 239985;
224 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
225 gid = 242018;
226 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
227 gid = 244480;
228 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
229 gid = 2415001;
230 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
231 gid = 2415002;
232 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
234 gid = 240;
235 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
236 gid = 245;
237 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
238 gid = 418;
239 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
240 gid = 571;
241 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
242 gid = 583;
243 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
244 gid = 584;
245 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
246 gid = 585;
247 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
248 gid = 724;
```

```
249 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
250 gid = 740;  
251 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
252 gid = 741;  
253 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
254 gid = 1467;  
255 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
256 gid = 2028;  
257 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
258 gid = 2070;  
259 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
260 gid = 2092;  
261 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
262 gid = 2210;  
263 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
264 gid = 2230;  
265 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
266 gid = 1404;  
267 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
268 gid = 21503;  
269 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
270 gid = 21504;  
271 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
272 gid = 2546;  
273 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
274 gid = 113084;  
275 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
276 gid = 2670;  
277 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
278 gid = 4461;  
279 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
280 gid = 4472;  
281 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
282 gid = 4730;  
283 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
284 gid = 4779;  
285 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
286 gid = 4801;  
287 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
288 gid = 9492;
```

```
289 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
290 gid = 9493;  
291 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
292 gid = 10119;  
293 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
294 gid = 10120;  
295 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
296 gid = 10160;  
297 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
298 gid = 46714;  
299 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
300 gid = 102623;  
301 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
302 gid = 113086;  
303 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
304 gid = 113105;  
305 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
306 gid = 113106;  
307 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
