

## SKRIPSI

### APLIKASI PENCARIAN RUTE TERDEKAT BERBASIS OPENSTREETMAP



ANDREAS

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**

**NPM: 2010730071**

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



## **LEMBAR PENGESAHAN**

### **APLIKASI PENCARIAN RUTE TERDEKAT BERBASIS OPENSTREETMAP**

**ANDREAS**

**NPM: 2010730071**

**Bandung, 27 Mei 2015**

**Menyetujui,**

**Pembimbing Tunggal**

**Dr. rer. nat. Cecilia Esti Nugraheni**

**Ketua Tim Penguji**

**Anggota Tim Penguji**

**Chandra Wijaya, M.T.**

**Joanna Helga, M.Sc.**

**Mengetahui,**

**Ketua Program Studi**

**Thomas Anung Basuki, Ph.D.**



## **PERNYATAAN**

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

### **APLIKASI PENCARIAN RUTE TERDEKAT BERBASIS OPENSTREETMAP**

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,  
Tanggal 27 Mei 2015

Meterai

Andreas  
NPM: 2010730071



## **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.

Berdasarkan pengujian yang telah dilakukan, aplikasi telah berhasil untuk memodelkan OSMXML ke dalam bentuk graf dan mencari rute terdekat antara titik asal dan tujuan menggunakan algoritma Dijkstra. Aplikasi juga telah berhasil membuat visualisasi graf dan rute terdekat menggunakan Google Maps Javascript API.

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



## ABSTRACT

This research aims to develop an application that can find the closest route, using map data provided by OpenStreetMap. OpenStreetMap is an opensource map portal that provides data in the form of a map image and XML document, namely OSMXML. The application can be used by the driver to help looking for a closest route to the destination quicker. The Application uses 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.

Based on testing has been done, the application has succeed to model OSMXML into the form of graph and search the closest route between the source and destination point using Dijkstra algorithm. The application also has succeeded in making graph visualization and closest route using Google Maps Javascript API.

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



*«kepada siapa anda mempersembahkan skripsi ini. . . ?»*



## KATA PENGANTAR

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Bandung, Mei 2015

Penulis



# DAFTAR ISI

<b>KATA PENGANTAR</b>	<b>xv</b>
<b>DAFTAR ISI</b>	<b>xvii</b>
<b>DAFTAR GAMBAR</b>	<b>xix</b>
<b>DAFTAR TABEL</b>	<b>xxi</b>
<b>1 PENDAHULUAN</b>	<b>1</b>
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
<b>2 DASAR TEORI</b>	<b>5</b>
2.1 OpenStreetMap . . . . .	5
2.2 XML . . . . .	6
2.2.1 OSMXML . . . . .	8
2.3 Javascript . . . . .	11
2.3.1 XMLHttpRequest . . . . .	14
2.3.2 XML DOM . . . . .	15
2.3.3 Google Maps Javascript API . . . . .	16
2.4 Graf . . . . .	24
2.4.1 Graf Tidak Berarah . . . . .	24
2.4.2 Graf Berarah . . . . .	25
2.4.3 Graf Berbobot . . . . .	25
2.4.4 Representasi Graf . . . . .	26
2.5 Algoritma Dijkstra . . . . .	27
2.6 Haversine <i>Formula</i> . . . . .	28
<b>3 ANALISIS</b>	<b>31</b>
3.1 Analisis OpenStreetMap . . . . .	31
3.1.1 Langkah-Langkah Pengambilan Data OSMXML . . . . .	31
3.1.2 OSMXML . . . . .	32
3.2 Analisis Javascript . . . . .	34
3.3 Menghitung Jarak Antara Dua Titik . . . . .	37
3.4 Pemodelan OSMXML Menjadi Graf . . . . .	39
3.5 Visualisasi . . . . .	41
3.6 Algoritma Dijkstra . . . . .	44
3.7 Analisis Berorientasi Objek . . . . .	45
3.7.1 Diagram <i>Use Case</i> . . . . .	45

3.7.2	Skenario . . . . .	46
3.7.3	Diagram Kelas Sederhana . . . . .	47
<b>4</b>	<b>PERANCANGAN</b>	<b>51</b>
4.1	Perancangan Antar Muka . . . . .	51
4.2	Perancangan Kelas . . . . .	52
4.3	Diagram Sekuens . . . . .	57
<b>5</b>	<b>IMPLEMENTASI DAN PENGUJIAN</b>	<b>61</b>
5.1	Implementasi . . . . .	61
5.2	Pengujian . . . . .	64
5.2.1	Lingkungan Pengujian . . . . .	64
5.2.2	Pengujian Fungsional . . . . .	65
5.2.3	Pengujian Eksperimental . . . . .	68
<b>6</b>	<b>KESIMPULAN DAN SARAN</b>	<b>79</b>
6.1	Kesimpulan . . . . .	79
6.2	Saran . . . . .	79
<b>DAFTAR REFERENSI</b>		<b>81</b>
<b>A</b>	<b>KODE PROGRAM</b>	<b>83</b>
<b>B</b>	<b>BANDUNG 1 OSMXML</b>	<b>93</b>

## 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 . . . . .	22
2.3	Info Window pada Peta . . . . .	25
2.4	Contoh Graf . . . . .	25
2.5	Contoh Graf Berarah . . . . .	26
2.6	Contoh Graf Berbobot . . . . .	26
2.7	Contoh Adjacency List . . . . .	27
2.8	Contoh Adjacency List . . . . .	27
2.9	Contoh Adjacency Matrix . . . . .	28
2.10	Contoh Adjacency Matrix . . . . .	28
3.1	Fitur search pada situs OpenStreetMap . . . . .	32
3.2	Fitur search pada situs OpenStreetMap . . . . .	32
3.3	Parsing XML menggunakan Javascript . . . . .	37
3.4	Perhitungan Jarak Dengan Haversine Formula . . . . .	38
3.5	Perhitungan Jarak Dengan Geometry Library . . . . .	39
3.6	Pemodelan OSMXML menjadi Graf . . . . .	42
3.7	Visualisasi . . . . .	44
3.8	Info Window . . . . .	45
3.9	Keluaran fungsi Dijkstra . . . . .	45
3.10	Visualisasi Rute Terdekat . . . . .	46
3.11	Diagram Use Case . . . . .	47
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 . . . . .	53
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 . . . . .	65
5.2	Dokumen test.xml . . . . .	66
5.3	Pengujian Fungsi OSMXML . . . . .	66
5.4	Dokumen test.xml . . . . .	67
5.5	Pengujian Jarak . . . . .	67
5.6	Pengujian Fungsi OSMXML Menjadi Graf . . . . .	68
5.7	Pengujian Fungsi Visualisasi . . . . .	69
5.8	Pengujian Titik Asal dan Tujuan . . . . .	70

5.9 Pengujian Rute Terdekat . . . . .	71
5.10 Pengujian Visualisasi Rute Terdekat . . . . .	72
5.11 Pengujian Rute Terdekat . . . . .	72
5.12 Pengujian Bandung 1 . . . . .	73
5.13 Pengujian Bandung 1 . . . . .	73
5.14 Pengujian Bandung 1 . . . . .	74
5.15 bandung1.xml . . . . .	74
5.16 Pengujian Bandung 2 . . . . .	75
5.17 Pengujian Bandung 2 . . . . .	75
5.18 Pengujian Bandung 2 . . . . .	76

## DAFTAR TABEL

3.1 Skenario memilih titik asal . . . . .	46
3.2 Skenario memilih titik tujuan . . . . .	47
3.3 Skenario cari rute terdekat . . . . .	48
5.1 Dokumen OSMXML . . . . .	76



