

SKRIPSI

APLIKASI PENCARIAN RUTE TERDEKAT BERBASIS OPENSTREETMAP



ANDREAS HARYAWAN

NPM: 2010730071

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2015

UNDERGRADUATE THESIS

**CLOSEST ROUTE SEARCH APPLICATION BASED ON
OPENSTREETMAP**



ANDREAS HARYAWAN

NPM: 2010730071

**DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2015**

ABSTRAK

Pada penelitian ini, dibahas cara pembangunan aplikasi untuk mencari rute terdekat dengan memanfaatkan data peta yang disediakan oleh OpenStreetMap. OpenStreetMap adalah portal peta terbuka yang menyediakan data dalam bentuk gambar peta maupun dokumen XML yang disebut dengan OSMXML. Aplikasi dapat digunakan oleh pengemudi untuk membantu mencari rute terdekat agar sampai ke tempat tujuan lebih cepat. Aplikasi menggunakan data peta dalam OSMXML dan algoritma dijkstra untuk pencarian rute terdekat. Aplikasi menampilkan seluruh node dan edge yang didapatkan dari OSMXML pada peta digital sehingga pengguna dapat memilih titik asal dan titik tujuan untuk mencari rute terdekat.

Kata-kata kunci: Rute Terdekat, OSMXML, Algoritma Dijkstra, Peta Digital

ABSTRACT

This research aims to develop an application that can find the closest route, use map data provided by OpenStreetMap. OpenStreetMap is an opensource map portal that provides data in the form of a map image and XML document which is called by OSMXML. The application can be used by the driver to help looking for a closest route to the destination more quickly. The Application use data in OSMXML and dijkstra algorithm to find the closest route. The Application show every node and edge obtained from OSMXML on a digital map so users can choose the source and destination point to find the closest route.

Keywords: Shortest Path, OSMXML, Dijkstra Algorithm, Digital Map

DAFTAR ISI

DAFTAR ISI	ix
DAFTAR GAMBAR	xi
DAFTAR TABEL	xiii
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	2
1.4 Batasan Masalah	3
1.5 Metodologi Penelitian	3
1.6 Sistematika Pembahasan	3
2 DASAR TEORI	5
2.1 OpenStreetMap	5
2.2 XML	6
2.2.1 OSMXML	8
2.3 Javascript	11
2.3.1 XMLHttpRequest	13
2.3.2 XML DOM	14
2.3.3 Google Maps Javascript API	15
2.4 Graf	24
2.4.1 Graf Tidak Berarah	24
2.4.2 Graf Berarah	24
2.4.3 Graf Berbobot	25
2.4.4 Representasi Graf	26
2.5 Algoritma Dijkstra	27
2.6 Haversine <i>Formula</i>	28
3 ANALISIS	29
3.1 Analisis OpenStreetMap	29
3.1.1 Langkah-Langkah Pengambilan Data OSMXML	29
3.1.2 OSMXML	30
3.2 Analisis Javascript	32
3.3 Menghitung Jarak Antara Dua Titik	35
3.4 Pemodelan OSMXML Menjadi Graf	37
3.5 Visualisasi	41
3.6 Algoritma Dijkstra	43
3.7 Analisis Berorientasi Objek	46
3.7.1 Diagram <i>Use Case</i>	46
3.7.2 Skenario	47
3.7.3 Diagram Kelas Sederhana	48

4 PERANCANGAN	51
4.1 Perancangan Antar Muka	51
4.2 Perancangan Kelas	52
4.3 Diagram Sekuens	57
5 IMPLEMENTASI DAN PENGUJIAN	61
5.1 Implementasi	61
5.2 Pengujian	61
5.2.1 Pengujian Fungsional	62
5.2.2 Pengujian Eksperimental	67
6 KESIMPULAN DAN SARAN	73
6.1 Kesimpulan	73
6.2 Saran	73
DAFTAR REFERENSI	75
A KODE PROGRAM	77
B TEST OSMXML	87

DAFTAR GAMBAR

1.1	Tampilan website OpenStreetMap	1
1.2	Contoh Graf	2
2.1	Ekspor data pada situs OpenStreetMap	5
2.2	Polyline pada Peta	21
2.3	Info Window pada Peta	23
2.4	Contoh Graf	24
2.5	Contoh Graf Berarah	25
2.6	Contoh Graf Berbobot	25
2.7	Contoh Adjacency List	26
2.8	Contoh Adjacency List	26
2.9	Contoh Adjacency Matrix	27
2.10	Contoh Adjacency Matrix	27
3.1	Fitur search pada situs OpenStreetMap	30
3.2	Fitur search pada situs OpenStreetMap	30
3.3	Parsing XML menggunakan Javascript	35
3.4	Perhitungan Jarak Dengan Haversine Formula	36
3.5	Perhitungan Jarak Dengan Geometry Library	36
3.6	Pemodelan OSMXML menjadi Graf	40
3.7	Visualisasi	43
3.8	Info Window	43
3.9	Keluaran fungsi Dijkstra	45
3.10	Visualisasi Rute Terdekat	45
3.11	Diagram Use Case	46
3.12	Diagram Kelas Sederhana	49
4.1	Rancangan Antar Muka Awal Aplikasi	51
4.2	Rancangan Pemilihan Titik	52
4.3	Rancangan Cari Rute	52
4.4	Diagram Kelas	54
4.5	Diagram Sekuens Pemilihan Titik Asal	58
4.6	Diagram Sekuens Pemilihan Titik Tujuan	58
4.7	Diagram Sekuens Pencarian Rute Terdekat	59
5.1	Pengujian Antar Muka	62
5.2	Dokumen test.xml	63
5.3	Pengujian Fungsi OSMXML	63
5.4	Dokumen test.xml	64
5.5	Pengujian Jarak	64
5.6	Pengujian Fungsi OSMXML Menjadi Graf	65
5.7	Pengujian Fungsi Visualisasi	65
5.8	Pengujian Titik Asal dan Tujuan	66

5.9 Pengujian Rute Terdekat	66
5.10 Pengujian Visualisasi Rute Terdekat	67
5.11 Pengujian Bandung 1	68
5.12 Pengujian Bandung 1	68
5.13 Pengujian Bandung 1	69
5.14 bandung1.xml	69
5.15 Pengujian Bandung 2	70
5.16 Pengujian Bandung 2	70
5.17 Pengujian Bandung 2	71
5.18 Pengujian Bandung 3	71

DAFTAR TABEL

3.1 Skenario memilih titik asal	47
3.2 Skenario memilih titik tujuan	47
3.3 Skenario cari rute terdekat	47
5.1 Dokumen OSMXML	72

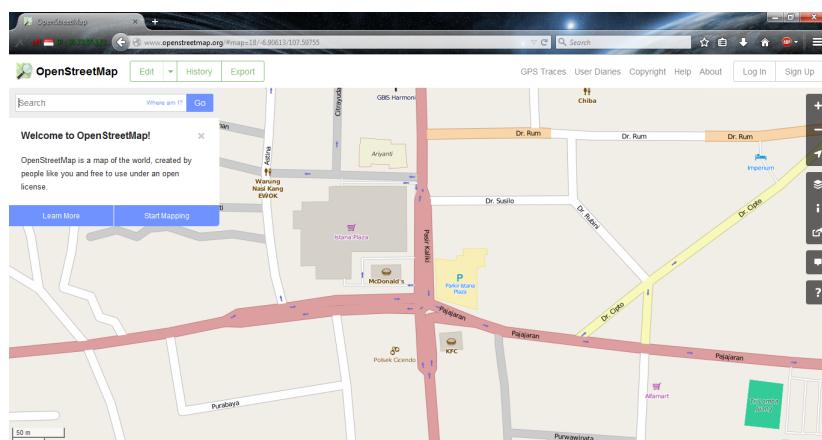
BAB 1

PENDAHULUAN

1.1 Latar Belakang

Mengemudi merupakan salah satu pilihan bagi masyarakat untuk bepergian dari suatu tempat ke tempat lain yang dituju. Contohnya adalah seorang wanita karir yang mengemudikan kendaraan pribadi dari rumah menuju kantor atau tempat kerjanya. Contoh lainnya adalah seorang sopir taksi yang mengemudikan kendaraannya untuk mengantar penumpang hingga sampai ke tujuan. Untuk dapat sampai ke titik tujuan, banyak rute yang dapat dilalui oleh seorang pengemudi. Seorang pengemudi, tentu saja akan mencari rute terdekat yang dapat dilalui, hal tersebut bertujuan untuk menghemat penggunaan bahan bakar dan juga waktu. Pemilihan rute terdekat untuk dapat sampai ke tujuan menjadi cukup penting, karena saat ini mobilitas masyarakat yang semakin tinggi. Aplikasi pencarian rute terdekat dapat membantu seorang pengemudi untuk menemukan rute terdekat agar sampai ke tempat tujuan lebih cepat. Aplikasi dapat menunjukkan rute mengemudi terdekat dari satu tempat ke tempat lain.

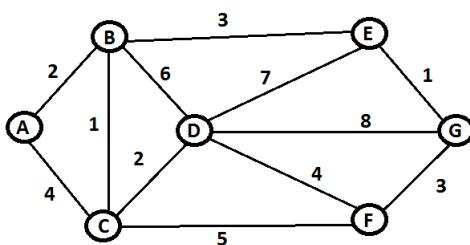
Aplikasi yang dibuat akan berbasis OpenStreetMap dan menggunakan algoritma Dijkstra. OpenStreetMap adalah portal peta terbuka yang menyediakan data dalam bentuk peta maupun XML, pengguna dapat mencari lokasi dan memilih area yang diinginkan. Setelah pengguna memilih area yang diinginkan, pengguna dapat menggunakan fitur export untuk mengunduh data XML pada area tersebut. Tampilan website OpenStreetMap dapat dilihat pada Gambar 1.1. Data yang disediakan oleh OpenStreetMap dalam bentuk XML biasa



Gambar 1.1: Tampilan website OpenStreetMap¹

¹<http://www.openstreetmap.org>

1 disebut dengan OpenStreetMap XML dan disingkat menjadi OSMXML. OSMXML adalah
 2 dokumen XML yang berisi data-data peta OSM. Pada dasarnya, OSMXML berisi data pri-
 3 mitif (node, way, dan relation) yang merupakan arsitektur dari model OSM. Node dapat
 4 diartikan sebagai titik pada peta digital, way merupakan informasi garis pada peta yang
 5 melambangkan jalan atau elemen lain seperti rel kereta, dan relation memberikan informasi
 6 node-node yang bersinggungan, elemen relation dapat menggambarkan suatu area seperti
 7 lapangan, taman bermain, rute bus, dan lain-lain. Sedangkan algoritma Dijkstra adalah
 8 algoritma untuk mencari jarak terpendek pada sebuah graf berarah dengan bobot yang ber-
 9 nilai tidak negatif pada setiap sisinya [1]. Graf adalah himpunan objek yang terdiri dari
 10 simpul(node) dan sisi (edge), graf digambarkan sebagai kumpulan titik yang dihubungkan
 oleh garis. Contoh graf dapat dilihat pada Gambar 1.2.



Gambar 1.2: Contoh Graf

11
 12 Aplikasi yang dibangun akan mengolah data yang disediakan oleh OpenStreetMap dalam
 13 bentuk XML dan memodelkannya ke dalam bentuk graf. Selanjutnya akan diimplementa-
 14 sikan algoritma Dijkstra untuk mencari rute terdekat pada graf tersebut dan menunjukkan
 15 hasilnya secara visual.

16 1.2 Rumusan Masalah

17 Berdasarkan latar belakang, maka rumusan masalah berikut:

- 18 • Bagaimana cara memodelkan data OSMXML menjadi sebuah graf?
 19 • Bagaimana cara menggunakan atau mengimplementasikan algoritma Dijkstra pada
 20 sebuah graf untuk mencari rute terdekat?
 21 • Bagaimana cara membuat visualisasi graf dan rute terdekat pada peta digital?

22 1.3 Tujuan

23 Berdasarkan rumusan masalah yang telah diuraikan di atas, maka tujuan dari penelitian
 24 yang dilakukan adalah:

- 25 • Mengetahui cara memodelkan data OSMXML menjadi sebuah graf.
 26 • Mempelajari cara kerja algoritma Dijkstra dan mengimplementasikannya pada sebuah
 27 graf.

- 1 • Mempelajari cara membuat visualisasi graf dan rute terdekat pada peta digital.

2 **1.4 Batasan Masalah**

3 Batasan permasalahan dari pembuatan aplikasi ini adalah :

- 4 • Aplikasi tidak mencari rute terdekat kedua dan seterusnya.
5 • Peta hanya menampilkan rute terdekat di wilayah Kota Bandung.
6 • Aplikasi tidak memperhitungkan faktor kemacetan jalan.

7 **1.5 Metodologi Penelitian**

8 Langkah-langkah yang akan dilakukan dalam melakukan penelitian adalah :

- 9 1. Melakukan studi pustaka untuk mengetahui teori-teori yang dapat mendukung proses
10 pembuatan aplikasi pencarian rute terdekat.
11 2. Melakukan analisis teori-teori yang mendukung proses pembuatan aplikasi.
12 3. Membuat rancangan aplikasi.
13 4. Melakukan implementasi berdasarkan rancangan yang telah dibuat.
14 5. Melakukan pengujian aplikasi.
15 6. Melakukan pengambilan kesimpulan berdasarkan analisis dan pengujian yang telah
16 dilakukan.

17 **1.6 Sistematika Pembahasan**

18 Pada setiap bab akan dibahas beberapa hal sebagai berikut :

- 19 1. Bab Pendahuluan
20 Bab 1 berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi
21 penelitian, dan sistematika pembahasan.
22 2. Bab Dasar Teori
23 Bab 2 berisi teori-teori dasar mengenai OpenStreetMap, algoritma Dijkstra, Google
24 Map Api, Graf, XML, dan beberapa teori lain yang mendukung pembuatan aplikasi.
25 3. Bab Analisis
26 Bab 3 berisi deskripsi sistem yang akan dibuat, analisis dasar teori, dan analisis cara
27 kerja algoritma Dijkstra pada graf.
28 4. Bab Perancangan
29 Bab 4 berisi perancangan antarmuka aplikasi disertai beberapa gambar, perancangan
30 kelas, dan diagram sekuen.

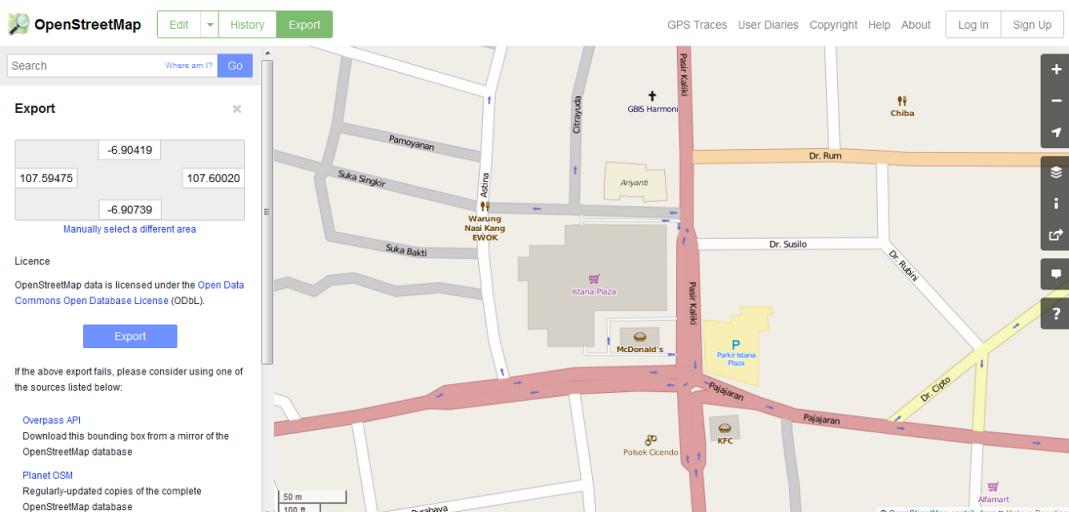
5. Bab Implementasi dan Pengujian
Bab 5 berisi laporan hasil implementasi yang dilakukan disertai dokumentasi mengenai penjelasan aplikasi tersebut dan hasil pengujian yang dilakukan berupa *screenshot*
 6. Bab Kesimpulan dan Saran
Bab 6 berisi kesimpulan dari seluruh hasil penelitian dan saran untuk pengembangan aplikasi yang akan datang.

BAB 2

DASAR TEORI

³ 2.1 OpenStreetMap

4 OpenStreetMap (OSM) adalah portal peta terbuka yang menyediakan data dalam bentuk
5 peta atau XML [2]. OSM menyediakan peta digital dan dapat diedit dari seluruh dunia,
6 juga memungkinkan pengguna untuk mengakses gambar peta yang terdapat pada situs
7 www.openstreetmap.org secara gratis. OSM terbentuk dan mendapatkan datanya dari
8 berbagai sukarelawan yang bersedia untuk berkontribusi, misalnya para pengguna OSM yang
9 menggunakan aplikasi untuk mengedit peta dan mengunggah data yang telah diedit ke situs
10 OSM. Selain itu, OSM menyediakan beberapa aplikasi bagi para pengguna untuk mengedit
11 peta, seperti iD online editor dan JOSM. Untuk mendapatkan gambar peta ataupun data
12 peta dalam bentuk lain, pengguna dapat menggunakan fitur export pada situs OSM. Fitur
export pada situs OSM dapat dilihat pada Gambar 2.1.



Gambar 2.1: Ekspor data pada situs OpenStreetMap

Berikut ini adalah beberapa data yang dapat diambil menggunakan fitur export [2]:

- ## 1. OpenStreetMap XML Data

OSM XML data dapat diperoleh dengan cara menggunakan tombol Export di bagian atas untuk membuka sidebar. Tombol Export mengarahkan langsung browser kepada OpenStreetMap API yang menyediakan data mentah OSM dalam bentuk XML.

- ## 19 2. Mapnik Image

1 Memungkinkan ekspor data OSM dalam bentuk PNG, JPEG, SVG, PDF dan peta
2 PostScript.

3 3. *Embeddable* HTML

4 Fitur ini memungkinkan pengguna untuk mendapatkan kode HTML yang dapat disalin
5 dan digunakan pada halaman web lain. Kode HTML tersebut akan menyisipkan peta
6 dalam sebuah iframe lengkap dengan javascript.

7 **2.2 XML**

8 XML adalah singkatan dari eXtensible Markup Language, XML adalah bahasa markup yang
9 dikembangkan oleh W3C (World Wide Web Consortium) [3]. Berikut ini adalah contoh
10 dokumen XML:

```
11 <?xml version="1.0" encoding="utf-8"?>
12 <catalog>
13   <book id="bk101">
14     <author>Gambardella , Matthew</author>
15     <title>XML Developer 's Guide</title>
16     <genre>Computer</genre>
17     <price>44.95</ price>
18     <publish_date>2000-10-01</publish_date>
19     <description>An in-depth look at creating applications
20       with XML.</description>
21   </book>
22   <book id="bk102">
23     <author>Ralls , Kim</author>
24     <title>Midnight Rain</title>
25     <genre>Fantasy</genre>
26     <price>5.95</ price>
27     <publish_date>2000-12-16</publish_date>
28     <description>A former architect battles corporate zombies ,
29       an evil sorceress , and her own childhood to become queen
30       of the world.</description>
31   </book>
32 <catalog>
```

33 Contoh di atas memberikan informasi mengenai katalog buku yang disimpan pada dokumen
34 XML. Pada awal dokumen tertera versi XML dan *encoding* yang digunakan. Setelah itu,
35 terdapat tag catalog yang memiliki *child* yaitu tag buku beserta informasinya. Terdapat
36 informasi id buku yang tertera pada atribut tag buku, seperti <book id="bk101"> dan
37 juga informasi lain seperti judul buku, penulis, genre, harga, tanggal terbit, dan deskripsi.

38 XML dikembangkan terutama untuk mengatasi keterbatasan pada HTML (Hypertext
39 Markup Language). HTML adalah salah satu bahasa markup yang paling populer dan
40 terus dikembangkan, banyak tag baru yang diperkenalkan. Pada versi pertama, HTML
41 memiliki satu lusin tag dan pada HTML pada versi 4.0 sudah hampir mencapai seratus

1 tag. Namun, pada aplikasi seperti *electronic commerce* dibutuhkan tag lebih untuk produk,
2 harga, nama, alamat, dan banyak lagi atau situs *streaming* memerlukan tag lebih untuk
3 mengontrol gambar dan suara.

4 HTML telah berkembang menjadi bahasa yang cukup kompleks, W3C memperkirakan
5 penggunaan komputer akan terus berkurang dan penggunaan gadget seperti smartphone
6 akan bertambah. Mesin tersebut tidak sekuat PC dan tidak bisa memproses bahasa yang
7 kompleks seperti HTML . Meskipun HTML adalah bahasa yang populer dan cukup suk-
8 ses, HTML memiliki beberapa kelemahan utama dan XML dikembangkan untuk mengatasi
9 kelemahan tersebut. XML adalah bahasa yang digunakan untuk menggambarkan dan me-
10 manipulasi dokumen terstruktur. Perubahan utama pada XML adalah tidak adanya tag
11 yang ditetapkan pada XML. Karena tidak ada tag yang ditetapkan, penulis dapat membuat
12 tag yang dibutuhkan. Beberapa ketentuan pada XML dapat dilihat pada uraian berikut:

13 1. Tag pada XML

14 Setiap elemen pada XML terdiri dari nama dan nilai, selain itu harus memiliki tag
15 pembuka dan tag penutup. Contoh:

16 <tel> 513-555-7098 </ tel>

17 Elemen untuk menyimpan nomor telepon memiliki nama tag tel, ditulis dengan <tel>
18 dan ditutup dengan </tel>.

19 2. Nama pada XML

20 Pemberian nama pada XML harus dimulai dengan huruf atau underscore (_) dan
21 sisanya diikuti huruf, angka, atau titik. Spasi tidak diperbolehkan pada pemberian
22 nama.

23 3. Atribut

24 Atribut memungkinkan untuk menyisipkan informasi tambahan, atribut juga memiliki
25 nama dan nilai. Contoh:

26 <tel preferred="true">513-555-8889</tel>
27 <tel>513-555-7098</tel>

28 Elemen tel dapat memiliki atribut *preferred*, memberikan informasi nomor telepon
29 yang lebih sering digunakan.

30 4. Elemen Kosong

31 Elemen yang tidak memiliki nilai atau isi disebut sebagai elemen kosong. Elemen
32 kosong biasanya memiliki atribut. Contoh:

33 <email href="mailto:jdoe@emailaholic.com"></email>

34 Elemen email tidak memiliki nilai atau isi.

35 5. *Nesting of Elements*

36 Sebuah elemen dapat memiliki elemen lain di dalamnya. Elemen yang berada di dalam
37 elemen lain disebut *child*, sedangkan elemen yang memiliki elemen lain disebut *parent*.
38 Contoh

```

1   <name>
2     <fname>Jack</fname>
3     <lname>Smith</lname>
4   </name>
```

5 Pada contoh berikut elemen name memiliki dua *child* yaitu fname dan lname dan
6 elemen name merupakan *parent* dari kedua elemen tersebut.

7 6. Root

8 *Root* merupakan elemen level tertinggi, pada dokumen XML harus ada satu elemen
9 pada level tertinggi. Dengan kata lain, elemen lain harus menjadi *child* dari *root*.

10 7. Deklarasi XML

11 Deklarasi XML dituliskan pada baris pertama dokumen. Pada deklarasi tersebut juga
12 dituliskan versi XML yang digunakan. Contoh:

```
13 <?xml version="1.0"?>
```

14 2.2.1 OSMXML

15 OpenStreetMap XML atau biasa disingkat dengan OSMXML merupakan dokumen XML
16 yang berisi data-data peta OSM. Pada dasarnya, OSMXML berisi data primitif (node, way,
17 dan relation) yang merupakan arsitektur dari model OSM [2]. Berikut ini adalah contoh
18 dokumen OSMXML:

```

19 <?xml version="1.0" encoding="UTF-8"?>
20 <osm version="0.6" generator="CGIImap_0.0.2">
21   <bounds minlat="54.0889580" minlon="12.2487570" maxlat="
22     54.0913900" maxlon="12.2524800"/>
23   <node id="298884269" lat="54.0901746" lon="12.2482632" user="
24     SvenHRO" uid="46882" visible="true" version="1" changeset="
25     676636" timestamp="2008-09-21T21:37:45Z"/>
26   <node id="261728686" lat="54.0906309" lon="12.2441924" user="
27     PikoWinter" uid="36744" visible="true" version="1" changeset="
28     323878" timestamp="2008-05-03T13:39:23Z"/>
29   <node id="1831881213" version="1" changeset="12370172" lat="
30     54.0900666" lon="12.2539381" user="lafkor" uid="75625" visible
31     ="true" timestamp="2012-07-20T09:43:19Z">
32     <tag k="name" v="Neu_Broderstorf"/>
33     <tag k="traffic_sign" v="city_limit"/>
34   </node>
35   ...
36   <node id="298884272" lat="54.0901447" lon="12.2516513" user="
37     SvenHRO" uid="46882" visible="true" version="1" changeset="
38     676636" timestamp="2008-09-21T21:37:45Z"/>
39   <way id="26659127" user="Masch" uid="55988" visible="true"
40     version="5" changeset="4142606" timestamp="2010-03-16
41     T11:47:08Z"/>
```

```

1  <nd ref="292403538"/>
2  <nd ref="298884289"/>
3  ...
4  <nd ref="261728686"/>
5  <tag k="highway" v="unclassified"/>
6  <tag k="name" v="Pastower_StraÃ§e"/>
7  <tag k="oneway" v="yes"/>
8  </way>
9  <relation id="56688" user="kmvar" uid="56190" visible="true"
10   version="28" changeset="6947637" timestamp="2011-01-12
11   T14:23:49Z">
12  <member type="node" ref="294942404" role="" />
13  ...
14  <member type="node" ref="364933006" role="" />
15  <member type="way" ref="4579143" role="" />
16  ...
17  <member type="node" ref="249673494" role="" />
18  <tag k="name" v="KÃijstenbus_Linie_123"/>
19  <tag k="network" v="VVW"/>
20  <tag k="operator" v="Regionalverkehr_KÃijste"/>
21  <tag k="ref" v="123"/>
22  <tag k="route" v="bus"/>
23  <tag k="type" v="route"/>
24  </relation>
25  ...
26  </osm>

```

27 Struktur OSMXML:

- 28 • Dokumen OSMXML diawali dengan tag xml yang menjelaskan versi xml dan encoding
29 yang digunakan, pada contoh di atas digunakan xml versi 1.0 dan encoding UTF-8.
 - 30 • Elemen osm memberikan informasi mengenai versi API dan generator yang digunakan.
31 Generator adalah alat untuk membuat dokumen OSMXML pada saat fitur export
32 digunakan.
 - 33 • Elemen bound memberikan informasi mengenai cakupan area pada dokumen OS-
34 MXML tersebut. Dilengkapi dengan atribut koordinat yaitu latitude dan longitude.
35 Data primitif pada OSM dibagi menjadi 3 bagian, yaitu node, way, dan relation.
- 36 1. Elemen Node merupakan informasi titik pada sebuah peta. Node memiliki beberapa
37 attribut yaitu:
- 38 – id
39 Merupakan id dari node tersebut.
 - 40 – user
41 Merupakan user yang melakukan editing pada node.

- 1 – uid
- 2 Id dari user.
- 3 – lat
- 4 berisi informasi koordinat pada garis lintang.
- 5 – lon
- 6 berisi informasi koordinat pada garis bujur.
- 7 – timestamp
- 8 Berisi informasi waktu saat node tersebut diperbarui.

9 Node juga memiliki elemen tag sebagai *child* yang memberikan informasi tambahan pada node tersebut, contoh:

11 `<tag k="name" v="Neu Broderstorf"/>`

12 nama dari node tersebut adalah Neu Broderstorf.

- 13 2. Elemen Way merupakan informasi garis yang dapat diartikan sebagai jalan atau pun elemen lain seperti rel kereta pada peta OSM. Way menyimpan informasi node-node terurut yang dilalui oleh garis dan juga sama seperti node dilengkapi atribut seperti id, uid, user, changeset, timestamp. Elemen way memiliki *child* elemen nd, contoh:

18 `<nd ref="292403538"/>`

19 atribut ref pada elemen nd mengacu pada node yang memiliki id 292403538, dan
20 elemen tag yang memberikan informasi tambahan pada elemen way. Selain itu,
21 elemen way memiliki informasi lain yang disimpan pada elemen tag, elemen tag
22 merupakan *child* dari elemen way dan menyimpan informasi jenis jalan yaitu *key*
23 *highway* dan *key oneway* yang memberikan informasi arah jalan, contoh:

24 `<tag k="highway" v="unclassified"/>`

25 `<tag k="oneway" v="yes"/>`

26 Key *oneway* memiliki 4 jenis *value*. Informasi arah jalan mengikuti node-node
27 yang telah terurut. Berikut ini adalah penjelasan dari keempat *value* tersebut:

- 28 – oneway=yes
- 29 Menunjukkan jalan satu arah
- 30 – oneway=no
- 31 Menunjukkan jalan dua arah
- 32 – oneway=-1
- 33 Menunjukkan jalan satu arah dan berlawanan
- 34 – oneway=reversible
- 35 Menunjukkan jalan satu arah dan dapat berubah arah menjadi berlawanan.
36 Contoh, pengalihan jalan untuk mengatasi kemacetan.

- 37 3. Elemen relation menyimpan informasi node-node dan way yang bersinggungan.
38 Elemen relation dapat menggambarkan suatu area seperti lapangan, taman bermain, atau pada contoh di atas menggambarkan rute bus.

2.3 Javascript

2 Javascript adalah bahasa pemrograman web yang mulai dikembangkan di perusahaan yang
3 bernama Netscape. Javascript memiliki lisensi dari Sun Microsystems yang sekarang sudah
4 berganti nama menjadi Oracle. Saat ini, mayoritas situs web sudah menggunakan javascript.
5 Berikut ini adalah contoh penggunaan javascript pada dokumen HTML:

```
6 <!DOCTYPE html>
7 <html>
8 <head>
9 <script>
10 function myFunction() {
11     document.getElementById("demo").innerHTML = "Paragraph changed
12         .";
13 }
14 </script>
15 </head>
16 <body>
17 <h1>JavaScript in Head</h1>
18 <p id="demo">A Paragraph.</p>
19 <button type="button" onclick="myFunction()">Try it</button>
20 </body>
21 </html>
```

22 Pada contoh di atas terdapat fungsi yang ditulis menggunakan javascript, fungsi tersebut
23 akan mengubah string “A Paragraph” pada tag `<p>` menjadi “Paragraph changed” jika
24 button atau tombol “Try it” di klik.

25 Seluruh browser yang terdapat pada komputer, konsol game, tablet, dan smartphone
26 sudah mendukung Javascript. Javascript adalah bagian yang cukup penting pada sebuah
27 halaman web, jika HTML berfungsi untuk menentukan isi dari halaman dan CSS untuk
28 menentukan tampilan pada halaman, javascript berfungsi untuk menentukan “behavior” dari
29 halaman web tersebut [4]. Berikut ini adalah uraian dari struktur javascript dan beberapa
30 contoh sintaks:

31 1. Struktur

32 • *Character Set*

33 Javascript ditulis menggunakan karakter Unicode. Unicode adalah superset ASCII
34 dan Latin-1 yang mendukung hampir seluruh bahasa di dunia.

35 • *Comments*

36 Javascript mendukung 2 jenis komentar yaitu komentar yang diletakkan setelah
37 garis miring ganda `//` dan komentar yang diletakkan antara karakter `/*` dan `*/`.

38 `// This is a single-line comment.`

39 `/* This is also a comment */ // and here is another comment.`

40 `/*`

41 `* This is yet another comment.`

```

1      * It has multiple lines.
2      */

```

3 • Literal

4 Literal adalah notasi untuk merepresentasikan nilai dan nilai yang dituliskan akan
 5 muncul secara langsung dalam program. Literal dapat berupa karakter, bilangan
 6 bulat, bilangan real, boolean. Berikut ini adalah contoh literal:

```

7      12 // The number twelve
8      1.2 // The number one point two
9      "hello world" // A string of text
10     'Hi' // Another string
11     true // A Boolean value
12     false // The other Boolean value
13     /javascript/gi // A "regular expression" literal (for pattern matching)
14     null // Absence of an object

```

15 • *Identifier*

16 *Identifier* pada javascript hanyalah nama yang digunakan untuk memberi nama
 17 pada variabel atau fungsi. Digit tidak diperbolehkan sebagai karakter pertama
 18 pada *identifier*.

19 • *Reserved words*

20 *Reserved words* adalah kata-kata yang tidak dapat digunakan sebagai identifier,
 21 karena digunakan oleh javascript sebagai keyword. Beberapa contoh keyword
 22 seperti break, delete, if, null, true, false, try, dan lain-lain.

23 • *Optional Semicolons*

24 Seperti banyak bahasa pemrograman lain, javascript menggunakan titik koma (;)
 25 untuk memisahkan perintah yang ditulis. Hal ini penting untuk membuat kode
 26 program menjadi jelas mengenai awal dan akhir. Pada javascript, titik koma
 27 dapat dihilangkan jika perintah ditulis pada baris yang berbeda, berikut adalah
 28 contoh penggunaan titik koma pada javascript:

```

29     a = 3;
30     b = 4;

```

31 titik koma pertama dapat dihilangkan, namun jika ditulis pada baris yang sama,
 32 titik koma tetap diperlukan

```

33     a = 3; b = 4;

```

34 2. Sintaks

35 • Deklarasi Variabel

36 Pembuatan variabel pada javascript menggunakan keyword var. Contoh deklarasi
 37 atau pembuatan variabel pada javascript:

```

38     var i;
39     var i, sum;
40     var message = "hello";
41     var i = 0, j = 0, k = 0;

```

1 ● Fungsi

2 Fungsi adalah blok kode program yang hanya dituliskan sekali, tetapi dapat di-
 3 panggil atau dijalankan berulang kali. Pada javascript, fungsi dapat dibuat meng-
 4 gunakan keyword function. Sebuah fungsi harus memiliki nama, sepasang tanda
 5 kurung untuk parameter, dan sepasang kurung kurawal. Berikut ini adalah beberapa contoh fungsi:

```
7 // Print the name and value of each property of o. Return undefined.
8 function printprops(o) {
9     for(var p in o)
10        console.log(p + ": " + o[p] + "\n");
11    }
12
13 // Compute the distance between Cartesian points (x1,y1) and (x2,y2).
14 function distance(x1, y1, x2, y2) {
15     var dx = x2 - x1;
16     var dy = y2 - y1;
17     return Math.sqrt(dx*dx + dy*dy);
18 }
```

18 2.3.1 XMLHttpRequest

19 XMLHttpRequest adalah salah satu objek pada javascript yang dapat digunakan untuk
 20 mendapatkan file XML dari server secara *asynchronous* atau *synchronous* [5]. *Asynchronous*
 21 berarti bahwa pertukaran data dilakukan tanpa harus memuat ulang seluruh halaman web,
 22 sedangkan pertukaran data *synchronous* harus memuat ulang seluruh halaman web. Berikut
 23 ini adalah contoh penggunaan XMLHttpRequest:

```
24 var objXMLHTTP = new XMLHttpRequest();
25
26 objXMLHTTP.open('GET', 'books.xml', false);
27 objXMLHTTP.send();
28
29 var objXML = objXMLHTTP.responseXML;
```

30 Langkah pertama adalah dengan membuat objek XMLHttpRequest. Selanjutnya, dengan
 31 memanggil fungsi open("method", "url", asynchronous). Parameter *method* menentukan me-
 32 tote yang digunakan, contoh "GET" untuk menerima data dan "POST" untuk mengirim
 33 data, parameter url adalah alamat file, dan parameter boolean "false" menunjukkan bahwa
 34 permintaan tersebut dilakukan secara *synchronous*. Langkah terakhir adalah mendapatkan
 35 respon dari server. Berikut ini penjelasan dari setiap *method* yang digunakan:

36 1. open("method","url", asynchronous,"username","password")

37 Melakukan inisialisasi permintaan

38 Parameter:

39 ● *method*

40 Method pada HTTP yang digunakan seperti "GET" dan "POST".

- 1 • *url*
 - 2 Alamat url tujuan
 - 3 • *asynchronous*
 - 4 boolean Opsional, secara default bernilai *true*. *True* menyatakan bahwa operasi
 - 5 yang dijalankan secara *asynchronous*. Nilai *false* menyatakan sebaliknya.
 - 6 • *username*
 - 7 Opsional, berisikan *username* yang digunakan untuk keperluan otentikasi. Secara
 - 8 default, berisi string kosong.
 - 9 • *password*
 - 10 Opsional, berisikan *password* yang digunakan untuk keperluan otentikasi. Secara
 - 11 default, berisi string kosong.
- 12 2. `send(content)`
- 13 Mengirimkan permintaan
- 14 Parameter:
- 15 • *content*
 - 16 Opsional, *content* dapat berisi string atau data lainnya seperti Array, dokumen,
 - 17 dan lain-lain.
- 18 3. `responseXML`
- 19 Respon dari permintaan
- 20 Return:
- 21 DOM Object

22 2.3.2 XML DOM

23 DOM adalah singkatan dari *Document Object Model*, XML DOM adalah API umum untuk menangani dokumen XML [5]. API adalah singkatan dari *Application Programming Interface* merupakan fungsi atau perintah yang dapat digunakan untuk menangani masalah pemrograman tertentu. XML DOM menyediakan fungsi standar untuk mengakses, memodifikasi, dan menciptakan berbagai bagian dari sebuah dokumen XML. Contoh:

```
28 var myNodeset = objXML.getElementsByTagName('plant');
29 var name = myNodeset[0].getAttribute('name');
```

30 Pemanggilan fungsi `getElementsByTagName('plant')` akan mengembalikan satu set node
31 yang memiliki nama tag 'plant'. Contoh lain, pemanggilan fungsi `getAttribute()` akan mengembalikan nilai atribut. Berikut ini penjelasan dari setiap *method* yang digunakan:

33 1. `getElementsByTagName('tagName')`

34 Mengembalikan elemen-elemen yang memiliki kesesuaian nama.

35 Parameter:

- 36 • *tagName*
- 37 String yang menentukan nama elemen yang dicari.

38 Return:

39 objek berisi elemen yang memiliki nama sesuai dengan yang dicari.

```

1   2. getAttribute('name')
2     Mengembalikan nilai atribut
3     Parameter:
4       • name
5         String yang menentukan nama atribut yang dicari.
6     Return:
7     Mengembalikan string jika atribut memiliki nilai, jika tidak mengembalikan null.

```

8 2.3.3 Google Maps Javascript API

9 Google Maps Javascript API memungkinkan untuk sebuah halaman web menampilkan peta
10 dunia yang datanya didapat dari server google [6]. Google menyediakan fungsi atau perintah
11 untuk menampilkan dan menyesuaikan peta sesuai dengan kebutuhan.

12 2.3.3.1 Elemen Dasar Google Maps

13 Google Maps Javascript API menyediakan fungsi dan kelas untuk memuat sebuah peta pada
14 halaman html. Berikut ini adalah contoh halaman web yang menampilkan peta di lokasi
15 Sydney, Australia:

```

16 <!DOCTYPE html>
17 <html>
18   <head>
19     <style type="text/css">
20       html, body, #map-canvas { height: 100%; margin: 0; padding:
21         0; }
22     </style>
23     <script type="text/javascript"
24       src="https://maps.googleapis.com/maps/api/js?key=API_KEY">
25     </script>
26     <script type="text/javascript">
27       function initialize() {
28         var mapOptions = {
29           center: { lat: -34.397, lng: 150.644 },
30           zoom: 8
31         };
32         var map = new google.maps.Map(document.getElementById('map-
33           -canvas'),
34           mapOptions);
35       }
36       google.maps.event.addDomListener(window, 'load', initialize)
37       ;
38     </script>
39   </head>
40   <body>

```

```

1 <div id="map-canvas"></div>
2 </body>
3 </html>
```

4 ● Declaring

5 Google menyarankan untuk membuat deklarasi tipe dokumen pada awal dokumen
6 yaitu dengan menulis <!DOCTYPE html>. Setelah itu diperlukan CSS yang bekerja
7 untuk mengatur tampilan peta pada halaman web.

```

8 <style type="text/css">
9   html { height: 100% }
10  body { height: 100%; margin: 0; padding: 0 }
11  #map-canvas { height: 100% }
12 </style>
```

13 Kode CSS pada contoh menunjukkan tag yang memiliki id map-canvas akan memiliki
14 tinggi 100% pada saat ditampilkan dan juga menunjukkan persentase yang sama pada
15 <html> dan <body>.

16 ● Loading Google Maps API

17 Untuk dapat menampilkan peta diperlukan juga melakukan *load javascript*. URL yang
18 terdapat pada tag script adalah lokasi file javascript yang akan memuat seluruh simbol
19 dan definisi yang dibutuhkan untuk menggunakan Google Maps API ini. Parameter
20 key berisi API key yang dimiliki oleh pengguna.

```

21 <html>
22   <head>
23     <script type="text/javascript"
24       src="https://maps.googleapis.com/maps/api/js?key=API_KEY">
25   </script>
```

26 ● Initialize

27 Setelah melakukan load javascript, diperlukan pemanggilan fungsi initialize. Di dalam
28 fungsi tersebut dapat ditambahkan beberapa variabel yang dibutuhkan.

```
29 function initialize() {}
```

30 Untuk inisialisasi peta, diperlukan variabel *map options*

```
31 var mapOptions = {};
```

32 Selanjutnya diperlukan koordinat pusat peta yang akan ditampilkan, sedangkan zoom
33 menunjukkan level zoom yang ingin ditampilkan

```

34   center: new google.maps.LatLng(-34.397, 150.644),
35   zoom: 8
```

```
1   • Map Object
2     objek peta perlu dibuat dengan cara melakukan inisialisasi kelas google.maps.Map.
3     Pada contoh, peta diletakkan pada <div> yang memiliki id map-canvas.

4     var map = new google.maps.Map(document.getElementById("map-canvas"),
5       mapOptions);

6   • Loading the Map
7     Google Maps API menyediakan fungsi untuk memuat peta. Pada potongan kode di
8     bawah, fungsi listener akan memanggil fungsi initialize ketika halaman dimuat.

9     google.maps.event.addDomListener(window, 'load', initialize);

10 Berikut ini adalah penjelasan kelas dan fungsi yang digunakan:
11 1. google.maps.Map class
12    Membuat peta baru pada halaman html.
13    Konstruktor:
14      • mapDiv:Node
15        node yang digunakan untuk membuat peta.
16      • opts?:MapOptions
17        Opsi dari map yang akan dibuat.

18 2. google.maps.LatLng class
19    Membuat objek LatLng yang merepresentasikan titik geografis.
20    Konstruktor:
21      • lat:number
22        Latitude dalam derajat.
23      • lng:number
24        Longitude dalam derajat.
25      • noWrap?:boolean
26        Latitude ditentukan dalam rentang derajat -90 hingga 90 dan longitude diten-
27        tukan dalam rentang derajat -180 hingga 180. Nilai true pada boolean noWrap
28        untuk mengaktifkan nilai di luar batas tersebut.

29 3. google.maps.event.addDomListener(instance:Object, eventName:string, handler:Function)
30    Menambahkan fungsi listener
31    Parameter:
32      • instance:Object
33        Objek yang ditambahkan listener.
34      • eventName:string
35        Nama Event.
36      • handler:Function
37        Fungsi yang dipanggil ketika event terjadi.

38    Return:
39    MapsEventListener
```

1 2.3.3.2 Menggambar pada Peta

2 Peta pada Google Maps API dapat ditambahkan objek seperti titik, garis, area, atau objek
3 lainnya. objek tersebut dinamakan *overlay*. Terdapat beberapa jenis *overlay* yang dapat
4 ditambahkan pada peta yaitu *marker* dan *polyline*. Berikut ini adalah penjelasan kelas dan
5 fungsi yang digunakan:

6 1. google.maps.Marker class

7 Membuat *marker* pada peta dengan opsi tertentu.

8 Konstruktor:

9 • opts?:MarkerOptions

10 Opsi dari *marker* yang dibuat.

11 2. google.maps.Polyline class

12 Membuat *polyline* pada peta dengan opsi tertentu.

13 Konstruktor:

14 • opts?:PolylineOptions

15 Opsi dari *polyline* yang dibuat.

16 3. setMap(map:Map)

17 Menyisipkan *marker* atau *polyline* pada peta tertentu.

18 Parameter:

19 • map:Map

20 Peta yang disisipkan *marker* atau *polyline*.

21 4. setIcon(icon:string|Icon|Symbol)

22 Mengubah *icon* pada *marker*.

23 Parameter:

24 • icon:string|Icon|Symbol

25 *Icon* yang digunakan.