308 gid = 107347;  
309 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
310 gid = 107351;  
311 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
312 gid = 5101;  
313 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
314 gid = 10154;  
315 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
316 gid = 49017;  
317 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
318 gid = 538150;  
319 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
320 gid = 54965;  
321 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
322 gid = 107169;  
323 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
324 gid = 108459;  
325 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
326 gid = 108677;  
327 UPDATE backup_ways SET reverse_cost_s = 150 WHERE  
328 gid = 108774;
```

```
329 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
330 gid = 109767;
331 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
332 gid = 109771;
333 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
334 gid = 110400;
335 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
336 gid = 110626;
337 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
338 gid = 110747;
339 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
340 gid = 111345;
341 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
342 gid = 1114150;
343 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
344 gid = 112892;
345 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
346 gid = 114191;
347 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
348 gid = 114194;
349 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
350 gid = 114839;
351 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
352 gid = 115511;
353 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
354 gid = 115534;
355 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
356 gid = 115535;
357 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
358 gid = 115764;
359 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
360 gid = 116415;
361 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
362 gid = 117154;
363 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
364 gid = 117158;
365 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
366 gid = 116908;
367 UPDATE backup_ways SET reverse_cost_s = 150 WHERE
368 gid = 119964;
```

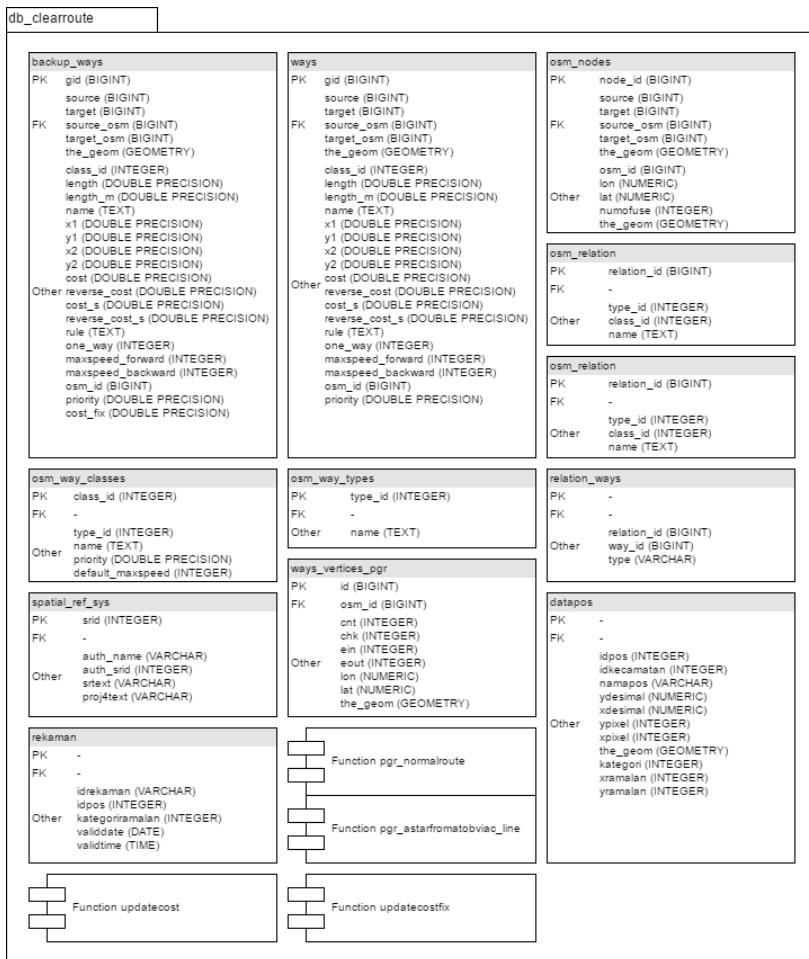
369	UPDATE backup_ways SET reverse_cost_s = 150 WHERE
370	gid = 119995;
371	UPDATE backup_ways SET reverse_cost_s = 150 WHERE
372	gid = 120877;
373	UPDATE backup_ways SET reverse_cost_s = 150 WHERE
374	gid = 125608;
375	UPDATE backup_ways SET reverse_cost_s = 150 WHERE
376	gid = 125609;
377	UPDATE backup_ways SET reverse_cost_s = 150 WHERE
378	gid = 125610;
379	UPDATE backup_ways SET reverse_cost_s = 150 WHERE
380	gid = 125628;
381	UPDATE backup_ways SET reverse_cost_s = 150 WHERE
382	gid = 127008;
383	UPDATE backup_ways SET reverse_cost_s = 150 WHERE
384	gid = 127022;
385	UPDATE backup_ways SET reverse_cost_s = 150 WHERE
386	gid = 185256;
387	UPDATE backup_ways SET reverse_cost_s = 150 WHERE
388	gid = 158299;
389	UPDATE backup_ways SET reverse_cost_s = 150 WHERE
390	gid = 2426