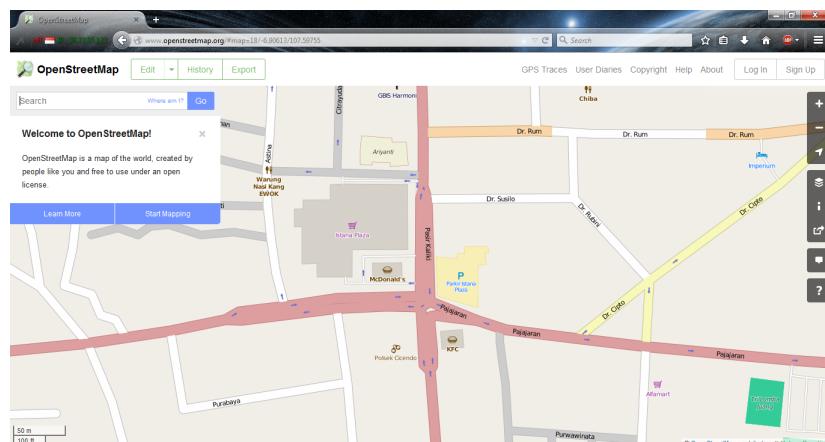
# BAB 1

## 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 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.

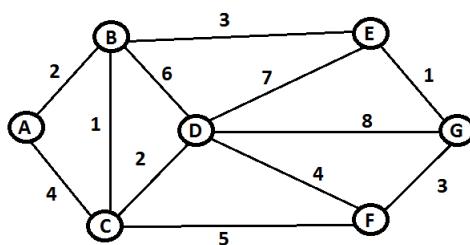
Aplikasi yang dibuat akan berbasis OpenStreetMap dan menggunakan algoritma Dijkstra. OpenStreetMap adalah portal peta terbuka yang menyediakan data dalam bentuk peta maupun XML, pengguna dapat mencari lokasi dan memilih area yang diinginkan. Setelah pengguna memilih area yang diinginkan, pengguna dapat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



Gambar 1.1: Tampilan website OpenStreetMap<sup>1</sup>

<sup>1</sup><http://www.openstreetmap.org>

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



Gambar 1.2: Contoh Graf

Aplikasi yang dibangun akan mengolah data yang disediakan oleh OpenStreetMap dalam bentuk XML dan memodelkannya ke dalam bentuk graf. Selanjutnya akan diimplementasikan algoritma Dijkstra untuk mencari rute terdekat pada graf tersebut dan menunjukkan hasilnya secara visual.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang, maka muncul rumusan masalah berikut:

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

## 1.3 Tujuan

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

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

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

## 1.4 Batasan Masalah

Batasan permasalahan dari pembuatan aplikasi ini adalah :

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

## 1.5 Metodologi Penelitian

Langkah-langkah yang akan dilakukan dalam melakukan penelitian adalah :

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

## 1.6 Sistematika Pembahasan

Pada setiap bab akan dibahas beberapa hal sebagai berikut :

1. Bab Pendahuluan  
Bab 1 berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, dan sistematika pembahasan.
2. Bab Dasar Teori  
Bab 2 berisi teori-teori dasar mengenai OpenStreetMap, algoritma Dijkstra, Google Map Api, Graf, XML, dan beberapa teori lain yang mendukung pembuatan aplikasi.
3. Bab Analisis  
Bab 3 berisi deskripsi sistem yang akan dibuat, analisis dasar teori, dan analisis cara kerja algoritma Dijkstra pada graf.
4. Bab Perancangan  
Bab 4 berisi perancangan antarmuka aplikasi disertai beberapa gambar, perancangan 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

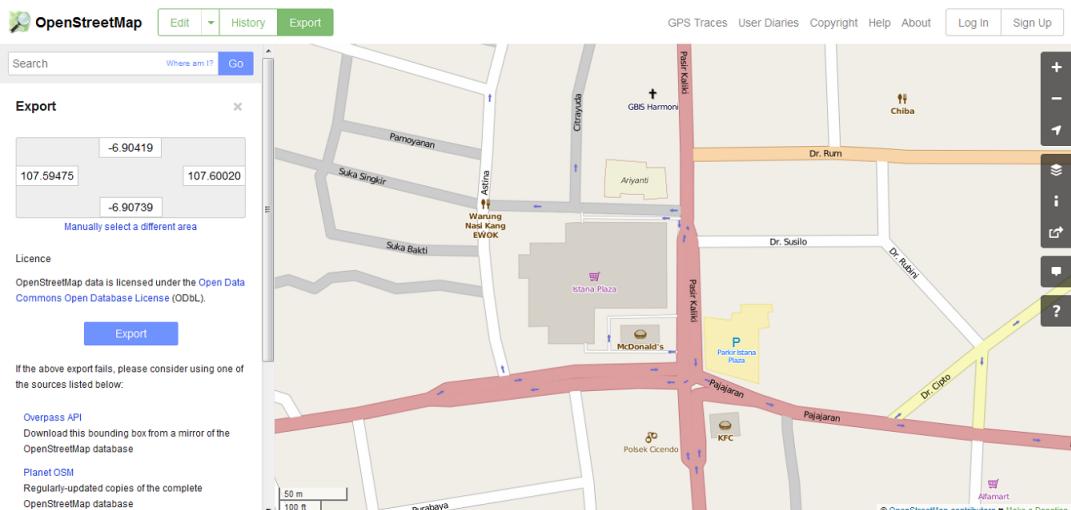
Bab 6 berisi kesimpulan dari seluruh hasil penelitian dan saran untuk pengembangan aplikasi yang akan datang.

## BAB 2

### DASAR TEORI

#### 2.1 OpenStreetMap

OpenStreetMap (OSM) adalah portal peta terbuka yang menyediakan data dalam bentuk peta atau XML [2]. OSM menyediakan peta digital dan dapat diedit dari seluruh dunia, juga memungkinkan pengguna untuk mengakses gambar peta yang terdapat pada situs [www.openstreetmap.org](http://www.openstreetmap.org) secara gratis. OSM terbentuk dan mendapatkan datanya dari berbagai sukarelawan yang bersedia untuk berkontribusi, misalnya para pengguna OSM yang menggunakan aplikasi untuk mengedit peta dan mengunggah data yang telah diedit ke situs OSM. Selain itu, OSM menyediakan beberapa aplikasi bagi para pengguna untuk mengedit peta, seperti iD online editor dan JOSM. Untuk mendapatkan gambar peta ataupun data 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: Eksport 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.

##### 2. Mapnik *Image*

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

### 3. *Embeddable* HTML

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

## 2.2 XML

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

```

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

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

XML dikembangkan terutama untuk mengatasi keterbatasan pada HTML (Hypertext Markup Language). HTML adalah salah satu bahasa markup yang paling populer dan terus dikembangkan, banyak tag baru yang diperkenalkan. Pada versi pertama, HTML memiliki satu lusin tag dan pada

HTML pada versi 4.0 sudah hampir mencapai seratus tag. Namun, pada aplikasi seperti *electronic commerce* dibutuhkan tag lebih untuk produk, harga, nama, alamat, dan banyak lagi atau situs *streaming* memerlukan tag lebih untuk mengontrol gambar dan suara.

HTML telah berkembang menjadi bahasa yang cukup kompleks, W3C memperkirakan penggunaan komputer akan terus berkurang dan penggunaan gadget seperti smartphone akan bertambah. Mesin tersebut tidak sekuat PC dan tidak bisa memproses bahasa yang kompleks seperti HTML . Meskipun HTML adalah bahasa yang populer dan cukup sukses, HTML memiliki beberapa kelemahan utama dan XML dikembangkan untuk mengatasi kelemahan tersebut. XML adalah bahasa yang digunakan untuk menggambarkan dan memanipulasi dokumen terstruktur. Perubahan utama pada XML adalah tidak adanya tag yang ditetapkan pada XML. Karena tidak ada tag yang ditetapkan, penulis dapat membuat tag yang dibutuhkan. Beberapa ketentuan pada XML dapat dilihat pada uraian berikut:

#### 1. Tag pada XML

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

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

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

#### 2. Nama pada XML

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

#### 3. Atribut

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

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

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

#### 4. Elemen Kosong

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

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

Elemen email tidak memiliki nilai atau isi.

#### 5. *Nesting of Elements*

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