26 Berikut ini adalah contoh penggunaan *marker* dan *polyline* pada peta:

27 1. *Marker*

28 Lokasi tunggal pada peta ditunjukkan oleh *Marker*.

29 • Menambahkan *Marker*

30 Untuk menampilkan *marker* pada peta harus membuat objek google.maps.Marker.

31 Berikut ini adalah atribut penting pada saat membuat objek *marker*:

32 (a) *position*

33 atribut *position* diperlukan untuk mengatur letak *marker* pada peta.

34 (b) *map*

35 atribut *map* bersifat opsional, untuk menentukan marker tersebut akan dile-
36 takukan pada peta. Jika atribut *map* tidak diatur, maka *marker* akan tetap
37 dibuat tetapi tidak akan ditampilkan pada peta.

38 Berikut ini adalah contoh kode program untuk menambahkan *marker* pada peta:

```

1      var myLatlng = new google.maps.LatLng(-25.363882,131.044922);
2      var mapOptions = {
3          zoom: 4,
4          center: myLatlng
5      }
6      var map = new google.maps.Map(document.getElementById
7          ("map-canvas"), mapOptions);
8
9      // To add the marker to the map, use the 'map' property
10     var marker = new google.maps.Marker({
11         position: myLatlng,
12         map: map,
13         title:"Hello World!"
14     });

```

Pada contoh, objek `google.maps.Marker` yang dibuat disimpan pada variabel *marker*, terdapat atribut *position* menggunakan variabel *myLatlng* yang berisi koordinat (-25.363882,131.044922), atribut *map* menunjukkan bahwa *marker* akan ditampilkan pada objek *map* yang tersimpan pada variabel *map*, dan atribut yang menunjukkan judul *marker*.

- Mengubah *icon marker*

Untuk mengubah *icon*, diperlukan pengaturan pada konstruktor *marker* tersebut. Pada contoh, *icon marker* diubah menjadi `beachflag.png`.

```

23     var image = 'images/beachflag.png';
24     var myLatLng = new google.maps.LatLng(-33.890542, 151.274856);
25     var beachMarker = new google.maps.Marker({
26         position: myLatLng,
27         map: map,
28         icon: image
29     });

```

Selain pengaturan pada konstruktor, pengubahan *icon* juga dapat dilakukan dengan cara memanggil fungsi `setIcon()`

```
32     beachMarkers.setIcon('images/beachflag.png');
```

- Menghapus *Marker* pada peta

Untuk menghapus *marker* pada peta, hanya diperlukan pemanggilan fungsi `setMap()` dan mengisi parameter fungsi tersebut dengan `null`. Contoh:

```
36     marker.setMap(null);
```

Pada contoh di atas hanya menghilangkan *marker* dari peta dan tidak menghapus objek *marker*.

- Animasi *Marker*

Menambahkan animasi pada *marker*, hanya memerlukan pengaturan atribut pada konstruktor `google.maps.Marker`. Contoh:

```

1      var marker = new google.maps.Marker({
2          position: myLatlng,
3          map: map,
4          animation: google.maps.Animation.BOUNCE,
5          title:"Hello World!"
6      });

```

7 Pada contoh, menambahkan animasi *bounce* pada marker sehingga *marker* ber-
8 gerak melompat-lompat pada peta.

9 • Mengubah Ikon

10 Gambar *marker* pada peta dapat diubah sesuai keinginan, hanya memerlukan
11 pengaturan atribut pada konstruktor google.maps.Marker. Contoh:

```

12     var image = 'images/beachflag.png';
13     var myLatLng = new google.maps.LatLng(-33.890542, 151.274856);
14     var beachMarker = new google.maps.Marker({
15         position: myLatLng,
16         map: map,
17         icon: image
18     });

```

19 Pada contoh, ikon *marker* akan ditampilkan menggunakan *file* gambar beachflag.png

20 • *Draggable*

21 Draggable memungkinkan pengguna untuk menyeret marker ke lokasi yang berbe-
22 da, hanya memerlukan pengaturan atribut pada konstruktor google.maps.Marker.
23 Contoh:

```

24     var marker = new google.maps.Marker({
25         position: myLatLng,
26         map: map,
27         draggable: true,
28         title:"Hello World!"
29     });

```

30 2. *Polyline*

31 Objek *polyline* adalah serangkaian garis pada peta, *polyline* berguna untuk menun-
32 jukkan dari satu titik ke titik lain. *Polyline* memiliki atribut yang dapat diubah sesuai
33 kebutuhan seperti warna, *opacity*, dan *weight*. Berikut ini penjelasan dari beberapa
34 atribut tersebut:

35 • *strokeColor*

36 Atribut *strokeColor* menentukan warna dalam format heksadesimal, contoh
37 "#FFFFFF".

38 • *strokeOpacity*

39 Atribut *strokeOpacity* menentukan *opacity* dalam nilai antara 0.0 dan 1.0.

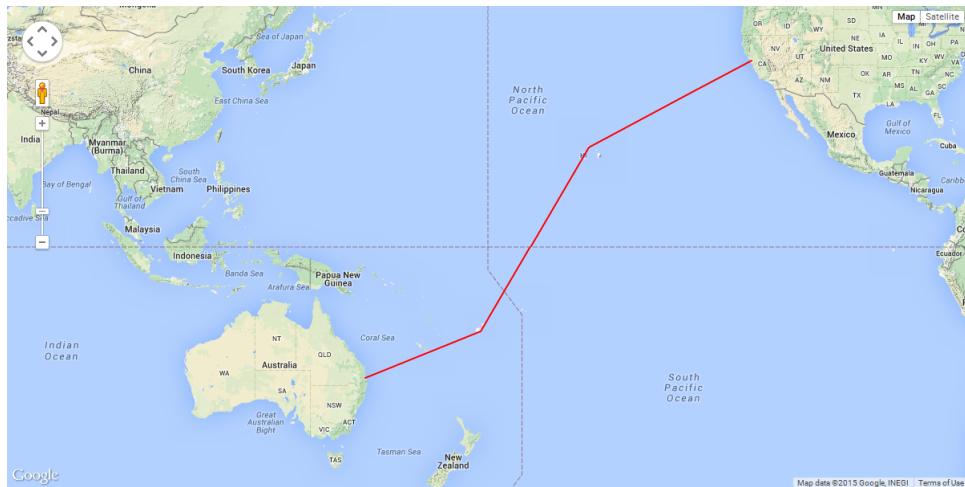
40 • *strokeWeight*

41 Atribut *strokeWeight* menentukan lebar garis dalam piksel.

1 Berikut ini adalah contoh potongan kode program untuk menampilkan *polyline* pada
 2 peta:

```
3 var flightPlanCoordinates = [
4   new google.maps.LatLng(37.772323, -122.214897),
5   new google.maps.LatLng(21.291982, -157.821856),
6   new google.maps.LatLng(-18.142599, 178.431),
7   new google.maps.LatLng(-27.46758, 153.027892)
8 ];
9 var flightPath = new google.maps.Polyline({
10   path: flightPlanCoordinates,
11   strokeColor: '#FF0000',
12   strokeOpacity: 1.0,
13   strokeWeight: 2
14 });
15
16 flightPath.setMap(map);
```

17 Pada contoh, akan menampilkan polyline pada peta yang akan menghubungkan setiap
 18 koordinat yang terdapat pada variabel *flightPlanCoordinates*. *Polyline* yang ditam-
 pilkan pada peta dapat dilihat pada Gambar 2.2.



Gambar 2.2: Polyline pada Peta

19

20 2.3.3.3 Geometry Library

21 Gambar pada Google Maps adalah dua dimensi, sedangkan bumi adalah tiga dimensi yang
 22 menyerupai bentuk bola. Hal ini tentu akan berbeda ketika mengukur suatu jarak dari
 23 satu titik ke titik lain, misalnya jarak terpendek antara dua titik pada bola bukanlah garis
 24 lurus, tetapi menyerupai lingkaran besar atau busur. Karena perbedaan tersebut, diperlukan
 25 *spherical geometry* untuk menghitung data geometris pada permukaan bumi seperti sudut,
 26 jarak, dan area yang berdasarkan garis lintang dan garis bujur. Google Maps JavaScript
 27 API menyediakan *geometry library* yang memiliki fungsi utilitas tersebut, fungsi utilitas

¹ tersebut dinamakan google.maps.geometry.spherical. Untuk menghitung jarak antara dua
² titik dapat memanggil fungsi computeDistanceBetween().

³ 1. google.maps.geometry.spherical namespace

⁴ Fungsi utilitas untuk menghitung sudut, jarak, dan area. Secara *default*, radius bumi
⁵ yang digunakan adalah 6378137 meter.

⁶ 2. computeDistanceBetween(from:LatLng, to:LatLng, radius?:number)

⁷ Menghitung jarak antara dua titik.

⁸ Parameter:

⁹ • from:LatLng

¹⁰ Koordinat titik pertama.

¹¹ • to:LatLng

¹² Koordinat titik kedua.

¹³ • radius?:number

¹⁴ Radius yang digunakan.

¹⁵ Berikut ini adalah contoh penggunaan fungsi computeDistanceBetween() untuk menghitung
¹⁶ jarak antara koordinat Kota Jakarta dan koordinat Kota Bandung:

```
17 var jakarta = new google.maps.LatLng(-6.1745,106.8227);
```

```
18 var bandung = new google.maps.LatLng(-6.9167,107.6000);
```

```
19
```

```
20 var distance = google.maps.geometry.spherical
```

```
21 .computeDistanceBetween(jakarta, bandung);
```

²² setelah menggunakan fungsi computeDistanceBetween(), didapatkan jarak antara dua titik
²³ koordinat tersebut adalah 119231.23264342443 meter atau lebih kurang 119,2 kilometer.

²⁴ 2.3.3.4 Info Window

²⁵ *Info window* adalah kelas yang disediakan Google Maps untuk menampilkan konten (biasanya berupa teks atau gambar) pada jendela *popup*. *Info window* memiliki ujung yang melekat ke lokasi tertentu pada peta. Biasanya *info window* diletakkan pada *marker* yang ada pada peta, tetapi *info window* juga dapat diletakkan pada koordinat peta tertentu. Berikut ini adalah contoh potongan kode program yang menampilkan *marker* beserta *info window*:

```
31 var contentString = 'Info Window';
```

```
32
```

```
33 var infowindow = new google.maps.InfoWindow({
```

```
34     content: contentString
```

```
35 });
```

```
36
```

```
37 var marker = new google.maps.Marker({
```

```
38     position: myLatlng,
```

```
39     map: map,
```

```

1     title: 'Uluru (Ayers Rock)',
2   });
3   google.maps.event.addListener(marker, 'click', function() {
4     infowindow.open(map,marker);
5   });

```

6 Pada contoh, terdapat variabel *contentString* yang berisi teks yang akan dimuat pada *info window*. Selanjutnya, diperlukan variabel *infowindow* yang menginisialisasi *info window*,
7 variabel *marker* yang menginisialisasi *marker*, dan *listener* yang memanggil fungsi *open*
8 ketika *marker* tersebut diklik. Berikut ini adalah penjelasan dari kelas dan fungsi yang
9 digunakan:
10

11 1. google.maps.InfoWindow class

12 Membuat *overlay* yang berbentuk seperti gelembung dan memuat konten seperti teks
13 atau gambar.

14 Konstruktor:

- 15 • opts?:InfoWindowOptions

16 Opsi dari *info window* yang dibuat.

17 2. open(map?:Map|StreetViewPanorama, anchor?:MVCObject)

18 Membuka *info window* pada peta.

19 Parameter

- 20 • map?:Map|StreetViewPanorama

21 Membuka *info window* pada peta yang diberikan.

- 22 • anchor?:MVCObject

23 Objek yang berasosiasi dengan *info window*, contoh: *marker*.

Info Window yang ditampilkan pada peta dapat dilihat pada Gambar 2.3.

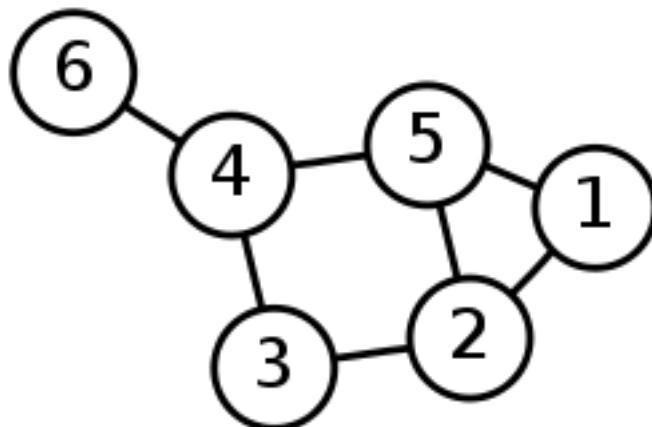


Gambar 2.3: Info Window pada Peta

2.4 Graf

Graf adalah himpunan objek yang terdiri dari node (simpul) dan edge (sisi), graf digambarkan sebagai node yang dihubungkan oleh edge. Terdapat berbagai macam jenis graf, tetapi pada subbab ini akan dibahas beberapa jenis graf seperti graf tidak berarah, graf berarah, dan graf berbobot. Contoh graf dapat dilihat pada Gambar 2.4. Graf mengikuti aturan berikut ²:

1. Graf terdiri dari dua bagian yang disebut node dan edge.
2. Node digambarkan berdasarkan tipenya dan nilainya mungkin terbatas atau tidak terbatas.
3. Setiap node menghubungkan dua buah edge.
4. Node digambarkan sebagai kotak atau lingkaran dan edge digambarkan sebagai garis atau busur.



Gambar 2.4: Contoh Graf

Berdasarkan contoh pada Gambar 2.4 didapatkan informasi tipe dari node adalah bulatan bulat

Himpunan node = 1,2,3,4,5,6

Himpunan edge = (6,4),(4,5),(4,3),(3,2),(5,2),(2,1),(5,1)

2.4.1 Graf Tidak Berarah

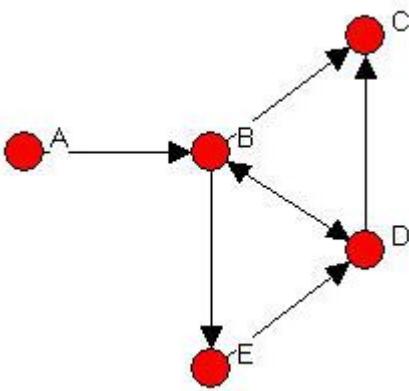
Graf tidak berarah adalah graf yang tidak memiliki arah pada setiap edgenya, sehingga setiap node tidak memiliki urutan. Graf tidak berarah digambarkan dengan garis lurus antara node. Contoh graf berarah dapat dilihat pada Gambar 2.4.

2.4.2 Graf Berarah

Graf berarah memiliki arah pada setiap edgenya. Pada graf berarah, edge biasanya digambarkan dengan panah sesuai arahnya. Contoh graf berarah dapat dilihat pada Gambar 2.5.

Berdasarkan contoh pada Gambar 2.5 didapatkan informasi tipe dari node adalah huruf

²<http://web.cecs.pdx.edu/sheard/course/Cs163/Doc/Graphs.html>



Gambar 2.5: Contoh Graf Berarah

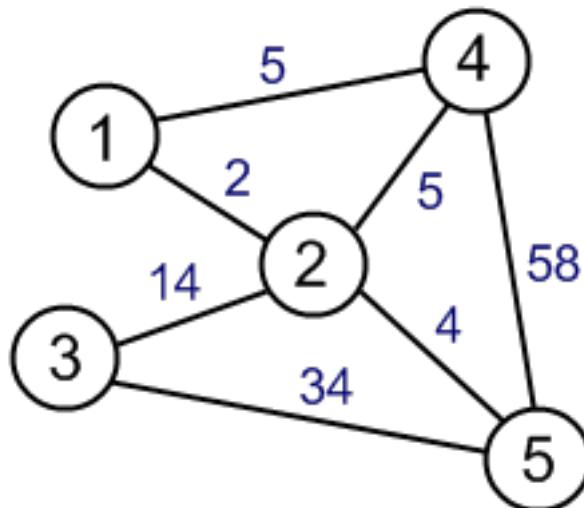
¹ kapital.

² Himpunan node = A, B, C, D, E

³ Himpunan edge = (A, B), (B, C), (D, C), (B, D), (D, B), (E, D), (B, E)

⁴ 2.4.3 Graf Berbobot

⁵ Graf berbobot adalah graf yang memiliki nilai pada setiap edgenya. Nilai tersebut dapat
⁶ berupa bilangan bulat ataupun bilangan pecahan desimal. Nilai tersebut dapat digunakan
⁷ untuk menyimpan jarak dari suatu node ke node lain. Contoh graf berbobot dapat dilihat
 pada Gambar 2.6. Berdasarkan contoh pada Gambar 2.6 didapatkan informasi tipe dari



Gambar 2.6: Contoh Graf Berbobot

⁸

⁹ node adalah bilangan bulat dan tipe dari bobot adalah bilangan bulat.

¹⁰ Himpunan node = 1,2,3,4,5

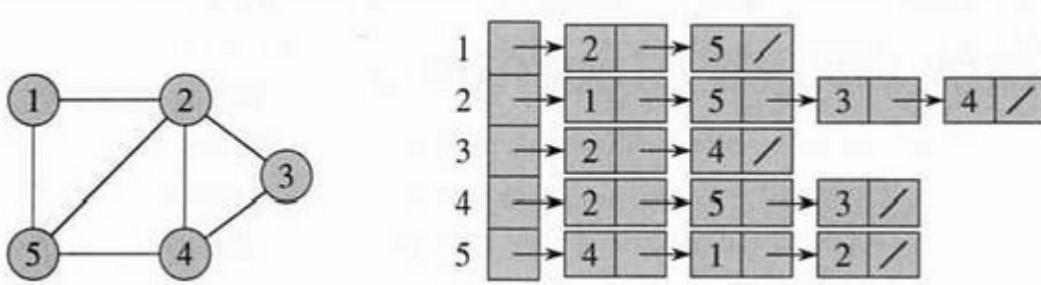
¹¹ Himpunan edge = (1,4,5) , (4,5,58) , (3,5,34) , (2,4,5) , (2,5,4) , (3,2,14) , (1,2,2)

2.4.4 Representasi Graf

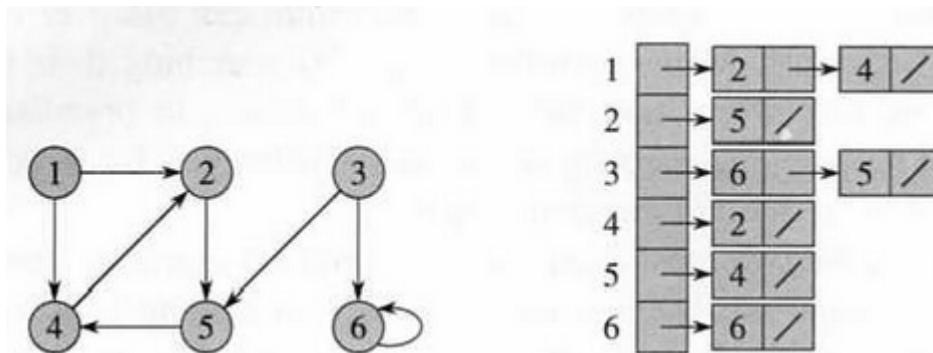
Terdapat dua cara untuk merepresentasikan graf yaitu dengan *adjacency list* dan *adjacency matrix* [1]. Keduanya dapat merepresentasikan graf berarah ataupun graf tidak berarah. *Adjacency list* merepresentasikan graf ke dalam bentuk array, sedangkan *adjacency matrix* merepresentasikan graf ke dalam bentuk matriks.

- *Adjacency List*

Adjacency List merupakan representasi graf ke dalam bentuk array, panjang array sesuai dengan jumlah node pada graf. Setiap index pada array mengacu pada setiap node graf, setiap index array tersebut memiliki list yang merepresentasikan hubungan dengan node-node lainnya. Contoh representasi graf tidak berarah dalam bentuk *adjacency list* dapat dilihat pada Gambar 2.7 dan representasi graf berarah dalam bentuk *adjacency list* dapat dilihat pada Gambar 2.8.



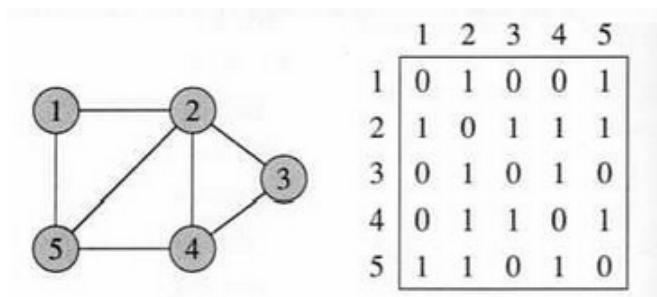
Gambar 2.7: Contoh Adjacency List (Graf Tidak Berarah)



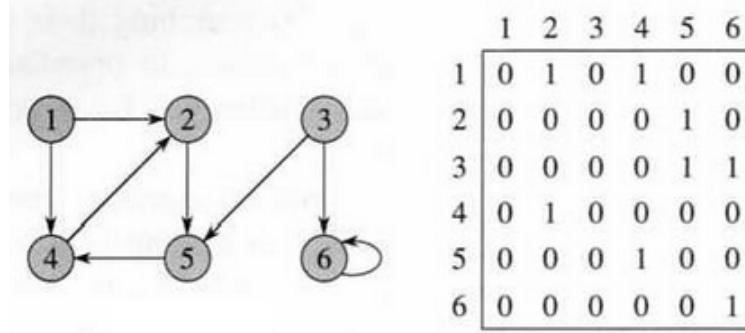
Gambar 2.8: Contoh Adjacency List (Graf Berarah)

- *Adjacency Matrix*

Adjacency Matrix merupakan representasi graf ke dalam bentuk matriks $n \times n$, pada matriks tersebut menyatakan hubungan antar node atau pada graf. Nilai n pada matriks $n \times n$ sesuai dengan jumlah node pada graf. Nilai 1 pada matriks menandakan terdapat hubungan pada node dan sebaliknya jika bernilai 0. Contoh representasi graf tidak berarah dalam bentuk *adjacency matrix* dapat dilihat pada Gambar 2.9 dan representasi graf berarah dalam bentuk *adjacency matrix* dapat dilihat pada Gambar 2.10.



Gambar 2.9: Contoh Adjacency Matrix (Graf Tidak Berarah)



Gambar 2.10: Contoh Adjacency Matrix (Graf Berarah)

2.5 Algoritma Dijkstra

- 2 Algoritma dijkstra adalah algoritma yang dapat mencari jalur terpendek pada graf berarah dengan persamaan $G=(V,E)$ untuk kasus pada setiap sisinya bernilai tidak negatif [1].
- 4 Algoritma ini menggunakan prinsip greedy. Prinsip greedy pada algoritma dijkstra adalah memilih sisi yang memiliki bobot paling kecil dan memasukannya dalam himpunan solusi.
- 5 Berikut ini adalah pseudocode dari algoritma dijkstra:

Algorithm 1 Dijkstra

```

 $dist[s] \leftarrow 0$ 
for all  $v \in V$  do
     $dist[v] \leftarrow \infty$ 
end for
 $S \leftarrow \emptyset$ 
 $Q \leftarrow V$ 
while  $Q \neq \emptyset$  do
     $u \leftarrow minDistance(Q, dist)$ 
     $S \leftarrow u$ 
    for all  $v \in neighbors[u]$  do
        if  $dist[v] > dist[u] + w(u, v)$  then
             $d[v] \leftarrow d[u] + w(u, v)$ 
        end if
    end for
end while
return  $dist$ 

```

¹ 2.6 Haversine *Formula*

Haversine *formula* adalah persamaan yang dapat memberikan jarak antara dua titik berdasarkan *latitude* atau garis lintang dan *longitude* atau garis bujur [7]. Haversine *formula* dinyatakan dalam persamaan berikut ini:

$$d = 2r \sin^{-1} \left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1) \cos(\phi_2) \sin^2\left(\frac{\psi_2 - \psi_1}{2}\right)} \right)$$

² dimana:

- ³ • d : jarak antara dua buah titik
- ⁴ • r : radius bumi (6371 km)
- ⁵ • ϕ_1, ϕ_2 : *latitude* dari titik 1 and *latitude* dari titik 2
- ⁶ • ψ_1, ψ_2 : *longitude* dari titik 1 and *longitude* dari titik 2

¹

BAB 3

²

ANALISIS

³ Pada bab ini akan dipaparkan analisis yang dilakukan dalam pembuatan aplikasi ini. Ba-
⁴ gaimana data XML didapatkan dari situs www.openstreetmap.org yang akan dibahas pada
⁵ subbab 3.1 dan membacanya menggunakan beberapa fungsi javascript yang akan dibahas
⁶ pada subbab 3.2. Setelah itu, data tersebut disimpan atau dikonversi ke dalam bentuk graf
⁷ sehingga dapat diimplementasikan algoritma dijkstra untuk mencari rute terpendek dari
⁸ satu node ke node lain. Berdasarkan informasi yang telah diolah, maka dapat dibuat visua-
⁹ lisasi data atau informasi tersebut menggunakan Google Maps Javascript API. Pada akhir
¹⁰ bab ini, juga akan dibahas mengenai diagram *use case* dan skenario untuk memperjelas apa
¹¹ saja yang dapat dilakukan oleh *user* pada aplikasi ini.

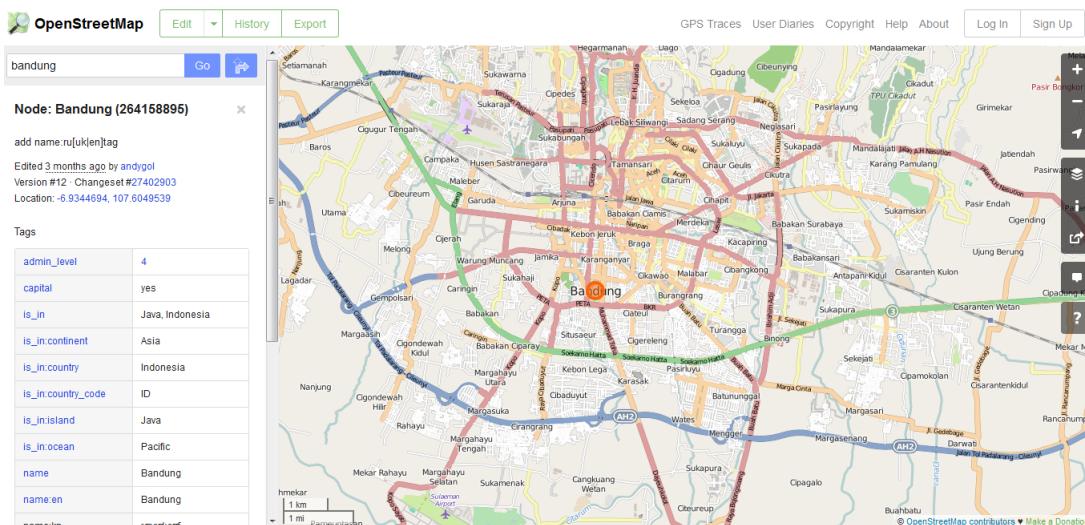
¹² **3.1 Analisis OpenStreetMap**

¹³ OpenStreetMap adalah portal peta terbuka yang menyediakan data dalam bentuk peta
¹⁴ ataupun dokumen XML. Aplikasi yang dibuat akan berbasis OpenStreetMap, hal ini berarti
¹⁵ aplikasi yang dibuat akan menggunakan data yang diperoleh dari situs www.openstreetmap.org.
¹⁶ Untuk mendapatkan data peta pada situs OpenStreetMap, user harus mengunjungi
¹⁷ situs tersebut dan menggunakan fitur *export*. Data yang digunakan adalah data peta yang
¹⁸ berbentuk dokumen XML atau biasa disebut dengan OSMXML. Selanjutnya, informasi
¹⁹ yang terkandung di dalam dokumen OSMXML tersebut akan diproses untuk mengetahui
²⁰ node dan edge pada peta. Informasi tersebut akan diubah ke dalam bentuk graf yang akan
²¹ diproses lebih lanjut menggunakan algoritma dijkstra untuk mengetahui jarak terpendek
²² dari satu node ke node lain.

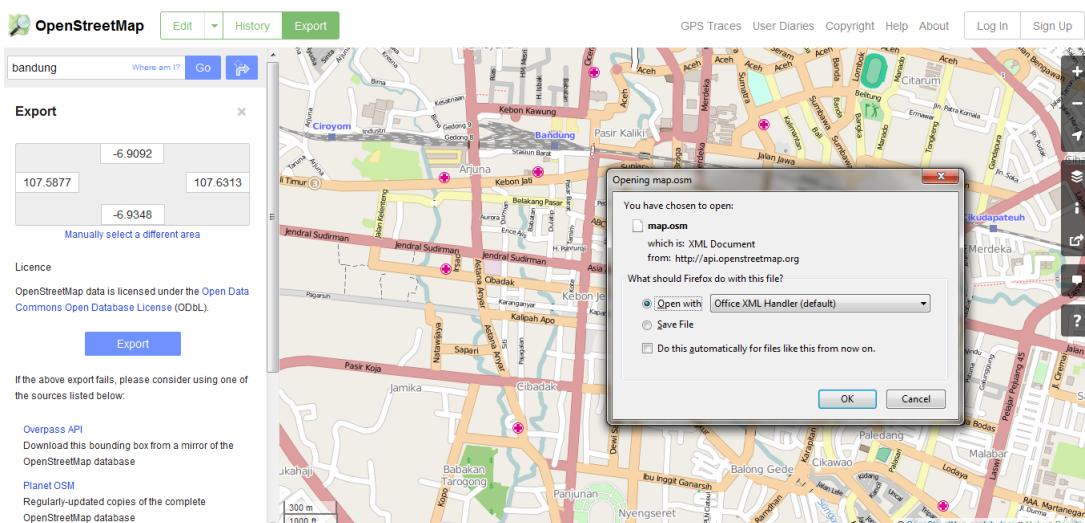
²³ **3.1.1 Langkah-Langkah Pengambilan Data OSMXML**

²⁴ Berikut ini adalah langkah-langkah pengambilan data OSMXML yang akan digunakan:

- ²⁵ 1. Mengunjungi situs www.openstreetmap.org.
- ²⁶ 2. Menggunakan fitur *search* untuk mencari area lokasi yang diinginkan. penggunaan
²⁷ fitur ini dapat dilihat pada Gambar 3.1 .
- ²⁸ 3. Menggunakan fitur *export* untuk mengunduh data dalam bentuk dokumen OSMXML.
²⁹ penggunaan fitur ini dapat dilihat pada Gambar 3.2.



Gambar 3.1: Fitur search pada situs OpenStreetMap



Gambar 3.2: Fitur export pada situs OpenStreetMap

3.1.2 OSMXML

2 Sesuai dengan pembahasan pada subbab 2.2.1, OSMXML merupakan dokumen XML yang
 3 mengandung data-data peta OpenStreetMap. Berikut ini adalah dokumen OSMXML yang
 4 sudah diunduh dan digunakan pada proses analisis.

```

5 <?xml version="1.0" encoding="UTF-8"?>
6 <osm version="0.6" generator="CGIImap_0.3.3_(29805_thorn-03.
  openstreetmap.org)" copyright="OpenStreetMap_and_contributors"
  attribution="http://www.openstreetmap.org/copyright" license=
  http://opendatacommons.org/licenses/odbl/1-0/">
10 <bounds minlat="-6.9076500" minlon="107.5961800" maxlat="
  -6.9044500" maxlon="107.6016300"/>
12 <node id="25418868" visible="true" version="6" changeset="
  27915808" timestamp="2015-01-04T17:54:58Z" user="isonpurba"
  uid="2552445" lat="-6.9064389" lon="107.5976351"/>
14
  
```

```

1 <node id="25433683" visible="true" version="3" changeset="839915"
2   timestamp="2009-03-21T14:18:48Z" user="adhitya" uid="7748"
3   lat="-6.9067659" lon="107.5989458"/>
4 ...
5 <node id="25433687" visible="true" version="2" changeset="839915"
6   timestamp="2009-03-21T14:18:36Z" user="adhitya" uid="7748"
7   lat="-6.9040267" lon="107.5969508"/>
8 <node id="25433688" visible="true" version="2" changeset="839915"
9   timestamp="2009-03-21T14:18:58Z" user="adhitya" uid="7748"
10  lat="-6.9039393" lon="107.5963723"/>
11 <node id="25500626" visible="true" version="3" changeset="839915"
12   timestamp="2009-03-21T14:22:17Z" user="adhitya" uid="7748"
13   lat="-6.9070329" lon="107.6019401"/>
14 ...
15 <node id="3030289971" visible="true" version="1" changeset=
16   24892866 timestamp="2014-08-20T18:40:31Z" user="albahrimaraxsa"
17   uid="2162153" lat="-6.9066710" lon="107.5982569"/>
18 <node id="2325451442" visible="true" version="4" changeset=
19   27916144 timestamp="2015-01-04T18:06:33Z" user="isonpurba"
20   uid="2552445" lat="-6.9045011" lon="107.6024922"/>
21 <way id="4567626" visible="true" version="4" changeset="15861148"
22   timestamp="2013-04-25T13:56:12Z" user="mrdooggie94" uid=
23   1331966">
24   <nd ref="25433682"/>
25   <nd ref="25433681"/>
26   <nd ref="25433680"/>
27   <tag k="avgspeed" v="15"/>
28   <tag k="highway" v="residential"/>
29   <tag k="name" v="Dr. Rubini"/>
30 </way>
31 <way id="4567634" visible="true" version="2" changeset="7821743"
32   timestamp="2011-04-10T11:15:30Z" user="evo2mind" uid="234610">
33   <nd ref="25433681"/>
34   <nd ref="28802396"/>
35   <tag k="avgspeed" v="15"/>
36   <tag k="highway" v="residential"/>
37   <tag k="name" v="Dr. Susilo"/>
38 </way>
39 ...
40 </osm>
```

41 Node dan way memiliki informasi penting yang akan digunakan pada aplikasi. Pada tag
42 node terdapat atribut id yang menunjukkan id pada setiap node, kemudian terdapat atribut
43 lat dan lon yang memberikan informasi titik koordinat (*latitude* dan *longitude*) pada node
44 tersebut. Informasi yang didapatkan akan disimpan ke dalam bentuk node pada graf. Tag

way akan menunjukkan hubungan pada node-node yang terdapat pada dokumen, dan akan disimpan sebagai edge pada graf. Selain itu, tag way tidak hanya memberikan informasi jalan raya atau jalan besar saja, tetapi juga beberapa elemen peta lain seperti area sekitar ling bangunan atau area sekitar tempat parkir. Maka dari itu, diperlukan *filter* pada tag way, karena hanya informasi jalan raya atau jalan besar saja yang diperlukan oleh aplikasi. Data atau dokumen OSMXML yang telah diperoleh, selanjutnya akan dibaca menggunakan javascript yang akan dibahas pada subbab 3.2.

3.2 Analisis Javascript

Javascript diperlukan untuk membaca dokumen OSMXML, sehingga seluruh informasi yang diperlukan dapat diubah ke dalam bentuk graf yang akan diproses lebih lanjut. XMLHttpRequest adalah salah satu objek pada javascript yang dapat digunakan untuk mendapatkan file XML. Berikut ini adalah langkah-langkah yang dilakukan untuk membaca dokumen OSMXML menggunakan javascript:

1. Membuat Objek XMLHttpRequest.

```
xmlhttp=new XMLHttpRequest();
xmlhttp.open("GET","map.xml",false);
xmlhttp.send();
 xmlDoc=xmlhttp.responseXML;
```

Objek XMLHttpRequest digunakan untuk membaca file XML yang telah diunduh sebelumnya. Tahap-tahap yang dilakukan adalah sebagai berikut:

- (a) Membuat objek XMLHttpRequest.
- (b) Objek ini akan menginisialisasi permintaan dengan memanggil fungsi open(), pada kode program di atas menggunakan *method* “GET” yaitu meminta dokumen “map.xml” dan parameter “false” menunjukkan bahwa permintaan tersebut *synchronous*.
- (c) Objek mengirimkan permintaan yang telah diinisialisasi sebelumnya dengan memanggil *method* send();
- (d) Tahap terakhir adalah mendapatkan respon dari permintaan yang telah dikirimkan. Objek mengakses atribut responseXML, jika permintaan berhasil maka variabel xmlDoc akan berisi dokumen XML yang diminta, jika gagal maka variabel akan bernilai *null*.

2. Menampilkan informasi node yang didapat pada layar.

```
document.write("<div style='float: left'>");
document.write("<table><tr><th>Node</th><th>Id</th>
<th>Latitude</th><th>Longitude</th></tr>");
document.write("<caption>Node</caption>");
var node=xmlDoc.getElementsByTagName("node");
```

```

1   for (i=0;i<node.length;i++){
2       document.write("<tr><td>");
3       document.write(i);
4       document.write("</td><td>");
5       document.write(node[i].getAttribute('id'));
6       document.write("</td><td>");
7       document.write(node[i].getAttribute('lat'));
8       document.write("</td><td>");
9       document.write(node[i].getAttribute('lon'));
10      document.write("</td></tr>");
11  }
12  document.write("</table>");
13  document.write("</div>");
```

14 Kode diatas menampilkan informasi node pada dokumen OSMXML dalam bentuk
 15 tabel. Berikut ini adalah tahap-tahap yang dilakukan:

- 16 (a) Membuat tag `<div>` sebagai tempat tabel.
- 17 (b) Membuat tabel.
- 18 (c) Membuat variabel node, variabel ini berisi informasi seluruh node yang terdapat
 19 pada variabel xmlDoc, *method* yang digunakan adalah `getElementsByTagName()`. Sebelumnya xmlDoc sudah berisi dengan dokumen “map.xml”.
- 20 (d) Melakukan *print* pada tabel, beberapa atribut yang ditampilkan adalah id node,
 21 *latitude*, dan *longitude*.

23 3. Membuat fungsi untuk melakukan *filter* pada elemen way.

```

24 function isHighway(way,index){
25     var tag = way[index].getElementsByTagName("tag");
26     for (hg=0;hg<tag.length;hg++){
27         if(tag[hg].getAttribute('k') == "highway"){
28             return true;
29         }
30     }
31     return false;
32 }
```

33 *Filter* dilakukan karena hanya elemen way yang berjenis *highway* saja yang akan di-
 34 gunakan. Berikut ini adalah tahap-tahap yang dilakukan:

- 35 (a) Membuat fungsi `isHighway()` dengan parameter *input* way dan index. Parameter
 36 way berisi informasi seluruh way, sedangkan parameter index menunjukkan tag
 37 way pada index tersebut yang akan dicari.
- 38 (b) Membuat variabel yang menyimpan informasi way pada index yang dicari, pada
 39 kode program variabel tersebut diberi nama “tag”.

- 1 (c) Lakukan pengulangan untuk mencari setiap *child* pada variabel “tag” yang me-
 2 miliki atribut “k=highway”, jika ditemukan fungsi akan mengembalikan *true* dan
 3 *false* jika tidak ditemukan.
- 4 4. Menampilkan informasi way yang didapat pada layar.

```

5       document.write("<div style='margin-left: 20px;float: left'>");
6       document.write("<table><tr><th>Way</th><th>Id Way</th>
7       <th>Edge</th><th>Id Node 1</th><th>Id Node 2</th></tr>"); 
8       document.write("<caption>Edge</caption>");

9
10      var way = xmlDoc.getElementsByTagName("way");
11      var nd;
12      for (i=0;i<way.length;i++){
13         nd = way[i].getElementsByTagName("nd");
14         if(isHighway(way,i)){
15             for (j=0;j<nd.length-1;j++){
16                 document.write("<tr><td>");
17                 document.write(i);
18                 document.write("</td><td>");
19                 document.write(way[i].getAttribute('id'));
20                 document.write("</td><td>");
21                 document.write(j);
22                 document.write("</td><td>");
23                 document.write(nd[j].getAttribute('ref'));
24                 document.write("</td><td>");
25                 document.write(nd[j+1].getAttribute('ref'));
26                 document.write("</td></tr>");
27             }
28         }
29      }
30      document.write("</div>");


```

31 Kode diatas menampilkan informasi way pada dokumen OSMXML dalam bentuk ta-
 32 bel.

- 33 (a) Membuat tag <div> sebagai tempat tabel.
 34 (b) Membuat tabel.
 35 (c) Membuat variabel way, variabel ini berisi informasi seluruh way yang terdapat pa-
 36 da variabel xmlDoc, *method* yang digunakan adalah getElementsByTagName().
 37 Sebelumnya xmlDoc sudah berisi dengan dokumen “map.xml”.
 38 (d) Membuat variabel yang berisi informasi way pada index tertentu, dan lakukan
 39 *filter* dengan menggunakan fungsi isHighway().
 40 (e) Melakukan *print* pada tabel, beberapa atribut yang ditampilkan adalah id way,
 41 id node pertama, dan id node kedua.

The screenshot shows a web browser window with two tables displayed side-by-side. The left table is titled 'Node' and has columns: 'Node', 'Id', 'Latitude', and 'Longitude'. The right table is titled 'Edge' and has columns: 'Way', 'Id Way', 'Edge', 'Id Node 1', and 'Id Node 2'. Both tables contain 17 rows of data.

Node				Edge				
Node	Id	Latitude	Longitude	Way	Id Way	Edge	Id Node 1	Id Node 2
0	25418868	-6.9064389	107.5976351	0	4567626	0	25433682	25433681
1	25433683	-6.9067659	107.5989458	0	4567626	1	25433681	25433680
2	25433687	-6.9040267	107.5969508	1	4567634	0	25433681	28802396
3	25433688	-6.9039393	107.5963723	2	4625182	0	29376826	364364242
4	25433690	-6.9052824	107.5961768	2	4625182	1	364364242	29376827
5	25433685	-6.9049404	107.5975738	2	4625182	2	29376827	25433690
6	25433678	-6.9039784	107.5985467	3	4627111	0	29356177	29356178
7	25433679	-6.9049265	107.5985843	3	4627111	1	29356178	29356179
8	25433680	-6.9062500	107.5995945	3	4627111	2	29356179	29356180
9	25433681	-6.9055235	107.5989193	3	4627111	3	29356180	25433684
10	25434115	-6.9097812	107.5978508	4	4628057	0	29392373	29392374
11	25500626	-6.9070329	107.6019401	4	4628057	1	29392374	29392377
12	28802396	-6.9055299	107.5976092	6	247058985	0	2325451442	364364086
13	29356177	-6.9102898	107.5994584	6	247058985	1	364364086	364364087
14	29356178	-6.9090157	107.5994925	6	247058985	2	364364087	2325451578
15	29356374	-6.9081596	107.5987289	6	247058985	3	2325451578	2325451571
16	29356381	-6.9082496	107.5977288	9	32388779	0	364364184	364364194
17	29356499	-6.9099650	107.5978505	9	32388779	1	364364194	364364195

Gambar 3.3: Parsing XML menggunakan Javascript

- 1 Hasil dari kode program di atas dapat dilihat pada Gambar 3.3. Pada Gambar 3.3 dapat
- 2 dilihat terdiri dari dua tabel yang menunjukkan informasi node dan edge yang sudah dibaca.
- 3 Tabel node menunjukkan atribut penting yang akan digunakan yaitu id node, *latitude*, dan
- 4 *longitude*. Sedangkan tabel edge menujukkan informasi yang didapatkan dari tag way yang
- 5 sudah dilakukan *filter*, yaitu hanya tag way yang memiliki *child* highway saja yang akan
- 6 digunakan. Pada tabel edge terdapat informasi penting yang akan digunakan, yaitu id way,
- 7 id node pertama, dan id node kedua. Informasi yang sudah didapatkan, selanjutnya akan
- 8 dimodelkan ke dalam bentuk graf yang akan dibahas pada subbab 3.4.