391	UPDATE backup_ways SET reverse_cost_s = 150 WHERE
	gid = 113094;

Lampiran B Mengubah nilai reverse_cost node perempatan

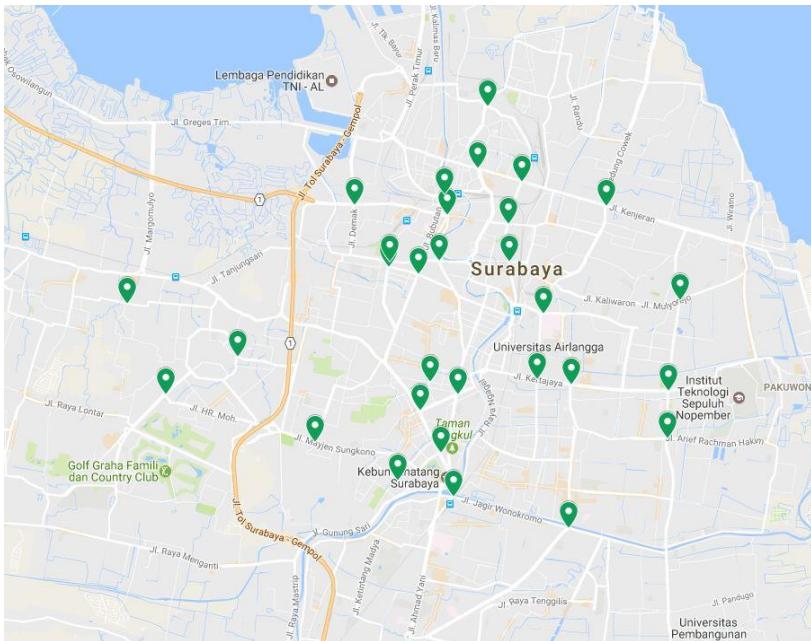
1	-7.289531, 112.780668
2	-7.263421, 112.783000
3	-7.265862, 112.756939
4	-7.245464, 112.769071
5	-7.278384, 112.755705
6	-7.256045, 112.750413
7	-7.248910, 112.750240
8	-7.240817, 112.752909
9	-7.245269, 112.720889
10	-7.226516, 112.746276
11	-7.257032, 112.727515

12	-7.238301, 112.744495
13	-7.263977, 112.677443
14	-7.274035, 112.698559
15	-7.292236, 112.737336
16	-7.290259, 112.713414
17	-7.300722, 112.739863
18	-7.297621, 112.729203
19	-7.255819, 112.736988
20	-7.258467, 112.733184
21	-7.255982, 112.727616
21	-7.247188, 112.738638
23	-7.281329, 112.740619
24	-7.243293, 112.738173
25	-7.281308, 112.684835
26	-7.284363, 112.733404
27	-7.278844, 112.735276
28	-7.306613, 112.761847
29	-7.279416, 112.762269
30	-7.28056, 112.78087

Lampiran C Koordinat perempatan diabaikan



Lampiran D Basis Data Clearroute



Lampiran E Titik Perempatan Diabaikan

1	{
2	"0": [
3	{
4	"json": "{\"type\": \"Feature\",
5	\"geometry\": {\"type\": \"LineString\",
6	\"coordinates\": [[[112.7977632, -7.280136],
7	[112.7978328, -7.2787877], [112.7971935, -
8	7.278096], [112.7968307, -7.2777086]]},
9	\"properties\": \"{}\""
10	},
11	{
12	"json": "{\"type\": \"Feature\",
13	\"geometry\": {\"type\": \"LineString\",
14	\"coordinates\": [[112.7968307, -7.2777086],
15	[112.7968307, -7.2777086]]}

```
16 [112.7959773, -7.2767973]]}, \"properties\":  
17 \"{}\""  
18 },  
19 {  
20     \"json\": \"{\\\"type\\\": \\\"Feature\\\",  
21 \\\"geometry\\\": {\\\"type\\\": \\\"LineString\\\",  
21 \\\"coordinates\\\": [[112.7959773, -7.2767973],  
23 [112.7954096, -7.2761912], [112.7950259, -  
24 7.275813]]}, \\\"properties\\\": \"{}\""  
25 },  
26 {  
27     \"json\": \"{\\\"type\\\": \\\"Feature\\\",  
28 \\\"geometry\\\": {\\\"type\\\": \\\"LineString\\\",  
29 \\\"coordinates\\\": [[112.7950259, -7.275813],  
30 [112.7949605, -7.2757305]]}, \\\"properties\\\":  
31 \"{}\""  
32 },  
33 ...  
34 ...  
35 ...  
36 {  
37     \"json\": \"{\\\"type\\\": \\\"Feature\\\",  
38 \\\"geometry\\\": {\\\"type\\\": \\\"LineString\\\",  
39 \\\"coordinates\\\": [[112.737643, -7.262573],  
40 [112.7376382, -7.2620616]]}, \\\"properties\\\":  
41 \"{}\""  
42 },  
43 {  
44     \"json\": \"{\\\"type\\\": \\\"Feature\\\",  
45 \\\"geometry\\\": {\\\"type\\\": \\\"LineString\\\",  
46 \\\"coordinates\\\": [[112.7376517, -7.2629921],  
47 [112.737643, -7.262573]]}, \\\"properties\\\":  
48 \"{}\""  
49 }  
],
```

Lampiran F Hasil GeoJSON Rute Clearroute API

BIODATA PENULIS



Ridho Perdana, lahir pada tanggal 30 April 1995 di Jakarta. Biasa dipanggil Ridho oleh masyarakat dan teman – teman sekitar. Saat ini sedang menempuh pendidikan S1 Teknik Informatika – Institut Teknologi Sepuluh Nopember Surabaya. Tertarik terhadap pemrograman (terutama pemrograman perangkat bergerak). Memperoleh beberapa pengalaman dalam hal pemrograman dari tugas – tugas kuliah dan beberapa proyek dari luar yang dilakukan bersama beberapa teman kuliah. Penulis dapat dihubungi melalui email ke **ridhoperdana@ymail.com**