```
<name>
  <fname>Jack</fname>
  <lname>Smith</lname>
</name>
```

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

#### 6. Root

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

#### 7. Deklarasi XML

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

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

### 2.2.1 OSMXML

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

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <osm version="0.6" generator="CGIMap_0.0.2">
3   <bounds minlat="54.0889580" minlon="12.2487570" maxlat="54.0913900"
        maxlon="12.2524800"/>
4   <node id="298884269" lat="54.0901746" lon="12.2482632" user="SvenHRO"
         uid="46882" visible="true" version="1" changeset="676636" timestamp
         ="2008-09-21T21:37:45Z"/>
5   <node id="261728686" lat="54.0906309" lon="12.2441924" user=
         PikoWinter" uid="36744" visible="true" version="1" changeset="
         323878" timestamp="2008-05-03T13:39:23Z"/>
6   <node id="1831881213" version="1" changeset="12370172" lat="54.0900666
         " lon="12.2539381" user="lafkor" uid="75625" visible="true"
         timestamp="2012-07-20T09:43:19Z">
7     <tag k="name" v="Neu_Broderstorf"/>
8     <tag k="traffic_sign" v="city_limit"/>
9   </node>
10  ...
11  <node id="298884272" lat="54.0901447" lon="12.2516513" user="SvenHRO"
        uid="46882" visible="true" version="1" changeset="676636" timestamp
        ="2008-09-21T21:37:45Z"/>
12  <way id="26659127" user="Masch" uid="55988" visible="true" version="5"
        changeset="4142606" timestamp="2010-03-16T11:47:08Z">
```

```

13  <nd ref="292403538"/>
14  <nd ref="298884289"/>
15  ...
16  <nd ref="261728686"/>
17  <tag k="highway" v="unclassified"/>
18  <tag k="name" v="Pastower_StraÃ§e"/>
19  <tag k="oneway" v="yes"/>
20  </way>
21  <relation id="56688" user="kmvar" uid="56190" visible="true" version="
22    28" changeset="6947637" timestamp="2011-01-12T14:23:49Z">
23  <member type="node" ref="294942404" role="" />
24  ...
25  <member type="node" ref="364933006" role="" />
26  ...
27  <member type="node" ref="249673494" role="" />
28  <tag k="name" v="KÃijstenbus_Linie_123"/>
29  <tag k="network" v="VVW"/>
30  <tag k="operator" v="Regionalverkehr_KÃijste"/>
31  <tag k="ref" v="123"/>
32  <tag k="route" v="bus"/>
33  <tag k="type" v="route"/>
34  </relation>
35  ...
36</osm>
```

Struktur OSMXML:

- Dokumen OSMXML diawali dengan tag xml yang menjelaskan versi xml dan encoding yang digunakan, pada contoh di atas digunakan xml versi 1.0 dan encoding UTF-8.
- Elemen osm memberikan informasi mengenai versi API dan generator yang digunakan. Generator adalah alat untuk membuat dokumen OSMXML pada saat fitur export digunakan.
- Elemen bound memberikan informasi mengenai cakupan area pada dokumen OSMXML tersebut. Dilengkapi dengan atribut koordinat yaitu latitude dan longitude. Data primitif pada OSM dibagi menjadi 3 bagian, yaitu node, way, dan relation.
  1. Elemen Node merupakan informasi titik pada sebuah peta. Node memiliki beberapa atribut yaitu:
    - id  
Merupakan id dari node tersebut.
    - user  
Merupakan user yang melakukan editing pada node.

- uid  
Id dari user.
- lat  
berisi informasi koordinat pada garis lintang.
- lon  
berisi informasi koordinat pada garis bujur.
- timestamp  
Berisi informasi waktu saat node tersebut diperbarui.

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

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

nama dari node tersebut adalah Neu Broderstorf.

2. Elemen Way merupakan informasi garis yang dapat diartikan sebagai jalan ataupun 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:

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

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

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

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

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

- oneway=yes  
Menunjukkan jalan satu arah
- oneway=no  
Menunjukkan jalan dua arah
- oneway=-1  
Menunjukkan jalan satu arah dan berlawanan
- oneway=reversible  
Menunjukkan jalan satu arah dan dapat berubah arah menjadi berlawanan. Contoh, pengalihan jalan untuk mengatasi kemacetan.

3. Elemen relation menyimpan informasi node-node dan way yang bersinggungan. Elemen relation dapat menggambarkan suatu area seperti lapangan, taman bermain.

```

1<relation id="56688" user="kmvar" uid="56190" visible="true"
2  version="28" changeset="6947637" timestamp="2011-01-12
3    T14:23:49Z">
4  <member type="node" ref="294942404" role="" />
5  ...
6  <member type="node" ref="364933006" role="" />
7  <member type="way" ref="4579143" role="" />
8  ...
9  <member type="node" ref="249673494" role="" />
10 <tag k="name" v="KÃijstenbus_Linie_123" />
11 <tag k="network" v="VVW" />
12 <tag k="operator" v="Regionalverkehr_KÃijste" />
13 <tag k="ref" v="123" />
14 <tag k="route" v="bus" />
15 <tag k="type" v="route" />
16 </relation>

```

Pada contoh di atas menggambarkan rute bus.

## 2.3 Javascript

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

```

1<!DOCTYPE html>
2<html>
3<head>
4<script>
5function myFunction() {
6  document.getElementById("demo").innerHTML = "Paragraph changed .";
7}
8</script>
9</head>
10<body>
11<h1>JavaScript in Head</h1>
12<p id="demo">A Paragraph.</p>
13<button type="button" onclick="myFunction () ">Try it</button>
14</body>
15</html>

```

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

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

### 1. Struktur

- *Character Set*

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

- *Comments*

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

```
// This is a single-line comment.  
/* This is also a comment */ // and here is another comment.  
/*  
 * This is yet another comment.  
 * It has multiple lines.  
 */
```

- *Literal*

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

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

- *Identifier*

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

- *Reserved words*

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

- *Optional Semicolons*

Seperti banyak bahasa pemrograman lain, javascript menggunakan titik koma (;) untuk

memisahkan perintah yang ditulis. Hal ini penting untuk membuat kode program menjadi jelas mengenai awal dan akhir. Pada javascript, titik koma dapat dihilangkan jika perintah ditulis pada baris yang berbeda, berikut adalah contoh penggunaan titik koma pada javascript:

```
a = 3;  
b = 4;
```

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

```
a = 3; b = 4;
```

## 2. Sintaks

- Deklarasi Variabel

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

```
var i;  
var i, sum;  
var message = "hello";  
var i = 0, j = 0, k = 0;
```

- Fungsi

Fungsi adalah blok kode program yang hanya dituliskan sekali, tetapi dapat dipanggil atau dijalankan berulang kali. Pada javascript, fungsi dapat dibuat menggunakan keyword function. Sebuah fungsi harus memiliki nama, sepasang tanda kurung untuk parameter, dan sepasang kurung kurawal. Berikut ini adalah beberapa contoh fungsi:

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

- Kelas

Pada javascript, kelas adalah objek. Cara untuk membuat kelas pada javascript adalah dengan membuat fungsi dan membuat objek dari fungsi tersebut menggunakan *keyword* “new”. Berikut ini adalah contoh pembuatan kelas pada javascript:

```

function Range(from, to) {
    // Store the start and end points (state) of this new range object.
    // These are noninherited properties that are unique to this object.
    this.from = from;
    this.to = to;
}

// Contoh pembuatan objek
var r = new Range(1,3);

```

### 2.3.1 XMLHttpRequest

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

```

var objXMLHTTP = new XMLHttpRequest();

objXMLHTTP.open('GET', 'books.xml', false);
objXMLHTTP.send();

var objXML = objXMLHTTP.responseXML;

```

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

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

Melakukan inisialisasi permintaan

Parameter:

- *method*

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

- *url*

Alamat url tujuan

- *asynchronous*

boolean Opsional, secara default bernilai *true*. *True* menyatakan bahwa operasi yang dijalankan secara *asynchronous*. Nilai *false* menyatakan sebaliknya.

- *username*

Opsional, berisikan *username* yang digunakan untuk keperluan otentikasi. Secara default, berisi string kosong.

- *password*

Opsional, berisikan *password* yang digunakan untuk keperluan otentikasi. Secara default, berisi string kosong.

## 2. send(content)

Mengirimkan permintaan

Parameter:

- *content*

Opsional, *content* dapat berisi string atau data lainnya seperti Array, dokumen, dan lain-lain.

## 3. responseXML

Respon dari permintaan

Return:

DOM Object

### 2.3.2 XML DOM

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:

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

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

## 1. getElementsByTagName('tagName')

Mengembalikan elemen-elemen yang memiliki kesesuaian nama.

Parameter:

- *tagName*

String yang menentukan nama elemen yang dicari.

Return:

objek berisi elemen yang memiliki nama sesuai dengan yang dicari.

## 2. getAttribute('name')

Mengembalikan nilai atribut

Parameter:

- *name*

String yang menentukan nama atribut yang dicari.

Return:

Mengembalikan string jika atribut memiliki nilai, jika tidak mengembalikan *null*.