3.3 Menghitung Jarak Antara Dua Titik

- 10 Untuk menghitung jarak antara dua titik dapat menggunakan rumus haversine atau dikenal dengan haversine *formula*. Analisis rumus haversine dilakukan dengan implementasi
- 11 rumus tersebut dengan contoh kasus perhitungan jarak antara koordinat Kota Bandung (-
- 12 6.9167,107.6000) dan koordinat Kota Jakarta (-6.1745,106.8227). Berikut ini adalah rumus
- 13 haversine yang telah diimplementasikan:

```

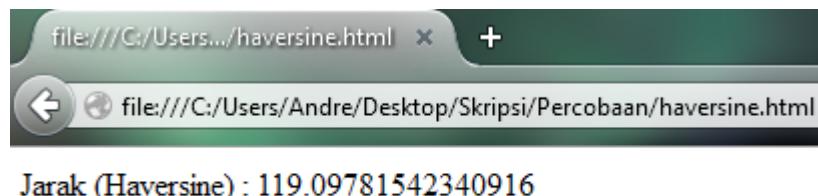
15 function getDistance(lat1,lon1,lat2,lon2){
16     var R = 6371;
17     var dLat = deg2rad(lat2-lat1);
18     var dLon = deg2rad(lon2-lon1);
19     var a =
20         Math.sin(dLat/2) * Math.sin(dLat/2) +
21         Math.cos(deg2rad(lat1)) * Math.cos(deg2rad(lat2)) *
22         Math.sin(dLon/2) * Math.sin(dLon/2);
23     var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));

```

```

1  var d = R * c;
2  return d;
3 }
4
5 function deg2rad(deg){
6   return deg * (Math.PI/180);
7 }
```

8 Hasil yang ditunjukkan dari rumus haversine adalah 119.09781542340916 km dapat dilihat
9 pada Gambar 3.4.



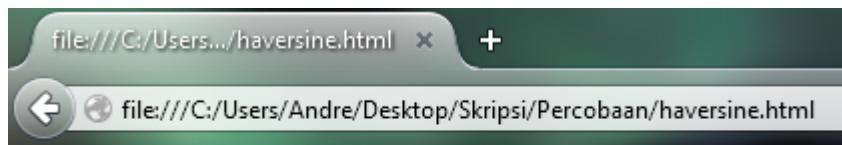
Gambar 3.4: Perhitungan Jarak Dengan Haversine Formula

10 Saat proses analisis, ternyata Google Maps Javascript API menyediakan suatu *library*
11 untuk mengukur jarak antara dua titik. *Library* tersebut adalah *geometry library* yang
12 menyediakan fungsi utilitas yaitu google.maps.geometry.spherical. Untuk menghitung jarak
13 antara dua titik digunakan fungsi computeDistanceBetween(). Sama seperti rumus haversi-
14 ne, analisis fungsi computeDistanceBetween() dilakukan dengan implementasi contoh kasus
15 perhitungan jarak antara koordinat Kota Bandung dan Jakarta. Berikut ini adalah fungsi
16 computeDistanceBetween() yang telah diimplementasikan:

```

17 var jakarta = new google.maps.LatLng(-6.1745,106.8227);
18 var bandung = new google.maps.LatLng(-6.9167,107.6000);
19 var distance = google.maps.geometry.spherical.
20 computeDistanceBetween(jakarta, bandung);
```

21 Hasil yang ditunjukkan dari fungsi computeDistanceBetween() adalah 119.23123264342443
22 km dapat dilihat pada Gambar 3.5.



Gambar 3.5: Perhitungan Jarak Dengan Geometry Library

23 Terdapat perbedaan atau selisih yang dihasilkan oleh rumus haversine dan fungsi com-
24 puteDistanceBetween() sebesar 0.133417220015275 km. Fungsi computeDistanceBetween()
25 yang akan digunakan untuk pembuatan aplikasi, bukan rumus haversine. Hal ini karena
26 fungsi tersebut lebih mudah digunakan.

3.4 Pemodelan OSMXML Menjadi Graf

Pada subbab 3.2, dokumen OSMXML sudah dibaca dan tahap selanjutnya adalah memodelkan data OSMXML tersebut ke dalam bentuk graf. Seperti yang sudah diketahui, graf terdiri dari node dan edge. Informasi node dan edge yang sudah didapatkan akan dimodelkan ke dalam bentuk graf berarah, hal ini karena jalan yang menghubungkan node-node tersebut memiliki arah, baik searah maupun dua arah, arah jalan diketahui dengan melihat informasi tag oneway. Untuk merepresentasikan graf tersebut akan digunakan *adjacency list*. Penggunaan *adjacency list* disebabkan oleh penggunaan memori yang lebih kecil dibandingkan dengan *adjacency matrix*. Pada *adjacency list* memori yang digunakan adalah sebesar jumlah node yang terdapat pada OSMXML, sedangkan pada *adjacency matrix* memori yang digunakan adalah nxn. Berikut ini adalah langkah-langkah yang dilakukan untuk memodelkan OSMXML menjadi graf:

1. Membuat kelas Node dan kelas Neighbor.

```

14     function Node(id, neighbors){
15         this.id = id;
16         this.adjList = neighbors;
17     }
18
19     function Neighbor(vnum, nbr, weight){
20         this.vertexNum = vnum;
21         this.weight = weight;
22         this.next = nbr;
23     }

```

Kedua kelas tersebut digunakan untuk menyimpan informasi id node dan jarak. Id node disimpan pada kelas node menggunakan atribut id, sedangkan jarak disimpan pada kelas neighbor menggunakan atribut weight.

2. Membaca seluruh informasi node dan menyimpan pada kelas node.

```

28     var adjLists = [];
29     for(i=0;i<node.length;i++){
30         adjLists.push(new Node(node[i].getAttribute('id'),null));
31     }

```

Berikut ini adalah tahap-tahap yang dilakukan untuk menyimpan seluruh informasi tersebut:

- (a) Membuat variabel adjLists, variabel ini bertipe array.
- (b) Lakukan pengulangan untuk memasukkan informasi node satu persatu.
3. Membuat fungsi untuk menentukan arah pada setiap node yang terhubung dengan node lain.

```

1   function wayDirection(way,index){
2       var tag = way[index].getElementsByTagName("tag");
3       for (hg=0;hg<tag.length;hg++){
4           if(tag[hg].getAttribute('k') == "oneway"){
5               return tag[hg].getAttribute('v');
6           }
7       }
8       return false;
9   }

```

10 Berikut ini adalah tahap-tahap yang dilakukan:

- 11 (a) Membuat fungsi wayDirection() dengan parameter *input* way dan index. Parameter way berisi informasi seluruh way, sedangkan parameter index menunjukkan tag way pada index tersebut yang akan dicari informasi arahnya.
 - 14 (b) Membuat variabel yang menyimpan informasi way pada index yang dicari, pada kode program variabel tersebut diberi nama “tag”.
 - 16 (c) Lakukan pengulangan untuk mencari setiap *child* pada variabel “tag” untuk mencari atribut “k=oneway”, jika ditemukan fungsi akan mengembalikan *value* dari *key* tersebut dan *false* jika tidak ditemukan.
- 19 4. Membuat fungsi untuk mendapatkan informasi koordinat node (*latitude* dan *longitude*).

```

21     function getLatByAtt(str){
22         for (n=0;n<node.length;n++){
23             if(node[n].getAttribute('id') == str){
24                 return node[n].getAttribute('lat');
25             }
26         }
27         return -1;
28     }
29
30     function getLonByAtt(str){
31         for (m=0;m<node.length;m++){
32             if(node[m].getAttribute('id') == str){
33                 return node[m].getAttribute('lon');
34             }
35         }
36         return -1;
37     }

```

38 Kode di atas terdiri dari dua fungsi, fungsi getLatByAtt() mencari informasi *latitude* 39 dan fungsi getLatByAtt() mencari informasi *longitude*. Kedua fungsi tersebut mela-40 kukan pencarian node berdasarkan parameter *input* str yang merupakan id node, jika

1 ditemukan fungsi akan mengembalikan nilai *latitude* atau *longitude* dan mengembalikan
2 nilai -1 jika tidak ditemukan.

3 5. Membaca seluruh informasi edge yang terdapat pada tag way

```

4   for (i=0;i<way.length;i++){
5     nd = way[i].getElementsByTagName("nd");
6     if(isHighway(way,i)){
7       oneway = wayDirection(way,i);
8       for (j=0;j<nd.length-1;j++){
9         //cari jarak
10        v1 = indexForId(adjLists,nd[j].getAttribute('ref'));
11        v2 = indexForId(adjLists,nd[j+1].getAttribute('ref'));
12        lat1 = getLatByAtt(nd[j].getAttribute('ref'));
13        lon1 = getLonByAtt(nd[j].getAttribute('ref'));
14        lat2 = getLatByAtt(nd[j+1].getAttribute('ref'));
15        lon2 = getLonByAtt(nd[j+1].getAttribute('ref'));
16
17        point1 = new google.maps.LatLng(lat1,lon1);
18        point2 = new google.maps.LatLng(lat2,lon2);
19        distance = google.maps.geometry.spherical.
20        computeDistanceBetween(point1,point2);
21
22        //memasukkan informasi edge
23        if(oneway == "yes"){
24          adjLists[v1].adjList = new Neighbor(v2,
25            adjLists[v1].adjList,distance);
26        }else if(oneway == "no"){
27          adjLists[v1].adjList = new Neighbor(v2,
28            adjLists[v1].adjList,distance);
29          adjLists[v2].adjList = new Neighbor(v1,
30            adjLists[v2].adjList,distance);
31        }else if(oneway == "-1"){
32          adjLists[v2].adjList = new Neighbor(v1,
33            adjLists[v2].adjList,distance);
34        }else{
35          adjLists[v1].adjList = new Neighbor(v2,
36            adjLists[v1].adjList,distance);
37          adjLists[v2].adjList = new Neighbor(v1,
38            adjLists[v2].adjList,distance);
39        }
40      }
41    }
42  }

```

43 Berikut ini adalah tahap-tahap yang dilakukan:

- 1 (a) Melakukan pengulangan untuk setiap way.
- 2 (b) Melakukan *filter* untuk setiap way, hanya “highway” saja yang akan dimasukkan
3 informasinya ke dalam *adjacency list*.
- 4 (c) Melakukan pemanggilan fungsi wayDirection() dan memasukkan nilai didapatkan
5 dari fungsi tersebut ke variabel, pada kode di atas menggunakan variabel
6 “oneway”.
- 7 (d) Cari jarak antar node.
- 8 (e) Memasukkan informasi edge berdasarkan arah yang dimasukkan ke dalam variabel
9 “oneway”.

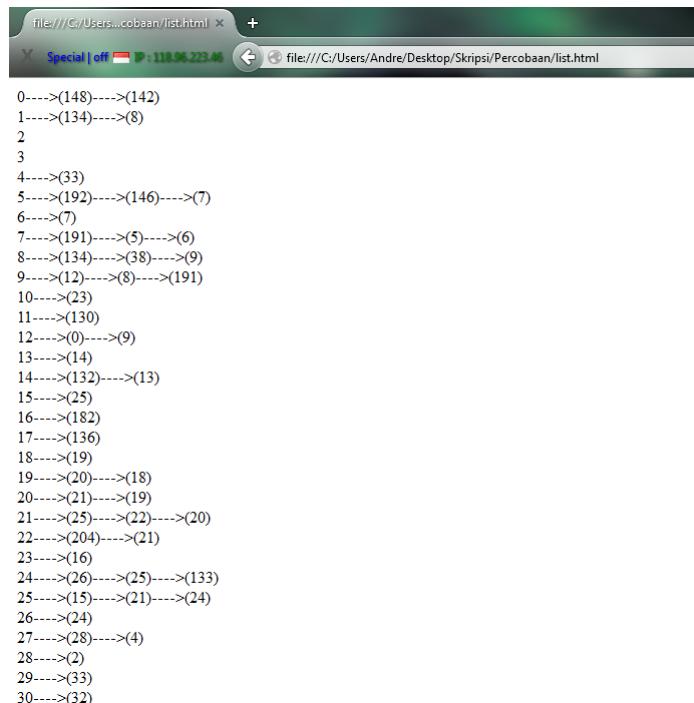
10 6. Membuat fungsi untuk menampilkan graf ke layar

```

11     function print(list){
12       for (j=0; j < list.length; j++){
13         document.write(j);
14         for (nbr=list[j].adjList; nbr != null;nbr=nbr.next) {
15           document.write("---->");
16           document.write('('+nbr.vertexNum+')');
17         }
18         document.write("<br>");
19     }
20 }

```

21 Fungsi akan menerima parameter *input* berupa array, yaitu variabel adjLists. Selanjutnya dilakukan pembacaan setiap index array dan hasilnya ditampilkan di layar.



Gambar 3.6: Pemodelan OSMXML menjadi Graf

- 1 Hasil dari kode program di atas dapat dilihat pada Gambar 3.6. Pada Gambar 3.6, data pada
 2 OSMXML sudah dimodelkan ke dalam bentuk graf menggunakan representasi *adjacency list*.
 3 Tahap selanjutnya selanjutnya akan divisualisasikan menggunakan Google Maps Javascript
 4 API yang akan dibahas pada subbab 3.5.

5 **3.5 Visualisasi**

6 Data OSMXML yang sudah dimodelkan ke dalam bentuk graf, selanjutnya akan divisualisa-
 7 sikan menggunakan Google Maps Javascript API. Peta yang akan ditampilkan dalam bentuk
 8 *roadmap*, karena peta jenis ini memberikan informasi mengenai nama jalan, sehingga peta
 9 jenis ini lebih cocok untuk aplikasi yang akan dibangun. Berikut ini langkah-langkah yang
 10 dilakukan untuk membuat visualisasi graf:

- 11 1. Melakukan *load* Google Maps Javascript API.

```
12 <script src="https://maps.googleapis.com/maps/api
13 /js?v=3.exp&libraries=geometry"></script>
```

- 14 2. Membuat elemen div sebagai wadah atau tempat untuk peta.

```
15 <div id="googleMap" style="width:75%;height:600px;float:left"></div>
```

- 16 3. Membuat objek google.maps.Map dengan parameter *input map options* (variabel map-
 17 Prop). Peta disisipkan pada elemen div yang memiliki id googleMap.

```
18 var mapProp = {
19   center:new google.maps.LatLng(-6.906845432118958,107.59851515293121),
20   zoom:17,
21   mapTypeId:google.maps.MapTypeId.ROADMAP
22 };
23
24 var map=new google.maps.Map(document.getElementById("googleMap"),mapProp);
```

25 Berikut ini tahap-tahap yang dilakukan:

- 26 (a) Membuat variabel untuk mendefinisikan properti peta, pada kode di atas dilala-
 27 kukan pengaturan titik tengah, zoom pada level 17, dan tipe peta yaitu “ROAD-
 28 MAP”.

- 29 (b) Membuat objek google.maps.Map, objek tersebut memiliki parameter properti
 30 peta (variabel mapProp), selanjutnya peta disisipkan pada elemen div yang me-
 31 miliki id googleMap.

- 32 4. Membuat objek *marker* untuk setiap node pada peta.

```
33 for (j=0;j<nd.length;j++){
34   id = uniqueId();
35   marker = new google.maps.Marker({
```

```

1      id: id,
2      position: new google.maps.LatLng(getLatByAtt(nd[j].getAttribute('ref')),
3          getLonByAtt(nd[j].getAttribute('ref'))), map: map,
4      icon: image,
5  });
6  markers[id] = marker;

```

7 Dilakukan pengulangan untuk setiap node dan membuat objek *marker*, membaca atribut *latitude* dan *longitude*, lalu menampilkan seluruh *marker* pada peta.

8 5. Membuat objek *polyline* yang menghubungkan setiap node pada peta.

```

10 for (k=0;k<nd.length-1;k++){
11     line = new google.maps.Polyline({
12         path: [new google.maps.LatLng(getLatByAtt(nd[k].getAttribute('ref')),
13             getLonByAtt(nd[k].getAttribute('ref'))), new
14             google.maps.LatLng(getLatByAtt(nd[k+1].getAttribute('ref')),
15                 getLonByAtt(nd[k+1].getAttribute('ref')))],
16             strokeColor: "#000000",
17             strokeOpacity: 1.0,
18             strokeWeight: 3,
19             map: map
20     });
21 }

```

22 6. Membuat fungsi untuk menambahkan *info window* pada setiap node untuk memberikan informasi id dan index node.

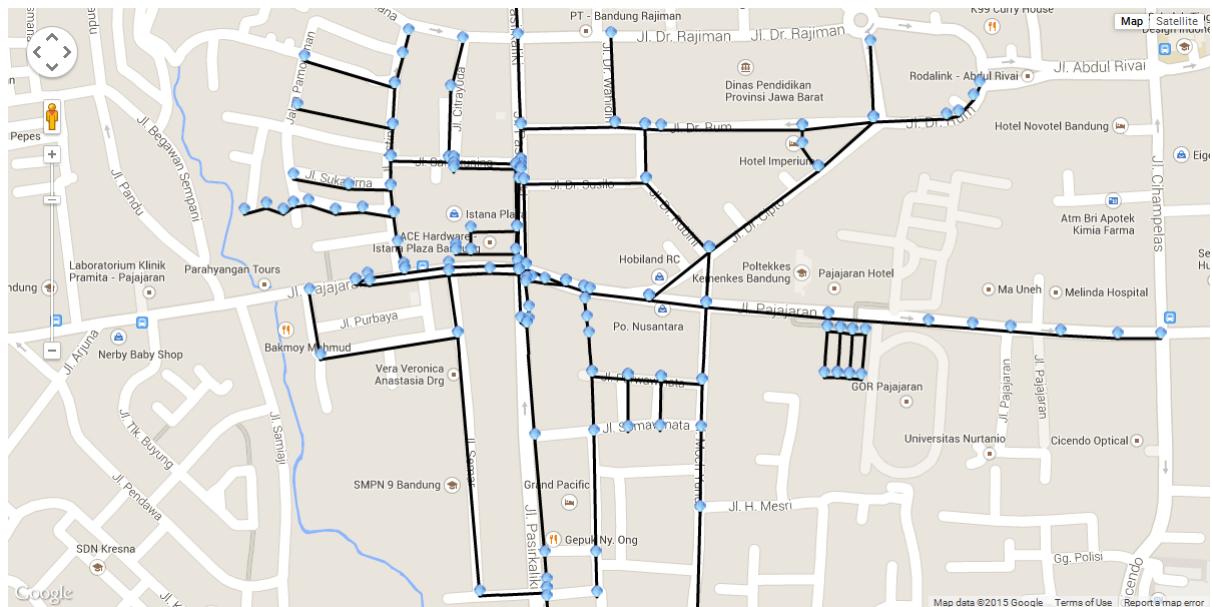
```

24 function addInfoWindow(marker, message) {
25     var infoWindow = new google.maps.InfoWindow({
26         content: message
27     });
28
29     google.maps.event.addListener(marker, 'click', function () {
30         infoWindow.open(map, marker);
31     });
32 }

```

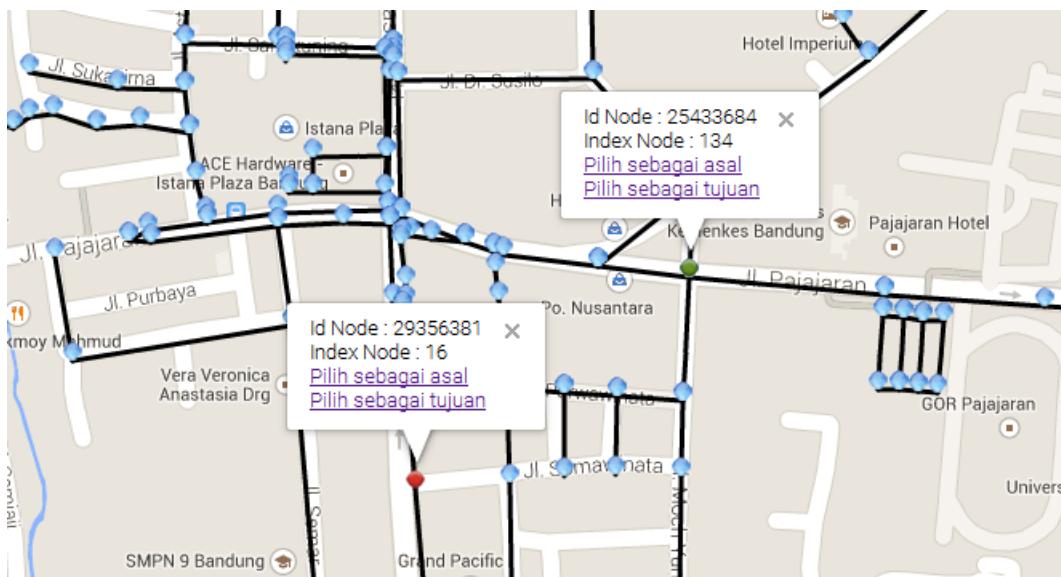
33 Berikut ini adalah tahap-tahap yang dilakukan:

- 34 (a) Membuat fungsi *addInfoWindow* dengan parameter *input* yaitu *marker* dan pesan.
- 35
- 36 (b) Membuat objek *google.maps.InfoWindow*.
- 37 (c) Menambahkan objek *google.maps.InfoWindow* pada *listener* yang berasosiasi dengan parameter *marker*.
- 38



Gambar 3.7: Visualisasi

- 1 Hasil dari langkah-langkah di atas dapat dilihat pada Gambar 3.7. Setiap node pada graf yang sudah dilakukan *filter* akan diwakili oleh marker. Setiap marker tersebut akan memiliki *info window* yang akan memberikan informasi seperti id node, index node pada graf, dan dua buah *hyperlink* yang berfungsi untuk menjadikan marker yang dipilih menjadi asal atau tujuan. Contoh *info window* yang ditampilkan pada peta dapat dilihat pada Gambar 3.8.
- 2 Setiap edge pada graf akan menjadi garis pada peta yang dibuat menggunakan *polyline*.



Gambar 3.8: Info Window

6

3.6 Algoritma Dijkstra

- 7 Untuk pencarian rute terdekat, aplikasi menggunakan algoritma dijkstra. Algoritma tersebut diimplementasikan pada graf yang sudah dimodelkan sebelumnya. Berikut ini adalah langkah-langkah algoritma dijkstra yang digunakan:

1 1. Membuat kelas Dijkstra dan inisialisasi variabel.

```
2     function Dijkstra(){
3         var INFINITY = 1/0;
4         this.nodes = {};
5     }
```

6 2. Membuat fungsi untuk menambahkan node.

```
7     this.addNode = function(name,edges){
8         this.nodes[name] = edges;
9     }
```

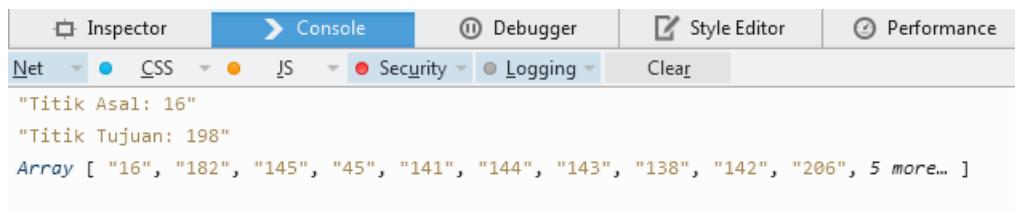
10 3. Implementasi algoritma dijkstra

```
11     this.shortestPath = function(asal, tujuan){
12         var PQ = new PriorityQueue();
13         var distances = {};
14         var previous = {};
15         var path = [];
16         var smallest, node, neighbor, alt;
17
18         for(node in this.nodes){
19             if(node === asal){
20                 distances[node] = 0;
21                 PQ.enqueue(0, node);
22             }
23             else{
24                 distances[node] = INFINITY;
25                 PQ.enqueue(INFINITY, node);
26             }
27             previous[node] = null;
28         }
29
30         while(!PQ.isEmpty()){
31             smallest = PQ.dequeue();
32             if(smallest === tujuan){
33                 while(previous[smallest]){
34                     path.push(smallest);
35                     smallest = previous[smallest];
36                 }
37                 break;
38             }
39
40             if(!smallest || distances[smallest] === INFINITY){
41                 continue;
```

```

1         }
2
3     for(neighbor in this.nodes[smallest]){
4         alt = distances[smallest] + this.nodes[smallest][neighbor];
5         if(alt < distances[neighbor]) {
6             distances[neighbor] = alt;
7             previous[neighbor] = smallest;
8             PQ.enqueue(alt, neighbor);
9         }
10    }
11 }
12 return path;
13 }
```

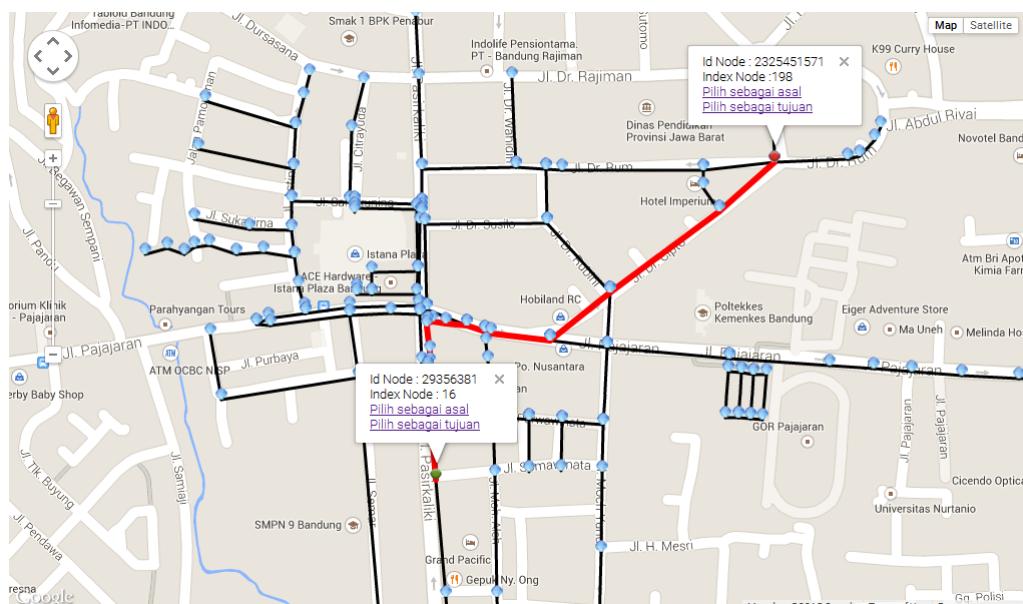
14 Fungsi dijkstra akan menerima *input* berupa objek “edge” yang berisi informasi dari graf,
15 selanjutnya fungsi akan mengeluarkan *output* yaitu jalur terpendek dari satu titik ke titik
16 lain dalam bentuk array. Contoh kasus yang digunakan, yaitu mencari rute terdekat dari
17 titik asal “16” dan titik tujuan “198”, output yang dihasilkan dapat dilihat pada Gambar 3.9.
Visualisasi rute terdekat dapat dilihat pada Gambar 3.10.



```

Inspector Console Debugger Style Editor Performance
Net CSS JS Security Logging Clear
"Titik Asal: 16"
"Titik Tujuan: 198"
Array [ "16", "182", "145", "45", "141", "144", "143", "138", "142", "206", 5 more... ]
```

Gambar 3.9: Keluaran fungsi Dijkstra



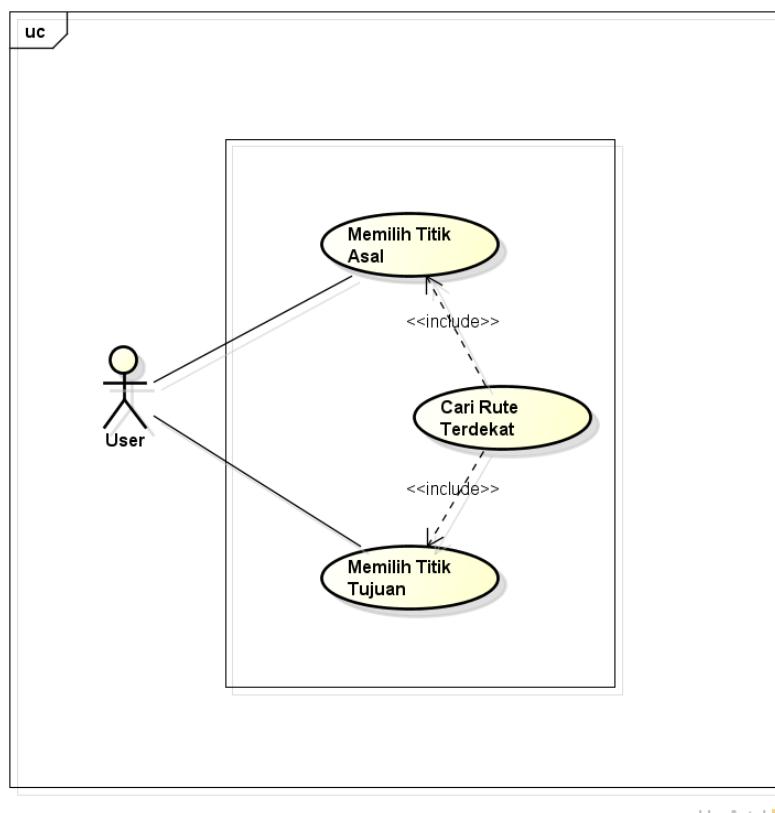
Gambar 3.10: Visualisasi Rute Terdekat

¹ 3.7 Analisis Berorientasi Objek

² Aplikasi pencarian rute terdekat yang dibangun akan mengolah data yang disediakan oleh
³ OpenStreetMap dalam bentuk XML dan memodelkannya ke dalam bentuk graf. *User* dapat
⁴ memilih titik asal dan titik tujuan, selanjutnya akan diimplementasikan algoritma Dijkstra
⁵ untuk mencari rute terdekat antara kedua titik tersebut dan menunjukkan hasilnya secara vi-
⁶ sual menggunakan Google Maps Javascript API. Pada subbab ini akan membahas interaksi
⁷ antara *user* dengan sistem yaitu menggunakan diagram *use case* dan skenario. Setelah-
⁸ nya, dibahas juga diagram kelas untuk menunjukkan kelas-kelas yang ada pada sistem dan
⁹ hubungannya.

¹⁰ 3.7.1 Diagram *Use Case*

¹¹ Diagram *use case* adalah pemodelan yang berfungsi memperjelas interaksi antara aktor atau
¹² *user* dengan sistem. Diagram *use case* aplikasi pencarian rute terdekat dapat dilihat pada
[Gambar 3.11](#).



Gambar 3.11: Diagram *use case*

¹³
¹⁴ Berdasarkan analisis yang telah dilakukan, maka *user* dapat melakukan interaksi sebagai
¹⁵ berikut:

¹⁶ 1. Memilih titik asal.

¹⁷ *User* dapat menekan salah satu *marker* yang ada pada peta dan menekan *link* "pilih
¹⁸ titik asal". Selanjutnya, akan ditampilkan informasi titik yang sudah dipilih pada sisi
¹⁹ kanan layar.

- ¹ 2. Memilih titik tujuan.

² *User* dapat menekan salah satu *marker* yang ada pada peta dan menekan *link* “pilih titik tujuan”. Selanjutnya, akan ditampilkan informasi titik yang sudah dipilih pada sisi kanan layar.

- ⁵ 3. Mencari rute terdekat dari titik asal ke titik tujuan.

⁶ *User* dapat menekan tombol “Cari” untuk mencari rute terdekat dari kedua titik yang telah dipilih sebelumnya.

⁸ 3.7.2 Skenario

- ⁹ Berikut ini adalah skenario untuk setiap *use case*:

1. Skenario memilih titik asal dapat dilihat pada Tabel 3.1.

Tabel 3.1: Skenario memilih titik asal

Nama	Memilih titik asal
Aktor	User
Kondisi Awal	Titik asal belum terpilih
Kondisi Akhir	Titik asal sudah terpilih dan ditampilkan di sisi kanan layar
Skenario	User menekan marker pada peta dan menekan link pilih titik asal
Deskripsi	User memilih titik pada peta untuk dijadikan titik asal
Eksepsi	-

¹⁰

2. Skenario memilih titik tujuan dapat dilihat pada Tabel 3.2.

Tabel 3.2: Skenario memilih titik tujuan

Nama	Memilih titik tujuan
Aktor	User
Kondisi Awal	Titik tujuan belum terpilih
Kondisi Akhir	Titik tujuan sudah terpilih dan ditampilkan di sisi kanan layar
Skenario	User menekan marker pada peta dan menekan link pilih titik tujuan
Deskripsi	User memilih titik pada peta untuk dijadikan titik tujuan
Eksepsi	-

¹¹

3. Skenario cari rute terdekat dapat dilihat pada Tabel 3.3.

Tabel 3.3: Skenario cari rute terdekat

Nama	Cari rute terdekat
Aktor	User
Kondisi Awal	Titik asal dan titik tujuan sudah terpilih
Kondisi Akhir	Sistem menampilkan rute terdekat menggunakan polyline
Skenario	User menekan tombol Cari!
Deskripsi	User menekan tombol Cari! dan sistem menampilkan rute terdekat
Eksepsi	Jika user belum memilih titik asal atau tujuan akan ditampilkan alert atau peringatan

¹²

3.7.3 Diagram Kelas Sederhana

Pada bagian ini, akan dijelaskan diagram kelas yang digunakan untuk memenuhi kebutuhan *user* yang sudah dijelaskan pada bagian diagram *use case* dan skenario. Berikut ini adalah diagram kelas sederhana yang dapat dilihat pada Gambar 3.12. Berikut ini adalah penjelasan dari setiap kelas yang terdapat pada diagram kelas sederhana:

- Kelas XMLHttpRequest

Kelas ini berfungsi untuk melakukan *load* dokumen OSMXML.

- Kelas Node dan Neighbor

Kedua kelas ini akan menyimpan informasi yang didapatkan dari OSMXML sebagai representasi dari graf yaitu *adjacency list*.

- Kelas Graph

Kelas ini berfungsi untuk mengubah informasi yang didapatkan dari OSMXML menjadi graf.

- Kelas Map

Kelas ini berfungsi untuk melakukan *generate* peta, visualisasi graf, dan visualisasi rute terdekat.

- Kelas google.maps.Map

Kelas ini berfungsi untuk membuat objek peta.

- Kelas google.maps.Marker

Kelas ini berfungsi untuk membuat objek *marker* yang akan digunakan sebagai visualisasi node pada peta.

- Kelas google.maps.Polyline

Kelas ini berfungsi untuk membuat objek *polyline* yang akan digunakan sebagai visualisasi edge pada peta.

- Kelas google.maps.InfoWindow

Kelas ini berfungsi untuk membuat objek *InfoWindow* yang akan disisipkan pada setiap objek *marker*.

- Kelas google.maps.LatLng

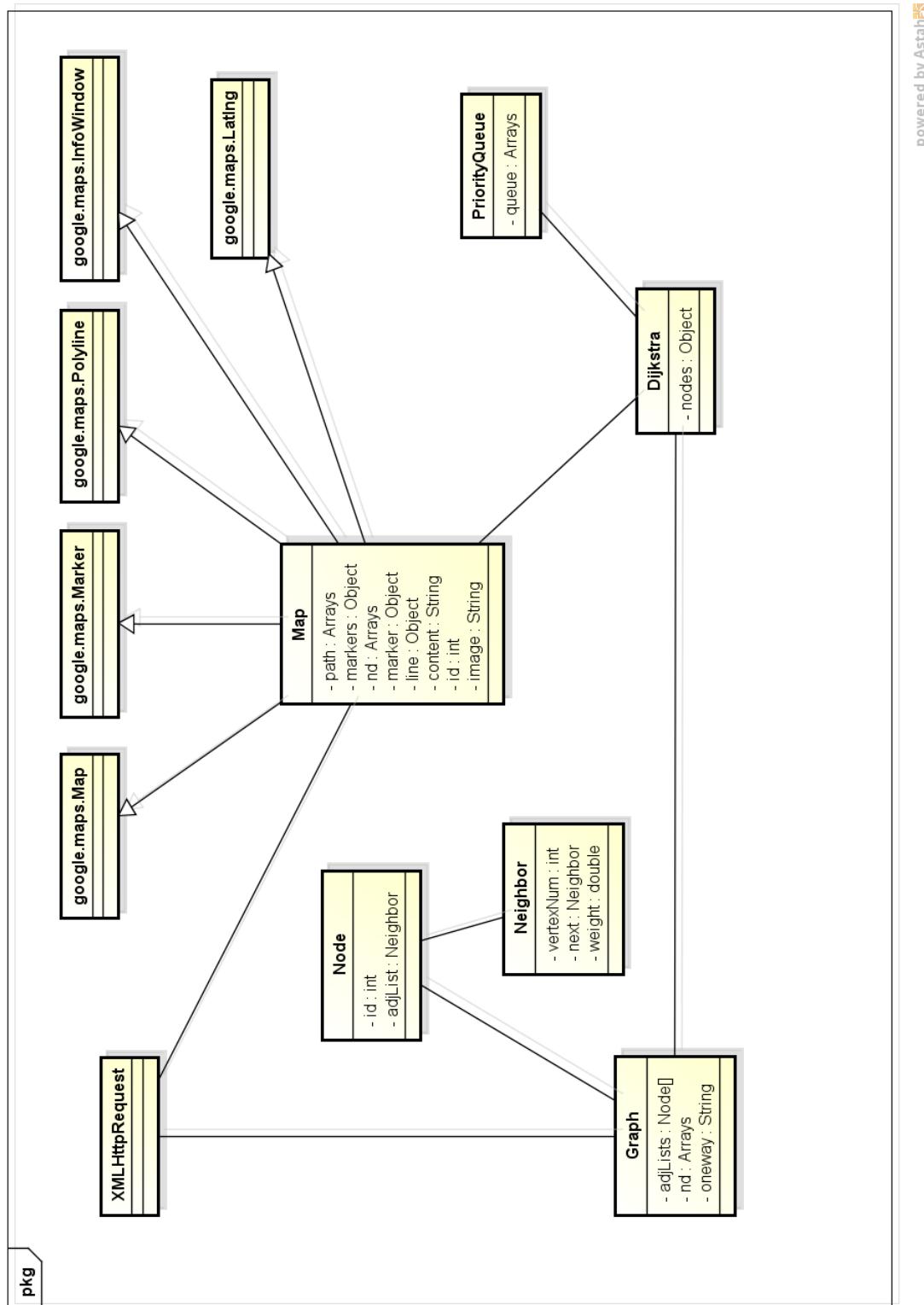
Kelas ini berfungsi untuk membuat objek *Latlng*. *Latlng* merupakan objek yang berisi informasi koordinat (*latitude* dan *longitude*).

- Kelas Dijkstra

Kelas ini berfungsi untuk mencari rute terdekat berdasarkan *input* titik asal dan titik tujuan.

- Kelas PriorityQueue

Kelas ini merupakan struktur data *queue* yang digunakan pada kelas dijkstra.



Gambar 3.12: Diagram Kelas Sederhana

¹

BAB 4

²

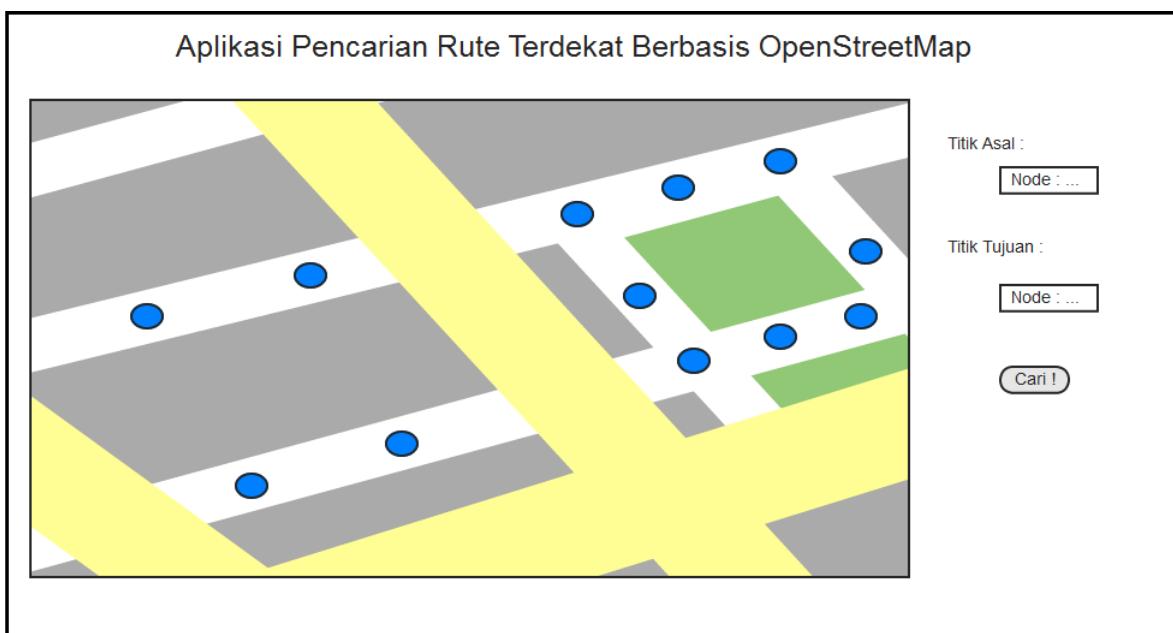
PERANCANGAN

³ Pada bab ini akan dijelaskan perancangan aplikasi pencarian rute terdekat berbasis Open-
⁴ StreetMap. Pada bagian pertama akan dijelaskan perancangan antar muka untuk aplikasi
⁵ yang akan dibangun. Diawali *user* dengan membuka aplikasi, selanjutnya *user* dapat memi-
⁶ lih titik-titik yang terdapat pada peta sebagai titik asal ataupun titik tujuan. Setelah itu,
⁷ *user* dapat mencari rute terdekat antara kedua titik tersebut.

⁸ Pada bagian kedua akan dijelaskan rancangan untuk aplikasi agar dapat menjalankan
⁹ fungsinya melalui diagram kelas. Pada bagian ketiga akan dijelaskan bagaimana alur aplikasi
¹⁰ dari *user* hingga mengeluarkan *output* yaitu rute terdekat menggunakan diagram *sequence*.

¹¹ 4.1 Perancangan Antar Muka

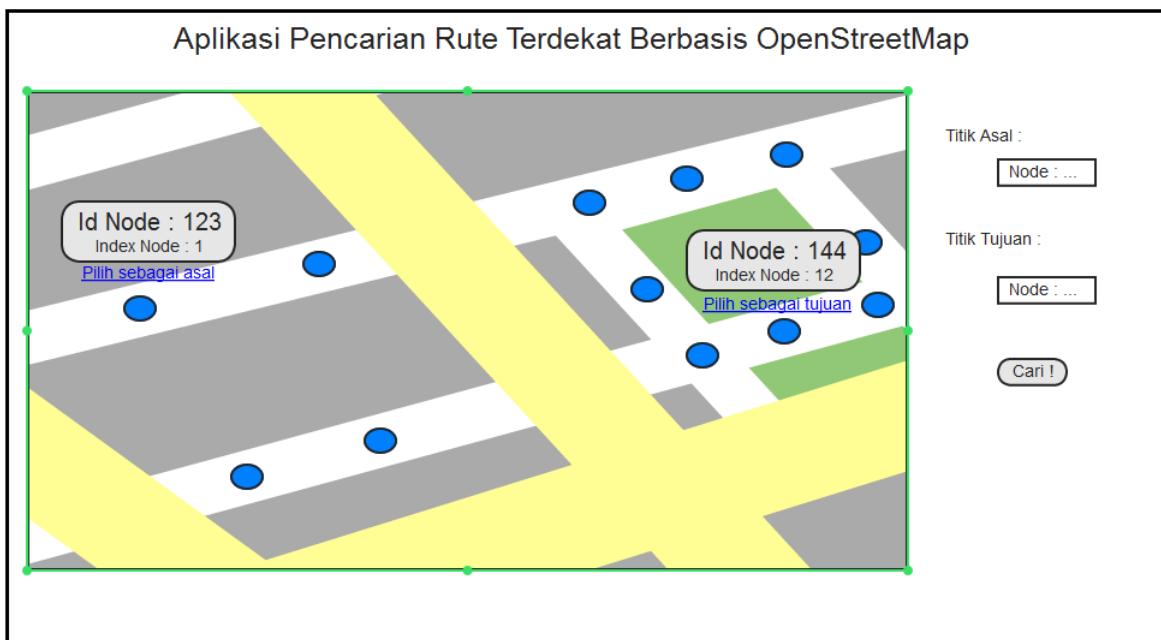
¹² Pada subbab ini akan dijelaskan rancangan antar muka untuk aplikasi pencarian rute ter-
dekat. Aplikasi yang dibangun akan memiliki tampilan awal seperti pada Gambar 4.1.



Gambar 4.1: Rancangan Antar Muka Awal Aplikasi

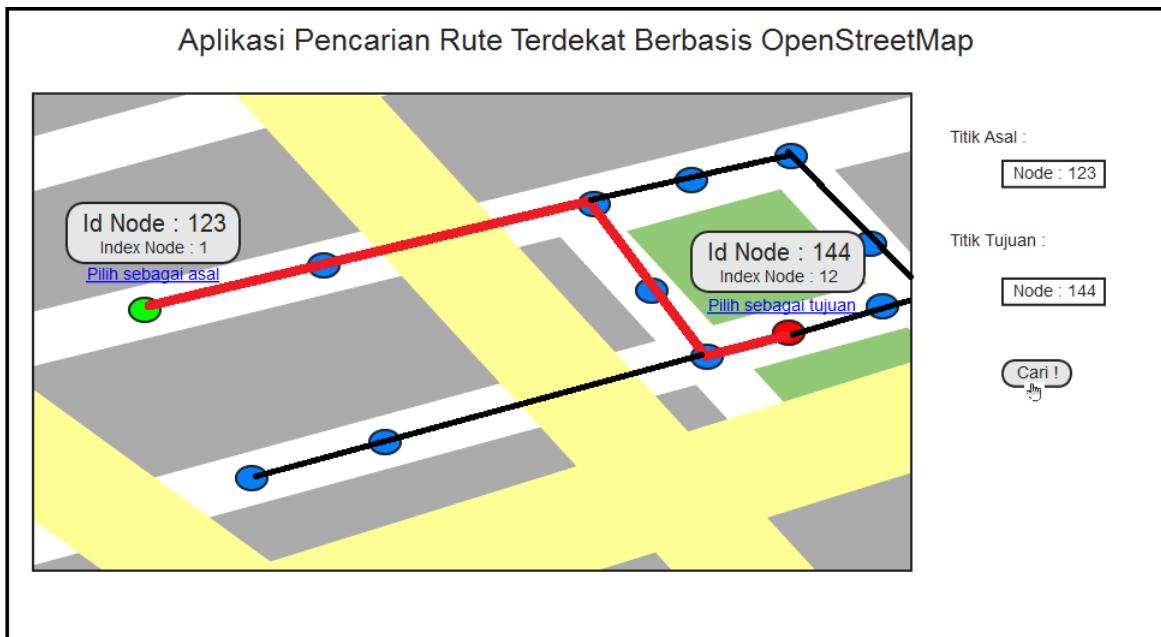
¹³

¹⁴ Setelah halaman awal terbuka, *User* dapat menekan setiap titik tersebut dan akan me-
¹⁵ nampilkan *info window* yang memberikan informasi titik beserta *link*. *User* dapat menekan
¹⁶ *link* tersebut untuk menjadikannya sebagai titik asal ataupun titik tujuan. Rancangan antar
¹⁷ muka saat user memilih titik, dapat dilihat pada Gambar 4.2.



Gambar 4.2: Rancangan Pemilihan Titik

- 1 Setelah user memilih titik asal dan titik tujuan, user dapat menekan tombol “Cari!”
- 2 untuk melihat rute terdekat antara kedua titik tersebut. Rute tersebut divisualisasikan
- 3 dengan menggunakan *polyline* yang berwarna merah. Rancangan pada tahap ini dapat dilihat pada Gambar 4.3.



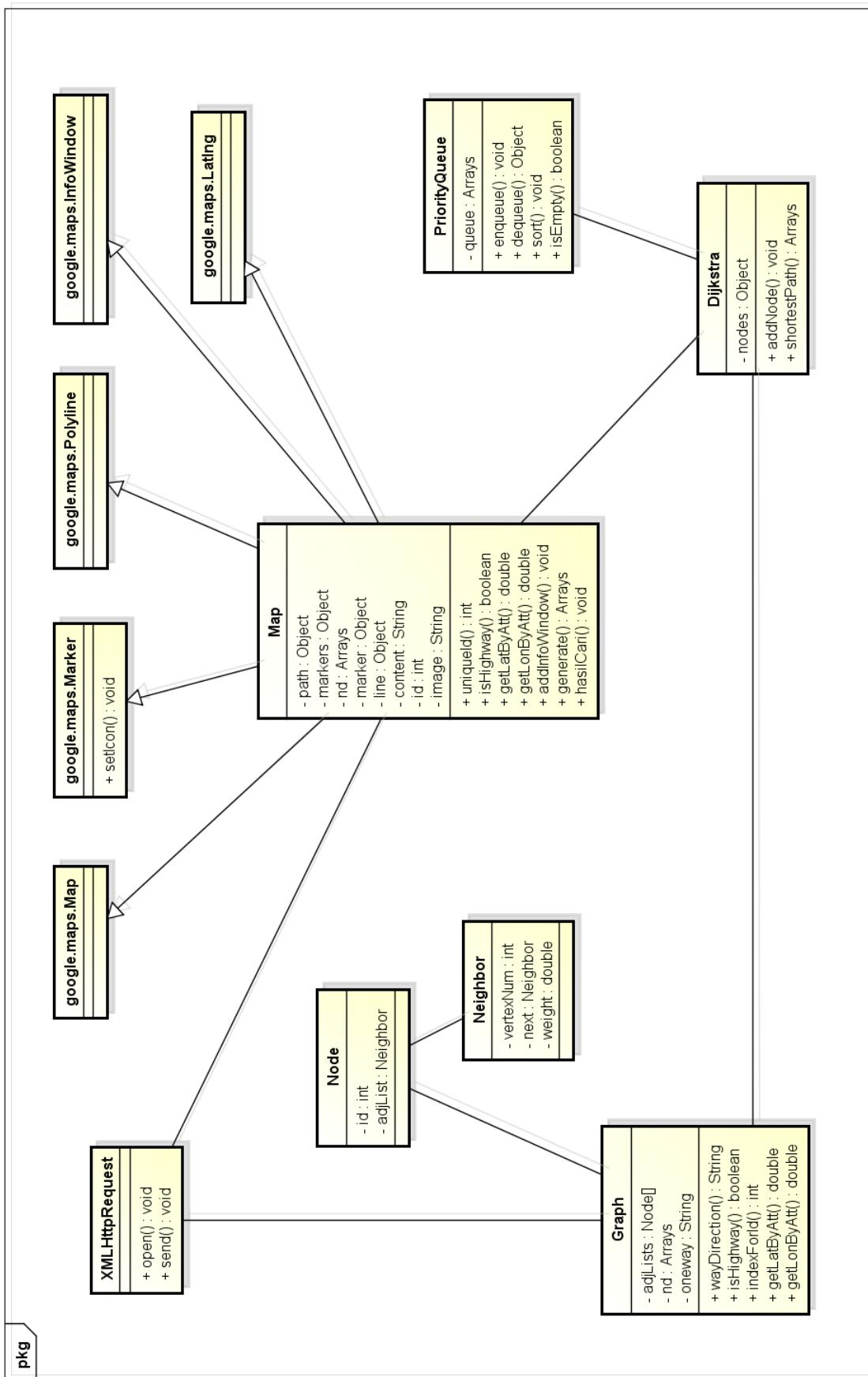
Gambar 4.3: Rancangan Cari Rute

4

5 4.2 Perancangan Kelas

- 6 Pada bagian ini akan dibahas perancangan kelas yang dilakukan untuk aplikasi yang akan
- 7 dibangun. Perancangan kelas bertujuan untuk menjelaskan seluruh atribut dan fungsi yang
- 8 akan diimplementasikan pada setiap kelas yang sudah dibahas pada bab analisis. Peran-

- 1 cangan kelas yang dibuat dapat dilihat pada Gambar 4.4. Berikut ini adalah penjelasan
2 dari setiap kelas beserta atribut dan fungsi yang digunakan:
- 3 ● Kelas XMLHttpRequest
- 4 Kelas ini berfungsi untuk melakukan *load* dokumen OSMXML.
- 5 – Fungsi
- 6 * open()
7 Fungsi open() digunakan untuk membuka dokumen OSMXML.
- 8 * send()
9 Fungsi send() digunakan untuk mengirimkan dokumen OSMXML yang sudah
10 dibuka.
- 11 ● Kelas Node dan Neighbor
- 12 Kedua kelas ini akan menyimpan informasi yang didapatkan dari OSMXML sebagai
13 representasi dari graf yaitu *adjacency list*.
- 14 – Atribut Kelas Node
- 15 * id : int
16 Atribut id menyimpan id node yang didapatkan dari dokumen OSMXML.
- 17 * adjList : Neighbor
18 Atribut adjList menyimpan objek Neighbor.
- 19 – Atribut Kelas Neighbor
- 20 * vertexNum : int
21 Atribut vertexNum menyimpan index dari node.
- 22 * next : Neighbor
23 Atribut next menyimpan objek Neighbor.
- 24 * weight : double
25 Atribut weight menyimpan jarak antar node.
- 26 ● Kelas Graph
- 27 Kelas ini berfungsi untuk mengubah informasi yang didapatkan dari OSMXML menjadi graf.
- 28 – Atribut
- 29 * adjLists : Array of Node
30 Atribut adjLists merupakan *array* yang menyimpan objek node.
- 31 * nd : Array
32 Atribut nd menyimpan informasi berupa id node yang terdapat pada tag way
33 di dalam dokumen OSMXML, atribut nd bertipe array.
- 34 * oneway : String
35 Atribut nd menyimpan informasi berupa *value* dari *key* oneway yang terdapat
36 pada tag way di dalam dokumen OSMXML, atribut oneway bertipe String.



Gambar 4.4: Diagram Kelas

1 – Fungsi

2 * wayDirection() : String
3 Fungsi ini digunakan untuk mengetahui arah pada setiap node berdasarkan
4 value yang terdapat pada OSMXML.

5 * isHighway() : boolean
6 Fungsi ini digunakan untuk melakukan *filter* pada tag way, yaitu hanya way
7 yang bertipe “Highway” saja yang akan digunakan pada pemodelan graf.

8 * indexForId() : int
9 Fungsi ini digunakan untuk mengetahui index node pada graf berdasarkan id
10 node.

11 * getLatByAtt() : double
12 Fungsi ini digunakan untuk mendapatkan informasi *latitude* pada sebuah
13 node berdasarkan atributnya, yaitu id node.

14 * getLonByAtt() : double
15 Fungsi ini digunakan untuk mendapatkan informasi *longitude* pada sebuah
16 node berdasarkan atributnya, yaitu id node.

17 ● Kelas Map

18 Kelas ini berfungsi untuk melakukan *generate* peta, visualisasi graf, dan visualisasi
19 rute terdekat.

20 – Atribut

21 * path : Object
22 Atribut path adalah variabel yang menyimpan informasi titik koordinat da-
23 lam bentuk objek, atribut path digunakan untuk pembuatan *marker* dan
24 .

25 * markers : Object
26 Atribut ini menyimpan seluruh objek *marker*.

27 * nd : Arrays
28 Atribut nd menyimpan informasi berupa id node yang terdapat pada tag way
29 di dalam dokumen OSMXML, atribut nd bertipe array.

30 * marker : Object
31 Atribut ini menyimpan objek *marker* untuk ditampilkan pada peta.

32 * line : Object
33 Atribut ini menyimpan objek *polyline* untuk ditampilkan pada peta.

34 * content : String
35 Atribut ini merupakan isi dari *info window* yang disisipkan pada setiap mar-
36 ker. Atribut ini berisi id node, index node, *link* titik asal, dan *link* titik
37 tujuan.

38 * id : int
39 Atribut ini merupakan id yang digunakan untuk melakukan *generate* id pada
40 fungsi *uniqueId*.

```
1      * image : String  
2          Atribut ini menyimpan alamat dari image atau gambar yang digunakan oleh  
3          marker.  
  
4      – Fungsi  
5          * uniqueId() : id  
6              Fungsi ini digunakan untuk melakukan generate id, id tersebut digunakan  
7              oleh marker.  
8          * isHighway() : boolean  
9              Fungsi ini digunakan untuk melakukan filter pada tag way, hanya way yang  
10             bertipe “Highway” saja yang digunakan. Fungsi ini digunakan pada saat  
11             pembuatan objek marker pada peta.  
12          * getLatByAtt() : double  
13              Fungsi ini digunakan untuk mendapatkan informasi latitude pada sebuah  
14              node berdasarkan atributnya, yaitu id node.  
15          * getLonByAtt() : double  
16              Fungsi ini digunakan untuk mendapatkan informasi longitude pada sebuah  
17              node berdasarkan atributnya, yaitu id node.  
18          * addInfoWindow()  
19              Fungsi ini digunakan untuk menambahkan objek info window pada setiap  
20              marker.  
21          * generate() : Arrays  
22              Fungsi ini digunakan untuk menampilkan secara keseluruhan marker dan  
23              polyline pada peta. Fungsi akan mengembalikan objek marker di dalam  
24              bentuk array.  
25          * hasilCari()  
26              Fungsi ini digunakan untuk melakukan visualisasi rute terdekat menggunakan  
27              polyline.  
  
28      • Kelas google.maps.Map  
29          Kelas ini berfungsi untuk membuat objek peta.  
  
30      • Kelas google.maps.Marker  
31          Kelas ini berfungsi untuk membuat objek marker yang akan digunakan sebagai visu-  
32          alisasi node pada peta.  
  
33      – Fungsi  
34          * SetIcon()  
35              Fungsi ini digunakan untuk mengubah icon dari marker.  
  
36      • Kelas google.maps.Polyline  
37          Kelas ini berfungsi untuk membuat objek polyline yang akan digunakan sebagai visu-  
38          alisasi edge pada peta.  
  
39      • Kelas google.maps.InfoWindow  
40          Kelas ini berfungsi untuk membuat objek InfoWindow yang akan disisipkan pada  
41          setiap objek marker.
```

- 1 ● Kelas google.maps.LatLng
- 2 Kelas ini berfungsi untuk membuat objek LatLng. LatLng merupakan objek yang berisi
- 3 informasi koordinat (*latitude* dan *longitude*).
- 4 ● Kelas Dijkstra
- 5 Kelas ini berfungsi untuk mencari rute terdekat berdasarkan *input* titik asal dan titik
- 6 tujuan.
- 7 – Atribut
 - 8 * nodes : Object
 - 9 Atribut ini menyimpan informasi node dalam bentuk objek.
- 10 – Fungsi
 - 11 * addNode()
 - 12 Fungsi ini digunakan untuk menambahkan satu node beserta edgenya.
 - 13 * shortestPath() : Arrays
 - 14 Fungsi ini digunakan untuk pencarian rute terdekat, fungsi akan mengembalikan rute atau jalur terpendek dalam bentuk array.
- 16 ● Kelas PriorityQueue
- 17 Kelas ini merupakan struktur data *queue* yang digunakan pada kelas dijkstra.
- 18 – Atribut
 - 19 * queue : Array
 - 20 Atribut ini merupakan *queue* dari kelas PriorityQueue.
- 21 – Fungsi
 - 22 * enqueue()
 - 23 Fungsi ini digunakan untuk menambahkan objek ke dalam *queue*.
 - 24 * dequeue() : Object
 - 25 Fungsi ini digunakan untuk mengeluarkan objek dari dalam *queue*.
 - 26 * sort()
 - 27 Fungsi ini digunakan untuk menyusun *queue*.
 - 28 * isEmpty() : boolean
 - 29 Fungsi ini digunakan untuk pengecekan isi dari *queue*. jika *queue* kosong,
 - 30 fungsi akan mengembalikan “true” dan sebaliknya “false” jika *queue* tidak
 - 31 kosong.

32 4.3 Diagram Sekuens

33 Diagram sekuens adalah diagram yang digunakan untuk menggambarkan interaksi antara

34 objek dengan waktu, interaksi tersebut digambarkan dengan grafik dua dimensi. Kedua

35 dimensi tersebut adalah dimensi horizontal dan dimensi vertikal. Dimensi horizontal meng-

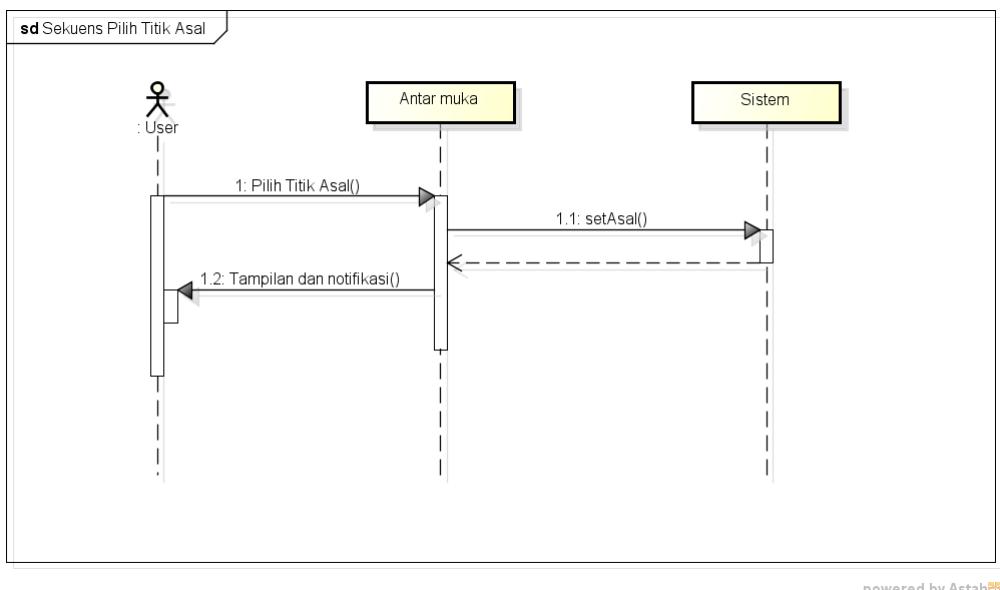
36 gambarkan objek-objek yang berperan, sedangkan dimensi vertikal menggambarkan waktu.

37 Pesan yang dikirimkan oleh suatu objek digambarkan dengan panah dan panah dengan garis

- 1 putus-putus menggambarkan pesan balasan. Diagram sekuens pada bagian ini mengacu ke-
 2 pada diagram *use case* yang terdapat pada bab 3, dapat dilihat pada Gambar 3.11. Berikut
 3 ini adalah diagram sekuens berdasarkan diagram *use case* tersebut:

4 1. Pemilihan Titik Asal

Diagram sekuens untuk pemilihan titik asal dapat dilihat pada Gambar 4.5.



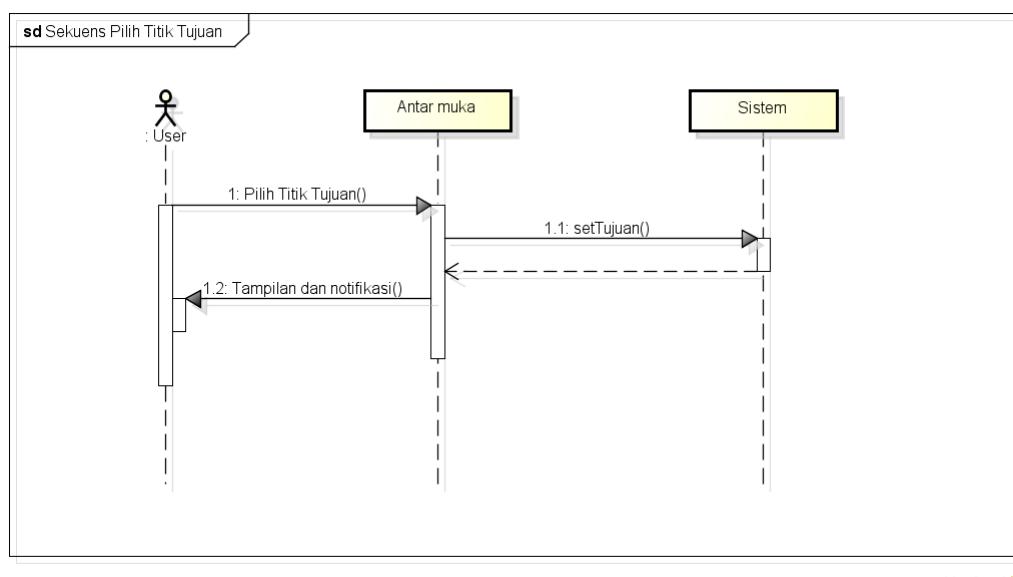
powered by Astah

Gambar 4.5: Diagram Sekuens Pemilihan Titik Asal

5

6 2. Pemilihan Titik Tujuan

Diagram sekuens untuk pemilihan titik tujuan dapat dilihat pada Gambar 4.6.



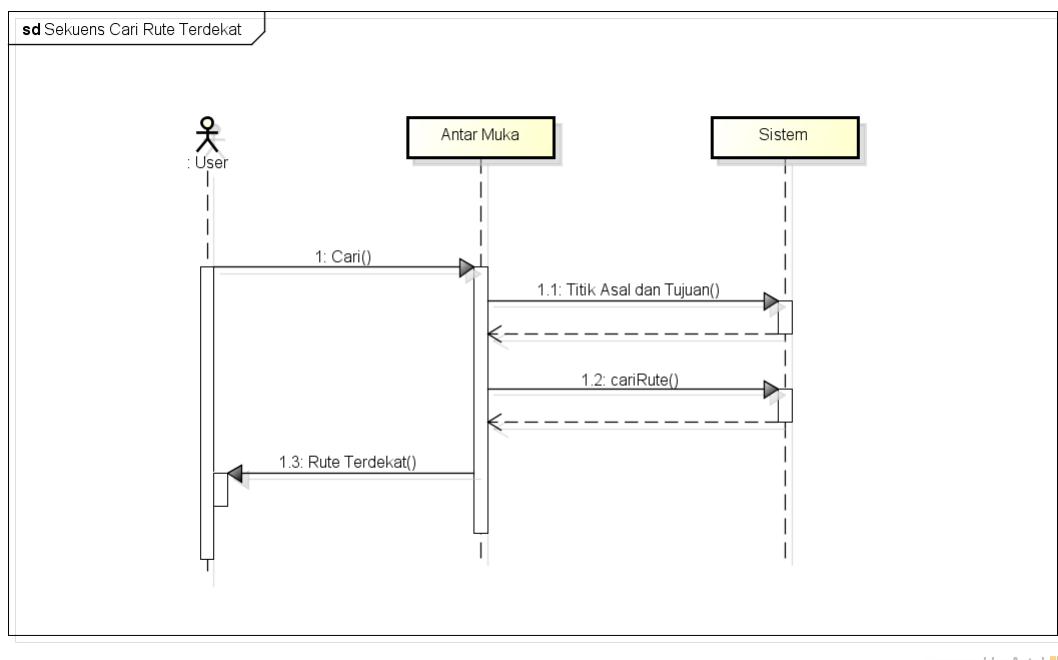
powered by Astah

Gambar 4.6: Diagram Sekuens Pemilihan Titik Tujuan

7

8 3. Pencarian Rute Terdekat

9 Diagram sekuens untuk pencarian rute terdekat dapat dilihat pada Gambar 4.7.



Gambar 4.7: Diagram Sekuens Pencarian Rute Terdekat

¹ **BAB 5**

² **IMPLEMENTASI DAN PENGUJIAN**

³ Pada bagian ini akan dijelaskan tentang implementasi dan pengujian yang dilakukan. Implementasi adalah tahapan setelah proses perancangan selesai, rancangan yang sudah dibuat, diimplementasikan menggunakan kode program. Bahasa pemrograman yang digunakan adalah javascript. Setelah implementasi dilakukan, akan dilakukan juga beberapa pengujian.
⁷ Pengujian dilakukan untuk mengetahui apakah fungsi-fungsi utama aplikasi sudah berjalan dengan baik dan juga untuk mengetahui kekurangan yang dimiliki aplikasi tersebut.

⁹ **5.1 Implementasi**

¹⁰ Implementasi dilakukan menggunakan bahasa pemrograman javascript, hasil implementasi ¹¹ adalah dokumen HTML. Berikut ini adalah lingkungan pembangunan aplikasi pada saat ¹² implementasi dilakukan:

- ¹³ 1. Processor
¹⁴ Intel Core i5-2410M CPU @ 2.30Ghz (4 CPU)
- ¹⁵ 2. Memory
¹⁶ 4096MB RAM
- ¹⁷ 3. Display
¹⁸ AMD Radeon HD 6470M
- ¹⁹ 4. Operating System
²⁰ Windows 7 Ultimate 64-bit
- ²¹ 5. Browser
²² Mozilla Firefox Version 37.0 dan Google Chrome Version 41.0.2272.118 m
- ²³ 6. Bahasa Pemrograman
²⁴ Javascript
- ²⁵ 7. Teks Editor
²⁶ Notepad++

²⁷ **5.2 Pengujian**

²⁸ Pada subbab ini akan dibahas pengujian yang dilakukan. Pengujian yang dilakukan dibagi menjadi dua tahap yaitu pengujian fungsional dan pengujian eksperimental. Pengujian

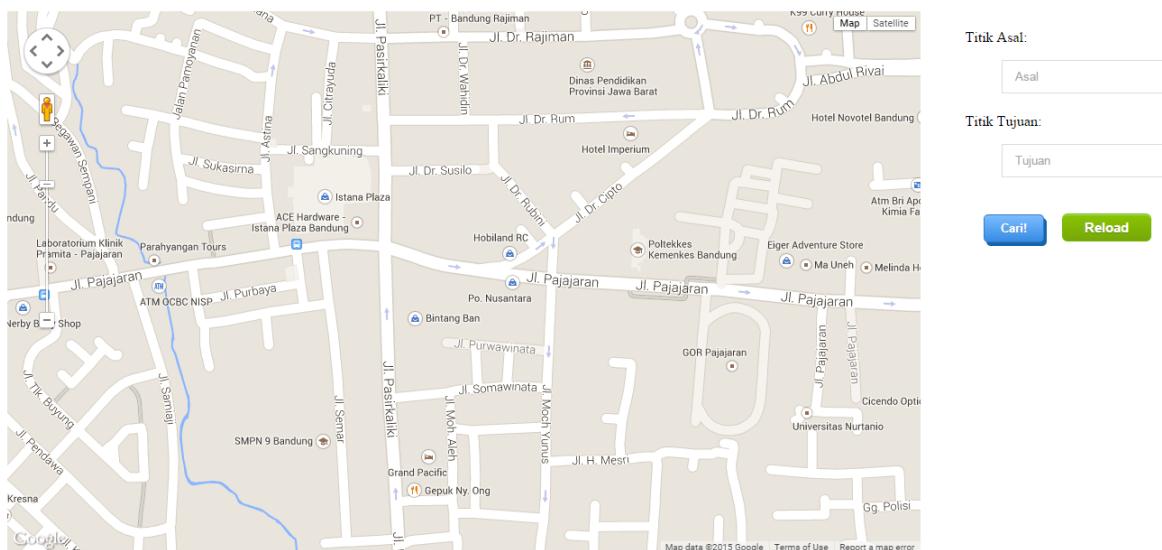
fungsional menguji tampilan antar muka aplikasi beserta method atau fungsi dasar, sedangkan pengujian eksperimental dilakukan dengan menggunakan tiga dokumen OSMXML yang berbeda.

5.2.1 Pengujian Fungsional

Pengujian fungsional menguji tampilan antar muka aplikasi beserta method atau fungsi dasar. Seluruh pengujian fungsional menggunakan dokumen OSMXML yang bernama “test.xml”, dokumen ini memiliki batas koordinat yaitu “<bounds minlat="-6.9131000" minlon="107.5931000" maxlat="-6.9003000" maxlon="107.6149000"/>”, cakupan area tersebut berlokasi di Kota Bandung. Berikut ini adalah daftar fungsi yang diujikan:

1. Tampilan antar muka.

Pada tampilan antar muka aplikasi terdapat elemen “<div>” sebagai tempat untuk menampilkan peta dengan ukuran lebar 75% dari lebar layar dan tinggi 600px. Di sebelah kanan layar terdapat 2 buah *TextBox* yang menampilkan informasi titik asal dan titik tujuan ketika *user* memilih. Selain itu, terdapat 2 buah tombol yaitu “Cari!” untuk mencari rute terdekat antara kedua titik dan tombol kedua adalah “Reload” untuk memuat ulang aplikasi. Hasil pengujian tampilan antar muka dapat dilihat pada Gambar 5.1.



Gambar 5.1: Pengujian Antar Muka

17

2. Fungsi untuk membaca dokumen OSMXML.

Aplikasi mendapatkan informasi dari dokumen OSMXML. Pengujian fungsi ini dilakukan untuk memastikan bahwa dokumen OSMXML dapat dibaca dengan baik. Pengujian dilakukan dengan cara membandingkan dokumen OSMXML dengan informasi yang sudah dibaca. Contoh kasus adalah membandingkan 10 node pertama, potongan informasi 10 node pertama pada “test.xml” dapat dilihat pada Gambar 5.2 dan hasil pengujian dapat dilihat pada Gambar 5.3. Pada Gambar 5.2 menunjukkan bahwa hasil pembacaan sama dengan “test.xml” pada Gambar 5.3, hal ini menunjukkan fungsi untuk membaca dokumen OSMXML sudah berjalan dengan baik.

```

▼<osm version="0.6" generator="CGIImap 0.3.3 (29805 thorn-03.openstreetmap.org)"
copyright="OpenStreetMap and contributors"
attribution="http://www.openstreetmap.org/copyright"
license="http://opendatacommons.org/licenses/odbl/1-0/"
<bounds minlat="-6.9076500" minlon="107.5961800" maxlat="-6.9044500"
maxlon="107.6016300"/>
<node id="25418868" visible="true" version="6" changeset="27915808" timestamp="2015-01-
04T17:54:58Z" user="isonpurba" uid="2552445" lat="-6.9064389" lon="107.5976351"/>
<node id="25433683" visible="true" version="3" changeset="839915" timestamp="2009-03-
21T14:18:48Z" user="adhitya" uid="7748" lat="-6.9067659" lon="107.5989458"/>
<node id="25433687" visible="true" version="2" changeset="839915" timestamp="2009-03-
21T14:18:36Z" user="adhitya" uid="7748" lat="-6.9040267" lon="107.5969508"/>
<node id="25433688" visible="true" version="2" changeset="839915" timestamp="2009-03-
21T14:18:58Z" user="adhitya" uid="7748" lat="-6.9039393" lon="107.5963723"/>
<node id="25433690" visible="true" version="2" changeset="839915" timestamp="2009-03-
21T14:19:02Z" user="adhitya" uid="7748" lat="-6.9052824" lon="107.5961768"/>
<node id="25433685" visible="true" version="4" changeset="13860930" timestamp="2012-11-
13T15:42:42Z" user="yudiwbs" uid="268765" lat="-6.9049404" lon="107.5975738"/>
<node id="25433678" visible="true" version="3" changeset="14069373" timestamp="2012-11-
28T07:04:31Z" user="dadan dany dipoera" uid="922816" lat="-6.9039784"
lon="107.5985467"/>
<node id="25433679" visible="true" version="3" changeset="14069373" timestamp="2012-11-
28T07:04:31Z" user="dadan dany dipoera" uid="922816" lat="-6.9049265"
lon="107.5985843"/>
<node id="25433680" visible="true" version="4" changeset="14069373" timestamp="2012-11-
28T07:04:31Z" user="dadan dany dipoera" uid="922816" lat="-6.9062500"
lon="107.5995945"/>
<node id="25433681" visible="true" version="3" changeset="14069373" timestamp="2012-11-
28T07:04:31Z" user="dadan dany dipoera" uid="922816" lat="-6.9055235"
lon="107.5989193"/>
```

Gambar 5.2: Dokumen test.xml

Node

Node	Id	Latitude	Longitude
0	25418868	-6.9064389	107.5976351
1	25433683	-6.9067659	107.5989458
2	25433687	-6.9040267	107.5969508
3	25433688	-6.9039393	107.5963723
4	25433690	-6.9052824	107.5961768
5	25433685	-6.9049404	107.5975738
6	25433678	-6.9039784	107.5985467
7	25433679	-6.9049265	107.5985843
8	25433680	-6.9062500	107.5995945
9	25433681	-6.9055235	107.5989193

Gambar 5.3: Pengujian Fungsi OSMXML

3. Fungsi untuk mengukur jarak antara dua titik.
 Pengukuran jarak antara dua titik menggunakan *geometry spherical*. Contoh kasus adalah mencari jarak setiap titik yang terdapat pada tag way dengan id 190605660, hasil pengujian dapat dilihat pada Gambar 5.4 dan 5.5.

```

▼<way id="190605660" visible="true" version="2" changeset="18998018" timestamp="2013-11-19T17:23:36Z" user="ubanovic" uid="1784103">
  <nd ref="2012498680"/>
  <nd ref="2012498697"/>
  <nd ref="364364179"/>
  <nd ref="29376826"/>
  <nd ref="25433685"/>
  <nd ref="25433686"/>
  <nd ref="1666615070"/>
  <nd ref="32041828"/>
  <nd ref="2325451286"/>
  <tag k="avgspeed" v="30"/>
  <tag k="bicycle" v="yes"/>
  <tag k="foot" v="yes"/>
  <tag k="highway" v="primary"/>
  <tag k="name" v="Pasir Kaliki"/>
  <tag k="oneway" v="no"/>
</way>
```

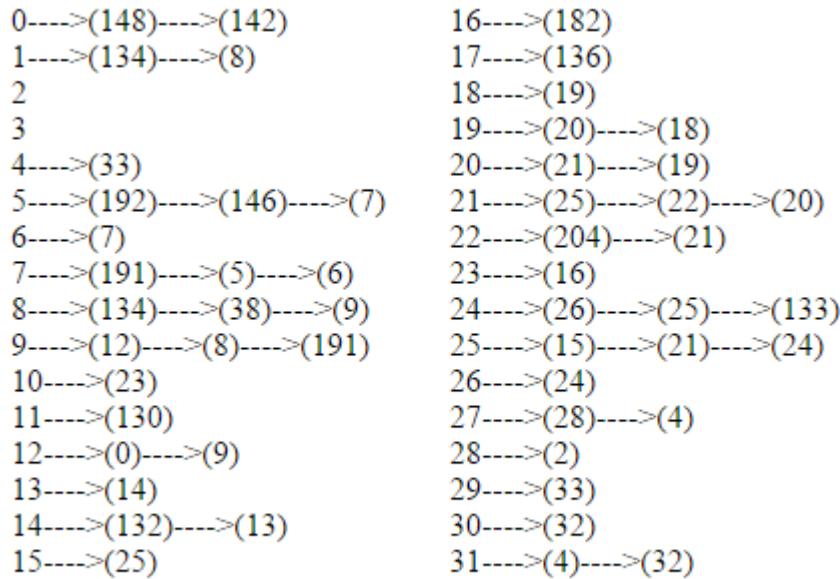
Gambar 5.4: Dokumen test.xml

4

Id Way	Edge	Id Node 1	Id Node 2	Jarak dalam meter
190605660	0	2012498680	2012498697	1.0557581008749386
190605660	1	2012498697	364364179	5.523873381905242
190605660	2	364364179	29376826	5.343438410609913
190605660	3	29376826	25433685	42.3014425903523
190605660	4	25433685	25433686	105.871646845099
190605660	5	25433686	1666615070	74.93190701831529
190605660	6	1666615070	32041828	55.467187471383795
190605660	7	32041828	2325451286	174.54957627536282

Gambar 5.5: Pengujian Jarak

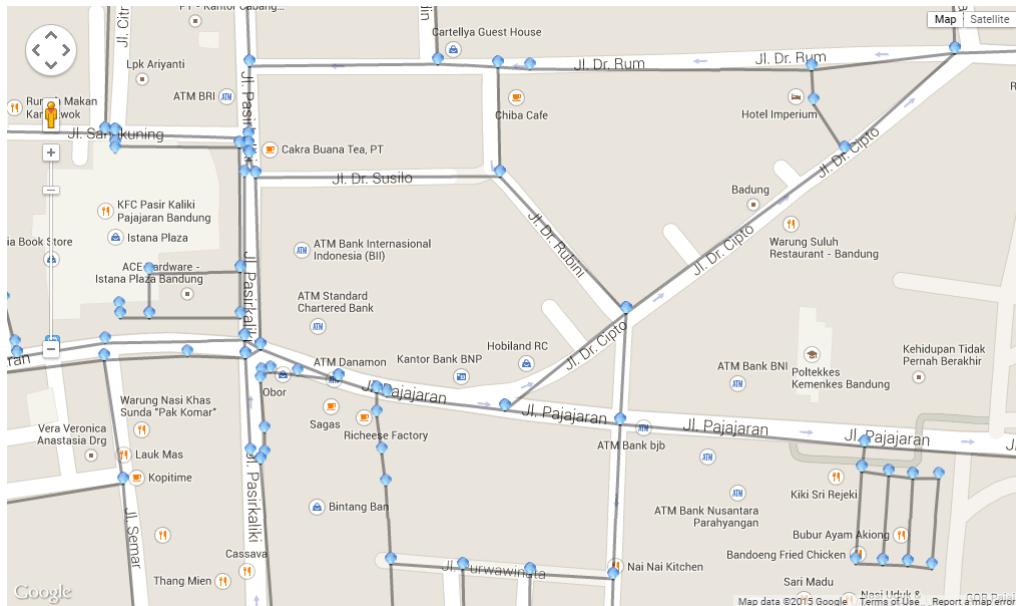
- 5
 6. Fungsi untuk mengubah informasi OSMXML menjadi graf.
 Informasi yang didapatkan dari OSMXML, selanjutnya diubah menjadi graf. Pengujian dilakukan dengan mengubah seluruh “test.xml” menjadi graf menggunakan *adjacency list*. Berikut hasil pengujian, dapat dilihat pada Gambar 5.6. Pada Gambar 5.6 hanya ditampilkan hingga 32 node, hal ini dikarenakan “test.xml” memiliki 207 buah node dan terlalu banyak jika ditampilkan seluruhnya.



Gambar 5.6: Pengujian Fungsi OSMXML Menjadi Graf

1 5. Fungsi untuk melakukan visualisasi.

2 Fungsi ini menggambarkan setiap node menggunakan *marker* dan setiap edge menggunakan polyline pada peta, hasil pengujian dapat dilihat pada Gambar 5.7.

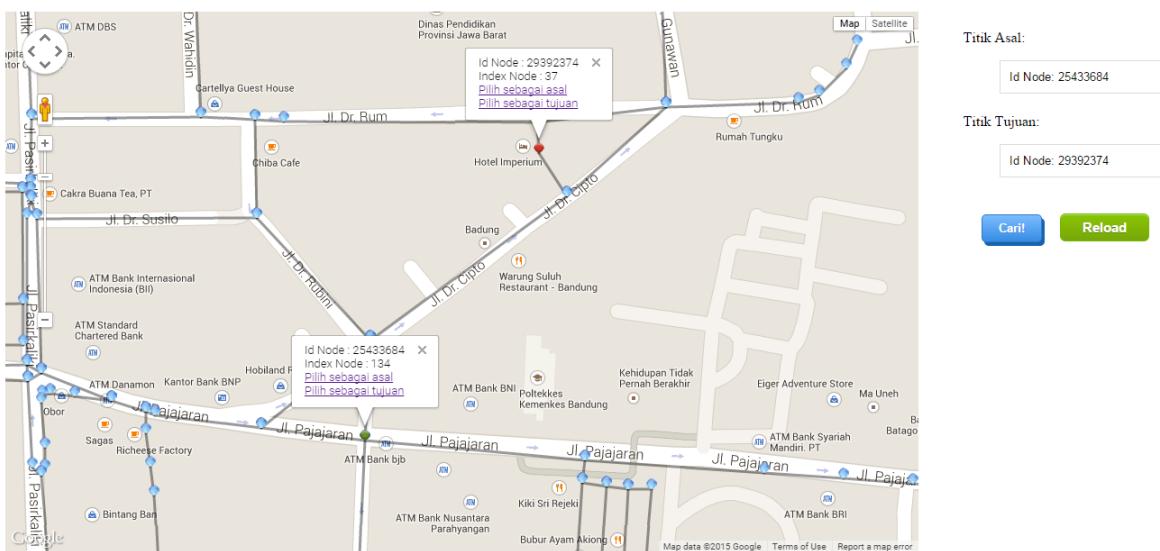


Gambar 5.7: Pengujian Fungsi Visualisasi

3

4 6. Fungsi untuk memilih titik asal dan titik tujuan.

5 Pengujian dilakukan dengan memilih titik asal dan titik tujuan. Titik asal yang sudah
6 dipilih berganti *icon* dengan warna hijau dan titik tujuan yang sudah dipilih berganti
7 *icon* dengan warna merah. Informasi Id Node dari kedua titik yang sudah dipilih,
8 ditampilkan di sebelah kanan layar. Hasil pengujian dapat dilihat pada Gambar 5.8.



Gambar 5.8: Pengujian Titik Asal dan Tujuan

- 1 7. Fungsi untuk mencari rute terdekat antar dua titik menggunakan algoritma dijkstra.
- 2 Pengujian dilakukan dengan contoh kasus mencari rute terdekat dari titik asal (id node : 29356381, index node : 16) ke titik tujuan (id node : 25500626, index node : 11),
- 3 hasil pengujian dapat dilihat pada Gambar 5.9.

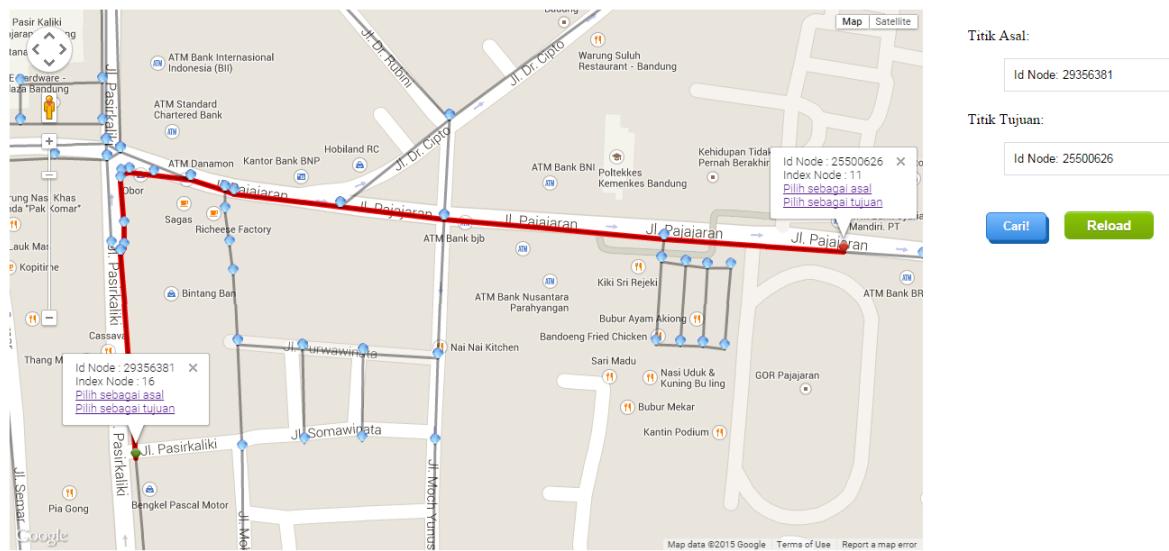
```

Elements Network Sources Timeline Profiles Resources Audits Console
  ↗ <top frame> ▾ Preserve log
  ▲ Synchronous XMLHttpRequest on the main thread is deprecated because of its detrimental effects to the end user's experience. For more help, check http://xhr.spec.whatwg.org/. aplikasi.html:142
  Titik Asal: 16 aplikasi.html:536
  Titik Tujuan: 11 aplikasi.html:537
  ▼ Array[15] ⓘ aplikasi.html:539
    0: "16"
    1: "182"
    2: "145"
    3: "45"
    4: "141"
    5: "144"
    6: "143"
    7: "138"
    8: "142"
    9: "206"
    10: "147"
    11: "1"
    12: "134"
    13: "82"
    14: "11"
    length: 15

```

Gambar 5.9: Pengujian Rute Terdekat

- 4
- 5 8. Fungsi untuk visualisasi rute terdekat.
- 6 Pengujian dilakukan dengan contoh kasus yang sama pada fungsi pencari rute terdekat,
- 7 rute terdekat digambarkan dengan *polyline* berwarna merah. Hasil pengujian dapat
- 8 dilihat pada Gambar 5.10.