### 2.3.3 Google Maps Javascript API

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

#### 2.3.3.1 Elemen Dasar Google Maps

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

```

1<!DOCTYPE html>
2<html>
3  <head>
4    <style type="text/css">
5      html, body, #map-canvas { height: 100%; margin: 0; padding: 0; }
6    </style>
7    <script type="text/javascript"
8      src="https://maps.googleapis.com/maps/api/js?key=API_KEY">
9    </script>
10   <script type="text/javascript">
11     function initialize() {
12       var mapOptions = {
13         center: { lat: -34.397, lng: 150.644 },
14         zoom: 8
15       };
16       var map = new google.maps.Map(document.getElementById('map-
17         canvas'),
18         mapOptions);
19       google.maps.event.addDomListener(window, 'load', initialize);
20     </script>
21   </head>
22   <body>
23 <div id="map-canvas"></div>
24 </body>
25</html>
```

- Declaring

Google menyarankan untuk membuat deklarasi tipe dokumen pada awal dokumen yaitu de-

ngan menulis <!DOCTYPE html>. Setelah itu diperlukan CSS yang bekerja untuk mengatur tampilan peta pada halaman web.

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

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

- Loading Google Maps API

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

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

- Initialize

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

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

Untuk inisialisasi peta, diperlukan variabel *map options*

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

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

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

- Map Object

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

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

- Loading the Map

Google Maps API menyediakan fungsi untuk memuat peta. Pada potongan kode di bawah, fungsi *listener* akan memanggil fungsi *initialize* ketika halaman dimuat.

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

Berikut ini adalah penjelasan kelas dan fungsi yang digunakan:

1. google.maps.Map class

Membuat peta baru pada halaman html.

Konstruktor:

- mapDiv:Node  
node yang digunakan untuk membuat peta.
- opts?:MapOptions  
Opsi dari *map* yang akan dibuat.

2. google.maps.LatLng class

Membuat objek LatLng yang merepresentasikan titik geografis.

Konstruktor:

- lat:number  
*Latitude* dalam derajat.
- lng:number  
*Longitude* dalam derajat.
- noWrap?:boolean  
*Latitude* ditentukan dalam rentang derajat -90 hingga 90 dan *longitude* ditentukan dalam rentang derajat -180 hingga 180. Nilai *true* pada boolean *noWrap* untuk mengaktifkan nilai di luar batas tersebut.

3. google.maps.event.addDomListener(instance:Object, eventName:string, handler:Function)

Menambahkan fungsi *listener*

Parameter:

- instance:Object  
Objek yang ditambahkan *listener*.
- eventName:string  
Nama *Event*.
- handler:Function  
Fungsi yang dipanggil ketika *event* terjadi.

Return:

MapsEventListener

### 2.3.3.2 Menggambar pada Peta

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

#### 1. google.maps.Marker class

Membuat *marker* pada peta dengan opsi tertentu.

Konstruktor:

- opts?:MarkerOptions

Opsi dari *marker* yang dibuat.

#### 2. google.maps.Polyline class

Membuat *polyline* pada peta dengan opsi tertentu.

Konstruktor:

- opts?:PolylineOptions

Opsi dari *polyline* yang dibuat.

#### 3. setMap(map:Map)

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

Parameter:

- map:Map

Peta yang disisipkan *marker* atau *polyline*.

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

Mengubah *icon* pada *marker*.

Parameter:

- icon:string|Icon|Symbol

*Icon* yang digunakan.

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

#### 1. Marker

Lokasi tunggal pada peta ditunjukkan oleh *Marker*.

- Menambahkan *Marker*

Untuk menampilkan *marker* pada peta harus membuat objek google.maps.Marker. Berikut ini adalah atribut penting pada saat membuat objek *marker*:

##### (a) *position*

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

##### (b) *map*

atribut *map* bersifat opsional, untuk menentukan marker tersebut akan diletakkan pada peta. Jika atribut *map* tidak diatur, maka *marker* akan tetap dibuat tetapi tidak akan ditampilkan pada peta.

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

```
var myLatlng = new google.maps.LatLng(-25.363882,131.044922);
var mapOptions = {
  zoom: 4,
  center: myLatlng
}
var map = new google.maps.Map(document.getElementById
("map-canvas"), mapOptions);

// To add the marker to the map, use the 'map' property
var marker = new google.maps.Marker({
  position: myLatlng,
  map: map,
  title:"Hello World!"
});
```

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`.

```
var image = 'images/beachflag.png';
var myLatLng = new google.maps.LatLng(-33.890542, 151.274856);
var beachMarker = new google.maps.Marker({
  position: myLatLng,
  map: map,
  icon: image
});
```

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

```
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:

```
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:

```
var marker = new google.maps.Marker({
    position: myLatlng,
    map: map,
    animation: google.maps.Animation.BOUNCE,
    title:"Hello World!"
});
```

Pada contoh, menambahkan animasi *bounce* pada marker sehingga *marker* bergerak melompat-lompat pada peta.

- Mengubah Ikon

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

```
var image = 'images/beachflag.png';
var myLatLng = new google.maps.LatLng(-33.890542, 151.274856);
var beachMarker = new google.maps.Marker({
    position: myLatLng,
    map: map,
    icon: image
});
```

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

- *Draggable*

*Draggable* memungkinkan pengguna untuk menyeret marker ke lokasi yang berbeda, hanya memerlukan pengaturan atribut pada konstruktor google.maps.Marker. Contoh:

```
var marker = new google.maps.Marker({
    position: myLatlng,
    map: map,
    draggable: true,
    title:"Hello World!"
});
```

## 2. *Polyline*

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

- *strokeColor*

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

- *strokeOpacity*

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

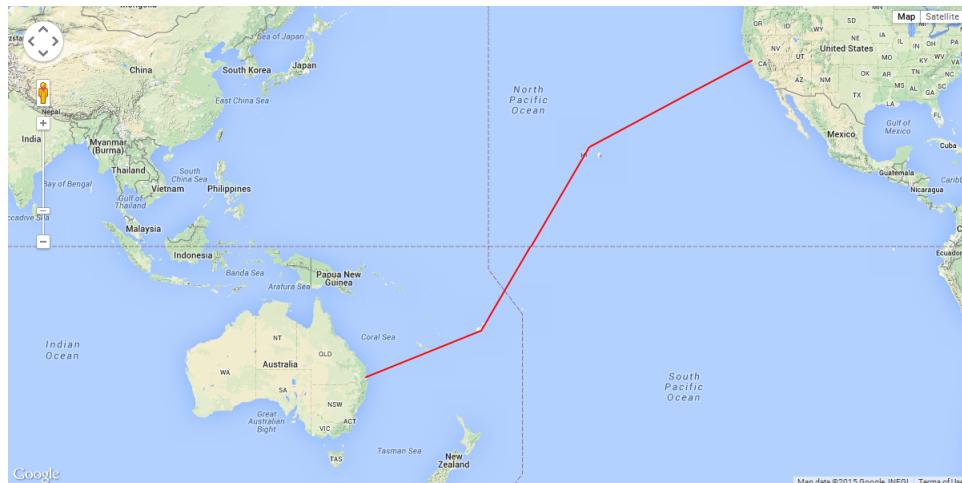
- *strokeWeight*

Atribut *strokeWeight* menentukan lebar garis dalam piksel.

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

```
var flightPlanCoordinates = [
    new google.maps.LatLng(37.772323, -122.214897),
    new google.maps.LatLng(21.291982, -157.821856),
    new google.maps.LatLng(-18.142599, 178.431),
    new google.maps.LatLng(-27.46758, 153.027892)
];
var flightPath = new google.maps.Polyline({
    path: flightPlanCoordinates,
    strokeColor: '#FF0000',
    strokeOpacity: 1.0,
    strokeWeight: 2
});
flightPath.setMap(map);
```

Pada contoh, akan menampilkan polyline pada peta yang akan menghubungkan setiap koordinat yang terdapat pada variabel *flightPlanCoordinates*. *Polyline* yang ditampilkan pada peta dapat dilihat pada Gambar 2.2.



Gambar 2.2: Polyline pada Peta

### 2.3.3.3 Geometry Library

Gambar pada Google Maps adalah dua dimensi, sedangkan bumi adalah tiga dimensi yang menyerupai bentuk bola. Hal ini tentu akan berbeda ketika mengukur suatu jarak dari satu titik ke titik lain, misalnya jarak terpendek antara dua titik pada bola bukanlah garis lurus, tetapi menyerupai lingkaran besar atau busur. Karena perbedaan tersebut, diperlukan *spherical geometry* untuk menghitung data geometris pada permukaan bumi seperti sudut, jarak, dan area yang berdasarkan garis

lintang dan garis bujur. Google Maps JavaScript API menyediakan *geometry library* yang memiliki fungsi utilitas tersebut, fungsi utilitas tersebut dinamakan `google.maps.geometry.spherical`. Untuk menghitung jarak antara dua titik dapat memanggil fungsi `computeDistanceBetween()`.

1. `google.maps.geometry.spherical` namespace

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

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

Menghitung jarak antara dua titik.

Parameter:

- `from:LatLng`  
Koordinat titik pertama.
- `to:LatLng`  
Koordinat titik kedua.
- `radius?:number`  
Radius yang digunakan.

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

```
var jakarta = new google.maps.LatLng(-6.1745,106.8227);
var bandung = new google.maps.LatLng(-6.9167,107.6000);

var distance = google.maps.geometry.spherical
.computeDistanceBetween(jakarta, bandung);
```

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

#### 2.3.3.4 Info Window

*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*:

```
var contentString = 'Info Window';

var infowindow = new google.maps.InfoWindow({
    content: contentString
});

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

```

    position: myLatlng,
    map: map,
    title: 'Uluru (Ayers Rock)'
});
google.maps.event.addListener(marker, 'click', function() {
  infowindow.open(map,marker);
});

```

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

1. google.maps.InfoWindow class

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

Konstruktor:

- opts?:InfoWindowOptions

Opsi dari *info window* yang dibuat.

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

Membuka *info window* pada peta.

Parameter

- map?:Map|StreetViewPanorama

Membuka *info window* pada peta yang diberikan.

- anchor?:MVCObject

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

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

## 2.4 Graf

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

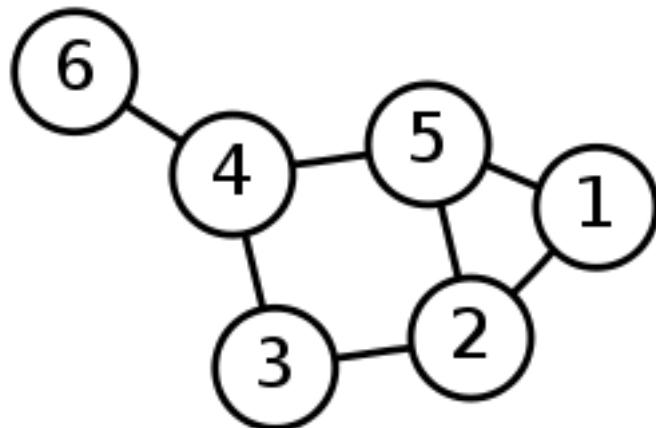
### 2.4.1 Graf Tidak Berarah

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

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



Gambar 2.3: Info Window pada Peta



Gambar 2.4: Contoh Graf

$$\text{Himpunan node} = \{1, 2, 3, 4, 5, 6\}$$

$$\text{Himpunan edge} = \{(6, 4), (4, 5), (4, 3), (3, 2), (5, 2), (2, 1), (5, 1)\}$$

### 2.4.2 Graf Berarah

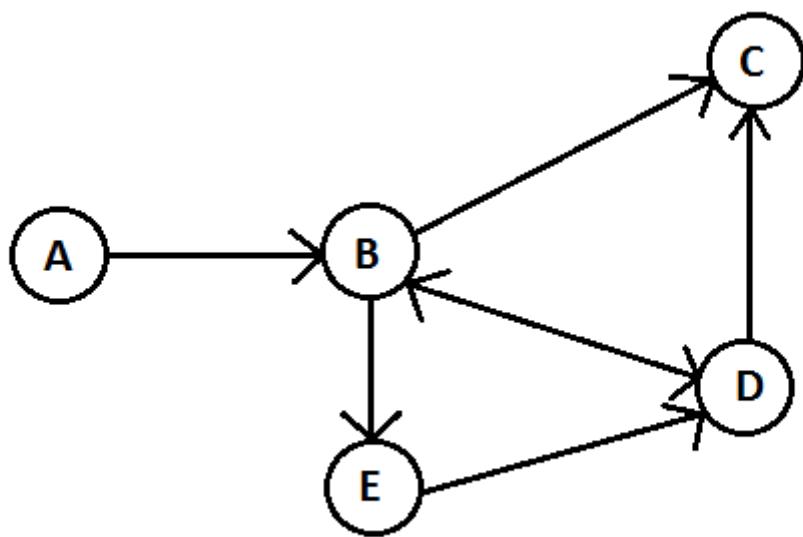
Graf berarah memiliki arah pada setiap edgenya. Pada graf berarah, edge biasanya digambarkan dengan panah sesuai arahnya. Contoh graf berarah dapat dilihat pada Gambar 2.5. Berdasarkan contoh pada Gambar 2.5 didapatkan informasi tipe dari node adalah huruf kapital.

$$\text{Himpunan node} = \{A, B, C, D, E\}$$

$$\text{Himpunan edge} = \{(A, B), (B, C), (D, C), (B, D), (D, B), (E, D), (B, E)\}$$

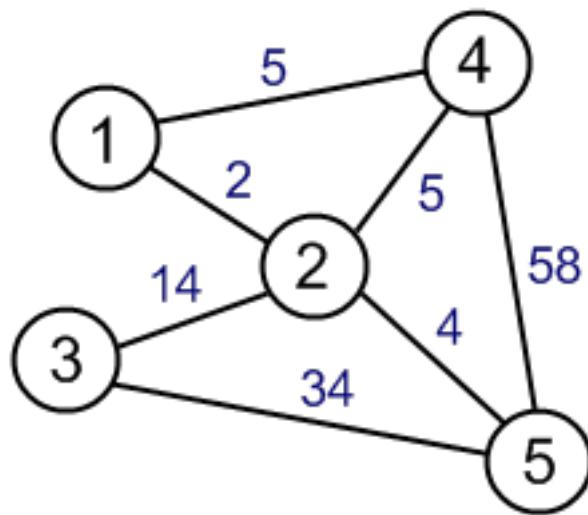
### 2.4.3 Graf Berbobot

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



Gambar 2.5: Contoh Graf Berarah

Berdasarkan contoh pada Gambar 2.6 didapatkan informasi tipe dari node adalah bilangan bulat



Gambar 2.6: Contoh Graf Berbobot

dan tipe dari bobot adalah bilangan bulat.

$$\text{Himpunan node} = \{1, 2, 3, 4, 5\}$$

$$\text{Himpunan edge} = \{(1, 4, 5), (4, 5, 58), (3, 5, 34), (2, 4, 5), (2, 5, 4), (3, 2, 14), (1, 2, 2)\}$$

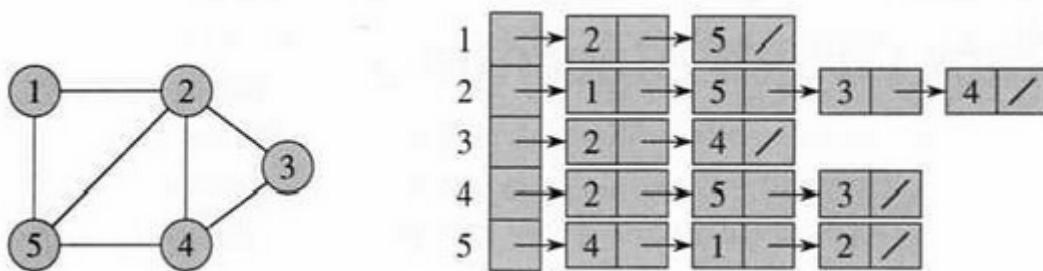
#### 2.4.4 Representasi Graf

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*

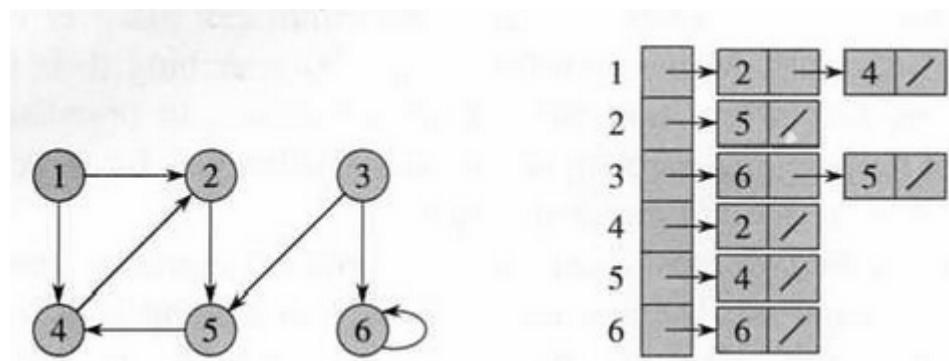
merepresentasikan graf ke dalam bentuk array, sedangkan *adjacency matrix* merepresentasikan graf ke dalam bentuk matriks.

- *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)



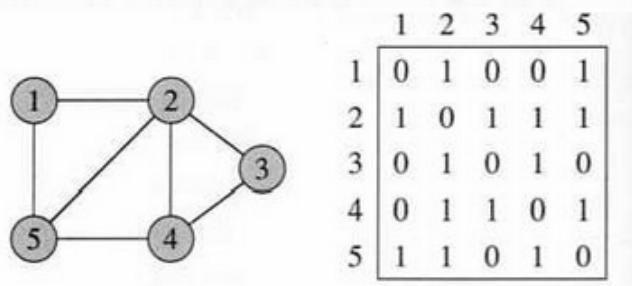
Gambar 2.8: Contoh Adjacency List (Graf Berarah)

- *Adjacency Matrix*

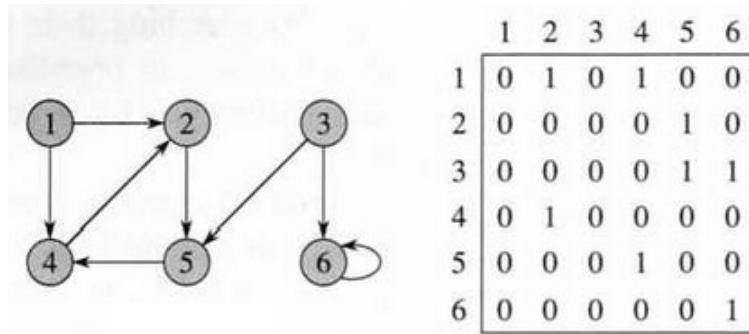
*Adjacency Matrix* merupakan representasi graf ke dalam bentuk matriks  $n \times n$ , pada matriks tersebut menyatakan hubungan antar node atau pada graf. Nilai  $n$  pada matriks  $n \times n$  sesuai dengan jumlah node pada graf. Nilai 1 pada matriks menandakan terdapat hubungan pada node dan sebaliknya 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.

## 2.5 Algoritma Dijkstra

Algoritma dijkstra adalah algoritma yang dapat mencari jalur terpendek pada graf berarah  $G = (V, E)$  untuk kasus pada setiap sisinya bernilai tidak negatif [1].  $V$  adalah himpunan tidak kosong dari



Gambar 2.9: Contoh Adjacency Matrix (Graf Tidak Berarah)



Gambar 2.10: Contoh Adjacency Matrix (Graf Berarah)

node dan E adalah himpunan sisi yang menghubungkan sepasang node. Algoritma ini menggunakan prinsip greedy, prinsip greedy pada algoritma dijkstra adalah memilih sisi yang memiliki bobot paling kecil dan memasukannya dalam himpunan solusi. Berikut ini adalah *pseudocode* dari algoritma dijkstra: Langkah-langkah yang dilakukan adalah:

1. Inisialisasi node asal dengan nilai 0, dan inisialisasi seluruh node lainnya dengan nilai tak terhingga.
2. S adalah himpunan node yang sudah dikunjungi, untuk awal inisialisasi dengan himpunan kosong.
3. Q adalah struktur data *queue*, inisialisasi dengan seluruh node.
4. Keluarkan node yang memiliki jarak paling kecil dari dalam *queue* dan tambahkan node tersebut ke dalam himpunan node yang sudah dikunjungi.
5. Lakukan pengulangan untuk seluruh node tetangga dan periksa jika jarak terpendek lain ditemukan tandai node tersebut.
6. Ulangi langkah 4 dan 5 hingga seluruh node di dalam *queue* habis.

## 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

**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  $dist$ 