Gambar 5.10: Pengujian Visualisasi Rute Terdekat

5.2.2 Pengujian Eksperimental

Pengujian eksperimental dilakukan dengan menggunakan tiga dokumen OSMXML yang berbeda. Ketiga dokumen tersebut diberi nama bandung1.xml, bandung2.xml, dan bandung3.xml, berikut ini adalah cakupan area dari ketiga OSMXML tersebut:

1. Bandung 1

- Batas Atas : -6.9044500
- Batas Bawah : -6.9076500
- Batas Kiri : 107.5961800
- Batas Kanan : 107.6016300

2. Bandung 2

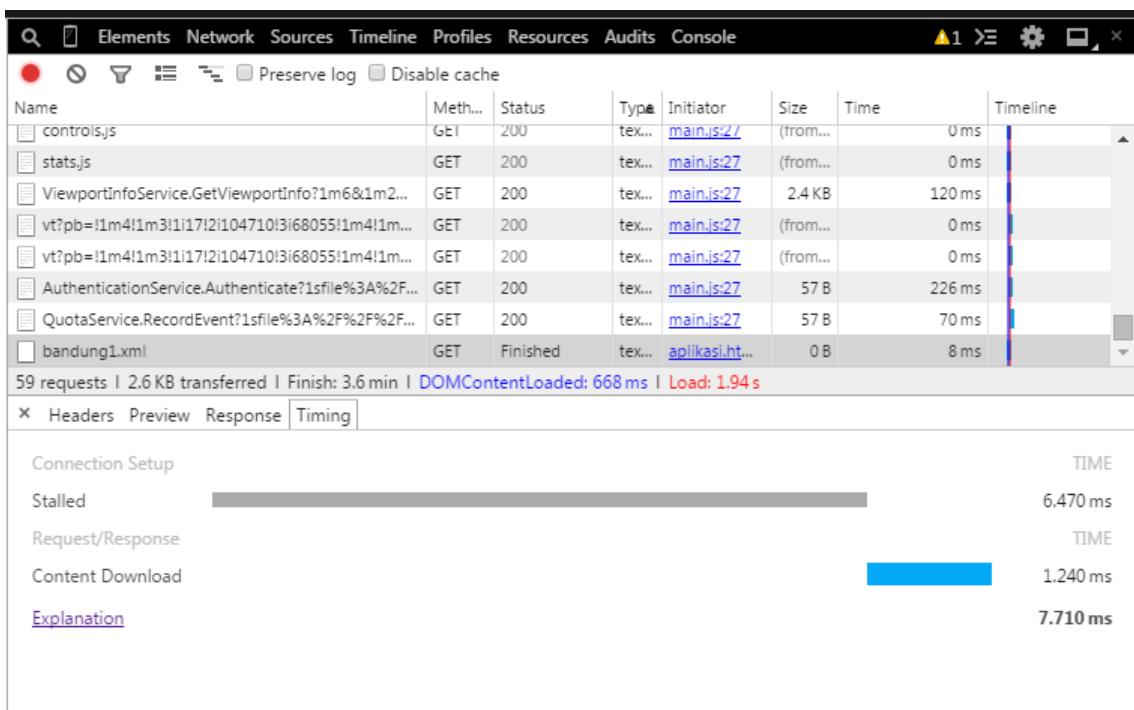
- Batas Atas : -6.9003000
- Batas Bawah : -6.9131000
- Batas Kiri : 107.5931000
- Batas Kanan : 107.6149000

3. Bandung 3

- Batas Atas : -6.8194
- Batas Bawah : -6.9959
- Batas Kiri : 107.553
- Batas Kanan : 107.745

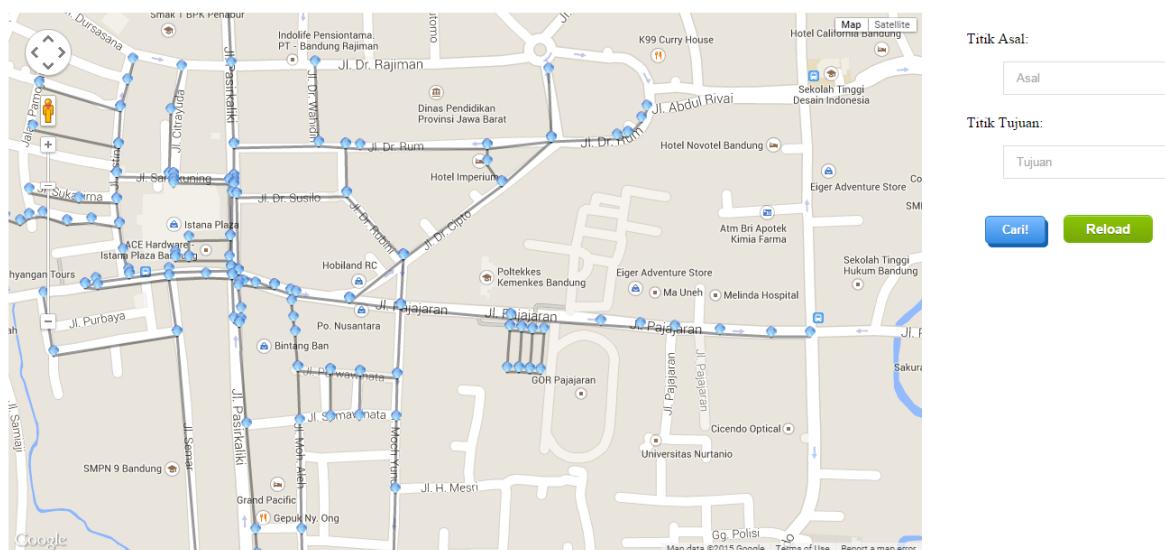
5.2.2.1 Pengujian Eksperimental Bandung 1

Pengujian dilakukan dengan melihat waktu *load* "bandung1.xml" (ukuran file sebesar 98Kb), hasil pengujian dapat dilihat pada Gambar 5.11. Pada Gambar 5.11 menunjukkan total



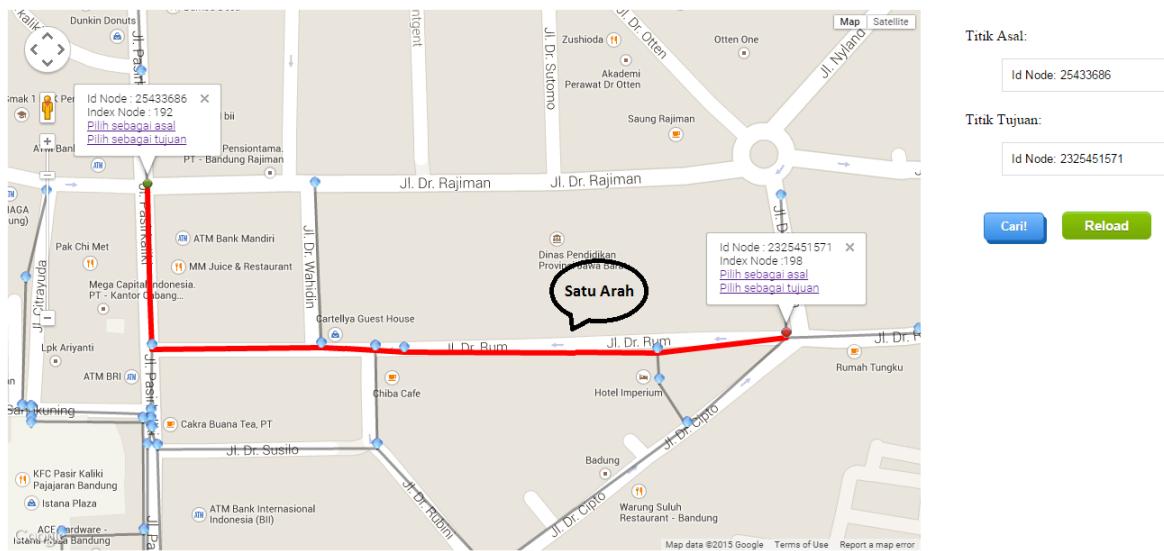
Gambar 5.11: Pengujian Bandung 1

- 1 waktu yang dibutuhkan untuk melakukan *load* "bandung1.xml" adalah 7.710ms. Berikut ini
- 2 adalah aplikasi yang menggunakan dokumen "bandung1.xml" dapat dilihat pada Gambar 5.12.



Gambar 5.12: Pengujian Bandung 1

- 3
- 4 Setelah melakukan *load* aplikasi menggunakan "bandung1.xml", pengujian dilanjutkan dengan mencari rute terdekat dari titik asal (Id Node : 25433686, Index Node : 192) ke titik tujuan (Id Node : 2325451571, Index Node : 198). Pada Gambar 5.13 ditemukan bug yaitu aplikasi menunjukkan rute terdekat (dalam kasus pengujian melalui Jl. Dr Rum) walaupun rute tersebut melawan arah (pada peta Google Maps menunjukkan bahwa Jl. Dr Rum adalah jalan satu arah). Setelah dilakukan penelitian lebih lanjut, ternyata kesalahan dite-



Gambar 5.13: Pengujian Bandung 1

- 1 mukan pada dokumen OSMXML yaitu “bandung1.xml” yang tidak memberikan informasi “oneway”, sehingga aplikasi memasukkan informasi yang salah tersebut ke dalam graf sebagai
- 2 jalan dua arah. Berikut ini adalah potongan dokumen “bandung1.xml” yang memberikan informasi Jl. Dr Rum, dapat dilihat pada Gambar 5.14

```

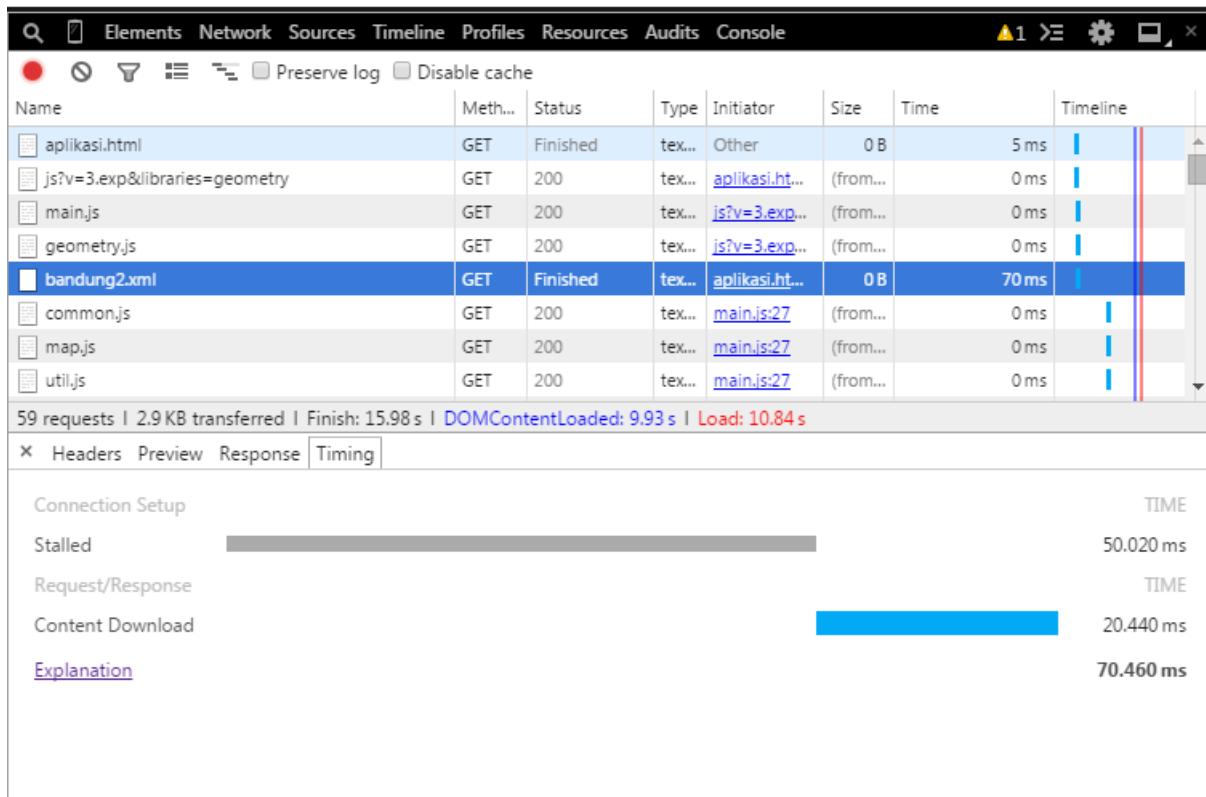
▼<way id="4567503" visible="true" version="6" changeset="27916265" timestamp="2015-01-04T18:11:06Z" user="isonpurba" uid="2552445">
  <nd ref="25433685"/>
  <nd ref="25433679"/>
  <nd ref="25433682"/>
  <nd ref="32041794"/>
  <nd ref="29392373"/>
  <nd ref="2325451571"/>
  <tag k="avgspeed" v="30"/>
  <tag k="bicycle" v="yes"/>
  <tag k="foot" v="yes"/>
  <tag k="highway" v="secondary"/>
  <tag k="name" v="Dr. Rum"/>
</way>
```

Gambar 5.14: bandung1.xml

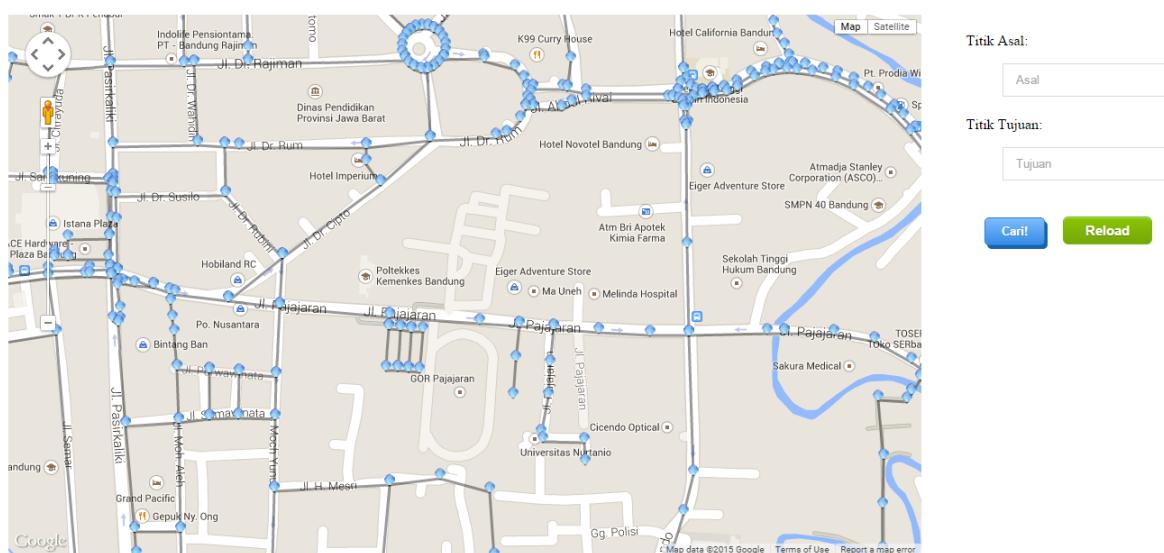
4

5.2.2.2 Pengujian Eksperimental Bandung 2

- 6 Pengujian dilakukan dengan melihat waktu *load* “bandung2.xml” (ukuran file sebesar 821Kb),
- 7 hasil pengujian dapat dilihat pada Gambar 5.15. Pada Gambar 5.15 menunjukkan total wak-
- 8 tu yang dibutuhkan untuk melakukan *load* “bandung2.xml” adalah 70.460ms. Berikut ini
- 9 adalah aplikasi yang menggunakan dokumen “bandung2.xml” dapat dilihat pada Gambar
- 10 5.16. Setelah melakukan *load* aplikasi menggunakan “bandung2.xml”, pengujian dilanjutkan
- 11 dengan mencari rute terdekat dari titik asal (Id Node : 29356503, Index Node : 47) ke titik
- 12 tujuan (Id Node : 1700554920, Index Node : 1163). Hasil pengujian dapat dilihat pada
- 13 Gambar 5.17.



Gambar 5.15: Pengujian Bandung 2



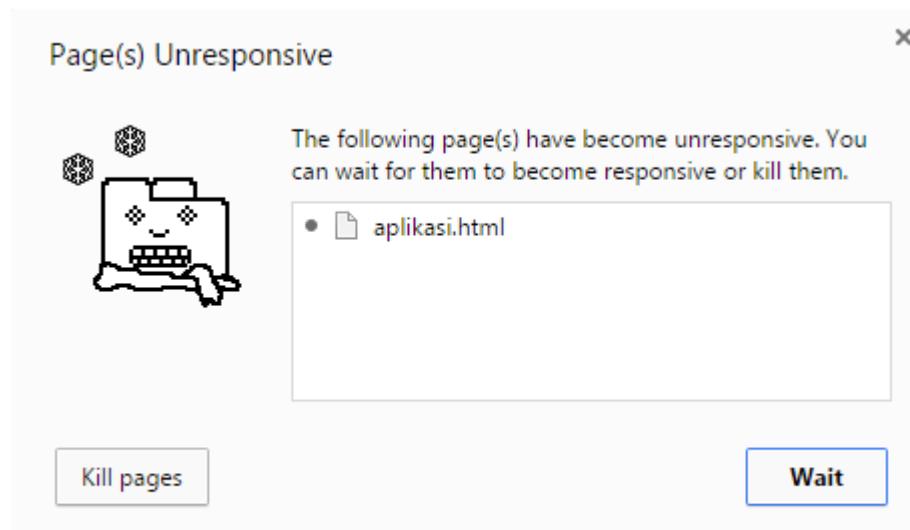
Gambar 5.16: Pengujian Bandung 2



Gambar 5.17: Pengujian Bandung 2

5.2.2.3 Pengujian Eksperimental Bandung 3

- 2 Pengujian dilakukan dengan melihat waktu *load* “bandung3.xml” (ukuran file sebesar 17,631Kb).
- 3 Dokumen “bandung3.xml” adalah OSMXML yang mencakup seluruh wilayah Kota Bandung,
- 4 berdasarkan pengujian yang telah dilakukan, aplikasi tidak berhasil untuk menggunakan
- 5 “bandung3.xml”. Hal ini disebabkan oleh dokumen “bandung3.xml” yang terlalu besar”, wak-
- 6 tu *load* yang terlalu lama sehingga browser menampilkan kotak dialog yang dapat dilihat pada Gambar 5.18.



Gambar 5.18: Pengujian Bandung 3

¹ **5.2.2.4 Hasil Pengujian**

² Berdasarkan pengujian eksperimental yang telah dilakukan, diketahui beberapa hal yaitu:

³ 1. *Load* dokumen OSMXML

Hasil pengujian ketiga dokumen OSMXML dapat dilihat pada Tabel 5.1.

Tabel 5.1: Dokumen OSMXML

	Ukuran File	Waktu Load	Jumlah Node	Jumlah Edge
Bandung 1	98 Kb	7.710 ms	208	141
Bandung 2	821 Kb	70.460 ms	2680	1073
Bandung 3	17,631 Kb	-	-	-

⁴

⁵ Berdasarkan pengujian tersebut, "bandung3.xml" tidak berhasil dibuka karena ukurannya yang terlalu besar dan diketahui bahwa semakin besar dokumen OSMXML,
⁶ semakin besar pula waktu *load* yang diperlukan.
⁷

⁸ 2. Bug

⁹ Bug ditemukan ketika pengujian berlangsung, yaitu rute terdekat yang ditunjukkan
¹⁰ oleh aplikasi melawan arus jalan, hal ini disebabkan oleh dokumen OSMXML yang
¹¹ tidak memberikan informasi "oneway".

¹

BAB 6

²

KESIMPULAN DAN SARAN

³

6.1 Kesimpulan

⁴

1. OSMXML merupakan data peta yang disediakan oleh situs OpenStreetMap dalam bentuk dokumen XML. OSMXML memiliki informasi node dan edge, informasi tersebut dapat dibaca menggunakan Javascript dan dimodelkan ke dalam bentuk graf berarah, graf tersebut dimodelkan menggunakan *adjacency list*.

⁵

⁶

⁷

2. Algoritma Dijkstra dapat mencari rute terdekat pada graf berarah. Aplikasi menggunakan algoritma dijkstra pada graf berarah yang sudah dimodelkan.

⁸

⁹

3. Visualisasi graf dapat dibuat menggunakan Google Maps Javascript API. Aplikasi menampilkan informasi node dan edge pada OSMXML dengan objek *marker* dan *polyline*. Rute terdekat didapatkan dari hasil algoritma dijkstra yang telah diimplementasikan dan divisualkan dengan *polyline*.

¹⁰

¹¹

¹²

¹³

¹⁴

6.2 Saran

¹⁵

- OSMXML memiliki informasi selain node dan edge yaitu “relation”. *Relation* menyimpan informasi seperti rute angkutan, rute bus, rute *hiking*, dan lain-lain. Untuk pengembangan aplikasi yang juga menggunakan data peta dari OpenStreetMap dapat menggunakan informasi tersebut, sehingga tidak hanya rute mengemudi saja, tetapi juga dapat mencari rute terdekat angkutan, bus, atau rute lainnya.
- ¹⁶
- ¹⁷
- ¹⁸
- ¹⁹

1

DAFTAR REFERENSI

- 2 [1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*,
3 *Second Edition*. MIT Press and McGrawHill, 2001.
- 4 [2] OpenStreetMap, “OpenStreetMap Wiki.” <http://wiki.openstreetmap.org/>, 2014.
5 [Online; accessed 30-January-2015].
- 6 [3] B. Marchal, *XML by Example*. John Pierce, 2000.
- 7 [4] D. Flanagan, *JavaScript: The Definitive Guide, Sixth Edition*. O'Reilly Media, Inc, 2011.
- 8 [5] E. Woychowsky, *Ajax: Creating Web Pages with Asynchronous JavaScript and XML*.
9 Prentice Hall, 2006.
- 10 [6] Google, “Google Maps JavaScript API v3.” <https://developers.google.com/maps/documentation/javascript/>, 2015. [Online; accessed 31-January-2015].
- 12 [7] N. R.Chopde and M. K. Nichat, “Landmark based shortest path detection by using a*
13 and haversine formula,” *International Journal of Innovative Research in Computer and*
14 *Communication Engineering*, vol. 1, 2013.

1

LAMPIRAN A

2

KODE PROGRAM

Listing A.1: aplikasi.html

```

3 <!DOCTYPE html>
4 <html>
5 <head>
6 <script src="https://maps.googleapis.com/maps/api/js?v=3.exp&libraries=geometry"></script>
7 </head>
8 <style>
9 .style-1 input[type="text"] {
10   padding: 10px;
11   border: solid 1px #dcdcdc;
12   transition: box-shadow 0.3s, border 0.3s;
13 }
14 }
15 .style-1 input[type="text"]:focus,
16 .style-1 input[type="text"].focus {
17   border: solid 1px #707070;
18   box-shadow: 0 0 5px 1px #969696;
19 }
20 }
21 .cari {
22   -moz-box-shadow: 3px 4px 0px 0px #1564ad;
23   -webkit-box-shadow: 3px 4px 0px 0px #1564ad;
24   box-shadow: 3px 4px 0px 0px #1564ad;
25   background:-webkit-gradient(linear, left top, left bottom, color-stop(0.05, #79bbff), color-
26     stop(1, #378de5));
27   background:-moz-linear-gradient(top, #79bbff 5%, #378de5 100%);
28   background:-webkit-linear-gradient(top, #79bbff 5%, #378de5 100%);
29   background:-o-linear-gradient(top, #79bbff 5%, #378de5 100%);
30   background:-ms-linear-gradient(top, #79bbff 5%, #378de5 100%);
31   background:linear-gradient(to bottom, #79bbff 5%, #378de5 100%);
32   filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#79bbff', endColorstr='#378
33     de5', GradientType=0);
34   background-color:#79bbff;
35   -moz-border-radius:5px;
36   -webkit-border-radius:5px;
37   border-radius:5px;
38   border:1px solid #337bc4;
39   display:inline-block;
40   cursor:pointer;
41   color:#ffffff;
42   font-family: arial;
43   font-size:13px;
44   font-weight:bold;
45   padding:7px 18px;
46   text-decoration:none;
47   text-shadow:0px 1px 0px #528ecc;
48 }
49 }
50 .cari:hover {
51   background:-webkit-gradient(linear, left top, left bottom, color-stop(0.05, #378de5), color-
52     stop(1, #79bbff));
53   background:-moz-linear-gradient(top, #378de5 5%, #79bbff 100%);
54   background:-webkit-linear-gradient(top, #378de5 5%, #79bbff 100%);
55   background:-o-linear-gradient(top, #378de5 5%, #79bbff 100%);
56   background:-ms-linear-gradient(top, #378de5 5%, #79bbff 100%);
57   background:linear-gradient(to bottom, #378de5 5%, #79bbff 100%);
58   filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#378de5', endColorstr='#79
59     bbff', GradientType=0);
60   background-color:#378de5;
61 }
62 }
63 .cari:active {
64   position: relative;
65   top:1px;
66 }
67 }
68 .reload {
69   -moz-box-shadow:inset 0px 1px 0px 0px #a4e271;
70   -webkit-box-shadow:inset 0px 1px 0px 0px #a4e271;
71   box-shadow:inset 0px 1px 0px 0px #a4e271;
72   background:-webkit-gradient(linear, left top, left bottom, color-stop(0.05, #89c403), color-
73     stop(1, #77a809));
74   background:-moz-linear-gradient(top, #89c403 5%, #77a809 100%);
75   background:-webkit-linear-gradient(top, #89c403 5%, #77a809 100%);
76   background:-o-linear-gradient(top, #89c403 5%, #77a809 100%);
77   background:-ms-linear-gradient(top, #89c403 5%, #77a809 100%);
78 }
```

```

1  background: linear-gradient(to bottom, #89c403 5%, #77a809 100%);
2  filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#89c403', endColorstr='#77
3   a809', GradientType=0);
4  background-color:#89c403;
5  -moz-border-radius:6px;
6  -webkit-border-radius:6px;
7  border-radius:6px;
8  border:1px solid #74b807;
9  display:inline-block;
10 cursor:pointer;
11 color:#fffff;
12 font-family:arial;
13 font-size:15px;
14 font-weight:bold;
15 padding:6px 24px;
16 text-decoration:none;
17 text-shadow:0px 1px 0px #528009;
18 }
19
20 .reload:hover {
21  background:-webkit-gradient(linear, left top, left bottom, color-stop(0.05, #77a809), color-
22  stop(1, #89c403));
23  background:-moz-linear-gradient(top, #77a809 5%, #89c403 100%);
24  background:-webkit-linear-gradient(top, #77a809 5%, #89c403 100%);
25  background:-o-linear-gradient(top, #77a809 5%, #89c403 100%);
26  background:-ms-linear-gradient(top, #77a809 5%, #89c403 100%);
27  background:linear-gradient(to bottom, #77a809 5%, #89c403 100%);
28  filter:progid:DXImageTransform.Microsoft.gradient(startColorstr='#77a809', endColorstr='#89
29   c403', GradientType=0);
30  background-color:#77a809;
31 }
32
33 .reload:active {
34  position:relative;
35  top:1px;
36 }
37 </style>
38 <body>
39 <div id="googleMap" style="width:75%;height:600px;float:left"></div>
40 <div id="cari" style="width:25%;height:60px;float:left">
41 <div style="margin-left:50px;margin-top:20px">
42  <table style="border:1">
43   <tr>
44    <label>Titik Asal:</label>
45   </tr>
46   <tr>
47    <ul class="input-list-style-l clearfix">
48     <input type="text" id="asal" placeholder="Asal">
49    </ul>
50   </tr>
51  </table>
52 </div>
53 <div style="margin-left:50px">
54 <table>
55  <tr>
56   <label>Titik Tujuan:</label>
57  </tr>
58  <tr>
59   <ul class="input-list-style-l clearfix">
60    <input type="text" id="tujuan" placeholder="Tujuan">
61   </ul>
62  </tr>
63 </table>
64 </div>
65 <br/>
66 <div style="margin-left:70px">
67  <a href="#" onclick="result()" class="cari">Cari!</a> &nbsp &nbsp
68  <a href="javascript:location.reload(true)" class="reload">Reload</a>
69 </div>
70 </div>
71 <script>
72 //Load XML
73 xmlhttp=new XMLHttpRequest();
74 xmlhttp.open("GET","bandung2.xml",false);
75 xmlhttp.send();
76 xmlDoc=xmlhttp.responseXML;
77
78 // Adjacency List
79 //Kelas Neighbor
80 function Neighbor(vnum, nbr, weight){
81  this.vertexNum = vnum;
82  this.next = nbr;
83  this.weight = weight;
84 }
85
86 //Kelas Node
87 function Node(id, neighbors){
88  this.id = id;
89  this.adjList = neighbors;
90 }
91
92 //Kelas Graph
93 function Graph(node,way){
94  var adjLists = [];
95  var nd;
96  var oneway;
97
98  function wayDirection(way,index){
99   var tag = way[index].getElementsByName("tag");
100   for (hg=0;hg<tag.length;hg++)
101   {
102      if(tag[hg].getAttribute('k') == "oneway"){
103         return tag[hg].getAttribute('v');
104      }
105   }
106  }
107
108  node.adjList = adjLists;
109  node.way = way;
110  node.oneway = oneway;
111
112  for (i=0;i<node.adjList.length;i++)
113  {
114    for (j=0;j<node.adjList[i].length;j++)
115    {
116      if(node.adjList[i][j].id == nbr)
117      {
118        node.adjList[i][j].weight = weight;
119        break;
120      }
121    }
122  }
123
124  for (i=0;i<node.way.length;i++)
125  {
126    if(node.way[i].id == nbr)
127    {
128      node.way[i].index = index;
129      break;
130    }
131  }
132
133  if(oneway == "true")
134  {
135    wayDirection(way,i+1);
136  }
137
138  if(oneway == "false")
139  {
140    wayDirection(way,i+1);
141    wayDirection(way,i);
142  }
143
144  if(oneway == "bidirectional")
145  {
146    wayDirection(way,i+1);
147    wayDirection(way,i);
148  }
149
150  if(oneway == "none")
151  {
152    wayDirection(way,i+1);
153  }
154
155  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
156  {
157    if(node.adjList[i].length == 0)
158    {
159      node.adjList[i].push(nd);
160    }
161  }
162
163  if(oneway == "none")
164  {
165    if(node.adjList[i].length == 0)
166    {
167      node.adjList[i].push(nd);
168    }
169  }
170
171  if(oneway == "true" || oneway == "false" || oneway == "bidirectional")
172  {
173    if(node.adjList[i].length == 0)
174    {
175      node.adjList[i].push(nd);
176    }
177  }
178
179  if(oneway == "none")
180  {
181    if(node.adjList[i].length == 0)
182    {
183      node.adjList[i].push(nd);
184    }
185  }
186
187  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
188  {
189    if(node.adjList[i].length == 0)
190    {
191      node.adjList[i].push(nd);
192    }
193  }
194
195  if(oneway == "none")
196  {
197    if(node.adjList[i].length == 0)
198    {
199      node.adjList[i].push(nd);
200    }
201  }
202
203  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
204  {
205    if(node.adjList[i].length == 0)
206    {
207      node.adjList[i].push(nd);
208    }
209  }
210
211  if(oneway == "none")
212  {
213    if(node.adjList[i].length == 0)
214    {
215      node.adjList[i].push(nd);
216    }
217  }
218
219  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
220  {
221    if(node.adjList[i].length == 0)
222    {
223      node.adjList[i].push(nd);
224    }
225  }
226
227  if(oneway == "none")
228  {
229    if(node.adjList[i].length == 0)
230    {
231      node.adjList[i].push(nd);
232    }
233  }
234
235  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
236  {
237    if(node.adjList[i].length == 0)
238    {
239      node.adjList[i].push(nd);
240    }
241  }
242
243  if(oneway == "none")
244  {
245    if(node.adjList[i].length == 0)
246    {
247      node.adjList[i].push(nd);
248    }
249  }
250
251  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
252  {
253    if(node.adjList[i].length == 0)
254    {
255      node.adjList[i].push(nd);
256    }
257  }
258
259  if(oneway == "none")
260  {
261    if(node.adjList[i].length == 0)
262    {
263      node.adjList[i].push(nd);
264    }
265  }
266
267  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
268  {
269    if(node.adjList[i].length == 0)
270    {
271      node.adjList[i].push(nd);
272    }
273  }
274
275  if(oneway == "none")
276  {
277    if(node.adjList[i].length == 0)
278    {
279      node.adjList[i].push(nd);
280    }
281  }
282
283  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
284  {
285    if(node.adjList[i].length == 0)
286    {
287      node.adjList[i].push(nd);
288    }
289  }
290
291  if(oneway == "none")
292  {
293    if(node.adjList[i].length == 0)
294    {
295      node.adjList[i].push(nd);
296    }
297  }
298
299  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
300  {
301    if(node.adjList[i].length == 0)
302    {
303      node.adjList[i].push(nd);
304    }
305  }
306
307  if(oneway == "none")
308  {
309    if(node.adjList[i].length == 0)
310    {
311      node.adjList[i].push(nd);
312    }
313  }
314
315  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
316  {
317    if(node.adjList[i].length == 0)
318    {
319      node.adjList[i].push(nd);
320    }
321  }
322
323  if(oneway == "none")
324  {
325    if(node.adjList[i].length == 0)
326    {
327      node.adjList[i].push(nd);
328    }
329  }
330
331  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
332  {
333    if(node.adjList[i].length == 0)
334    {
335      node.adjList[i].push(nd);
336    }
337  }
338
339  if(oneway == "none")
340  {
341    if(node.adjList[i].length == 0)
342    {
343      node.adjList[i].push(nd);
344    }
345  }
346
347  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
348  {
349    if(node.adjList[i].length == 0)
350    {
351      node.adjList[i].push(nd);
352    }
353  }
354
355  if(oneway == "none")
356  {
357    if(node.adjList[i].length == 0)
358    {
359      node.adjList[i].push(nd);
360    }
361  }
362
363  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
364  {
365    if(node.adjList[i].length == 0)
366    {
367      node.adjList[i].push(nd);
368    }
369  }
370
371  if(oneway == "none")
372  {
373    if(node.adjList[i].length == 0)
374    {
375      node.adjList[i].push(nd);
376    }
377  }
378
379  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
380  {
381    if(node.adjList[i].length == 0)
382    {
383      node.adjList[i].push(nd);
384    }
385  }
386
387  if(oneway == "none")
388  {
389    if(node.adjList[i].length == 0)
390    {
391      node.adjList[i].push(nd);
392    }
393  }
394
395  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
396  {
397    if(node.adjList[i].length == 0)
398    {
399      node.adjList[i].push(nd);
400    }
401  }
402
403  if(oneway == "none")
404  {
405    if(node.adjList[i].length == 0)
406    {
407      node.adjList[i].push(nd);
408    }
409  }
410
411  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
412  {
413    if(node.adjList[i].length == 0)
414    {
415      node.adjList[i].push(nd);
416    }
417  }
418
419  if(oneway == "none")
420  {
421    if(node.adjList[i].length == 0)
422    {
423      node.adjList[i].push(nd);
424    }
425  }
426
427  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
428  {
429    if(node.adjList[i].length == 0)
430    {
431      node.adjList[i].push(nd);
432    }
433  }
434
435  if(oneway == "none")
436  {
437    if(node.adjList[i].length == 0)
438    {
439      node.adjList[i].push(nd);
440    }
441  }
442
443  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
444  {
445    if(node.adjList[i].length == 0)
446    {
447      node.adjList[i].push(nd);
448    }
449  }
450
451  if(oneway == "none")
452  {
453    if(node.adjList[i].length == 0)
454    {
455      node.adjList[i].push(nd);
456    }
457  }
458
459  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
460  {
461    if(node.adjList[i].length == 0)
462    {
463      node.adjList[i].push(nd);
464    }
465  }
466
467  if(oneway == "none")
468  {
469    if(node.adjList[i].length == 0)
470    {
471      node.adjList[i].push(nd);
472    }
473  }
474
475  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
476  {
477    if(node.adjList[i].length == 0)
478    {
479      node.adjList[i].push(nd);
480    }
481  }
482
483  if(oneway == "none")
484  {
485    if(node.adjList[i].length == 0)
486    {
487      node.adjList[i].push(nd);
488    }
489  }
490
491  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
492  {
493    if(node.adjList[i].length == 0)
494    {
495      node.adjList[i].push(nd);
496    }
497  }
498
499  if(oneway == "none")
500  {
501    if(node.adjList[i].length == 0)
502    {
503      node.adjList[i].push(nd);
504    }
505  }
506
507  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
508  {
509    if(node.adjList[i].length == 0)
510    {
511      node.adjList[i].push(nd);
512    }
513  }
514
515  if(oneway == "none")
516  {
517    if(node.adjList[i].length == 0)
518    {
519      node.adjList[i].push(nd);
520    }
521  }
522
523  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
524  {
525    if(node.adjList[i].length == 0)
526    {
527      node.adjList[i].push(nd);
528    }
529  }
530
531  if(oneway == "none")
532  {
533    if(node.adjList[i].length == 0)
534    {
535      node.adjList[i].push(nd);
536    }
537  }
538
539  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
540  {
541    if(node.adjList[i].length == 0)
542    {
543      node.adjList[i].push(nd);
544    }
545  }
546
547  if(oneway == "none")
548  {
549    if(node.adjList[i].length == 0)
550    {
551      node.adjList[i].push(nd);
552    }
553  }
554
555  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
556  {
557    if(node.adjList[i].length == 0)
558    {
559      node.adjList[i].push(nd);
560    }
561  }
562
563  if(oneway == "none")
564  {
565    if(node.adjList[i].length == 0)
566    {
567      node.adjList[i].push(nd);
568    }
569  }
570
571  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
572  {
573    if(node.adjList[i].length == 0)
574    {
575      node.adjList[i].push(nd);
576    }
577  }
578
579  if(oneway == "none")
580  {
581    if(node.adjList[i].length == 0)
582    {
583      node.adjList[i].push(nd);
584    }
585  }
586
587  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
588  {
589    if(node.adjList[i].length == 0)
590    {
591      node.adjList[i].push(nd);
592    }
593  }
594
595  if(oneway == "none")
596  {
597    if(node.adjList[i].length == 0)
598    {
599      node.adjList[i].push(nd);
600    }
601  }
602
603  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
604  {
605    if(node.adjList[i].length == 0)
606    {
607      node.adjList[i].push(nd);
608    }
609  }
610
611  if(oneway == "none")
612  {
613    if(node.adjList[i].length == 0)
614    {
615      node.adjList[i].push(nd);
616    }
617  }
618
619  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
620  {
621    if(node.adjList[i].length == 0)
622    {
623      node.adjList[i].push(nd);
624    }
625  }
626
627  if(oneway == "none")
628  {
629    if(node.adjList[i].length == 0)
630    {
631      node.adjList[i].push(nd);
632    }
633  }
634
635  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
636  {
637    if(node.adjList[i].length == 0)
638    {
639      node.adjList[i].push(nd);
640    }
641  }
642
643  if(oneway == "none")
644  {
645    if(node.adjList[i].length == 0)
646    {
647      node.adjList[i].push(nd);
648    }
649  }
650
651  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
652  {
653    if(node.adjList[i].length == 0)
654    {
655      node.adjList[i].push(nd);
656    }
657  }
658
659  if(oneway == "none")
660  {
661    if(node.adjList[i].length == 0)
662    {
663      node.adjList[i].push(nd);
664    }
665  }
666
667  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
668  {
669    if(node.adjList[i].length == 0)
670    {
671      node.adjList[i].push(nd);
672    }
673  }
674
675  if(oneway == "none")
676  {
677    if(node.adjList[i].length == 0)
678    {
679      node.adjList[i].push(nd);
680    }
681  }
682
683  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
684  {
685    if(node.adjList[i].length == 0)
686    {
687      node.adjList[i].push(nd);
688    }
689  }
690
691  if(oneway == "none")
692  {
693    if(node.adjList[i].length == 0)
694    {
695      node.adjList[i].push(nd);
696    }
697  }
698
699  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
700  {
701    if(node.adjList[i].length == 0)
702    {
703      node.adjList[i].push(nd);
704    }
705  }
706
707  if(oneway == "none")
708  {
709    if(node.adjList[i].length == 0)
710    {
711      node.adjList[i].push(nd);
712    }
713  }
714
715  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
716  {
717    if(node.adjList[i].length == 0)
718    {
719      node.adjList[i].push(nd);
720    }
721  }
722
723  if(oneway == "none")
724  {
725    if(node.adjList[i].length == 0)
726    {
727      node.adjList[i].push(nd);
728    }
729  }
730
731  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
732  {
733    if(node.adjList[i].length == 0)
734    {
735      node.adjList[i].push(nd);
736    }
737  }
738
739  if(oneway == "none")
740  {
741    if(node.adjList[i].length == 0)
742    {
743      node.adjList[i].push(nd);
744    }
745  }
746
747  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
748  {
749    if(node.adjList[i].length == 0)
750    {
751      node.adjList[i].push(nd);
752    }
753  }
754
755  if(oneway == "none")
756  {
757    if(node.adjList[i].length == 0)
758    {
759      node.adjList[i].push(nd);
760    }
761  }
762
763  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
764  {
765    if(node.adjList[i].length == 0)
766    {
767      node.adjList[i].push(nd);
768    }
769  }
770
771  if(oneway == "none")
772  {
773    if(node.adjList[i].length == 0)
774    {
775      node.adjList[i].push(nd);
776    }
777  }
778
779  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
780  {
781    if(node.adjList[i].length == 0)
782    {
783      node.adjList[i].push(nd);
784    }
785  }
786
787  if(oneway == "none")
788  {
789    if(node.adjList[i].length == 0)
790    {
791      node.adjList[i].push(nd);
792    }
793  }
794
795  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
796  {
797    if(node.adjList[i].length == 0)
798    {
799      node.adjList[i].push(nd);
800    }
801  }
802
803  if(oneway == "none")
804  {
805    if(node.adjList[i].length == 0)
806    {
807      node.adjList[i].push(nd);
808    }
809  }
810
811  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
812  {
813    if(node.adjList[i].length == 0)
814    {
815      node.adjList[i].push(nd);
816    }
817  }
818
819  if(oneway == "none")
820  {
821    if(node.adjList[i].length == 0)
822    {
823      node.adjList[i].push(nd);
824    }
825  }
826
827  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
828  {
829    if(node.adjList[i].length == 0)
830    {
831      node.adjList[i].push(nd);
832    }
833  }
834
835  if(oneway == "none")
836  {
837    if(node.adjList[i].length == 0)
838    {
839      node.adjList[i].push(nd);
840    }
841  }
842
843  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
844  {
845    if(node.adjList[i].length == 0)
846    {
847      node.adjList[i].push(nd);
848    }
849  }
850
851  if(oneway == "none")
852  {
853    if(node.adjList[i].length == 0)
854    {
855      node.adjList[i].push(nd);
856    }
857  }
858
859  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
860  {
861    if(node.adjList[i].length == 0)
862    {
863      node.adjList[i].push(nd);
864    }
865  }
866
867  if(oneway == "none")
868  {
869    if(node.adjList[i].length == 0)
870    {
871      node.adjList[i].push(nd);
872    }
873  }
874
875  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
876  {
877    if(node.adjList[i].length == 0)
878    {
879      node.adjList[i].push(nd);
880    }
881  }
882
883  if(oneway == "none")
884  {
885    if(node.adjList[i].length == 0)
886    {
887      node.adjList[i].push(nd);
888    }
889  }
890
891  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
892  {
893    if(node.adjList[i].length == 0)
894    {
895      node.adjList[i].push(nd);
896    }
897  }
898
899  if(oneway == "none")
900  {
901    if(node.adjList[i].length == 0)
902    {
903      node.adjList[i].push(nd);
904    }
905  }
906
907  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
908  {
909    if(node.adjList[i].length == 0)
910    {
911      node.adjList[i].push(nd);
912    }
913  }
914
915  if(oneway == "none")
916  {
917    if(node.adjList[i].length == 0)
918    {
919      node.adjList[i].push(nd);
920    }
921  }
922
923  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
924  {
925    if(node.adjList[i].length == 0)
926    {
927      node.adjList[i].push(nd);
928    }
929  }
930
931  if(oneway == "none")
932  {
933    if(node.adjList[i].length == 0)
934    {
935      node.adjList[i].push(nd);
936    }
937  }
938
939  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
940  {
941    if(node.adjList[i].length == 0)
942    {
943      node.adjList[i].push(nd);
944    }
945  }
946
947  if(oneway == "none")
948  {
949    if(node.adjList[i].length == 0)
950    {
951      node.adjList[i].push(nd);
952    }
953  }
954
955  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
956  {
957    if(node.adjList[i].length == 0)
958    {
959      node.adjList[i].push(nd);
960    }
961  }
962
963  if(oneway == "none")
964  {
965    if(node.adjList[i].length == 0)
966    {
967      node.adjList[i].push(nd);
968    }
969  }
970
971  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
972  {
973    if(node.adjList[i].length == 0)
974    {
975      node.adjList[i].push(nd);
976    }
977  }
978
979  if(oneway == "none")
980  {
981    if(node.adjList[i].length == 0)
982    {
983      node.adjList[i].push(nd);
984    }
985  }
986
987  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
988  {
989    if(node.adjList[i].length == 0)
990    {
991      node.adjList[i].push(nd);
992    }
993  }
994
995  if(oneway == "none")
996  {
997    if(node.adjList[i].length == 0)
998    {
999      node.adjList[i].push(nd);
1000     }
1001  }
1002
1003  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
1004  {
1005    if(node.adjList[i].length == 0)
1006    {
1007      node.adjList[i].push(nd);
1008    }
1009  }
1010
1011  if(oneway == "none")
1012  {
1013    if(node.adjList[i].length == 0)
1014    {
1015      node.adjList[i].push(nd);
1016    }
1017  }
1018
1019  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
1020  {
1021    if(node.adjList[i].length == 0)
1022    {
1023      node.adjList[i].push(nd);
1024    }
1025  }
1026
1027  if(oneway == "none")
1028  {
1029    if(node.adjList[i].length == 0)
1030    {
1031      node.adjList[i].push(nd);
1032    }
1033  }
1034
1035  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
1036  {
1037    if(node.adjList[i].length == 0)
1038    {
1039      node.adjList[i].push(nd);
1040    }
1041  }
1042
1043  if(oneway == "none")
1044  {
1045    if(node.adjList[i].length == 0)
1046    {
1047      node.adjList[i].push(nd);
1048    }
1049  }
1050
1051  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
1052  {
1053    if(node.adjList[i].length == 0)
1054    {
1055      node.adjList[i].push(nd);
1056    }
1057  }
1058
1059  if(oneway == "none")
1060  {
1061    if(node.adjList[i].length == 0)
1062    {
1063      node.adjList[i].push(nd);
1064    }
1065  }
1066
1067  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
1068  {
1069    if(node.adjList[i].length == 0)
1070    {
1071      node.adjList[i].push(nd);
1072    }
1073  }
1074
1075  if(oneway == "none")
1076  {
1077    if(node.adjList[i].length == 0)
1078    {
1079      node.adjList[i].push(nd);
1080    }
1081  }
1082
1083  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
1084  {
1085    if(node.adjList[i].length == 0)
1086    {
1087      node.adjList[i].push(nd);
1088    }
1089  }
1090
1091  if(oneway == "none")
1092  {
1093    if(node.adjList[i].length == 0)
1094    {
1095      node.adjList[i].push(nd);
1096    }
1097  }
1098
1099  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
1100  {
1101    if(node.adjList[i].length == 0)
1102    {
1103      node.adjList[i].push(nd);
1104    }
1105  }
1106
1107  if(oneway == "none")
1108  {
1109    if(node.adjList[i].length == 0)
1110    {
1111      node.adjList[i].push(nd);
1112    }
1113  }
1114
1115  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
1116  {
1117    if(node.adjList[i].length == 0)
1118    {
1119      node.adjList[i].push(nd);
1120    }
1121  }
1122
1123  if(oneway == "none")
1124  {
1125    if(node.adjList[i].length == 0)
1126    {
1127      node.adjList[i].push(nd);
1128    }
1129  }
1130
1131  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
1132  {
1133    if(node.adjList[i].length == 0)
1134    {
1135      node.adjList[i].push(nd);
1136    }
1137  }
1138
1139  if(oneway == "none")
1140  {
1141    if(node.adjList[i].length == 0)
1142    {
1143      node.adjList[i].push(nd);
1144    }
1145  }
1146
1147  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
1148  {
1149    if(node.adjList[i].length == 0)
1150    {
1151      node.adjList[i].push(nd);
1152    }
1153  }
1154
1155  if(oneway == "none")
1156  {
1157    if(node.adjList[i].length == 0)
1158    {
1159      node.adjList[i].push(nd);
1160    }
1161  }
1162
1163  if(oneway == "true" || oneway == "false" || oneway == "bidirectional" || oneway == "none")
1164  {
1165    if(node.adjList[i].length == 0)
1166    {