```

---

persamaan berikut ini:

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

dimana:

- $d$  : jarak antara dua buah titik (dalam satuan km)
- $r$  : radius bumi (6371 km)
- $\phi_1, \phi_2$  : *latitude* dari titik 1 dan *latitude* dari titik 2
- $\psi_1, \psi_2$  : *longitude* dari titik 1 dan *longitude* dari titik 2



## BAB 3

### ANALISIS

Pada bab ini akan dipaparkan analisis yang dilakukan dalam pembuatan aplikasi ini. Bagaimana data XML didapatkan dari situs [www.openstreetmap.org](http://www.openstreetmap.org), dibahas pada subbab 3.1. Pembacaannya menggunakan beberapa fungsi javascript, dibahas pada subbab 3.2. Setelah itu, data tersebut disimpan atau dikonversi ke dalam bentuk graf sehingga dapat diimplementasikan algoritma Dijkstra untuk mencari rute terpendek dari satu node ke node lain. Berdasarkan informasi yang telah diolah, maka dapat dibuat visualisasi data atau informasi menggunakan Google Maps Javascript API. Pada akhir bab ini, juga dibahas mengenai diagram *use case* dan skenario untuk memperjelas apa saja yang dapat dilakukan oleh *user* pada aplikasi ini.

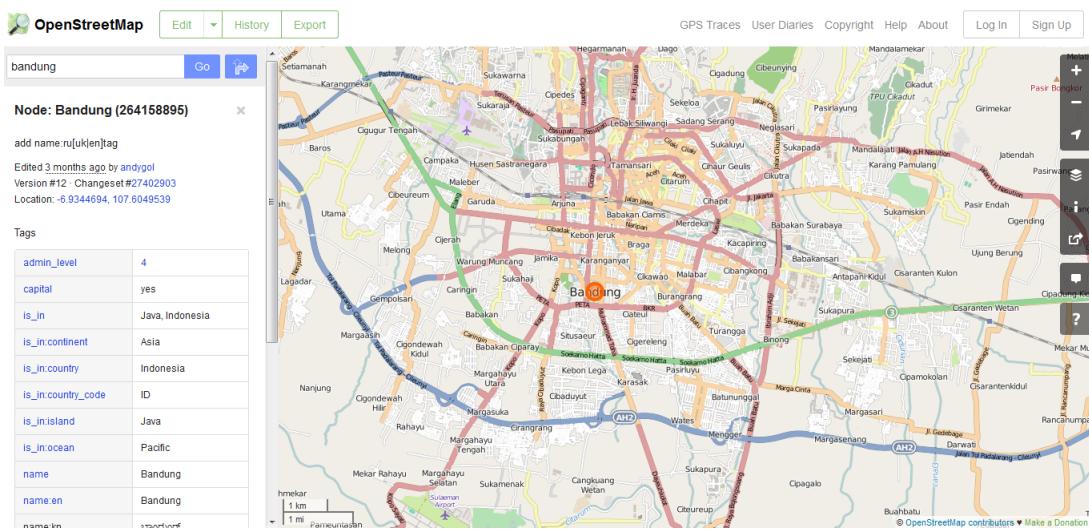
#### 3.1 Analisis OpenStreetMap

OpenStreetMap adalah portal peta terbuka yang menyediakan data dalam bentuk peta ataupun dokumen XML. Aplikasi yang dibuat berbasis OpenStreetMap, hal ini berarti aplikasi yang dibuat menggunakan data yang diperoleh dari situs [www.openstreetmap.org](http://www.openstreetmap.org). Untuk mendapatkan data peta pada situs OpenStreetMap, user harus mengunjungi situs tersebut dan menggunakan fitur *export*. Data yang digunakan adalah data peta yang berbentuk dokumen XML atau biasa disebut dengan OSMXML. Selanjutnya, informasi yang terkandung di dalam dokumen OSMXML tersebut diproses untuk mengetahui node dan edge pada peta. Informasi tersebut akan diubah ke dalam bentuk graf yang akan diproses lebih lanjut menggunakan algoritma Dijkstra untuk mengetahui jarak terpendek dari satu node ke node lain.