```

```

1         }
2     }
3     return false;
4 }
5
6     function isHighway(way, index){
7         var tag = way[index].getElementsByTagName("tag");
8         for (hg=0;hg<tag.length;hg++){
9             if (tag[hg].getAttribute('k') == "highway"){
10                 return true;
11             }
12         }
13     }
14     return false;
15 }
16
17     function indexForId(adjLists, id) {
18         for (var i=0; i < adjLists.length; i++){
19             if (adjLists[i].id == id){
20                 return i;
21             }
22         }
23     return -1;
24 }
25
26     function getLatByAtt(str)
27 {
28     for (n=0;n<node.length;n++)
29     {
30         if (node[n].getAttribute('id') == str)
31         {
32             return node[n].getAttribute('lat');
33         }
34     }
35 }
36
37     function getLonByAtt(str)
38 {
39     for (m=0;m<node.length;m++)
40     {
41         if (node[m].getAttribute('id') == str)
42         {
43             return node[m].getAttribute('lon');
44         }
45     }
46 }
47
48 for(v=0; v < node.length; v++){
49     adjLists.push(new Node(node[v].getAttribute('id'), null));
50 }
51
52 var lat1, lon1, lat2, lon2;
53 var point1, point2, distance;
54
55 for (i=0;i<way.length; i++)
56 {
57     nd = way[i].getElementsByTagName("nd");
58     if (isHighway(way, i)){
59         oneway = wayDirection(way, i);
60         for (j=0;j<nd.length-1;j++)
61         {
62             v1 = indexForId(adjLists, nd[j].getAttribute('ref'));
63             v2 = indexForId(adjLists, nd[j+1].getAttribute('ref'));
64
65             lat1 = getLatByAtt(nd[j].getAttribute('ref'));
66             lon1 = getLonByAtt(nd[j].getAttribute('ref'));
67             lat2 = getLatByAtt(nd[j+1].getAttribute('ref'));
68             lon2 = getLonByAtt(nd[j+1].getAttribute('ref'));
69
70             point1 = new google.maps.LatLng(lat1, lon1);
71             point2 = new google.maps.LatLng(lat2, lon2);
72             distance = google.maps.geometry.spherical.computeDistanceBetween(point1, point2);
73
74             if (oneway == "yes"){
75                 adjLists[v1].adjList = new Neighbor(v2, adjLists[v1].adjList, distance);
76             } else if (oneway == "no"){
77                 adjLists[v1].adjList = new Neighbor(v2, adjLists[v1].adjList, distance);
78                 adjLists[v2].adjList = new Neighbor(v1, adjLists[v2].adjList, distance);
79             } else if (oneway == "-1"){
80                 adjLists[v2].adjList = new Neighbor(v1, adjLists[v2].adjList, distance);
81             } else{
82                 adjLists[v1].adjList = new Neighbor(v2, adjLists[v1].adjList, distance);
83                 adjLists[v2].adjList = new Neighbor(v1, adjLists[v2].adjList, distance);
84             }
85         }
86     }
87 }
88 return adjLists;
89 }
90
91 // Kelas Map
92 function Map(){
93     var path;
94     var markers = {};
95     var nd, marker, line, content, id;
96     var image = 'icon/dot_blue.png';
97
98     var mapProp = {
99         center: new google.maps.LatLng(-6.906845432118958, 107.59851515293121),
100        zoom: 17,
101        mapTypeId: google.maps.MapTypeId.ROADMAP
102    };
103    var map=new google.maps.Map(document.getElementById("googleMap"), mapProp);

```

```

1  function indexForId(adjLists,id) {
2      for (var i=0; i < adjLists.length; i++){
3          if (adjLists[i].id == id){
4              return i;
5          }
6      }
7      return -1;
8  }
9
10
11 var currentId = 0;
12 var uniqueId = function() {
13     return ++currentId;
14 }
15
16 function isHighway(way,index){
17     var tag = way[index].getElementsByTagName("tag");
18     for (hg=0;hg<tag.length;hg++){
19         {
20             if(tag[hg].getAttribute('k') == "highway"){
21                 return true;
22             }
23         }
24     return false;
25 }
26
27 function getLatByAtt(str)
28 {
29     for (n=0;n<node.length;n++)
30     {
31         if(node[n].getAttribute('id') == str)
32         {
33             return node[n].getAttribute('lat');
34         }
35     }
36 }
37
38 function getLonByAtt(str)
39 {
40     for (m=0;m<node.length;m++)
41     {
42         if(node[m].getAttribute('id') == str)
43         {
44             return node[m].getAttribute('lon');
45         }
46     }
47 }
48
49 function addInfoWindow(marker, message) {
50     var infoWindow = new google.maps.InfoWindow({
51         content: message
52     });
53
54     google.maps.event.addListener(marker, 'click', function () {
55         infoWindow.open(map, marker);
56     });
57 }
58
59 this.hasilCari = function(rute){
60     for (i=0; i < rute.length-1; i++){
61         line = new google.maps.Polyline({
62             path: [new google.maps.LatLng(getLatByAtt(list[rute[i]].id), getLonByAtt(list[rute[i]].id)),
63                     new google.maps.LatLng(getLatByAtt(list[rute[i+1]].id), getLonByAtt(list[rute[i+1]].id))],
64             strokeColor: "#FF0000",
65             strokeOpacity: 1.0,
66             strokeWeight: 6,
67             map: map,
68         });
69     }
70 }
71
72
73 this.generate = function(){
74     for (i=0;i<way.length;i++)
75     {
76         nd = way[i].getElementsByTagName("nd");
77         if(isHighway(way,i))
78         {
79             for (j=0;j<nd.length;j++)
80             {
81                 id = uniqueId();
82                 marker = new google.maps.Marker({
83                     id: id,
84                     position: new google.maps.LatLng(getLatByAtt(nd[j].getAttribute('ref'))),
85                     getLonByAtt(nd[j].getAttribute('ref'))),
86                     map: map,
87                     icon: image,
88                 });
89                 markers[id] = marker;
90
91                 content = 'Id Node : '+nd[j].getAttribute('ref')+'<br>'+
92                 'Index Node : '+indexForId(list,nd[j].getAttribute('ref'))+'<br>'+
93                 '<a href="#" onclick="setAsal(\''+id+'\','+nd[j].getAttribute('ref')+')>Pilih
94                 sebagai asal</a>'+<br>+
95                 '<a href="#" onclick="setTujuan(\''+id+'\','+nd[j].getAttribute('ref')+')>Pilih
96                 sebagai tujuan</a>';
97
98                 addInfoWindow(marker, content);
99             }
100
101
102
103

```

```

1   for (k=0;k<nd.length -1;k++)
2   {
3       line = new google.maps.Polyline({
4           path: [new google.maps.LatLng(getLatByAtt(nd[k].getAttribute('ref')) ,
5               getLonByAtt(nd[k].getAttribute('ref'))),
6               new google.maps.LatLng(getLatByAtt(nd[k+1].getAttribute('ref')) ,
7                   getLonByAtt(nd[k+1].getAttribute('ref')))],
8               strokeColor: "#000000",
9               strokeOpacity: 0.4 ,
10              strokeWeight: 3 ,
11              map: map
12          });
13      }
14  }
15  return markers;
16 }
17 }
18 }
19
20 //Kelas PriorityQueue
21 function PriorityQueue(){
22     this._nodes = [];
23
24     this.enqueue = function(priority ,key){
25         this._nodes.push({key: key ,priority: priority});
26         this.sort();
27     }
28
29     this.dequeue = function () {
30         return this._nodes.shift().key;
31     }
32
33     this.sort = function(){
34         this._nodes.sort(function(a,b){
35             return a.priority - b.priority;
36         });
37     }
38
39     this.isEmpty = function(){
40         return !this._nodes.length;
41     }
42 }
43
44 //Kelas Dijkstra
45 function Dijkstra(){
46     var INFINITY = 1/0;
47     this.nodes = {};
48
49     this.addNode = function(name,edges){
50         this.nodes[name] = edges;
51     }
52
53     this.shortestPath = function(asal , tujuan){
54         var PQ = new PriorityQueue();
55         var distances = {};
56         var previous = {};
57         var path = [];
58         var smallest , vertex , neighbor , alt;
59
60         for(vertex in this.nodes){
61             if(vertex === asal){
62                 distances[vertex] = 0;
63                 PQ.enqueue(0 , vertex);
64             } else{
65                 distances[vertex] = INFINITY;
66                 PQ.enqueue(INFINITY , vertex);
67             }
68             previous[vertex] = null;
69         }
70
71         while (!PQ.isEmpty()){
72             smallest = PQ.dequeue();
73             if (smallest === tujuan){
74                 while (previous[smallest]){
75                     path.push(smallest);
76                     smallest = previous[smallest];
77                 }
78                 break;
79             }
80         }
81
82         if (!smallest || distances[smallest] === INFINITY){
83             continue;
84         }
85
86         for (neighbor in this.nodes[smallest]){
87             alt = distances[smallest] + this.nodes[smallest][neighbor];
88             if (alt < distances[neighbor]) {
89                 distances[neighbor] = alt;
90                 previous[neighbor] = smallest;
91                 PQ.enqueue(alt , neighbor);
92             }
93         }
94     }
95     return path;
96 }
97 }
98
99
100 var node = xmlDoc.getElementsByTagName("node");
101 var way = xmlDoc.getElementsByTagName("way");
102
103 var list = new Graph(node,way);

```

```

1 | var peta = new Map();
2 | var mark = peta.generate();
3 |
4 | //Fungsi SetAsal dan Tujuan
5 | var isAsalClicked = false;
6 | var isTujuanClicked = false;
7 | var idAsal,idTujuan,indexAsal ,indexTujuan ;
8 |
9 | function setAsal(id,idnode ,indexNode){
10 |     if (!isAsalClicked){
11 |         mark[id].setIcon('icon/dot_green.png');
12 |         isAsalClicked = true;
13 |         idAsal = id;
14 |         indexAsal = indexNode;
15 |     }else{
16 |         mark[idAsal].setIcon('icon/dot_blue.png');
17 |         mark[id].setIcon('icon/dot_green.png');
18 |         idAsal = id;
19 |         indexAsal = indexNode;
20 |     }
21 |     document.getElementById('asal').value = "Id_Node:"+ idnode;
22 | }
23 |
24 | function setTujuan(id,idnode ,indexNode){
25 |     if (!isTujuanClicked){
26 |         mark[id].setIcon('icon/dot_red.png');
27 |         isTujuanClicked = true;
28 |         idTujuan = id;
29 |         indexTujuan = indexNode;
30 |     }else{
31 |         mark[idTujuan].setIcon('icon/dot_blue.png');
32 |         mark[id].setIcon('icon/dot_red.png');
33 |         idTujuan = id;
34 |         indexTujuan = indexNode;
35 |     }
36 |     document.getElementById('tujuan').value = "Id_Node:"+ idnode;
37 | }
38 |
39 | function result(){
40 |     var cari = new Dijkstra();
41 |     var rute;
42 |     var edge;
43 |     for (j=0; j < list.length; j++){
44 |         edge = new Object();
45 |         for (nbr=list[j].adjList; nbr != null;nbr=nbr.next) {
46 |             edge[nbr.vertexNum] = nbr.weight;
47 |         }
48 |         cari.addNode(j,edge);
49 |     }
50 |
51 |     console.log("Titik_Asal:"+indexAsal);
52 |     console.log("Titik_Tujuan:"+indexTujuan);
53 |     rute = cari.shortestPath(indexAsal ,indexTujuan).concat([indexAsal]).reverse();
54 |     console.log(rute);
55 |     if(rute.length == 1){
56 |         if(indexAsal == null){
57 |             alert("Anda belum memilih titik_Asal");
58 |         }else if(indexTujuan == null){
59 |             alert("Anda belum memilih titik_Tujuan");
60 |         }else if(indexAsal != null && indexTujuan != null){
61 |             alert("Rute Tidak Ditemukan");
62 |         }
63 |     }else{
64 |         peta.hasilCari(rute);
65 |     }
66 | }
67 |
68 |</script>
69 |</body>
70 |</html>

```

Listing A.2: list.html

```

71 | <!DOCTYPE html>
72 | <html>
73 | <head>
74 | <script src="https://maps.googleapis.com/maps/api/js?v=3.exp&libraries=geometry"></script>
75 | </head>
76 | <body>
77 | <script>
78 | xmlhttp=new XMLHttpRequest();
79 | xmlhttp.open("GET","test.xml",false);
80 | xmlhttp.send();
81 | xmlDoc=xmlhttp.responseXML;
82 |
83 | function Neighbor(vnum, nbr, weight){
84 |     this.vertexNum = vnum;
85 |     this.weight = weight;
86 |     this.next = nbr;
87 | }
88 |
89 | function Node(id, neighbors){
90 |     this.id = id;
91 |     this.adjList = neighbors;
92 | }
93 |
94 | function isOneway(way,index){
95 |     var tag = way[index].getElementsByName("tag");
96 |     for (hg=0;hg<tag.length;hg++)
97 |     {
98 |         if(tag[hg].getAttribute('k') == "oneway"){
99 |             return tag[hg].getAttribute('v');

```

```

1      }
2  }
3  return false;
4 }
5
6 function isHighway(way,index){
7   var tag = way[index].getElementsByTagName("tag");
8   for (hg=0;hg<tag.length;hg++){
9     {
10       if(tag[hg].getAttribute('k') == "highway"){
11         return true;
12       }
13     }
14   }
15   return false;
16 }
17
18 function indexForId(adjLists,id) {
19   for (var i=0; i < adjLists.length; i++){
20     if (adjLists[i].id == id){
21       return i;
22     }
23   }
24   return -1;
25 }
26
27 function getLatByAtt(str)
28 {
29   for (n=0;n<node.length;n++)
30   {
31     if (node[n].getAttribute('id') == str)
32     {
33       return node[n].getAttribute('lat');
34     }
35   }
36 function getLonByAtt(str)
37 {
38   for (m=0;m<node.length;m++)
39   {
40     if (node[m].getAttribute('id') == str)
41     {
42       return node[m].getAttribute('lon');
43     }
44   }
45 }
46
47 function Graph(node,way){
48   var adjLists = [];
49   var nd;
50   var oneway;
51   var lat1,lon1,lat2,lon2;
52   var point1,point2,distance;
53
54   for(v=0; v < node.length; v++){
55     adjLists.push(new Node(node[v].getAttribute('id'),null));
56   }
57
58   for (i=0;i<way.length;i++)
59   {
60     nd = way[i].getElementsByTagName("nd");
61     if(isHighway(way,i)){
62       oneway = IsOneway(way,i);
63       for (j=0;j<nd.length-1;j++)
64       {
65         v1 = indexForId(adjLists,nd[j].getAttribute('ref'));
66         v2 = indexForId(adjLists,nd[j+1].getAttribute('ref'));
67
68         lat1 = getLatByAtt(nd[j].getAttribute('ref'));
69         lon1 = getLonByAtt(nd[j].getAttribute('ref'));
70         lat2 = getLatByAtt(nd[j+1].getAttribute('ref'));
71         lon2 = getLonByAtt(nd[j+1].getAttribute('ref'));
72
73         point1 = new google.maps.LatLng(lat1,lon1);
74         point2 = new google.maps.LatLng(lat2,lon2);
75         distance = google.maps.geometry.spherical.computeDistanceBetween(point1, point2);
76
77         if(oneway == "yes"){
78           adjLists[v1].adjList = new Neighbor(v2, adjLists[v1].adjList, distance);
79         } else if(oneway == "no"){
80           adjLists[v1].adjList = new Neighbor(v2, adjLists[v1].adjList, distance);
81           adjLists[v2].adjList = new Neighbor(v1, adjLists[v2].adjList, distance);
82         } else if(oneway == "-1"){
83           adjLists[v2].adjList = new Neighbor(v1, adjLists[v2].adjList, distance);
84         } else{
85           adjLists[v1].adjList = new Neighbor(v2, adjLists[v1].adjList, distance);
86           adjLists[v2].adjList = new Neighbor(v1, adjLists[v2].adjList, distance);
87         }
88       }
89     }
90   }
91   return adjLists;
92 }
93
94 function print(list){
95   for (j=0; j < list.length; j++){
96     document.write(j);
97     for (nbr=list[j].adjList; nbr != null; nbr=nbr.next) {
98       document.write("---->");
99       document.write('(' + nbr.vertexNum + ')');
100    }
101   }
102 }
103 }
```

```

1  function print2(list){
2      for (j=0; j < list.length; j++){
3          document.write(j);
4          document.write("—>");
5          document.write(list[j].id);
6          for (nbr=list[j].adjList; nbr != null; nbr=nbr.next) {
7              document.write("---->");
8              document.write(list[nbr.vertexNum].id);
9          }
10         document.write("<br>");
11     }
12 }
13 }
14
15 var node = xmlDoc.getElementsByTagName("node");
16 var way = xmlDoc.getElementsByTagName("way");
17
18 var list = new Graph(node,way);
19 print(list);
//print2(list);
20
21
22
23 </script>
24 </body>
25 </html>

```

Listing A.3: xml_parsing.html

```

26 <html>
27 <head>
28 <script src="https://maps.googleapis.com/maps/api/js?v=3.exp&libraries=geometry"></script>
29 <style>
30 table, th, td {
31     border: 1px solid black;
32     border-collapse: collapse;
33 }
34 th, td {
35     padding: 5px;
36 }
37 </style>
38 </head>
39 <body>
40 <script>
41 xmlhttp=new XMLHttpRequest();
42 xmlhttp.open("GET","test.xml",false);
43 xmlhttp.send();
44 xmlDoc=xmlhttp.responseXML;
45
46 function getDistance(lat1,lon1,lat2,lon2) {
47     var R = 6371; // Radius of the earth in km
48     var dLat = deg2rad(lat2-lat1); // deg2rad below
49     var dLon = deg2rad(lon2-lon1);
50     var a =
51         Math.sin(dLat/2) * Math.sin(dLat/2) +
52         Math.cos(deg2rad(lat1)) * Math.cos(deg2rad(lat2)) *
53         Math.sin(dLon/2) * Math.sin(dLon/2)
54     ;
55     var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
56     var d = R * c; // Distance in km
57     return d;
58 }
59
60 function deg2rad(deg) {
61     return deg * (Math.PI/180)
62 }
63
64 document.write("<div style='float:left'>");
65 document.write("<table><tr><th>Node</th><th>Id</th><th>Latitude </th><th>Longitude </th></tr>");
66 document.write("<caption>Node</caption>");
67 var node=xmlDoc.getElementsByTagName("node");
68 for (ct=0;ct<node.length;ct++)
69 {
70     document.write("<tr><td>");
71     document.write(ct);
72     document.write("</td><td>");
73     document.write(node[ct].getAttribute('id'));
74     document.write("</td><td>");
75     document.write(node[ct].getAttribute('lat'));
76     document.write("</td><td>");
77     document.write(node[ct].getAttribute('lon'));
78     document.write("</td></tr>");
79 }
80 document.write("</table>");
81 document.write("</div>");
82
83 function isHighway(way,index){
84     var tag = way[index].getElementsByTagName("tag");
85     for (hg=0;hg<tag.length;hg++)
86     {
87         if (tag[hg].getAttribute('k') == "highway"){
88             return true;
89         }
90     }
91     return false;
92 }
93
94 function getLatByAtt(str)
95 {
96     for (n=0;n<node.length;n++)
97     {
98         if (node[n].getAttribute('id') == str)
99         {

```

```

1         return node[n].getAttribute('lat');
2     }
3     return -1;
4 }
5 function getLonByAtt(str)
6 {
7     for (m=0;m<node.length;m++)
8     {
9         if (node[m].getAttribute('id') == str)
10        {
11            return node[m].getAttribute('lon');
12        }
13    }
14    return -1;
15 }
16 }
17 document.write("<div style='margin-left:20px;float:left'>");
18 document.write("<table><tr><th>Way</th><th>Id_Way</th><th>Edge</th><th>Id_Node_1</th><th>Id_Node_2</th><th>Jarak_dalam_meter</th></tr>");
19 document.write("<caption>Edge</caption>");
20 var way = xmlDoc.getElementsByTagName("way");
21 var nd;
22 var lat1;
23 var lon1;
24 var lat2;
25 var lon2;
26 var lon1;
27 var lon2;
28 for (i=0;i<way.length;i++)
29 {
30     nd = way[i].getElementsByTagName("nd");
31     if (isHighway(way, i)){
32         for (j=0;j<nd.length-1;j++)
33         {
34             document.write("<tr><td>");
35             document.write(i);
36             document.write("</td><td>");
37             document.write(way[i].getAttribute('id'));
38             document.write("</td><td>");
39             document.write(j);
40             document.write("</td><td>");
41             document.write(nd[j].getAttribute('ref'));
42             document.write("</td><td>");
43             document.write(nd[j+1].getAttribute('ref'));
44             document.write("</td><td>");
45             lat1 = getLatByAtt(nd[j].getAttribute('ref'));
46             lon1 = getLonByAtt(nd[j].getAttribute('ref'));
47             lat2 = getLatByAtt(nd[j+1].getAttribute('ref'));
48             lon2 = getLonByAtt(nd[j+1].getAttribute('ref'));
49
50             point1 = new google.maps.LatLng(lat1,lon1);
51             point2 = new google.maps.LatLng(lat2,lon2);
52             distance = google.maps.geometry.spherical.computeDistanceBetween(point1, point2);
53             //document.write(getDistance(lat1,lon1,lat2,lon2)*1000);
54             document.write(distance);
55             document.write("</td></tr>");
56         }
57     }
58 }
59 document.write("</div>");
60
61 </script>
62 </body>
63 </html>
```


LAMPIRAN B

TEST OSMXML

Listing B.1: test.xml

```

3  <?xml version="1.0" encoding="UTF-8"?>
4  <osm version="0.6" generator="CGImap_0.3.3_(29805_thorn-03.openstreetmap.org)" copyright=
5      "OpenStreetMap_and CONTRIBUTORS" attribution="http://www.openstreetmap.org/copyright" license=
6      "http://opendatacommons.org/licenses/odbl/1-0/">
7  <bounds minlat="-6.9076500" minlon="107.5961800" maxlat="-6.9044500" maxlon="107.6016300"/>
8  <node id="25418868" visible="true" version="6" changeset="27915808" timestamp="2015-01-04T17
9      :54:58Z" user="isonpurba" uid="2552445" lat="-6.9064389" lon="107.5976351"/>
10 <node id="25433683" visible="true" version="3" changeset="839915" timestamp="2009-03-21T14:18:48Z
11     " user="adhitya" uid="7748" lat="-6.9067659" lon="107.5989458"/>
12 <node id="25433687" visible="true" version="2" changeset="839915" timestamp="2009-03-21T14:18:36Z
13     " user="adhitya" uid="7748" lat="-6.9040267" lon="107.5969508"/>
14 <node id="25433688" visible="true" version="2" changeset="839915" timestamp="2009-03-21T14:18:58Z
15     " user="adhitya" uid="7748" lat="-6.9039393" lon="107.5963723"/>
16 <node id="25433690" visible="true" version="2" changeset="839915" timestamp="2009-03-21T14:19:02Z
17     " user="adhitya" uid="7748" lat="-6.9052824" lon="107.5961768"/>
18 <node id="25433685" visible="true" version="4" changeset="13860930" timestamp="2012-11-13T15
19      :42:42Z" user="yudiwbz" uid="268765" lat="-6.9049404" lon="107.5975738"/>
20 <node id="25433678" visible="true" version="3" changeset="14069373" timestamp="2012-11-28T07
21      :04:31Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9039784" lon="107.5985467"/>
22 <node id="25433679" visible="true" version="3" changeset="14069373" timestamp="2012-11-28T07
23      :04:31Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9049265" lon="107.5985843"/>
24 <node id="25433680" visible="true" version="4" changeset="14069373" timestamp="2012-11-28T07
25      :04:31Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9062500" lon="107.5995945"/>
26 <node id="25433681" visible="true" version="3" changeset="14069373" timestamp="2012-11-28T07
27      :04:31Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9055235" lon="107.5989193"/>
28 <node id="25434115" visible="true" version="2" changeset="839915" timestamp="2009-03-21T14:19:29Z
29     " user="adhitya" uid="7748" lat="-6.9097812" lon="107.5978508"/>
30 <node id="25500626" visible="true" version="3" changeset="839915" timestamp="2009-03-21T14:22:17Z
31     " user="adhitya" uid="7748" lat="-6.9070329" lon="107.6019401"/>
32 <node id="28802396" visible="true" version="2" changeset="839915" timestamp="2009-03-21T14:23:18Z
33     " user="adhitya" uid="7748" lat="-6.9055299" lon="107.5976092"/>
34 <tag k="created_by" v="YahooApplet_1.0"/>
35 </node>
36 <node id="29356177" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:47:33Z"
37     user="adhitya" uid="7748" lat="-6.9102898" lon="107.5994584"/>
38 <node id="29356178" visible="true" version="2" changeset="11905099" timestamp="2012-06-15T14
39      :05:57Z" user="andryono" uid="643030" lat="-6.9090157" lon="107.5994925"/>
40 <node id="29356374" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:50:18Z"
41     user="adhitya" uid="7748" lat="-6.9081596" lon="107.5978289"/>
42 <node id="29356381" visible="true" version="2" changeset="27915800" timestamp="2015-01-04T17
43      :57:25Z" user="isonpurba" uid="2552445" lat="-6.9082496" lon="107.5977288"/>
44 <node id="29356499" visible="true" version="2" changeset="839915" timestamp="2009-03-21T14:20:07Z
45     " user="adhitya" uid="7748" lat="-6.9099650" lon="107.5978505"/>
46 <node id="29356500" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:49:27Z"
47     user="adhitya" uid="7748" lat="-6.9099148" lon="107.5983941"/>
48 <node id="29356501" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:49:27Z"
49     user="adhitya" uid="7748" lat="-6.9094973" lon="107.5983770"/>
50 <node id="29356502" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:50:18Z"
51     user="adhitya" uid="7748" lat="-6.9082022" lon="107.5983598"/>
52 <node id="29356503" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:49:27Z"
53     user="adhitya" uid="7748" lat="-6.9075802" lon="107.5983340"/>
54 <node id="29356504" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:49:27Z"
55     user="adhitya" uid="7748" lat="-6.9071626" lon="107.5983083"/>
56 <node id="29356528" visible="true" version="2" changeset="839915" timestamp="2009-03-21T14:20:07Z
57     " user="adhitya" uid="7748" lat="-6.9094946" lon="107.5978334"/>
58 <node id="29356534" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:49:55Z"
59     user="adhitya" uid="7748" lat="-6.9076313" lon="107.5990808"/>
60 <node id="29356535" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:49:55Z"
61     user="adhitya" uid="7748" lat="-6.9076057" lon="107.5987203"/>
62 <node id="29356540" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:50:15Z"
63     user="adhitya" uid="7748" lat="-6.9081511" lon="107.5990722"/>
64 <node id="29376827" visible="true" version="1" changeset="58597" timestamp="2007-05-20T11:57:31Z"
65     user="adhitya" uid="7748" lat="-6.9052966" lon="107.5968062"/>
66 <node id="29376829" visible="true" version="1" changeset="58597" timestamp="2007-05-20T11:57:31Z"
67     user="adhitya" uid="7748" lat="-6.9045382" lon="107.5968234"/>
68 <node id="29376832" visible="true" version="1" changeset="58597" timestamp="2007-05-20T11:57:15Z"
69     user="adhitya" uid="7748" lat="-6.9047342" lon="107.5951840"/>
70 <node id="29376835" visible="true" version="1" changeset="58597" timestamp="2007-05-20T11:57:15Z"
71     user="adhitya" uid="7748" lat="-6.9054755" lon="107.5951411"/>
72 <node id="29376836" visible="true" version="2" changeset="839915" timestamp="2009-03-21T14:21:11Z
73     " user="adhitya" uid="7748" lat="-6.9055948" lon="107.5961854"/>
74 <node id="29376837" visible="true" version="1" changeset="58597" timestamp="2007-05-20T11:57:12Z"
75     user="adhitya" uid="7748" lat="-6.9055863" lon="107.5957334"/>
76 <node id="29376839" visible="true" version="1" changeset="58597" timestamp="2007-05-20T11:56:55Z"
77     user="adhitya" uid="7748" lat="-6.9049472" lon="107.5962054"/>
78 <node id="29376842" visible="true" version="1" changeset="58597" timestamp="2007-05-20T11:57:47Z"
79     user="adhitya" uid="7748" lat="-6.9073927" lon="107.5954330"/>
```

```

1 <node id="29376841" visible="true" version="2" changeset="12686463" timestamp="2012-08-11T00
2   :36:04Z" user="ferdyodin" uid="796044" lat="-6.9071561" lon="107.5968980"/>
3 <node id="29392373" visible="true" version="2" changeset="839915" timestamp="2009-03-21T14:19:03Z
4   " user="adhitya" uid="7748" lat="-6.9049590" lon="107.6005861"/>
5 <node id="29392374" visible="true" version="1" changeset="147858" timestamp="2007-07-16T21:40:29Z
6   " user="adhitya" uid="7748" lat="-6.9051439" lon="107.6005958">
7   <tag k="name" v="Imperium"/>
8   <tag k="tourism" v="hotel"/>
9 </node>
10 <node id="29392377" visible="true" version="2" changeset="839915" timestamp="2009-03-21T14:19:03Z
11   " user="adhitya" uid="7748" lat="-6.9053946" lon="107.6007639"/>
12 <node id="29392413" visible="true" version="3" changeset="839915" timestamp="2009-03-21T14:17:04Z
13   " user="adhitya" uid="7748" lat="-6.9071747" lon="107.6039686"/>
14 <node id="32041785" visible="true" version="1" changeset="147858" timestamp="2007-07-16T21:20:58Z
15   " user="adhitya" uid="7748" lat="-6.9045608" lon="107.5990459">
16   <tag k="amenity" v="restaurant"/>
17   <tag k="created_by" v="JOSM"/>
18   <tag k="name" v="Chiba"/>
19 </node>
20 <node id="32041794" visible="true" version="2" changeset="839915" timestamp="2009-03-21T14:20:08Z
21   " user="adhitya" uid="7748" lat="-6.9049540" lon="107.5990764">
22   <tag k="created_by" v="JOSM"/>
23 </node>
24 <node id="32041828" visible="true" version="2" changeset="13856395" timestamp="2012-11-13T08
25   :27:46Z" user="yudiwbs" uid="268765" lat="-6.9028198" lon="107.5974882"/>
26 <node id="32042055" visible="true" version="1" changeset="147858" timestamp="2007-07-16T21:25:13Z
27   " user="adhitya" uid="7748" lat="-6.9045275" lon="107.5973424">
28   <tag k="amenity" v="place_of_worship"/>
29   <tag k="created_by" v="JOSM"/>
30   <tag k="name" v="GBIS_Harmoni"/>
31   <tag k="religion" v="christian"/>
32 </node>
33 <node id="32042131" visible="true" version="1" changeset="147858" timestamp="2007-07-16T21:25:27Z
34   " user="adhitya" uid="7748" lat="-6.9050969" lon="107.5973817">
35   <tag k="amenity" v="university"/>
36   <tag k="created_by" v="JOSM"/>
37   <tag k="name" v="Ariyanti"/>
38 </node>
39 <node id="32520365" visible="true" version="3" changeset="13860930" timestamp="2012-11-13T15
40   :42:42Z" user="yudiwbs" uid="268765" lat="-6.9068829" lon="107.5976578"/>
41 <node id="32529146" visible="true" version="3" changeset="839915" timestamp="2009-03-21T14:16:03Z
42   " user="adhitya" uid="7748" lat="-6.9071333" lon="107.6033567">
43   <tag k="created_by" v="JOSM"/>
44 </node>
45 <node id="364363837" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:02:46
46   " user="adhitya" uid="7748" lat="-6.9109164" lon="107.5979263"/>
47 <node id="364363838" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:02:48
48   " user="adhitya" uid="7748" lat="-6.9103801" lon="107.5978906"/>
49 <node id="364363888" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:09
50   " user="adhitya" uid="7748" lat="-6.9055397" lon="107.5965000"/>
51 <node id="364363889" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:10
52   " user="adhitya" uid="7748" lat="-6.9056185" lon="107.5965003"/>
53 <node id="364363890" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:10
54   " user="adhitya" uid="7748" lat="-6.9056188" lon="107.5964229"/>
55 <node id="364363895" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:14
56   " user="adhitya" uid="7748" lat="-6.9057778" lon="107.5964236"/>
57 <node id="364363896" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:15
58   " user="adhitya" uid="7748" lat="-6.9057775" lon="107.5964918"/>
59 <node id="364363897" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:17
60   " user="adhitya" uid="7748" lat="-6.9061063" lon="107.5964919"/>
61 <node id="364363898" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:18
62   " user="adhitya" uid="7748" lat="-6.9061031" lon="107.5966256"/>
63 <node id="364363900" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:21
64   " user="adhitya" uid="7748" lat="-6.9062220" lon="107.5966261"/>
65 <node id="364363901" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:23
66   " user="adhitya" uid="7748" lat="-6.9062225" lon="107.5969390"/>
67 <node id="364363903" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:27
68   " user="adhitya" uid="7748" lat="-6.9059809" lon="107.5969397"/>
69 <node id="364363948" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:31
70   " user="adhitya" uid="7748" lat="-6.9059787" lon="107.5974369"/>
71 <node id="364363954" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:32
72   " user="adhitya" uid="7748" lat="-6.9054018" lon="107.5974346"/>
73 <node id="364363962" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:33
74   " user="adhitya" uid="7748" lat="-6.9054047" lon="107.5967435"/>
75 <node id="364363968" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:34
76   " user="adhitya" uid="7748" lat="-6.9055386" lon="107.5967441"/>
77 <node id="364364075" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:04:26
78   " user="adhitya" uid="7748" lat="-6.9060974" lon="107.5971201"/>
79 <node id="364364076" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:04:28
80   " user="adhitya" uid="7748" lat="-6.9062244" lon="107.5971218"/>
81 <node id="364364077" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:04:30
82   " user="adhitya" uid="7748" lat="-6.9062210" lon="107.5973845"/>
83 <node id="364364078" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:04:30
84   " user="adhitya" uid="7748" lat="-6.9060940" lon="107.5973828"/>
85 <node id="364364087" visible="true" version="2" changeset="1899770" timestamp="2013-11-19T17
86   :08:36Z" user="ubanovic" uid="1784103" lat="-6.9048017" lon="107.6022644"/>
87 <node id="364364086" visible="true" version="2" changeset="12593148" timestamp="2012-08-03T02
88   :44:39Z" user="ivan_elangrawa" uid="771585" lat="-6.9046383" lon="107.6024234"/>
89 <node id="364364182" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:05:38
90   " user="adhitya" uid="7748" lat="-6.9053722" lon="107.5975201"/>
91 <node id="364364183" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:05:39
92   " user="adhitya" uid="7748" lat="-6.9062774" lon="107.5975276"/>
93 <node id="364364184" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:05:43
94   " user="adhitya" uid="7748" lat="-6.9062749" lon="107.5970425"/>
95 <node id="364364192" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:05:48
96   " user="adhitya" uid="7748" lat="-6.9062745" lon="107.5968821"/>
97 <node id="364364194" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:05:56
98   " user="adhitya" uid="7748" lat="-6.9060423" lon="107.5970375"/>
99 <node id="364364195" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:05:57
100  " user="adhitya" uid="7748" lat="-6.9060323" lon="107.5975301"/>
101 <node id="364364202" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:05:59
102  " user="adhitya" uid="7748" lat="-6.9053647" lon="107.5968575"/>
```

```

1  <node id="364364220" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:06:06
2    Z" user="adhitya" uid="7748" lat="-6.9053972" lon="107.5968557" />
3  <node id="364364237" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:06:10
4    Z" user="adhitya" uid="7748" lat="-6.9062223" lon="107.5968807" />
5  <node id="364364242" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:06:11
6    Z" user="adhitya" uid="7748" lat="-6.9053002" lon="107.5968551" />
7  <node id="364364247" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:06:11
8    Z" user="adhitya" uid="7748" lat="-6.9061909" lon="107.5962609" />
9  <node id="364364251" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:06:12
10   Z" user="adhitya" uid="7748" lat="-6.9058764" lon="107.5962152" />
11 <node id="364364179" visible="true" version="2" changeset="13860930" timestamp="2012-11-13T15
12 :42:43Z" user="yudiwbs" uid="268765" lat="-6.9053684" lon="107.5975736" />
13 <node id="364365012" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:19
14   Z" user="adhitya" uid="7748" lat="-6.9069591" lon="107.6008710" />
15 <node id="364365018" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:20
16   Z" user="adhitya" uid="7748" lat="-6.9075810" lon="107.6008234" />
17 <node id="364365025" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:21
18   Z" user="adhitya" uid="7748" lat="-6.9076078" lon="107.6012281" />
19 <node id="364365061" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:25
20   Z" user="adhitya" uid="7748" lat="-6.9071287" lon="107.6012668" />
21 <node id="364365069" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:25
22   Z" user="adhitya" uid="7748" lat="-6.9070930" lon="107.6008561" />
23 <node id="364365077" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:26
24   Z" user="adhitya" uid="7748" lat="-6.9075869" lon="107.6009692" />
25 <node id="364365090" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:34
26   Z" user="adhitya" uid="7748" lat="-6.9071108" lon="107.6009990" />
27 <node id="364365091" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:34
28   Z" user="adhitya" uid="7748" lat="-6.9075929" lon="107.6011001" />
29 <node id="364365096" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:35
30   Z" user="adhitya" uid="7748" lat="-6.9071257" lon="107.6011269" />
31 <node id="364365097" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:37
32   Z" user="adhitya" uid="7748" lat="-6.9071132" lon="107.6004979" />
33 <node id="364365098" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:41
34   Z" user="adhitya" uid="7748" lat="-6.9071511" lon="107.6007965" />
35 <node id="364365099" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:42
36   Z" user="adhitya" uid="7748" lat="-6.9075492" lon="107.6007585" />
37 <node id="364365133" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:49
38   Z" user="adhitya" uid="7748" lat="-6.9075161" lon="107.6004505" />
39 <node id="364365134" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:51
40   Z" user="adhitya" uid="7748" lat="-6.9076448" lon="107.6007866" />
41 <node id="364365135" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:55
42   Z" user="adhitya" uid="7748" lat="-6.9080087" lon="107.6007623" />
43 <node id="364365136" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:56
44   Z" user="adhitya" uid="7748" lat="-6.9080421" lon="107.6012506" />
45 <node id="364365137" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:58
46   Z" user="adhitya" uid="7748" lat="-6.9076812" lon="107.6012810" />
47 <node id="364365142" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:02
48   Z" user="adhitya" uid="7748" lat="-6.9071929" lon="107.6017996" />
49 <node id="364365143" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:02
50   Z" user="adhitya" uid="7748" lat="-6.9073415" lon="107.6014084" />
51 <node id="364365144" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:04
52   Z" user="adhitya" uid="7748" lat="-6.9083514" lon="107.6013295" />
53 <node id="364365145" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:05
54   Z" user="adhitya" uid="7748" lat="-6.9086244" lon="107.6016722" />
55 <node id="364365146" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:12
56   Z" user="adhitya" uid="7748" lat="-6.9084728" lon="107.6020513" />
57 <node id="364365151" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:16
58   Z" user="adhitya" uid="7748" lat="-6.9074021" lon="107.6021272" />
59 <node id="364365155" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:17
60   Z" user="adhitya" uid="7748" lat="-6.9072323" lon="107.6015873" />
61 <node id="364365159" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:18
62   Z" user="adhitya" uid="7748" lat="-6.9072656" lon="107.6020028" />
63 <node id="364365161" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:22
64   Z" user="adhitya" uid="7748" lat="-6.9085729" lon="107.6018997" />
65 <node id="364365162" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:23
66   Z" user="adhitya" uid="7748" lat="-6.9085334" lon="107.6014690" />
67 <node id="364365164" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:24
68   Z" user="adhitya" uid="7748" lat="-6.9071929" lon="107.6016935" />
69 <node id="364365165" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:26
70   Z" user="adhitya" uid="7748" lat="-6.9072687" lon="107.6014872" />
71 <node id="364365166" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:26
72   Z" user="adhitya" uid="7748" lat="-6.9072201" lon="107.6019088" />
73 <node id="364365167" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:27
74   Z" user="adhitya" uid="7748" lat="-6.9073202" lon="107.6020726" />
75 <node id="364365168" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:28
76   Z" user="adhitya" uid="7748" lat="-6.9085365" lon="107.6019786" />
77 <node id="364365169" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:30
78   Z" user="adhitya" uid="7748" lat="-6.9086123" lon="107.6017875" />
79 <node id="364365170" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:31
80   Z" user="adhitya" uid="7748" lat="-6.9085911" lon="107.6015600" />
81 <node id="364365171" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:33
82   Z" user="adhitya" uid="7748" lat="-6.9084788" lon="107.6013811" />
83 <node id="1666615973" visible="true" version="1" changeset="10914639" timestamp="2012-03-08T22
84 :37:16Z" user="juey" uid="617650" lat="-6.9057580" lon="107.5953036" />
85 <node id="1666616129" visible="true" version="1" changeset="10914639" timestamp="2012-03-08T22
86 :37:19Z" user="juey" uid="617650" lat="-6.9058491" lon="107.5955873" />
87 <node id="1666616062" visible="true" version="1" changeset="10914639" timestamp="2012-03-08T22
88 :37:18Z" user="juey" uid="617650" lat="-6.9058491" lon="107.5946111" />
89 <node id="1666615284" visible="true" version="1" changeset="10914639" timestamp="2012-03-08T22
90 :36:51Z" user="juey" uid="617650" lat="-6.9041841" lon="107.5963120" />
91 <node id="1666616021" visible="true" version="1" changeset="10914639" timestamp="2012-03-08T22
92 :37:17Z" user="juey" uid="617650" lat="-6.9057870" lon="107.5948489" />
93 <node id="1666616044" visible="true" version="1" changeset="10914639" timestamp="2012-03-08T22
94 :37:17Z" user="juey" uid="617650" lat="-6.9058242" lon="107.5958834" />
95 <node id="1666616103" visible="true" version="1" changeset="10914639" timestamp="2012-03-08T22
96 :37:19Z" user="juey" uid="617650" lat="-6.9058491" lon="107.5950283" />
97 <node id="1666616003" visible="true" version="1" changeset="10914639" timestamp="2012-03-08T22
98 :37:16Z" user="juey" uid="617650" lat="-6.9057745" lon="107.591367" />
99 <node id="1700554868" visible="true" version="1" changeset="11172636" timestamp="2012-04-01T03
100 :16:26Z" user="andryono" uid="643030" lat="-6.9068452" lon="107.5979369" />
101 <node id="1700554884" visible="true" version="1" changeset="11172636" timestamp="2012-04-01T03
102 :16:27Z" user="andryono" uid="643030" lat="-6.9068613" lon="107.5977407" />
```

```

1 <node id="1700554892" visible="true" version="1" changeset="11172636" timestamp="2012-04-01T03
2   :16:27Z" user="andryono" uid="643030" lat="-6.9067023" lon="107.5977275"/>
3 <node id="1700554901" visible="true" version="1" changeset="11172636" timestamp="2012-04-01T03
4   :16:27Z" user="andryono" uid="643030" lat="-6.9066862" lon="107.5979237"/>
5 <node id="1706564155" visible="true" version="1" changeset="11220464" timestamp="2012-04-08T02
6   :49:00Z" user="andryono" uid="643030" lat="-6.9070955" lon="107.6028239"/>
7 <node id="1706564168" visible="true" version="1" changeset="11220464" timestamp="2012-04-08T02
8   :49:01Z" user="andryono" uid="643030" lat="-6.9070663" lon="107.6024111"/>
9 <node id="25447003" visible="true" version="5" changeset="27916246" timestamp="2015-01-04T18
10  :10:18Z" user="isonpurba" uid="2552445" lat="-6.9040554" lon="107.6013227"/>
11 <node id="29356179" visible="true" version="2" changeset="11905099" timestamp="2012-06-15T14
12  :05:57Z" user="andryono" uid="643030" lat="-6.9081649" lon="107.5995085"/>
13 <node id="29356180" visible="true" version="2" changeset="11905099" timestamp="2012-06-15T14
14  :05:57Z" user="andryono" uid="643030" lat="-6.9076627" lon="107.5995236"/>
15 <node id="25433684" visible="true" version="4" changeset="11905099" timestamp="2012-06-15T14
16  :05:57Z" user="andryono" uid="643030" lat="-6.9068381" lon="107.5995016"/>
17 <node id="1860932973" visible="true" version="1" changeset="12686463" timestamp="2012-08-11T00
18  :36:04Z" user="ferdyodin" uid="796044" lat="-6.9099128" lon="107.5971365"/>
19 <node id="1860932975" visible="true" version="1" changeset="12686463" timestamp="2012-08-11T00
20  :36:04Z" user="ferdyodin" uid="796044" lat="-6.9098721" lon="107.5978507"/>
21 <node id="1860932974" visible="true" version="3" changeset="14069373" timestamp="2012-11-28T07
22  :04:18Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9065003" lon="107.5968009"/>
23 <node id="2012498678" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
24  :42:41Z" user="yudiwbs" uid="268765" lat="-6.9065807" lon="107.5978333"/>
25 <node id="2012498680" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
26  :42:41Z" user="yudiwbs" uid="268765" lat="-6.9054261" lon="107.5975749"/>
27 <node id="2012498682" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
28  :42:41Z" user="yudiwbs" uid="268765" lat="-6.9064253" lon="107.5963227"/>
29 <node id="2012498684" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
30  :42:41Z" user="yudiwbs" uid="268765" lat="-6.9066150" lon="107.5976332"/>
31 <node id="2012498693" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
32  :42:41Z" user="yudiwbs" uid="268765" lat="-6.9066061" lon="107.5980546"/>
33 <node id="2012498699" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
34  :42:42Z" user="yudiwbs" uid="268765" lat="-6.9065619" lon="107.5976909"/>
35 <node id="2012498701" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
36  :42:42Z" user="yudiwbs" uid="268765" lat="-6.9065713" lon="107.5976410"/>
37 <node id="2012498709" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
38  :42:42Z" user="yudiwbs" uid="268765" lat="-6.9070083" lon="107.5976580"/>
39 <node id="29376826" visible="true" version="2" changeset="13860930" timestamp="2012-11-13T15
40  :42:42Z" user="yudiwbs" uid="268765" lat="-6.9053204" lon="107.5975733"/>
41 <node id="25500712" visible="true" version="4" changeset="13860930" timestamp="2012-11-13T15
42  :42:42Z" user="yudiwbs" uid="268765" lat="-6.9066888" lon="107.5983123"/>
43 <node id="2012498673" visible="true" version="4" changeset="27915808" timestamp="2015-01-04T17
44  :54:58Z" user="isonpurba" uid="2552445" lat="-6.9064863" lon="107.5975537"/>
45 <node id="2012502549" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
46  :47:31Z" user="yudiwbs" uid="268765" lat="-6.9061123" lon="107.5981368"/>
47 <node id="2012502551" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
48  :47:31Z" user="yudiwbs" uid="268765" lat="-6.9059552" lon="107.5979383"/>
49 <node id="2012502553" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
50  :47:31Z" user="yudiwbs" uid="268765" lat="-6.9060146" lon="107.5978211"/>
51 <node id="2012502555" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
52  :47:31Z" user="yudiwbs" uid="268765" lat="-6.9069030" lon="107.5974055"/>
53 <node id="2012502557" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
54  :47:31Z" user="yudiwbs" uid="268765" lat="-6.9060119" lon="107.5979465"/>
55 <node id="2012502559" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
56  :47:31Z" user="yudiwbs" uid="268765" lat="-6.9069128" lon="107.5972456"/>
57 <node id="2012502561" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
58  :47:31Z" user="yudiwbs" uid="268765" lat="-6.9060617" lon="107.5978739"/>
59 <node id="2012502563" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
60  :47:31Z" user="yudiwbs" uid="268765" lat="-6.9067851" lon="107.5973982"/>
61 <node id="2012502565" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
62  :47:31Z" user="yudiwbs" uid="268765" lat="-6.9060412" lon="107.5976890"/>
63 <node id="2012502566" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
64  :47:31Z" user="yudiwbs" uid="268765" lat="-6.9063927" lon="107.5977024"/>
65 <node id="2012502567" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
66  :47:31Z" user="yudiwbs" uid="268765" lat="-6.9060039" lon="107.5981584"/>
67 <node id="2012502568" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
68  :47:31Z" user="yudiwbs" uid="268765" lat="-6.9059533" lon="107.5981530"/>
69 <node id="2012502569" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
70  :47:31Z" user="yudiwbs" uid="268765" lat="-6.9068199" lon="107.5973263"/>
71 <node id="2012502570" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
72  :47:31Z" user="yudiwbs" uid="268765" lat="-6.9061184" lon="107.5980645"/>
73 <node id="2012502572" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
74  :47:31Z" user="yudiwbs" uid="268765" lat="-6.9064931" lon="107.5980644"/>
75 <node id="2012502574" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
76  :47:31Z" user="yudiwbs" uid="268765" lat="-6.9068252" lon="107.5972402"/>
77 <node id="2012502576" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
78  :47:31Z" user="yudiwbs" uid="268765" lat="-6.9067896" lon="107.5973244"/>
79 <node id="2012502577" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
80  :47:31Z" user="yudiwbs" uid="268765" lat="-6.9060066" lon="107.5981235"/>
81 <node id="1700556141" visible="true" version="2" changeset="13860930" timestamp="2012-11-13T15
82  :47:32Z" user="yudiwbs" uid="268765" lat="-6.9067781" lon="107.5978365">
83   <tag k="amenity" v="fast_food"/>
84   <tag k="cuisine" v="american"/>
85   <tag k="name" v="KFC"/>
86 </node>
87 <node id="2012512606" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
88  :51:18Z" user="yudiwbs" uid="268765" lat="-6.9052099" lon="107.5972778"/>
89 <node id="2012512608" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
90  :51:18Z" user="yudiwbs" uid="268765" lat="-6.9050292" lon="107.5972977"/>
91 <node id="2012512610" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
92  :51:18Z" user="yudiwbs" uid="268765" lat="-6.9052257" lon="107.5970093"/>
93 <node id="2012512612" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
94  :51:18Z" user="yudiwbs" uid="268765" lat="-6.9052321" lon="107.5974396"/>
95 <node id="2012512614" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
96  :51:18Z" user="yudiwbs" uid="268765" lat="-6.9051312" lon="107.5970096"/>
97 <node id="2012512616" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
98  :51:18Z" user="yudiwbs" uid="268765" lat="-6.9049922" lon="107.5972962"/>
99 <node id="2012512618" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
100 :51:18Z" user="yudiwbs" uid="268765" lat="-6.9050263" lon="107.5974231"/>
101 <node id="2012512620" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
102 :51:18Z" user="yudiwbs" uid="268765" lat="-6.9052440" lon="107.5972790"/>

```

```

1  <node id="2012512622" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
2    :51:18Z" user="yudiwbs" uid="268765" lat="-6.9052298" lon="107.5971181"/>
3  <node id="2012512623" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
4    :51:18Z" user="yudiwbs" uid="268765" lat="-6.9052039" lon="107.5971181"/>
5  <node id="2012512624" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
6    :51:18Z" user="yudiwbs" uid="268765" lat="-6.9049711" lon="107.5970233"/>
7  <node id="1837514693" visible="true" version="3" changeset="14069373" timestamp="2012-11-28T07
8    :04:08Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9071711" lon="107.6044350"/>
9  <node id="2025969631" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07
10   :04:29Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9064106" lon="107.5968064"/>
11 <node id="1666615070" visible="true" version="3" changeset="14069373" timestamp="2012-11-28T07
12   :03:54Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9033176" lon="107.5975100"/>
13 <node id="2012498676" visible="true" version="3" changeset="14069373" timestamp="2012-11-28T07
14   :04:29Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9070495" lon="107.5976364"/>
15 <node id="2012498686" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07
16   :04:29Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9065298" lon="107.5959311"/>
17 <node id="2012498695" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07
18   :04:29Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9055222" lon="107.5975513"/>
19 <node id="2012498697" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07
20   :04:29Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9054179" lon="107.5975701"/>
21 <node id="2012498703" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07
22   :04:29Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9065988" lon="107.5958098"/>
23 <node id="2012498705" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07
24   :04:29Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9064822" lon="107.5963313"/>
25 <node id="2012498707" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07
26   :04:29Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9066042" lon="107.5959524"/>
27 <node id="2012498711" visible="true" version="3" changeset="14069373" timestamp="2012-11-28T07
28   :04:29Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9063931" lon="107.5975475"/>
29 <node id="2012498713" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07
30   :04:29Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9069982" lon="107.5975809"/>
31 <node id="25433682" visible="true" version="3" changeset="14069373" timestamp="2012-11-28T07
32   :04:32Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9049424" lon="107.5989062"/>
33 <node id="25433686" visible="true" version="4" changeset="14069373" timestamp="2012-11-28T07
34   :04:32Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9039897" lon="107.5975474"/>
35 <node id="25433689" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07
36   :04:32Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9045058" lon="107.5962277"/>
37 <node id="29376831" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07
38   :04:34Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9042246" lon="107.5952578"/>
39 <node id="29376843" visible="true" version="3" changeset="14069373" timestamp="2012-11-28T07
40   :04:34Z" user="dadan_dany_dipoera" uid="922816" lat="-6.9066987" lon="107.5953179"/>
41 <node id="32520364" visible="true" version="4" changeset="27915808" timestamp="2015-01-04T17
42   :54:58Z" user="isonpurba" uid="2552445" lat="-6.9064793" lon="107.5972458"/>
43 <node id="2325451578" visible="true" version="2" changeset="18997770" timestamp="2013-11-19T17
44   :08:36Z" user="ubanovic" uid="1784103" lat="-6.9048417" lon="107.6021410"/>
45 <node id="2325451571" visible="true" version="3" changeset="24131990" timestamp="2014-07-14T03
46   :03:32Z" user="brambanan" uid="2092576" lat="-6.9048702" lon="107.6013524"/>
47 <node id="2325451286" visible="true" version="2" changeset="18998018" timestamp="2013-11-19T17
48   :23:34Z" user="ubanovic" uid="1784103" lat="-6.9012529" lon="107.5974289"/>
49 <node id="2490686820" visible="true" version="1" changeset="18276242" timestamp="2013-10-10T07
50   :28:52Z" user="ArjanO" uid="38066" lat="-6.9071703" lon="107.5996685">
51 <tag k="name" v="Alfamart"/>
52 <tag k="shop" v="supermarket"/>
53 </node>
54 <node id="2628264132" visible="true" version="1" changeset="20081263" timestamp="2014-01-19T09
55   :38:29Z" user="Irfan_Muhammad" uid="646006" lat="-6.9116314" lon="107.5979581"/>
56 <node id="29376968" visible="true" version="3" changeset="20081263" timestamp="2014-01-19T09
57   :38:32Z" user="Irfan_Muhammad" uid="646006" lat="-6.9117870" lon="107.5979651"/>
58 <node id="2725732963" visible="true" version="1" changeset="21180806" timestamp="2014-03-18T19
59   :57:06Z" user="Warung_Nasi_Kang_EWOK" uid="1991592" lat="-6.9052749" lon="107.5961955">
60 <tag k="addr:city" v="Bandung"/>
61 <tag k="addr:housenumber" v="16"/>
62 <tag k="addr:street" v="Astina"/>
63 <tag k="amenity" v="restaurant"/>
64 <tag k="capacity" v="50"/>
65 <tag k="cuisine" v="sundanese"/>
66 <tag k="name" v="Warung_Nasi_Kang_EWOK"/>
67 <tag k="phone" v="0226038783_02292035359"/>
68 </node>
69 <node id="3030289969" visible="true" version="1" changeset="24892866" timestamp="2014-08-20T18
70   :40:31Z" user="albahrimaraxsa" uid="2162153" lat="-6.9069924" lon="107.5982831"/>
71 <node id="3030289970" visible="true" version="1" changeset="24892866" timestamp="2014-08-20T18
72   :40:31Z" user="albahrimaraxsa" uid="2162153" lat="-6.9067987" lon="107.5982587"/>
73 <node id="3030289971" visible="true" version="1" changeset="24892866" timestamp="2014-08-20T18
74   :40:31Z" user="albahrimaraxsa" uid="2162153" lat="-6.9066710" lon="107.5982569"/>
75 <node id="2325451442" visible="true" version="4" changeset="27916144" timestamp="2015-01-04T18
76   :06:33Z" user="isonpurba" uid="2552445" lat="-6.9045011" lon="107.6024922"/>
77 <way id="4567626" visible="true" version="4" changeset="15861148" timestamp="2013-04-25T13:56:12Z
78   " user="mrdoogieg94" uid="1331966">
79 <nd ref="25433682"/>
80 <nd ref="25433681"/>
81 <nd ref="25433680"/>
82 <tag k="avgspeed" v="15"/>
83 <tag k="highway" v="residential"/>
84 <tag k="name" v="Dr._Rubini"/>
85 </way>
86 <way id="4567634" visible="true" version="2" changeset="7821743" timestamp="2011-04-10T11:15:30Z"
87   user="evo2mind" uid="234610">
88 <nd ref="25433681"/>
89 <nd ref="28802396"/>
90 <tag k="avgspeed" v="15"/>
91 <tag k="highway" v="residential"/>
92 <tag k="name" v="Dr._Susilo"/>
93 </way>
94 <way id="4625182" visible="true" version="5" changeset="11905099" timestamp="2012-06-15T14:05:57Z
95   " user="andryono" uid="643030">
96 <nd ref="29376826"/>
97 <nd ref="3643642422"/>
98 <nd ref="29376827"/>
99 <nd ref="25433690"/>
100 <tag k="avgspeed" v="20"/>
101 <tag k="highway" v="living_street"/>
102 <tag k="oneway" v="yes"/>
103 </way>
```

```

1  <way id="4627111" visible="true" version="2" changeset="7821743" timestamp="2011-04-10T11:15:36Z"
2    user="evo2mind" uid="234610">
3    <nd ref="29356177"/>
4    <nd ref="29356178"/>
5    <nd ref="29356179"/>
6    <nd ref="29356180"/>
7    <nd ref="25433684"/>
8    <tag k="avgspeed" v="15"/>
9    <tag k="highway" v="residential"/>
10   <tag k="name" v="Muhammad_Yunus"/>
11  </way>
12 <way id="4628057" visible="true" version="3" changeset="11905099" timestamp="2012-06-15T14:05:58Z"
13   " user="andryono" uid="643030">
14   <nd ref="29392373"/>
15   <nd ref="29392374"/>
16   <nd ref="29392377"/>
17   <tag k="avgspeed" v="10"/>
18   <tag k="highway" v="service"/>
19   <tag k="service" v="parking_aisle"/>
20  </way>
21 <way id="4907167" visible="true" version="2" changeset="7821743" timestamp="2011-04-10T11:12:42Z"
22   user="evo2mind" uid="234610">
23   <nd ref="32041794"/>
24   <nd ref="32041785"/>
25   <tag k="avgspeed" v="10"/>
26  </way>
27 <way id="247058985" visible="true" version="2" changeset="24131990" timestamp="2014-07-14T03
28 :03:32Z" user="brambanan" uid="2092576">
29   <nd ref="2325451442"/>
30   <nd ref="364364086"/>
31   <nd ref="364364087"/>
32   <nd ref="2325451578"/>
33   <nd ref="2325451571"/>
34   <tag k="avgspeed" v="30"/>
35   <tag k="highway" v="secondary"/>
36   <tag k="name" v="Dr. Rum"/>
37   <tag k="oneway" v="yes"/>
38  </way>
39 <way id="32388775" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:15:09Z"
40   user="adhitya" uid="7748">
41   <nd ref="364363888"/>
42   <nd ref="364363889"/>
43   <nd ref="364363890"/>
44   <nd ref="364363895"/>
45   <nd ref="364363896"/>
46   <nd ref="364363897"/>
47   <nd ref="364363898"/>
48   <nd ref="364363900"/>
49   <nd ref="364364237"/>
50   <nd ref="364363901"/>
51   <nd ref="364363903"/>
52   <nd ref="364363948"/>
53   <nd ref="364363954"/>
54   <nd ref="364364220"/>
55   <nd ref="364363962"/>
56   <nd ref="364363968"/>
57   <nd ref="364363888"/>
58   <tag k="amenity" v="fast_food,restaurant"/>
59   <tag k="atm" v="yes"/>
60   <tag k="building" v="yes"/>
61   <tag k="name" v="Istana_Plaza"/>
62   <tag k="shop" v="supermarket"/>
63  </way>
64 <way id="32388776" visible="true" version="2" changeset="943187" timestamp="2009-04-25T10:24:30Z"
65   user="Andre68" uid="31231">
66   <nd ref="364364075"/>
67   <nd ref="364364076"/>
68   <nd ref="364364077"/>
69   <nd ref="364364078"/>
70   <nd ref="364364075"/>
71   <tag k="amenity" v="fast_food"/>
72   <tag k="building" v="yes"/>
73   <tag k="cuisine" v="burger"/>
74   <tag k="name" v="McDonald's"/>
75  </way>
76 <way id="32388779" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:15:19Z"
77   user="adhitya" uid="7748">
78   <nd ref="364364184"/>
79   <nd ref="364364194"/>
80   <nd ref="364364195"/>
81   <tag k="highway" v="service"/>
82   <tag k="oneway" v="yes"/>
83   <tag k="service" v="parking_aisle"/>
84  </way>
85 <way id="32388780" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:15:20Z"
86   user="adhitya" uid="7748">
87   <nd ref="364364182"/>
88   <nd ref="364364202"/>
89   <nd ref="364364220"/>
90   <tag k="highway" v="service"/>
91   <tag k="oneway" v="yes"/>
92   <tag k="service" v="parking_aisle"/>
93  </way>
94 <way id="32388782" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:15:21Z"
95   user="adhitya" uid="7748">
96   <nd ref="364364202"/>
97   <nd ref="364364242"/>
98   <tag k="highway" v="service"/>
99   <tag k="oneway" v="yes"/>
100  <tag k="service" v="parking_aisle"/>
101 </way>
102 <way id="32388783" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:15:26Z"
103   user="adhitya" uid="7748">
```

```

1  <nd ref="364365012" />
2  <nd ref="364365069" />
3  <nd ref="364365018" />
4  <nd ref="364365077" />
5  <nd ref="364365091" />
6  <nd ref="364365025" />
7  <nd ref="364365061" />
8  <nd ref="364365096" />
9  <nd ref="364365090" />
10 <nd ref="364365069" />
11 <tag k="highway" v="service" />
12 <tag k="service" v="parking_aisle" />
13 </way>
14 <way id="32388784" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:15:27Z"
15   user="adhitya" uid="7748">
16   <nd ref="364365077" />
17   <nd ref="364365090" />
18   <tag k="highway" v="service" />
19   <tag k="service" v="parking_aisle" />
20 </way>
21 <way id="32388785" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:15:29Z"
22   user="adhitya" uid="7748">
23   <nd ref="364365091" />
24   <nd ref="364365096" />
25   <tag k="highway" v="service" />
26   <tag k="service" v="parking_aisle" />
27 </way>
28 <way id="32388787" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:15:30Z"
29   user="adhitya" uid="7748">
30   <nd ref="364365134" />
31   <nd ref="364365135" />
32   <nd ref="364365136" />
33   <nd ref="364365137" />
34   <nd ref="364365134" />
35   <tag k="leisure" v="stadium" />
36   <tag k="name" v="GOR_Padjajaran" />
37   <tag k="sport" v="basketball" />
38 </way>
39 <way id="32388788" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:15:38Z"
40   user="adhitya" uid="7748">
41   <nd ref="364365142" />
42   <nd ref="364365164" />
43   <nd ref="364365155" />
44   <nd ref="364365165" />
45   <nd ref="364365143" />
46   <nd ref="364365144" />
47   <nd ref="364365171" />
48   <nd ref="364365162" />
49   <nd ref="364365170" />
50   <nd ref="364365145" />
51   <nd ref="364365169" />
52   <nd ref="364365161" />
53   <nd ref="364365168" />
54   <nd ref="364365146" />
55   <nd ref="364365151" />
56   <nd ref="364365167" />
57   <nd ref="364365159" />
58   <nd ref="364365166" />
59   <nd ref="364365142" />
60   <tag k="leisure" v="track" />
61   <tag k="name" v="Padjajaran" />
62   <tag k="sport" v="athletics" />
63 </way>
64 <way id="32388786" visible="true" version="2" changeset="11220464" timestamp="2012-04-08T02:49:11
65   Z" user="andryono" uid="643030">
66   <nd ref="364365097" />
67   <nd ref="364365098" />
68   <nd ref="364365099" />
69   <nd ref="364365133" />
70   <nd ref="364365097" />
71   <tag k="leisure" v="sports_centre" />
72   <tag k="name" v="Tri_Lomba_Juang" />
73   <tag k="sport" v="multi" />
74 </way>
75 <way id="32388777" visible="true" version="2" changeset="11905099" timestamp="2012-06-15T14:05:59
76   Z" user="andryono" uid="643030">
77   <nd ref="364364179" />
78   <nd ref="364364182" />
79   <nd ref="364364195" />
80   <nd ref="364364183" />
81   <nd ref="364364184" />
82   <nd ref="364364192" />
83   <tag k="highway" v="service" />
84   <tag k="oneway" v="yes" />
85   <tag k="service" v="parking_aisle" />
86 </way>
87 <way id="4625197" visible="true" version="2" changeset="7821743" timestamp="2011-04-10T11:14:24Z"
88   user="evo2mind" uid="234610" >
89   <nd ref="29376841" />
90   <nd ref="29376842" />
91   <nd ref="29376843" />
92   <tag k="avgspeed" v="15" />
93   <tag k="highway" v="residential" />
94   <tag k="name" v="Purabaya" />
95 </way>
96 <way id="4567527" visible="true" version="4" changeset="18997770" timestamp="2013-11-19T17:08:37Z
97   " user="ubanovic" uid="1784103" >
98   <nd ref="2325451571" />
99   <nd ref="29392377" />
100  <nd ref="25433680" />
101  <tag k="avgspeed" v="15" />
102  <tag k="highway" v="tertiary" />
103  <tag k="name" v="Dr._Cipto" />
```

```

1 <tag k="oneway" v="-1" />
2 </way>
3 <way id="4625184" visible="true" version="4" changeset="11905099" timestamp="2012-06-15T14:05:58Z
4   " user="andryono" uid="643030">
5   <nd ref="25433687" />
6   <nd ref="29376829" />
7   <nd ref="29376827" />
8   <tag k="avgsped" v="15" />
9   <tag k="highway" v="living_street" />
10  <tag k="name" v="Citrayuda" />
11  <tag k="oneway" v="-1" />
12 </way>
13 <way id="4625188" visible="true" version="2" changeset="7821743" timestamp="2011-04-10T11:12:21Z"
14   " user="evo2mind" uid="234610">
15   <nd ref="25433689" />
16   <nd ref="29376831" />
17   <tag k="avgsped" v="15" />
18   <tag k="highway" v="residential" />
19 </way>
20 <way id="154066530" visible="true" version="2" changeset="11884346" timestamp="2012-06-13T12
21 :22:04Z" user="andryono" uid="643030">
22   <nd ref="364364251" />
23   <nd ref="1666616044" />
24   <nd ref="1666616129" />
25   <nd ref="1666615973" />
26   <nd ref="1666616003" />
27   <nd ref="1666616103" />
28   <nd ref="1666616021" />
29   <nd ref="1666616062" />
30   <tag k="highway" v="living_street" />
31 </way>
32 <way id="4627113" visible="true" version="4" changeset="24892866" timestamp="2014-08-20T18:40:32Z
33   " user="albahrimaraxsa" uid="2162153">
34   <nd ref="29356500" />
35   <nd ref="29356501" />
36   <nd ref="29356502" />
37   <nd ref="29356503" />
38   <nd ref="29356504" />
39   <nd ref="3030289969" />
40   <nd ref="3030289970" />
41   <nd ref="3030289971" />
42   <tag k="avgsped" v="15" />
43   <tag k="highway" v="living_street" />
44   <tag k="name" v="Muhammad_Aleh" />
45 </way>
46 <way id="4567530" visible="true" version="3" changeset="11220464" timestamp="2012-04-08T02:49:10Z
47   " user="andryono" uid="643030">
48   <nd ref="25433678" />
49   <nd ref="25433679" />
50   <tag k="avgsped" v="15" />
51   <tag k="highway" v="residential" />
52 </way>
53 <way id="4627112" visible="true" version="3" changeset="11220464" timestamp="2012-04-08T02:49:12Z
54   " user="andryono" uid="643030">
55   <nd ref="29356180" />
56   <nd ref="29356534" />
57   <nd ref="29356535" />
58   <nd ref="29356503" />
59   <tag k="avgsped" v="15" />
60   <tag k="highway" v="living_street" />
61   <tag k="name" v="Purwawinata" />
62 </way>
63 <way id="4627146" visible="true" version="3" changeset="11220464" timestamp="2012-04-08T02:49:12Z
64   " user="andryono" uid="643030">
65   <nd ref="29356540" />
66   <nd ref="29356534" />
67   <tag k="avgsped" v="15" />
68   <tag k="highway" v="living_street" />
69 </way>
70 <way id="4627117" visible="true" version="3" changeset="11220464" timestamp="2012-04-08T02:49:13Z
71   " user="andryono" uid="643030">
72   <nd ref="29356374" />
73   <nd ref="29356535" />
74   <tag k="avgsped" v="15" />
75   <tag k="highway" v="living_street" />
76 </way>
77 <way id="4625190" visible="true" version="3" changeset="11220620" timestamp="2012-04-08T04:00:41Z
78   " user="andryono" uid="643030">
79   <nd ref="29376836" />
80   <nd ref="29376837" />
81   <nd ref="29376835" />
82   <tag k="avgsped" v="15" />
83   <tag k="highway" v="living_street" />
84   <tag k="name" v="Suka_Bakti" />
85 </way>
86 <way id="4625194" visible="true" version="3" changeset="11220620" timestamp="2012-04-08T04:00:42Z
87   " user="andryono" uid="643030">
88   <nd ref="29376839" />
89   <nd ref="29376832" />
90   <tag k="avgsped" v="15" />
91   <tag k="highway" v="living_street" />
92   <tag k="name" v="Pamoyanan" />
93 </way>
94 <way id="4567503" visible="true" version="6" changeset="27916265" timestamp="2015-01-04T18:11:06Z
95   " user="isonpurba" uid="2552445">
96   <nd ref="25433685" />
97   <nd ref="25433679" />
98   <nd ref="25433682" />
99   <nd ref="32041794" />
100  <nd ref="29392373" />
101  <nd ref="2325451571" />
102  <tag k="avgsped" v="30" />
103  <tag k="bicycle" v="yes" />
```

```

1   <tag k="foot" v="yes"/>
2   <tag k="highway" v="secondary"/>
3   <tag k="name" v="Dr. Rum"/>
4   </way>
5   <way id="4567635" visible="true" version="8" changeset="13860930" timestamp="2012-11-13T15:42:43Z
6     " user="yudiwbs" uid="268765">
7     <nd ref="25418868"/>
8     <nd ref="28802396"/>
9     <nd ref="2012498680"/>
10    <tag k="avgspeed" v="30"/>
11    <tag k="bicycle" v="yes"/>
12    <tag k="foot" v="yes"/>
13    <tag k="highway" v="primary"/>
14    <tag k="name" v="Pasir_Kaliki"/>
15    <tag k="oneway" v="-1"/>
16   </way>
17   <way id="167186330" visible="true" version="1" changeset="11884346" timestamp="2012-06-13T12
18 :22:01Z" user="andryono" uid="643030">
19     <nd ref="25433680"/>
20     <nd ref="25433684"/>
21     <tag k="avgspeed" v="15"/>
22     <tag k="highway" v="tertiary_link"/>
23     <tag k="oneway" v="yes"/>
24   </way>
25   <way id="4567529" visible="true" version="3" changeset="11884346" timestamp="2012-06-13T12:22:06Z
26   " user="andryono" uid="643030">
27     <nd ref="25433683"/>
28     <nd ref="25433680"/>
29     <tag k="avgspeed" v="15"/>
30     <tag k="highway" v="tertiary"/>
31     <tag k="name" v="Dr._Cipto"/>
32     <tag k="oneway" v="yes"/>
33   </way>
34   <way id="157803724" visible="true" version="3" changeset="13860930" timestamp="2012-11-13T15
35 :47:32Z" user="yudiwbs" uid="268765">
36     <nd ref="1700554884"/>
37     <nd ref="1700554892"/>
38     <nd ref="1700554901"/>
39     <nd ref="1700554868"/>
40     <nd ref="1700554884"/>
41     <tag k="building" v="retail"/>
42     <tag k="name" v="KFC"/>
43   </way>
44   <way id="167462800" visible="true" version="2" changeset="12389881" timestamp="2012-07-20T20
45 :14:38Z" user="OSMF_Redaction_Account" uid="722137">
46     <nd ref="364364237"/>
47     <nd ref="364364192"/>
48     <tag k="highway" v="service"/>
49     <tag k="oneway" v="no"/>
50     <tag k="service" v="parking_aisle"/>
51   </way>
52   <way id="4256037" visible="true" version="10" changeset="27611233" timestamp="2014-12-21T15:25:04
53 Z" user="gnocin" uid="2526082">
54     <nd ref="25418868"/>
55     <nd ref="2012498693"/>
56     <tag k="avgspeed" v="30"/>
57     <tag k="bicycle" v="yes"/>
58     <tag k="foot" v="yes"/>
59     <tag k="highway" v="primary"/>
60     <tag k="name" v="Pajajaran"/>
61     <tag k="oneway" v="yes"/>
62   </way>
63   <way id="175517312" visible="true" version="1" changeset="12686463" timestamp="2012-08-11T00
64 :36:04Z" user="ferdyodin" uid="796044">
65     <nd ref="1860932974"/>
66     <nd ref="29376841"/>
67     <nd ref="1860932973"/>
68     <nd ref="1860932975"/>
69     <tag k="highway" v="unclassified"/>
70     <tag k="name" v="Semar"/>
71   </way>
72   <way id="190605660" visible="true" version="2" changeset="18998018" timestamp="2013-11-19T17
73 :23:36Z" user="ubanovic" uid="1784103">
74     <nd ref="2012498680"/>
75     <nd ref="2012498697"/>
76     <nd ref="364364179"/>
77     <nd ref="29376826"/>
78     <nd ref="25433685"/>
79     <nd ref="25433686"/>
80     <nd ref="1666615070"/>
81     <nd ref="32041828"/>
82     <nd ref="2325451286"/>
83     <tag k="avgspeed" v="30"/>
84     <tag k="bicycle" v="yes"/>
85     <tag k="foot" v="yes"/>
86     <tag k="highway" v="primary"/>
87     <tag k="name" v="Pasir_Kaliki"/>
88     <tag k="oneway" v="no"/>
89   </way>
90   <way id="190605655" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
91 :42:42Z" user="yudiwbs" uid="268765">
92     <nd ref="2012498697"/>
93     <nd ref="2012498695"/>
94     <nd ref="2012498711"/>
95     <tag k="highway" v="primary"/>
96     <tag k="oneway" v="-1"/>
97   </way>
98   <way id="190605657" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
99 :42:42Z" user="yudiwbs" uid="268765">
100    <nd ref="2012498705"/>
101    <nd ref="2012498682"/>
102    <nd ref="364364247"/>
103    <nd ref="364364251"/>
```

```

1 <nd ref="29376836" />
2 <nd ref="25433690" />
3 <nd ref="29376839" />
4 <nd ref="25433689" />
5 <nd ref="1666615284" />
6 <nd ref="25433688" />
7 <tag k="avgspeed" v="15" />
8 <tag k="highway" v="unclassified" />
9 <tag k="name" v="Astina" />
10 <tag k="oneway" v="yes" />
11 </way>
12 <way id="1906065633" visible="true" version="2" changeset="13981717" timestamp="2012-11-22T06
13 :24:04Z" user="yudiwbs" uid="268765">
14 <nd ref="2012498676" />
15 <nd ref="2012498713" />
16 <nd ref="2012498673" />
17 <nd ref="2012498711" />
18 <tag k="highway" v="primary" />
19 <tag k="oneway" v="yes" />
20 </way>
21 <way id="1906061511" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
22 :47:31Z" user="yudiwbs" uid="268765">
23 <nd ref="2012502574" />
24 <nd ref="2012502569" />
25 <nd ref="2012502576" />
26 <nd ref="2012502563" />
27 <nd ref="2012502555" />
28 <nd ref="2012502559" />
29 <nd ref="2012502574" />
30 <tag k="amenity" v="police" />
31 <tag k="name" v="Polsek_Cicendo" />
32 </way>
33 <way id="1906061522" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
34 :47:31Z" user="yudiwbs" uid="268765">
35 <nd ref="2012502565" />
36 <nd ref="2012502561" />
37 <nd ref="2012502553" />
38 <nd ref="2012502557" />
39 <nd ref="2012502551" />
40 <nd ref="2012502568" />
41 <nd ref="2012502567" />
42 <nd ref="2012502577" />
43 <nd ref="2012502549" />
44 <nd ref="2012502570" />
45 <nd ref="2012502572" />
46 <nd ref="2012502566" />
47 <nd ref="2012502563" />
48 <tag k="amenity" v="parking" />
49 <tag k="name" v="Parkir_Istana_Plaza" />
50 <tag k="parking" v="surface" />
51 </way>
52 <way id="190606798" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15
53 :51:18Z" user="yudiwbs" uid="268765">
54 <nd ref="2012512624" />
55 <nd ref="2012512616" />
56 <nd ref="2012512608" />
57 <nd ref="2012512618" />
58 <nd ref="2012512612" />
59 <nd ref="2012512620" />
60 <nd ref="2012512606" />
61 <nd ref="2012512623" />
62 <nd ref="2012512622" />
63 <nd ref="2012512610" />
64 <nd ref="2012512614" />
65 <nd ref="2012512624" />
66 <tag k="amenity" v="college" />
67 <tag k="name" v="Ariyanti" />
68 </way>
69 <way id="190605650" visible="true" version="3" changeset="13981717" timestamp="2012-11-22T06
70 :24:04Z" user="yudiwbs" uid="268765">
71 <nd ref="2012498673" />
72 <nd ref="32520364" />
73 <nd ref="1860932974" />
74 <nd ref="2012498707" />
75 <nd ref="2012498703" />
76 <tag k="highway" v="primary" />
77 <tag k="oneway" v="yes" />
78 </way>
79 <way id="1906056633" visible="true" version="3" changeset="27592401" timestamp="2014-12-20T16
80 :49:58Z" user="gnocin" uid="2526082">
81 <nd ref="2012498703" />
82 <nd ref="2012498686" />
83 <nd ref="2012498705" />
84 <tag k="highway" v="primary" />
85 <tag k="oneway" v="yes" />
86 </way>
87 <way id="257227478" visible="true" version="4" changeset="27915834" timestamp="2015-01-04T17
88 :55:47Z" user="isonpurba" uid="2552445">
89 <nd ref="29376968" />
90 <nd ref="2628264132" />
91 <nd ref="364363837" />
92 <nd ref="364363838" />
93 <nd ref="29356499" />
94 <nd ref="1860932975" />
95 <nd ref="25434115" />
96 <nd ref="29356528" />
97 <nd ref="29356381" />
98 <nd ref="2012498676" />
99 <tag k="avgspeed" v="30" />
100 <tag k="bicycle" v="yes" />
101 <tag k="foot" v="yes" />
102 <tag k="highway" v="primary" />
103 <tag k="name" v="Jln .HOS . Tjokroaminoto" />
```

```

1   <tag k="oneway" v="yes" />
2   </way>
3   <way id="318100678" visible="true" version="1" changeset="27592401" timestamp="2014-12-20T16
4     :49:54Z" user="gnocin" uid="2526082">
5     <nd ref="2012498705"/>
6     <nd ref="2025969631"/>
7     <nd ref="2012498711"/>
8     <nd ref="25418868"/>
9     <tag k="highway" v="primary" />
10    <tag k="oneway" v="yes" />
11   </way>
12   <way id="318222725" visible="true" version="2" changeset="27726149" timestamp="2014-12-27T09
13     :47:32Z" user="gnocin" uid="2526082">
14     <nd ref="2012498693"/>
15     <nd ref="3030289971"/>
16     <nd ref="25500712"/>
17     <nd ref="25433683"/>
18     <tag k="avgspeed" v="30" />
19     <tag k="bicycle" v="yes" />
20     <tag k="foot" v="yes" />
21     <tag k="highway" v="primary" />
22     <tag k="name" v="Pajajaran" />
23     <tag k="oneway" v="yes" />
24   </way>
25   <way id="318892821" visible="true" version="1" changeset="27700068" timestamp="2014-12-26T00
26     :34:22Z" user="gnocin" uid="2526082">
27     <nd ref="25418868"/>
28     <nd ref="2012498673"/>
29     <tag k="highway" v="primary" />
30     <tag k="oneway" v="yes" />
31   </way>
32   <way id="319068174" visible="true" version="1" changeset="27726149" timestamp="2014-12-27T09
33     :47:29Z" user="gnocin" uid="2526082">
34     <nd ref="25433683"/>
35     <nd ref="25433684"/>
36     <nd ref="364365012"/>
37     <nd ref="25500626"/>
38     <nd ref="1706564168"/>
39     <nd ref="1706564155"/>
40     <nd ref="32529146"/>
41     <nd ref="29392413"/>
42     <nd ref="1837514693"/>
43     <tag k="avgspeed" v="30" />
44     <tag k="bicycle" v="yes" />
45     <tag k="foot" v="yes" />
46     <tag k="highway" v="primary" />
47     <tag k="name" v="Pajajaran" />
48     <tag k="oneway" v="yes" />
49   </way>
50   <way id="318670708" visible="true" version="3" changeset="27915808" timestamp="2015-01-04T17
51     :54:58Z" user="isonpurba" uid="2552445">
52     <nd ref="2012498676"/>
53     <nd ref="2012498709"/>
54     <nd ref="32520365"/>
55     <nd ref="2012498684"/>
56     <nd ref="2012498701"/>
57     <nd ref="2012498699"/>
58     <nd ref="2012498678"/>
59     <nd ref="2012498693"/>
60     <tag k="avgspeed" v="30" />
61     <tag k="bicycle" v="yes" />
62     <tag k="foot" v="yes" />
63     <tag k="highway" v="primary_link" />
64     <tag k="name" v="Jln.HOS.Cjokroaminoto" />
65     <tag k="oneway" v="yes" />
66     <tag k="surface" v="asphalt" />
67   </way>
68   <way id="320417107" visible="true" version="1" changeset="27916246" timestamp="2015-01-04T18
69     :10:18Z" user="isonpurba" uid="2552445">
70     <nd ref="2325451571"/>
71     <nd ref="25447003"/>
72     <tag k="highway" v="secondary" />
73   </way>
74   <relation id="4294473" visible="true" version="18" changeset="28004961" timestamp="2015-01-08T20
75     :02:57Z" user="isonpurba" uid="2552445">
76     <member type="way" ref="4627204" role="" />
77     <member type="way" ref="318756518" role="" />
78     <member type="way" ref="4760087" role="" />
79     <member type="way" ref="168942784" role="" />
80     <member type="way" ref="318231943" role="" />
81     <member type="way" ref="153586311" role="" />
82     <member type="way" ref="318756514" role="" />
83     <member type="way" ref="318756516" role="" />
84     <member type="way" ref="318231944" role="" />
85     <member type="way" ref="4936382" role="" />
86     <member type="way" ref="318231946" role="" />
87     <member type="way" ref="318729558" role="" />
88     <member type="way" ref="153586324" role="" />
89     <member type="way" ref="192917039" role="" />
90     <member type="way" ref="318054780" role="" />
91     <member type="way" ref="4623278" role="" />
92     <member type="way" ref="318729560" role="" />
93     <member type="way" ref="153315566" role="" />
94     <member type="way" ref="153315568" role="" />
95     <member type="node" ref="3246270247" role="" />
96     <member type="way" ref="4623279" role="" />
97     <member type="way" ref="186246670" role="" />
98     <member type="way" ref="174270181" role="" />
99     <member type="way" ref="318222726" role="" />
100    <member type="way" ref="318729562" role="" />
101    <member type="way" ref="318729561" role="" />
102    <member type="way" ref="4255951" role="" />
103    <member type="way" ref="174270180" role="" />
```

```

1 <member type="way" ref="299081906" role="" />
2 <member type="way" ref="174270184" role="" />
3 <member type="way" ref="174270182" role="" />
4 <member type="way" ref="318478990" role="" />
5 <member type="way" ref="4255981" role="" />
6 <member type="way" ref="4255990" role="" />
7 <member type="way" ref="32388792" role="" />
8 <member type="node" ref="3246270248" role="" />
9 <member type="way" ref="318233616" role="" />
10 <member type="way" ref="4256015" role="" />
11 <member type="way" ref="168569217" role="" />
12 <member type="way" ref="4251274" role="" />
13 <member type="way" ref="4567655" role="" />
14 <member type="node" ref="3246190446" role="stop" />
15 <member type="way" ref="257227459" role="" />
16 <member type="way" ref="4567652" role="" />
17 <member type="way" ref="4625235" role="" />
18 <member type="way" ref="4625237" role="" />
19 <member type="way" ref="4569060" role="" />
20 <member type="way" ref="190605663" role="" />
21 <member type="way" ref="190605657" role="" />
22 <member type="way" ref="4567636" role="" />
23 <member type="way" ref="318231955" role="" />
24 <member type="way" ref="318892822" role="" />
25 <member type="way" ref="257232273" role="" />
26 <member type="way" ref="318100679" role="" />
27 <member type="way" ref="318891912" role="" />
28 <member type="way" ref="257232272" role="" />
29 <member type="way" ref="257232274" role="" />
30 <member type="way" ref="318804217" role="" />
31 <member type="way" ref="192031850" role="" />
32 <member type="way" ref="4567524" role="" />
33 <member type="way" ref="247058986" role="" />
34 <member type="way" ref="318478989" role="" />
35 <member type="way" ref="292668836" role="" />
36 <member type="way" ref="292668641" role="" />
37 <member type="way" ref="321030145" role="" />
38 <member type="way" ref="235830734" role="" />
39 <member type="way" ref="257232271" role="" />
40 <member type="way" ref="318231951" role="" />
41 <member type="way" ref="4627108" role="" />
42 <member type="way" ref="4567685" role="" />
43 <member type="way" ref="318193812" role="" />
44 <member type="way" ref="318231952" role="" />
45 <member type="way" ref="4567680" role="" />
46 <member type="node" ref="2352383137" role="stop" />
47 <member type="way" ref="306556225" role="" />
48 <member type="way" ref="306586869" role="" />
49 <member type="way" ref="4625951" role="" />
50 <member type="way" ref="166752446" role="" />
51 <member type="way" ref="166752447" role="" />
52 <member type="way" ref="4625948" role="" />
53 <member type="way" ref="318191421" role="" />
54 <member type="way" ref="318193811" role="" />
55 <member type="way" ref="102775972" role="" />
56 <member type="way" ref="255736436" role="" />
57 <member type="way" ref="4637816" role="" />
58 <member type="way" ref="306586886" role="" />
59 <member type="way" ref="318231958" role="" />
60 <member type="node" ref="3246270249" role="" />
61 <member type="node" ref="3240012631" role="" />
62 <member type="node" ref="3249368131" role="" />
63 <member type="node" ref="3252877763" role="stop" />
64 <member type="way" ref="239929577" role="" />
65 <member type="way" ref="4569058" role="" />
66 <member type="node" ref="3252892639" role="stop" />
67 <member type="node" ref="3252929646" role="stop" />
68 <tag k="name" v="Rute_Angkot_34_Sadang_Serang_Caringin" />
69 <tag k="network" v="Angkot_Kota_Bandung" />
70 <tag k="ref" v="34" />
71 <tag k="route" v="bus" />
72 <tag k="type" v="route" />
73 <tag k="wikipedia" v="id:Daftar_Angkutan_Umum_di_Kota_Bandung" />
74 </relation>
75 <relation id="4308299" visible="true" version="8" changeset="27739817" timestamp="2014-12-27T20
:32:33Z" user="gnocin" uid="2526082">
76   <member type="way" ref="182987706" role="" />
77   <member type="way" ref="182628294" role="" />
78   <member type="way" ref="182628297" role="" />
79   <member type="way" ref="162762593" role="" />
80   <member type="way" ref="255739144" role="" />
81   <member type="node" ref="3248857676" role="" />
82   <member type="way" ref="172913717" role="" />
83   <member type="way" ref="318678696" role="" />
84   <member type="way" ref="167539549" role="" />
85   <member type="way" ref="318054785" role="" />
86   <member type="way" ref="223676585" role="" />
87   <member type="way" ref="153284261" role="" />
88   <member type="way" ref="4698221" role="" />
89   <member type="way" ref="318102560" role="" />
90   <member type="way" ref="318054784" role="" />
91   <member type="way" ref="4567624" role="" />
92   <member type="way" ref="318054782" role="" />
93   <member type="way" ref="319083572" role="" />
94   <member type="way" ref="319083569" role="" />
95   <member type="way" ref="292274154" role="" />
96   <member type="way" ref="153175500" role="" />
97   <member type="way" ref="292668299" role="" />
98   <member type="way" ref="318670704" role="" />
99   <member type="way" ref="153181146" role="" />
100  <member type="way" ref="153180024" role="" />
101  <member type="way" ref="318670705" role="" />
102  <member type="way" ref="318478988" role="" />
```

```

1 <member type="node" ref="3244264888" role="" />
2 <member type="way" ref="152949024" role="" />
3 <member type="way" ref="318538778" role="" />
4 <member type="way" ref="318478986" role="" />
5 <member type="way" ref="4623261" role="" />
6 <member type="way" ref="4623260" role="" />
7 <member type="way" ref="247064779" role="" />
8 <member type="way" ref="318670701" role="" />
9 <member type="way" ref="223676540" role="" />
10 <member type="way" ref="95608890" role="" />
11 <member type="way" ref="4623264" role="" />
12 <member type="way" ref="4256037" role="" />
13 <member type="way" ref="318222725" role="" />
14 <member type="way" ref="319068174" role="" />
15 <member type="way" ref="4251274" role="" />
16 <member type="way" ref="4567655" role="" />
17 <member type="way" ref="4567635" role="" />
18 <member type="way" ref="190605660" role="" />
19 <member type="way" ref="4567652" role="" />
20 <member type="way" ref="257227478" role="" />
21 <member type="way" ref="190605653" role="" />
22 <member type="way" ref="32360540" role="" />
23 <member type="way" ref="32360546" role="" />
24 <member type="way" ref="4567653" role="" />
25 <member type="way" ref="4567659" role="" />
26 <member type="way" ref="318222717" role="" />
27 <member type="way" ref="87148402" role="" />
28 <member type="way" ref="257227462" role="" />
29 <member type="way" ref="257227468" role="" />
30 <member type="way" ref="4627123" role="" />
31 <member type="way" ref="4627122" role="" />
32 <member type="way" ref="32388789" role="" />
33 <member type="way" ref="193626591" role="" />
34 <member type="way" ref="318670707" role="" />
35 <member type="way" ref="4623313" role="" />
36 <member type="node" ref="3246190446" role="" />
37 <member type="node" ref="1013407012" role="" />
38 <member type="way" ref="318670708" role="" />
39 <member type="node" ref="29391345" role="stop" />
40 <tag k="name" v="Rute_Angkot_Stasiun_Hall_-_Lembang"/>
41 <tag k="network" v="Angkot_Lintas_Kota_Bandung"/>
42 <tag k="ref" v="St.Hall-Lembang"/>
43 <tag k="route" v="bus"/>
44 <tag k="type" v="route"/>
45 <tag k="wikipedia" v="id:Daftar_Angkutan_Umum_di_Kota_Bandung"/>
46 </relation>
47 <relation id="4420962" visible="true" version="5" changeset="27744522" timestamp="2014-12-28T01
:11:33Z" user="gnocin" uid="2526082">
48 <member type="way" ref="153106229" role="" />
49 <member type="way" ref="4623276" role="" />
50 <member type="way" ref="153106230" role="" />
51 <member type="way" ref="318670697" role="" />
52 <member type="way" ref="292276222" role="" />
53 <member type="way" ref="318531181" role="" />
54 <member type="way" ref="318670699" role="" />
55 <member type="way" ref="318478988" role="" />
56 <member type="way" ref="223676540" role="" />
57 <member type="way" ref="292274152" role="" />
58 <member type="way" ref="318670698" role="" />
59 <member type="way" ref="318531182" role="" />
60 <member type="way" ref="292276221" role="" />
61 <member type="way" ref="318531184" role="" />
62 <member type="way" ref="318531179" role="" />
63 <member type="way" ref="318670695" role="" />
64 <member type="way" ref="292276706" role="" />
65 <member type="way" ref="4623249" role="" />
66 <member type="way" ref="318531180" role="" />
67 <member type="way" ref="4623250" role="" />
68 <member type="way" ref="4256037" role="" />
69 <member type="way" ref="318222725" role="" />
70 <member type="way" ref="319068174" role="" />
71 <member type="way" ref="4251274" role="" />
72 <member type="way" ref="4567655" role="" />
73 <member type="way" ref="4567635" role="" />
74 <member type="way" ref="190605660" role="" />
75 <member type="node" ref="3246190446" role="" />
76 <member type="way" ref="32360540" role="" />
77 <member type="way" ref="32360546" role="" />
78 <member type="way" ref="4567653" role="" />
79 <member type="way" ref="4567659" role="" />
80 <member type="way" ref="318222717" role="" />
81 <member type="way" ref="87148402" role="" />
82 <member type="node" ref="1013407012" role="" />
83 <member type="way" ref="257227462" role="" />
84 <member type="way" ref="318104555" role="" />
85 <member type="way" ref="319194428" role="" />
86 <member type="way" ref="4627124" role="" />
87 <member type="way" ref="4627123" role="" />
88 <member type="way" ref="318670707" role="" />
89 <member type="way" ref="4623313" role="" />
90 <member type="way" ref="193626591" role="" />
91 <member type="way" ref="32388789" role="" />
92 <member type="way" ref="4567652" role="" />
93 <member type="way" ref="257227478" role="" />
94 <member type="way" ref="190605653" role="" />
95 <member type="way" ref="190605655" role="" />
96 <member type="way" ref="55780163" role="" />
97 <member type="way" ref="55783893" role="" />
98 <member type="way" ref="143954415" role="" />
99 <member type="way" ref="95507069" role="" />
100 <member type="node" ref="3249368133" role="" />
101 <member type="node" ref="3250415165" role="" />
102 <tag k="name" v="Rute_Angkot_13._Stasiun_Hall_-_Sarijadi"/>
```

```

1 <tag k="network" v="Angkot_Kota_Bandung" />
2 <tag k="ref" v="13" />
3 <tag k="route" v="bus" />
4 <tag k="type" v="route" />
5 <tag k="wikipedia" v="id:Daftar_Angkutan_Umum_di_Kota_Bandung" />
6 </relation>
7 <relation id="4420961" visible="true" version="7" changeset="28001835" timestamp="2015-01-08T17
8 :46:26Z" user="isonpurba" uid="2552445">
9 <member type="node" ref="3250402136" role="/" />
10 <member type="way" ref="318670693" role="/" />
11 <member type="way" ref="125813777" role="/" />
12 <member type="way" ref="125813784" role="/" />
13 <member type="way" ref="125813781" role="/" />
14 <member type="way" ref="125813778" role="/" />
15 <member type="way" ref="4625407" role="/" />
16 <member type="way" ref="318531173" role="/" />
17 <member type="way" ref="292276703" role="/" />
18 <member type="way" ref="292276219" role="/" />
19 <member type="way" ref="318670697" role="/" />
20 <member type="way" ref="292276222" role="/" />
21 <member type="way" ref="318531181" role="/" />
22 <member type="way" ref="318670700" role="/" />
23 <member type="way" ref="318670702" role="/" />
24 <member type="way" ref="318670704" role="/" />
25 <member type="way" ref="179614119" role="/" />
26 <member type="way" ref="153180024" role="/" />
27 <member type="way" ref="292276224" role="/" />
28 <member type="way" ref="153134040" role="/" />
29 <member type="way" ref="165804161" role="/" />
30 <member type="way" ref="318531179" role="/" />
31 <member type="way" ref="318531184" role="/" />
32 <member type="way" ref="292276221" role="/" />
33 <member type="way" ref="318531182" role="/" />
34 <member type="way" ref="318670698" role="/" />
35 <member type="way" ref="186246671" role="/" />
36 <member type="way" ref="192575395" role="/" />
37 <member type="way" ref="192575392" role="/" />
38 <member type="way" ref="4567534" role="/" />
39 <member type="way" ref="179614116" role="/" />
40 <member type="way" ref="318670706" role="/" />
41 <member type="way" ref="4567524" role="/" />
42 <member type="way" ref="247058986" role="/" />
43 <member type="way" ref="318478989" role="/" />
44 <member type="way" ref="292668836" role="/" />
45 <member type="way" ref="4567490" role="/" />
46 <member type="way" ref="168569217" role="/" />
47 <member type="way" ref="292668468" role="/" />
48 <member type="way" ref="247058988" role="/" />
49 <member type="way" ref="247058985" role="/" />
50 <member type="way" ref="4567491" role="/" />
51 <member type="way" ref="168569234" role="/" />
52 <member type="way" ref="4256015" role="/" />
53 <member type="way" ref="4251274" role="/" />
54 <member type="way" ref="4567655" role="/" />
55 <member type="node" ref="3246190446" role="/" />
56 <member type="way" ref="32360540" role="/" />
57 <member type="way" ref="32360546" role="/" />
58 <member type="way" ref="4567653" role="/" />
59 <member type="way" ref="4567659" role="/" />
60 <member type="way" ref="318222717" role="/" />
61 <member type="way" ref="87148402" role="/" />
62 <member type="node" ref="1013407012" role="/" />
63 <member type="way" ref="257227462" role="/" />
64 <member type="way" ref="318104555" role="/" />
65 <member type="way" ref="319194428" role="/" />
66 <member type="way" ref="4627124" role="/" />
67 <member type="way" ref="4627123" role="/" />
68 <member type="way" ref="318670707" role="/" />
69 <member type="way" ref="4623313" role="/" />
70 <member type="way" ref="32388790" role="/" />
71 <member type="way" ref="4623315" role="/" />
72 <member type="way" ref="318896304" role="/" />
73 <member type="way" ref="318479413" role="/" />
74 <member type="way" ref="235830733" role="/" />
75 <member type="way" ref="292668884" role="/" />
76 <member type="way" ref="3182233616" role="/" />
77 <member type="node" ref="3249368133" role="/" />
78 <tag k="name" v="Rute_Angkot_14_Stasiun_Hall_Gunung_Batu" />
79 <tag k="network" v="Angkot_Kota_Bandung" />
80 <tag k="ref" v="14" />
81 <tag k="route" v="bus" />
82 <tag k="type" v="route" />
83 <tag k="wikipedia" v="id:Daftar_Angkutan_Umum_di_Kota_Bandung" />
84 </relation>
85 <relation id="4294474" visible="true" version="12" changeset="28005713" timestamp="2015-01-08T20
86 :32:47Z" user="isonpurba" uid="2552445">
87 <member type="node" ref="3245971834" role="/" />
88 <member type="way" ref="158851159" role="/" />
89 <member type="way" ref="244646404" role="/" />
90 <member type="way" ref="244646402" role="/" />
91 <member type="way" ref="127343945" role="/" />
92 <member type="way" ref="192917221" role="/" />
93 <member type="way" ref="28410794" role="/" />
94 <member type="way" ref="4910295" role="/" />
95 <member type="way" ref="153340105" role="/" />
96 <member type="way" ref="168549935" role="/" />
97 <member type="way" ref="168549936" role="/" />
98 <member type="way" ref="318222714" role="/" />
99 <member type="way" ref="192917040" role="/" />
100 <member type="way" ref="318054780" role="/" />
101 <member type="way" ref="192917222" role="/" />
102 <member type="way" ref="318222716" role="/" />
103 <member type="way" ref="119439452" role="/" />
```

```

1 <member type="node" ref="3240012631" role="" />
2 <member type="way" ref="4698220" role="" />
3 <member type="way" ref="166687310" role="" />
4 <member type="way" ref="4567624" role="" />
5 <member type="way" ref="318054782" role="" />
6 <member type="way" ref="319083572" role="" />
7 <member type="way" ref="319083569" role="" />
8 <member type="way" ref="292274154" role="" />
9 <member type="way" ref="247060527" role="" />
10 <member type="way" ref="321034670" role="" />
11 <member type="way" ref="292668612" role="" />
12 <member type="way" ref="292668641" role="" />
13 <member type="way" ref="321030145" role="" />
14 <member type="way" ref="292668470" role="" />
15 <member type="way" ref="4567491" role="" />
16 <member type="way" ref="168569234" role="" />
17 <member type="way" ref="4256015" role="" />
18 <member type="way" ref="168569217" role="" />
19 <member type="way" ref="318222725" role="" />
20 <member type="way" ref="319068174" role="" />
21 <member type="way" ref="318670708" role="" />
22 <member type="way" ref="4567655" role="" />
23 <member type="way" ref="193626591" role="" />
24 <member type="way" ref="32360540" role="" />
25 <member type="way" ref="32360546" role="" />
26 <member type="way" ref="4567653" role="" />
27 <member type="way" ref="4567659" role="" />
28 <member type="way" ref="4627139" role="backward" />
29 <member type="way" ref="318222718" role="" />
30 <member type="way" ref="318222719" role="" />
31 <member type="way" ref="215583979" role="" />
32 <member type="way" ref="318191410" role="" />
33 <member type="way" ref="318191412" role="" />
34 <member type="way" ref="318222722" role="" />
35 <member type="node" ref="3246190446" role="" />
36 <member type="node" ref="3246190447" role="" />
37 <member type="way" ref="306163524" role="" />
38 <member type="way" ref="318222723" role="" />
39 <member type="way" ref="318191414" role="" />
40 <member type="way" ref="318191415" role="" />
41 <member type="way" ref="318191416" role="" />
42 <member type="way" ref="4567660" role="" />
43 <member type="way" ref="4567705" role="" />
44 <member type="way" ref="318222717" role="" />
45 <member type="way" ref="318104555" role="" />
46 <member type="way" ref="319194428" role="" />
47 <member type="way" ref="4627124" role="" />
48 <member type="way" ref="4627123" role="" />
49 <member type="way" ref="318670707" role="" />
50 <member type="way" ref="257227459" role="" />
51 <member type="way" ref="257227462" role="" />
52 <member type="way" ref="4623313" role="" />
53 <member type="node" ref="364363784" role="" />
54 <member type="way" ref="32388789" role="" />
55 <member type="way" ref="4567652" role="" />
56 <member type="way" ref="257227478" role="" />
57 <member type="way" ref="4251274" role="" />
58 <member type="way" ref="4567490" role="" />
59 <member type="way" ref="235830734" role="" />
60 <member type="way" ref="318222726" role="backward" />
61 <member type="way" ref="318729562" role="" />
62 <member type="way" ref="318729561" role="" />
63 <member type="way" ref="174270181" role="" />
64 <member type="way" ref="186246670" role="" />
65 <member type="way" ref="4623279" role="" />
66 <member type="way" ref="153315563" role="" />
67 <member type="way" ref="153315566" role="" />
68 <member type="way" ref="318729560" role="" />
69 <member type="way" ref="4623278" role="" />
70 <member type="node" ref="1013407012" role="" />
71 <tag k="name" v="Rute_Angkot_26._Cisitu_~_Tegalega" />
72 <tag k="network" v="Angkot_Kota_Bandung" />
73 <tag k="ref" v="26" />
74 <tag k="route" v="bus" />
75 <tag k="type" v="route" />
76 <tag k="wikipedia" v="id:Daftar_Angkutan_Umum_di_Kota_Bandung" />
77 </relation>
78 <relation id="4423238" visible="true" version="12" changeset="28004961" timestamp="2015-01-08T20
:02:58Z" user="isonpurba" uid="2552445">
79   <member type="node" ref="3240012632" role="stop" />
80   <member type="way" ref="168942781" role="" />
81   <member type="way" ref="318756527" role="" />
82   <member type="way" ref="28446157" role="" />
83   <member type="way" ref="242633052" role="" />
84   <member type="way" ref="242633051" role="" />
85   <member type="way" ref="28446283" role="" />
86   <member type="way" ref="318756523" role="" />
87   <member type="way" ref="319191948" role="" />
88   <member type="way" ref="162812231" role="" />
89   <member type="way" ref="318760037" role="" />
90   <member type="node" ref="3248882521" role="stop" />
91   <member type="way" ref="153315564" role="" />
92   <member type="way" ref="153517385" role="" />
93   <member type="way" ref="318054779" role="" />
94   <member type="way" ref="319191949" role="" />
95   <member type="way" ref="292274159" role="" />
96   <member type="way" ref="4623279" role="" />
97   <member type="way" ref="241791694" role="" />
98   <member type="way" ref="153483321" role="" />
99   <member type="way" ref="186246670" role="" />
100  <member type="way" ref="174270180" role="" />
101  <member type="way" ref="299081906" role="" />
102
103