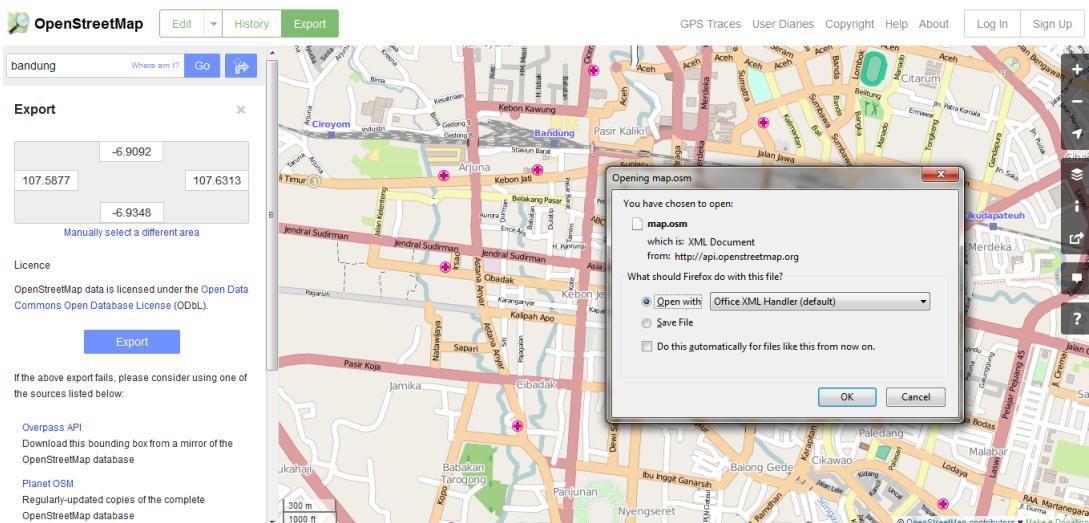
##### 3.1.1 Langkah-Langkah Pengambilan Data OSMXML

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

1. Mengunjungi situs [www.openstreetmap.org](http://www.openstreetmap.org).
2. Menggunakan fitur *search* untuk mencari area lokasi yang diinginkan. penggunaan fitur ini dapat dilihat pada Gambar 3.1 .
3. Menggunakan fitur *export* untuk mengunduh data dalam bentuk dokumen OSMXML. 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

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

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <osm version="0.6" generator="CGIMap_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/">
3 <bounds minlat="-6.9076500" minlon="107.5961800" maxlat="-6.9044500"
   maxlon="107.6016300"/>
4 <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" />
5 <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" />
6 ...
7 <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" />
8 <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" />
9 <node id="25500626" visible="true" version="3" changeset="839915"
     timestamp="2009-03-21T14:22:17Z" user="adhitya" uid="7748" lat="
     -6.9070329" lon="107.6019401" />
10 ...
11 <node id="3030289971" visible="true" version="1" changeset="24892866"
     timestamp="2014-08-20T18:40:31Z" user="albahrimaraxsa" uid="2162153"
     lat="-6.9066710" lon="107.5982569" />
12 <node id="2325451442" visible="true" version="4" changeset="27916144"
     timestamp="2015-01-04T18:06:33Z" user="isonpurba" uid="2552445" lat
     ="-6.9045011" lon="107.6024922" />
13 <way id="4567626" visible="true" version="4" changeset="15861148"
     timestamp="2013-04-25T13:56:12Z" user="mrdoggie94" uid="1331966">
14 <nd ref="25433682" />
15 <nd ref="25433681" />
16 <nd ref="25433680" />
17 <tag k="avgspeed" v="15" />
18 <tag k="highway" v="residential" />
19 <tag k="name" v="Dr. Rubini" />
20 </way>
21 <way id="4567634" visible="true" version="2" changeset="7821743"
     timestamp="2011-04-10T11:15:30Z" user="evo2mind" uid="234610">
22 <nd ref="25433681" />
23 <nd ref="28802396" />
24 <tag k="avgspeed" v="15" />
25 <tag k="highway" v="residential" />
26 <tag k="name" v="Dr. Susilo" />
27 </way>
28 ...
29 </osm>
```

Node dan way memiliki informasi penting yang digunakan pada aplikasi. Pada tag node terdapat atribut id yang menunjukkan id pada setiap node, kemudian terdapat atribut lat dan lon yang

memberikan informasi titik koordinat (*latitude* dan *longitude*) node tersebut. Informasi yang didapatkan, disimpan ke dalam bentuk node pada graf. Tag way akan menunjukkan hubungan pada node-node yang terdapat pada dokumen, dan 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 sekeliling 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 dibaca menggunakan javascript yang 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 dapat 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");
for (i=0;i<node.length;i++){
    document.write("<tr><td>");
    document.write(i);
    document.write("</td><td>");
    document.write(node[i].getAttribute('id'));
    document.write("</td><td>");
    document.write(node[i].getAttribute('lat'));
    document.write("</td><td>");
    document.write(node[i].getAttribute('lon'));
    document.write("</td></tr>");
}
document.write("</table>");
document.write("</div>");

```

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

- (a) Membuat tag `<div>` sebagai tempat tabel.
  - (b) Membuat tabel.
  - (c) Membuat variabel node, variabel ini berisi informasi seluruh node yang terdapat pada variabel xmlDoc, *method* yang digunakan adalah `getElementsByTagName()`. Sebelumnya xmlDoc sudah berisi dengan dokumen “map.xml”.
  - (d) Melakukan *print* pada tabel, beberapa atribut yang ditampilkan adalah id node, *latitude*, dan *longitude*.
3. Membuat fungsi untuk melakukan *filter* pada elemen way.

```

function isHighway(way,index){
    var tag = way[index].getElementsByTagName("tag");
    for (hg=0;hg<tag.length;hg++){
        if(tag[hg].getAttribute('k') == "highway"){
            return true;
        }
    }
    return false;
}

```

*Filter* dilakukan karena hanya elemen way yang berjenis *highway* saja yang akan digunakan. Berikut ini adalah tahap-tahap yang dilakukan:

- (a) Membuat fungsi `isHighway()` dengan parameter *input* way dan index. Parameter way berisi informasi seluruh way, sedangkan parameter index menunjukkan tag way pada index tersebut yang akan dicari.

- (b) Membuat variabel yang menyimpan informasi way pada index yang dicari, pada kode program variabel tersebut diberi nama “tag”.
- (c) Lakukan pengulangan untuk mencari setiap *child* pada variabel “tag” yang memiliki atribut “k=highway”, jika ditemukan fungsi akan mengembalikan *true* dan *false* jika tidak ditemukan.
4. Menampilkan informasi way yang didapat pada layar.

```

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

var way = xmlDoc.getElementsByTagName("way");
var nd;
for (i=0;i<way.length;i++){
  nd = way[i].getElementsByTagName("nd");
  if(isHighway(way,i)){
    for (j=0;j<nd.length-1;j++){
      document.write("<tr><td>");
      document.write(i);
      document.write("</td><td>");
      document.write(way[i].getAttribute('id'));
      document.write("</td><td>");
      document.write(j);
      document.write("</td><td>");
      document.write(nd[j].getAttribute('ref'));
      document.write("</td><td>");
      document.write(nd[j+1].getAttribute('ref'));
      document.write("</td></tr>");
    }
  }
}
document.write("</div>");
```

Kode diatas menampilkan informasi way pada dokumen OSMXML dalam bentuk tabel.

- (a) Membuat tag <div> sebagai tempat tabel.
- (b) Membuat tabel.
- (c) Membuat variabel way, variabel ini berisi informasi seluruh way yang terdapat pada variabel xmlDoc, *method* yang digunakan adalah *getElementsByTagName()*. Sebelumnya xmlDoc sudah berisi dengan dokumen “map.xml”.
- (d) Membuat variabel yang berisi informasi way pada index tertentu, dan lakukan *filter* dengan menggunakan fungsi *isHighway()*.

- (e) Melakukan *print* pada tabel, beberapa atribut yang ditampilkan adalah id way, id node pertama, dan id node kedua.