```

```

1 <member type="way" ref="174270184" role="" />
2 <member type="way" ref="174270182" role="" />
3 <member type="way" ref="318729562" role="" />
4 <member type="way" ref="318231948" role="" />
5 <member type="way" ref="174270183" role="" />
6 <member type="way" ref="4255986" role="" />
7 <member type="way" ref="4255981" role="" />
8 <member type="way" ref="318222726" role="" />
9 <member type="way" ref="4255951" role="" />
10 <member type="way" ref="318478990" role="" />
11 <member type="node" ref="3246270248" role="" />
12 <member type="way" ref="4255990" role="" />
13 <member type="way" ref="32388792" role="" />
14 <member type="way" ref="3182233616" role="" />
15 <member type="way" ref="4256015" role="" />
16 <member type="way" ref="168569217" role="" />
17 <member type="way" ref="4567490" role="" />
18 <member type="way" ref="4567655" role="" />
19 <member type="way" ref="4251274" role="" />
20 <member type="way" ref="319068174" role="" />
21 <member type="way" ref="292668470" role="" />
22 <member type="way" ref="292668641" role="" />
23 <member type="way" ref="321030145" role="" />
24 <member type="way" ref="292668837" role="" />
25 <member type="way" ref="318222725" role="" />
26 <member type="way" ref="318670708" role="" />
27 <member type="way" ref="4567652" role="" />
28 <member type="way" ref="257227478" role="" />
29 <member type="way" ref="190605653" role="" />
30 <member type="way" ref="190605650" role="" />
31 <member type="way" ref="192031850" role="" />
32 <member type="way" ref="318804216" role="" />
33 <member type="way" ref="4569062" role="" />
34 <member type="way" ref="4569064" role="" />
35 <member type="way" ref="318891912" role="" />
36 <member type="way" ref="318891913" role="" />
37 <member type="way" ref="4625292" role="" />
38 <member type="way" ref="318891914" role="" />
39 <member type="way" ref="318891916" role="" />
40 <member type="way" ref="318891920" role="" />
41 <member type="way" ref="318891919" role="" />
42 <member type="way" ref="318891915" role="" />
43 <member type="way" ref="4567662" role="" />
44 <member type="way" ref="257229251" role="" />
45 <member type="way" ref="257229252" role="" />
46 <member type="node" ref="3246190446" role="" />
47 <member type="node" ref="3249368131" role="stop" />
48 <member type="way" ref="257232271" role="" />
49 <member type="way" ref="318891917" role="" />
50 <member type="way" ref="318891918" role="" />
51 <member type="way" ref="180710757" role="" />
52 <member type="way" ref="318891933" role="" />
53 <member type="way" ref="318891935" role="" />
54 <member type="way" ref="318891928" role="" />
55 <member type="way" ref="318891925" role="" />
56 <member type="way" ref="223676554" role="" />
57 <member type="way" ref="4625286" role="" />
58 <member type="way" ref="4625283" role="" />
59 <member type="way" ref="4625281" role="" />
60 <member type="way" ref="4625256" role="" />
61 <member type="node" ref="3244745033" role="stop" />
62 <member type="way" ref="223676441" role="" />
63 <member type="way" ref="223676380" role="" />
64 <member type="way" ref="318891923" role="" />
65 <member type="way" ref="318891932" role="" />
66 <member type="way" ref="318891930" role="" />
67 <member type="way" ref="318891934" role="" />
68 <member type="node" ref="3252877763" role="stop" />
69 <member type="way" ref="172898300" role="" />
70 <member type="node" ref="3248969441" role="stop" />
71 <member type="node" ref="1662376687" role="stop" />
72 <member type="node" ref="3262619802" role="stop" />
73 <tag k="name" v="Rute_Angkot_17._Dago_-Pasar_Induk_Caringin" />
74 <tag k="network" v="Angkot_Kota_Bandung" />
75 <tag k="ref" v="17" />
76 <tag k="route" v="bus" />
77 <tag k="type" v="route" />
78 <tag k="wikipedia" v="id:Daftar_Angkutan_Umum_di_Kota_Bandung" />
79 </relation>
80 <relation id="4425899" visible="true" version="3" changeset="27740065" timestamp="2014-12-27T20
:46:39Z" user="gnocin" uid="2526082">
81 <member type="node" ref="3249461527" role="stop" />
82 <member type="way" ref="4626653" role="" />
83 <member type="way" ref="319083571" role="" />
84 <member type="way" ref="4716961" role="backward" />
85 <member type="way" ref="4716962" role="" />
86 <member type="way" ref="4623260" role="" />
87 <member type="way" ref="247064779" role="" />
88 <member type="way" ref="223676546" role="" />
89 <member type="way" ref="318670699" role="" />
90 <member type="way" ref="318478988" role="" />
91 <member type="way" ref="223676540" role="" />
92 <member type="way" ref="318670701" role="" />
93 <member type="way" ref="4567635" role="" />
94 <member type="way" ref="318892821" role="" />
95 <member type="way" ref="190605660" role="" />
96 <member type="way" ref="190605650" role="" />
97 <member type="way" ref="190605650" role="" />
98 <member type="way" ref="192031850" role="" />
99 <member type="way" ref="318804217" role="" />
100 <member type="way" ref="257232274" role="" />
101 <member type="way" ref="257232272" role="" />
102 <member type="way" ref="318891912" role="" />
```

```

1 | <member type="way" ref="318100679" role="" />
2 | <member type="way" ref="190605663" role="" />
3 | <member type="way" ref="190605657" role="" />
4 | <member type="way" ref="4567636" role="" />
5 | <member type="way" ref="239929577" role="" />
6 | <member type="way" ref="4569058" role="" />
7 | <member type="node" ref="3249368131" role="" />
8 | <member type="way" ref="257232271" role="" />
9 | <member type="way" ref="257232273" role="" />
10 | <member type="way" ref="318231955" role="" />
11 | <member type="way" ref="318892822" role="" />
12 | <member type="way" ref="318231951" role="" />
13 | <member type="way" ref="4627108" role="" />
14 | <member type="way" ref="4567685" role="" />
15 | <member type="way" ref="318193811" role="" />
16 | <member type="way" ref="318191421" role="" />
17 | <member type="way" ref="318193812" role="" />
18 | <member type="way" ref="318231952" role="" />
19 | <member type="way" ref="4567680" role="" />
20 | <member type="node" ref="2352383137" role="stop" />
21 | <member type="node" ref="3252892639" role="stop" />
22 | <member type="node" ref="29392015" role="stop" />
23 | <tag k="name" v="Rute_Angkot_23._Sederhana_-_Cijerah" />
24 | <tag k="network" v="Angkot_Kota_Bandung" />
25 | <tag k="ref" v="23" />
26 | <tag k="route" v="bus" />
27 | <tag k="type" v="route" />
28 | <tag k="wikipedia" v="id : Daftar_Angkutan_Umum_di_Kota_Bandung" />
29 | </relation>
30 | <relation id="4291189" visible="true" version="22" changeset="28769426" timestamp="2015-02-11T10
31 |   :05:22Z" user="shravan91" uid="1051550">
32 |   <member type="way" ref="4623277" role="" />
33 |   <member type="way" ref="318222716" role="" />
34 |   <member type="way" ref="318222715" role="" />
35 |   <member type="way" ref="318222714" role="" />
36 |   <member type="way" ref="318054781" role="" />
37 |   <member type="way" ref="318756515" role="" />
38 |   <member type="way" ref="175071635" role="" />
39 |   <member type="way" ref="153340105" role="" />
40 |   <member type="way" ref="168349935" role="" />
41 |   <member type="way" ref="168549936" role="" />
42 |   <member type="way" ref="4698220" role="" />
43 |   <member type="way" ref="166687310" role="" />
44 |   <member type="way" ref="4567624" role="" />
45 |   <member type="way" ref="318054782" role="" />
46 |   <member type="way" ref="319083572" role="" />
47 |   <member type="way" ref="319083569" role="" />
48 |   <member type="way" ref="292274154" role="" />
49 |   <member type="way" ref="153175500" role="" />
50 |   <member type="way" ref="292668298" role="" />
51 |   <member type="way" ref="292668301" role="" />
52 |   <member type="way" ref="292668297" role="" />
53 |   <member type="way" ref="318054783" role="" />
54 |   <member type="way" ref="318478987" role="" />
55 |   <member type="way" ref="318102560" role="" />
56 |   <member type="way" ref="153305903" role="" />
57 |   <member type="way" ref="192917039" role="" />
58 |   <member type="way" ref="318054780" role="" />
59 |   <member type="way" ref="4910295" role="" />
60 |   <member type="way" ref="192917041" role="" />
61 |   <member type="way" ref="318729557" role="" />
62 |   <member type="way" ref="4623302" role="" />
63 |   <member type="way" ref="175071632" role="" />
64 |   <member type="way" ref="318729556" role="" />
65 |   <member type="way" ref="175071643" role="" />
66 |   <member type="node" ref="3240012631" role="" />
67 |   <member type="way" ref="175071646" role="" />
68 |   <member type="way" ref="318100671" role="" />
69 |   <member type="way" ref="318102033" role="" />
70 |   <member type="way" ref="318051978" role="" />
71 |   <member type="way" ref="153315564" role="" />
72 |   <member type="way" ref="153517385" role="" />
73 |   <member type="way" ref="241791694" role="" />
74 |   <member type="way" ref="153483321" role="" />
75 |   <member type="way" ref="4627223" role="" />
76 |   <member type="way" ref="318481954" role="" />
77 |   <member type="way" ref="32360437" role="" />
78 |   <member type="way" ref="239944112" role="" />
79 |   <member type="way" ref="32360440" role="" />
80 |   <member type="way" ref="320405968" role="" />
81 |   <member type="way" ref="320605375" role="" />
82 |   <member type="node" ref="2352378157" role="" />
83 |   <member type="way" ref="318054779" role="" />
84 |   <member type="way" ref="319191949" role="" />
85 |   <member type="way" ref="318053623" role="" />
86 |   <member type="way" ref="318053626" role="" />
87 |   <member type="way" ref="4632671" role="" />
88 |   <member type="way" ref="318053625" role="" />
89 |   <member type="way" ref="318100669" role="" />
90 |   <member type="way" ref="175071657" role="" />
91 |   <member type="way" ref="292668296" role="" />
92 |   <member type="way" ref="318100677" role="" />
93 |   <member type="way" ref="4625730" role="" />
94 |   <member type="way" ref="4625729" role="" />
95 |   <member type="way" ref="4716962" role="" />
96 |   <member type="way" ref="4623260" role="" />
97 |   <member type="way" ref="247064779" role="" />
98 |   <member type="way" ref="223676546" role="" />
99 |   <member type="way" ref="318670699" role="" />
100 |  <member type="way" ref="318478988" role="" />
101 |  <member type="way" ref="190605660" role="" />
102 |  <member type="way" ref="4567635" role="" />
103 |  <member type="way" ref="318892821" role="" />
```

```

1 <member type="way" ref="4716961" role="" />
2 <member type="way" ref="4716963" role="" />
3 <member type="way" ref="190605650" role="" />
4 <member type="way" ref="190605663" role="" />
5 <member type="way" ref="190605657" role="" />
6 <member type="way" ref="4567636" role="" />
7 <member type="way" ref="192031850" role="" />
8 <member type="way" ref="318804217" role="" />
9 <member type="way" ref="257232274" role="" />
10 <member type="way" ref="257232272" role="" />
11 <member type="way" ref="318891912" role="" />
12 <member type="way" ref="318891913" role="" />
13 <member type="way" ref="318891922" role="" />
14 <member type="way" ref="4625281" role="" />
15 <member type="way" ref="4625254" role="" />
16 <member type="way" ref="4569057" role="" />
17 <member type="way" ref="257232271" role="" />
18 <member type="node" ref="3244745033" role="stop" />
19 <member type="way" ref="257229926" role="stop" />
20 <member type="node" ref="3244754744" role="" />
21 <member type="node" ref="3248969441" role="stop" />
22 <member type="node" ref="3249368131" role="" />
23 <member type="node" ref="3251007172" role="stop" />
24 <tag k="name" v="Rute_Angkot_06._Cicaheum-Ciroyom" />
25 <tag k="network" v="Angkot_Kota_Bandung" />
26 <tag k="ref" v="06" />
27 <tag k="route" v="bus" />
28 <tag k="type" v="route" />
29 <tag k="wikipedia" v="id:Daftar_Angkutan_Umum_di_Kota_Bandung" />
30 </relation>
31 <relation id="4430413" visible="true" version="4" changeset="28005713" timestamp="2015-01-08T20
:32:48Z" user="isonpurba" uid="2552445">
32 <member type="node" ref="3254870563" role="stop" />
33 <member type="way" ref="319068173" role="stop" />
34 <member type="way" ref="222733920" role="" />
35 <member type="way" ref="29064646" role="" />
36 <member type="way" ref="222733921" role="" />
37 <member type="way" ref="222733919" role="" />
38 <member type="way" ref="319083573" role="" />
39 <member type="way" ref="153306178" role="" />
40 <member type="way" ref="153305903" role="" />
41 <member type="way" ref="318102560" role="" />
42 <member type="way" ref="318478987" role="" />
43 <member type="way" ref="318054783" role="" />
44 <member type="way" ref="292668297" role="" />
45 <member type="way" ref="292668301" role="" />
46 <member type="way" ref="292668298" role="" />
47 <member type="way" ref="4567617" role="" />
48 <member type="way" ref="247058986" role="" />
49 <member type="way" ref="166687310" role="" />
50 <member type="way" ref="4567624" role="" />
51 <member type="way" ref="318054782" role="" />
52 <member type="way" ref="319083572" role="" />
53 <member type="way" ref="319083569" role="" />
54 <member type="way" ref="292274154" role="" />
55 <member type="way" ref="247060527" role="" />
56 <member type="way" ref="321034670" role="" />
57 <member type="way" ref="4567623" role="" />
58 <member type="way" ref="247058988" role="" />
59 <member type="way" ref="247058985" role="" />
60 <member type="way" ref="4567527" role="backward" />
61 <member type="way" ref="4567529" role="backward" />
62 <member type="way" ref="319068174" role="" />
63 <member type="way" ref="4251274" role="" />
64 <member type="way" ref="4567655" role="" />
65 <member type="way" ref="4567490" role="" />
66 <member type="way" ref="292668468" role="" />
67 <member type="way" ref="4567502" role="" />
68 <member type="way" ref="4567652" role="" />
69 <member type="way" ref="257227478" role="" />
70 <member type="way" ref="318670708" role="" />
71 <member type="way" ref="318222725" role="" />
72 <member type="way" ref="193626591" role="" />
73 <member type="way" ref="32388789" role="" />
74 <member type="way" ref="4623313" role="" />
75 <member type="node" ref="3246190446" role="stop" />
76 <member type="way" ref="32360540" role="" />
77 <member type="way" ref="32360546" role="" />
78 <member type="way" ref="4567653" role="" />
79 <member type="way" ref="4567659" role="" />
80 <member type="way" ref="318222717" role="" />
81 <member type="way" ref="87148402" role="" />
82 <member type="way" ref="257227468" role="stop" />
83 <member type="way" ref="257227462" role="stop" />
84 <member type="node" ref="1228645006" role="stop" />
85 <member type="way" ref="4627123" role="" />
86 <member type="way" ref="318670707" role="" />
87 <member type="way" ref="4627122" role="" />
88 <member type="node" ref="1013407012" role="stop" />
89 <tag k="name" v="Rute_Angkot_11B._Stasiun_Hall_- Ciumbeuleuit_via_Cihampelas;" />
90 <tag k="network" v="Angkot_Kota_Bandung" />
91 <tag k="ref" v="11B" />
92 <tag k="route" v="bus" />
93 <tag k="type" v="route" />
94 <tag k="wikipedia" v="id:Daftar_Angkutan_Umum_di_Kota_Bandung" />
95 </relation>
96 <relation id="4430737" visible="true" version="3" changeset="28005713" timestamp="2015-01-08T20
:32:48Z" user="isonpurba" uid="2552445">
97 <member type="way" ref="32360540" role="" />
98 <member type="way" ref="32360546" role="" />
99 <member type="way" ref="4567653" role="" />
100 <member type="way" ref="4567659" role="" />
101 <member type="way" ref="4567659" role="" />
102 <member type="way" ref="4567659" role="" />
103 <member type="way" ref="4567659" role="" />
```

```

1 <member type="way" ref="318222717" role="" />
2 <member type="way" ref="87148402" role="" />
3 <member type="way" ref="4627123" role="" />
4 <member type="way" ref="318670707" role="" />
5 <member type="way" ref="4623313" role="" />
6 <member type="way" ref="4627122" role="" />
7 <member type="node" ref="1013407012" role="stop" />
8 <member type="way" ref="257227462" role="stop" />
9 <member type="way" ref="4251274" role="" />
10 <member type="way" ref="319068174" role="" />
11 <member type="way" ref="318222725" role="" />
12 <member type="way" ref="4567655" role="" />
13 <member type="way" ref="193626591" role="" />
14 <member type="way" ref="32388789" role="" />
15 <member type="way" ref="4567652" role="" />
16 <member type="way" ref="257227478" role="" />
17 <member type="way" ref="318670708" role="" />
18 <member type="way" ref="4567490" role="" />
19 <member type="way" ref="292668468" role="" />
20 <member type="way" ref="247058988" role="" />
21 <member type="way" ref="4567502" role="" />
22 <member type="way" ref="247058986" role="" />
23 <member type="way" ref="4567617" role="" />
24 <member type="way" ref="292668298" role="" />
25 <member type="way" ref="292668301" role="" />
26 <member type="way" ref="292668297" role="" />
27 <member type="way" ref="318054783" role="" />
28 <member type="way" ref="318478987" role="" />
29 <member type="way" ref="318102560" role="" />
30 <member type="way" ref="153305903" role="" />
31 <member type="way" ref="319083573" role="" />
32 <member type="way" ref="153306178" role="" />
33 <member type="way" ref="247058985" role="" />
34 <member type="way" ref="4567527" role="backward" />
35 <member type="way" ref="4567529" role="backward" />
36 <member type="way" ref="4625729" role="" />
37 <member type="way" ref="319083571" role="" />
38 <member type="way" ref="4625722" role="" />
39 <member type="way" ref="4627197" role="" />
40 <member type="way" ref="318054782" role="" />
41 <member type="way" ref="4567624" role="" />
42 <member type="way" ref="166687310" role="" />
43 <member type="way" ref="4716963" role="" />
44 <member type="way" ref="318100677" role="" />
45 <member type="way" ref="179613937" role="" />
46 <member type="way" ref="319083569" role="" />
47 <member type="way" ref="292274154" role="" />
48 <member type="way" ref="247060527" role="" />
49 <member type="way" ref="321034670" role="" />
50 <member type="way" ref="4567623" role="" />
51 <member type="node" ref="3249461527" role="stop" />
52 <member type="way" ref="222733919" role="" />
53 <member type="way" ref="222733920" role="" />
54 <member type="way" ref="29064646" role="" />
55 <member type="node" ref="3254870563" role="stop" />
56 <member type="node" ref="29392015" role="stop" />
57 <tag k="name" v="Rute_Angkot_11A_Stasiun_Hall_Ciumbeuleuit_via_Eyckman&quot; ;BELOK&quot;" />
58 <tag k="network" v="Angkot_Kota_Bandung" />
59 <tag k="ref" v="11A" />
60 <tag k="route" v="bus" />
61 <tag k="type" v="route" />
62 <tag k="wikipedia" v="id : Daftar_Angkutan_Umum_di_Kota_Bandung" />
63 </relation>
<relation id="4432958" visible="true" version="4" changeset="28004961" timestamp="2015-01-08T20
:02:59Z" user="isopurba" uid="2552445">
<member type="node" ref="3256049561" role="stop" />
<member type="way" ref="319191947" role="" />
<member type="way" ref="319191948" role="" />
<member type="way" ref="162812231" role="" />
<member type="way" ref="318760037" role="" />
<member type="way" ref="159457434" role="" />
<member type="way" ref="318231944" role="" />
<member type="way" ref="175071635" role="" />
<member type="way" ref="192917041" role="" />
<member type="way" ref="318729557" role="" />
<member type="way" ref="4623302" role="" />
<member type="way" ref="175071632" role="" />
<member type="way" ref="4623282" role="" />
<member type="way" ref="318732962" role="" />
<member type="way" ref="318732963" role="" />
<member type="way" ref="319191949" role="" />
<member type="way" ref="292274159" role="" />
<member type="way" ref="4623279" role="" />
<member type="node" ref="3240012632" role="stop" />
<member type="node" ref="1662376687" role="stop" />
<member type="way" ref="318233618" role="" />
<member type="node" ref="3240012631" role="stop" />
<member type="node" ref="3251007172" role="stop" />
<member type="way" ref="153315568" role="" />
<member type="way" ref="4633273" role="" />
<member type="way" ref="186246670" role="" />
<member type="way" ref="174270181" role="" />
<member type="way" ref="174270180" role="" />
<member type="way" ref="299081906" role="" />
<member type="way" ref="174270184" role="" />
<member type="way" ref="174270182" role="" />
<member type="way" ref="318729561" role="" />
<member type="way" ref="318729562" role="" />
<member type="way" ref="318222726" role="" />
<member type="way" ref="4255951" role="" />
<member type="way" ref="318478990" role="" />
<member type="way" ref="4255981" role="" />
<member type="way" ref="4255990" role="" />
```

```

1 <member type="way" ref="32388792" role="" />
2 <member type="way" ref="318233616" role="" />
3 <member type="way" ref="4256015" role="" />
4 <member type="way" ref="168569217" role="" />
5 <member type="way" ref="4251274" role="" />
6 <member type="way" ref="4567655" role="" />
7 <member type="node" ref="3246270248" role="stop" />
8 <member type="way" ref="318100678" role="" />
9 <member type="way" ref="4256037" role="" />
10 <member type="way" ref="318222725" role="" />
11 <member type="way" ref="319068174" role="" />
12 <member type="way" ref="4567490" role="" />
13 <member type="way" ref="292668641" role="" />
14 <member type="way" ref="321030145" role="" />
15 <member type="way" ref="292668470" role="" />
16 <member type="way" ref="235830734" role="" />
17 <member type="way" ref="4567652" role="" />
18 <member type="way" ref="257227478" role="" />
19 <member type="way" ref="190605653" role="" />
20 <member type="node" ref="3246190446" role="stop" />
21 <member type="way" ref="190605650" role="" />
22 <member type="way" ref="192031850" role="" />
23 <member type="way" ref="318804217" role="" />
24 <member type="way" ref="257232274" role="" />
25 <member type="way" ref="257232272" role="" />
26 <member type="way" ref="318891912" role="" />
27 <member type="way" ref="318891913" role="" />
28 <member type="way" ref="318891922" role="" />
29 <member type="way" ref="4625281" role="" />
30 <member type="way" ref="4625254" role="" />
31 <member type="way" ref="4569057" role="" />
32 <member type="node" ref="3244745033" role="stop" />
33 <member type="way" ref="257229926" role="stop" />
34 <member type="node" ref="3262619802" role="stop" />
35 <tag k="name" v="Rute_Angkot_Ciburial_(Dago Atas)---Ciroiyom" />
36 <tag k="network" v="Angkot_Lintas_Kota_Bandung" />
37 <tag k="ref" v="Ciburial-Ciroiyom" />
38 <tag k="route" v="bus" />
39 <tag k="source" v="http://www.transportasiumum.com/content/rute-angkot-bandung" />
40 <tag k="type" v="route" />
41 </relation>
42 </osm>
```