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

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

### 3.3 Menghitung Jarak Antara Dua Titik

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

```
function getDistance(lat1,lon1,lat2,lon2){
    var R = 6371;
    var dLat = deg2rad(lat2-lat1);
    var dLon = deg2rad(lon2-lon1);
    var a =
        Math.sin(dLat/2) * Math.sin(dLat/2) +
        Math.cos(lat1) * Math.cos(lat2) *
        Math.sin(dLon/2) * Math.sin(dLon/2);
    var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
    var d = R * c;
    return d;
}
```

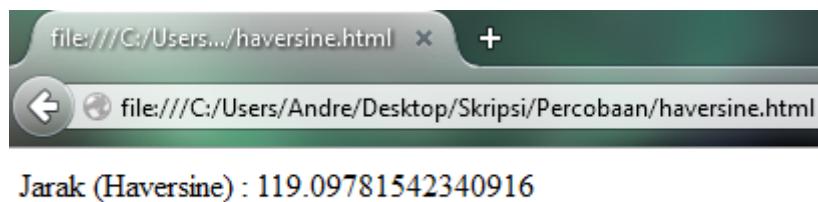
```

    Math.cos(deg2rad(lat1)) * Math.cos(deg2rad(lat2)) *
    Math.sin(dLon/2) * Math.sin(dLon/2);
var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
var d = R * c;
return d;
}

function deg2rad(deg){
  return deg * (Math.PI/180);
}

```

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



Gambar 3.4: Perhitungan Jarak Dengan Haversine Formula

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

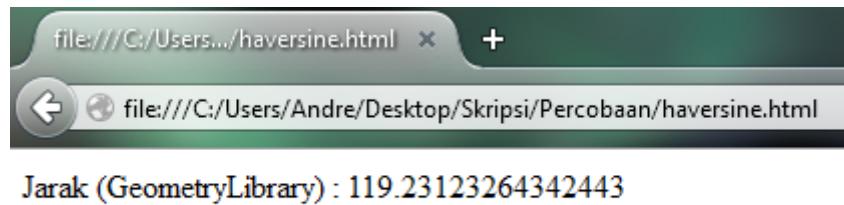
```

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

```

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

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



Gambar 3.5: Perhitungan Jarak Dengan Geometry Library

### 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, digunakan *adjacency list*. Penggunaan *adjacency list* disebabkan oleh penggunaan memori yang lebih kecil dibandingkan dengan *adjacency matrix*. Hal ini karena, setelah dilakukan *filter* diketahui bahwa graf tersebut berjenis *sparse*. Berikut ini adalah langkah-langkah yang dilakukan untuk memodelkan OSMXML menjadi graf:

1. Membuat kelas Node dan kelas Neighbor.

```
function Node(id, neighbors){
    this.id = id;
    this.adjList = neighbors;
}

function Neighbor(vnum, nbr, weight){
    this.vertexNum = vnum;
    this.weight = weight;
    this.next = nbr;
}
```

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.

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

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.

```
function wayDirection(way, index){
    var tag = way[index].getElementsByTagName("tag");
    for (hg=0; hg<tag.length; hg++){
        if(tag[hg].getAttribute('k') == "oneway"){
            return tag[hg].getAttribute('v');
        }
    }
    return false;
}
```

Berikut ini adalah tahap-tahap yang dilakukan:

- (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.
  - (b) Membuat variabel yang menyimpan informasi way pada index yang dicari, pada kode program variabel tersebut diberi nama “tag”.
  - (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.
4. Membuat fungsi untuk mendapatkan informasi koordinat node (*latitude* dan *longitude*).

```
function getLatByAtt(str){
    for (n=0;n<node.length;n++){
        if(node[n].getAttribute('id') == str){
            return node[n].getAttribute('lat');
        }
    }
    return -1;
}

function getLonByAtt(str){
    for (m=0;m<node.length;m++){
        if(node[m].getAttribute('id') == str){
            return node[m].getAttribute('lon');
        }
    }
}
```

```

    return -1;
}

```

Kode di atas terdiri dari dua fungsi, fungsi `getLatByAtt()` mencari informasi *latitude* dan fungsi `getLatByAtt()` mencari informasi *longitude*. Kedua fungsi tersebut melakukan pencarian node berdasarkan parameter *input* str yang merupakan id node, jika ditemukan fungsi akan mengembalikan nilai *latitude* atau *longitude* dan mengembalikan nilai -1 jika tidak ditemukan.

5. Membaca seluruh informasi edge yang terdapat pada tag way. Berikut ini adalah tahap-tahap yang dilakukan:
  - (a) Melakukan pengulangan untuk setiap way.
  - (b) Melakukan *filter* untuk setiap way, hanya “highway” saja yang akan dimasukkan informasinya ke dalam *adjacency list*.
  - (c) Melakukan pemanggilan fungsi `wayDirection()` dan memasukkan nilai didapatkan dari fungsi tersebut ke variabel, pada kode di atas menggunakan variabel “oneway”.
  - (d) Cari jarak antar node.
  - (e) Memasukkan informasi edge berdasarkan arah yang dimasukkan ke dalam variabel “one-way”.
6. Membuat fungsi untuk menampilkan graf ke layar

```

function print(list){
  for (j=0; j < list.length; j++){
    document.write(j);
    for (nbr=list[j].adjList; nbr != null;nbr=nbr.next) {
      document.write("---->");
      document.write('('+nbr.vertexNum+')');
    }
    document.write("<br>");
  }
}

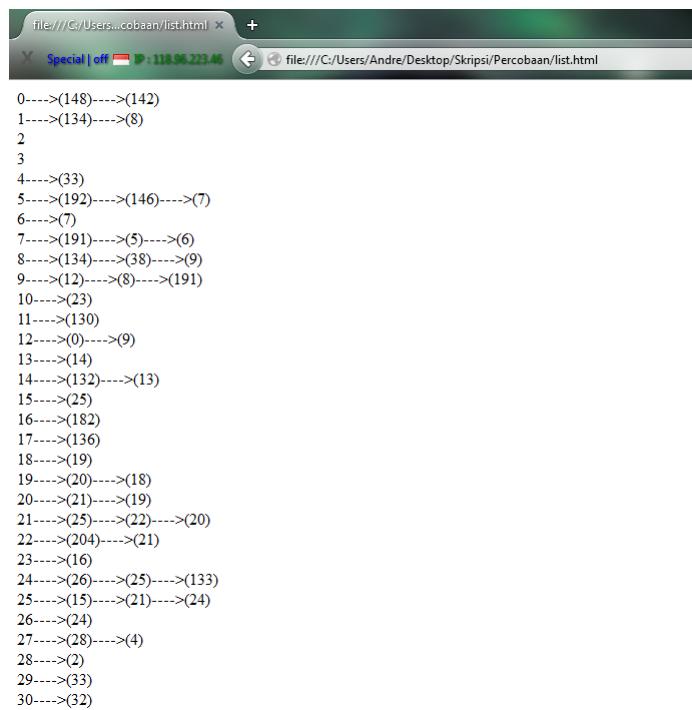
```

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

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

## 3.5 Visualisasi

Data OSMXML yang sudah dimodelkan ke dalam bentuk graf, selanjutnya divisualisasikan menggunakan Google Maps Javascript API. Peta ditampilkan dalam bentuk *roadmap*, karena peta jenis



Gambar 3.6: Pemodelan OSMXML menjadi Graf

ini memberikan informasi mengenai nama jalan, sehingga peta jenis ini lebih cocok untuk aplikasi yang dibangun. Berikut ini langkah-langkah yang dilakukan untuk membuat visualisasi graf:

1. Melakukan *load* Google Maps Javascript API.

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

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

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

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

```
var mapProp = {
  center:new google.maps.LatLng(-6.906845432118958,107.59851515293121),
  zoom:17,
  mapTypeId:google.maps.MapTypeId.ROADMAP
};

var map=new google.maps.Map(document.getElementById("googleMap"),mapProp);
```

Berikut ini tahap-tahap yang dilakukan:

- (a) Membuat variabel untuk mendefinisikan properti peta, pada kode di atas dilakukan pengaturan titik tengah, zoom pada level 17, dan tipe peta yaitu “ROADMAP”.

- (b) Membuat objek google.maps.Map, objek tersebut memiliki parameter properti peta (variabel mapProp), selanjutnya peta disisipkan pada elemen div yang memiliki id google-Map.
4. Membuat objek *marker* untuk setiap node pada peta.

```
for (j=0;j<nd.length;j++){
    id = uniqueId();
    marker = new google.maps.Marker({
        id: id,
        position: new google.maps.LatLng(getLatByAtt(nd[j].getAttribute('ref')),
            getLonByAtt(nd[j].getAttribute('ref'))),
        map: map,
        icon: image,
    });
    markers[id] = marker;
}
```

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

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

```
for (k=0;k<nd.length-1;k++){
    line = new google.maps.Polyline({
        path: [new google.maps.LatLng(getLatByAtt(nd[k].getAttribute('ref')),
            getLonByAtt(nd[k].getAttribute('ref'))),
            new google.maps.LatLng(getLatByAtt(nd[k+1].getAttribute('ref')),
            getLonByAtt(nd[k+1].getAttribute('ref')))],
        strokeColor: "#000000",
        strokeOpacity: 1.0,
        strokeWeight: 3,
        map: map
    });
}
```

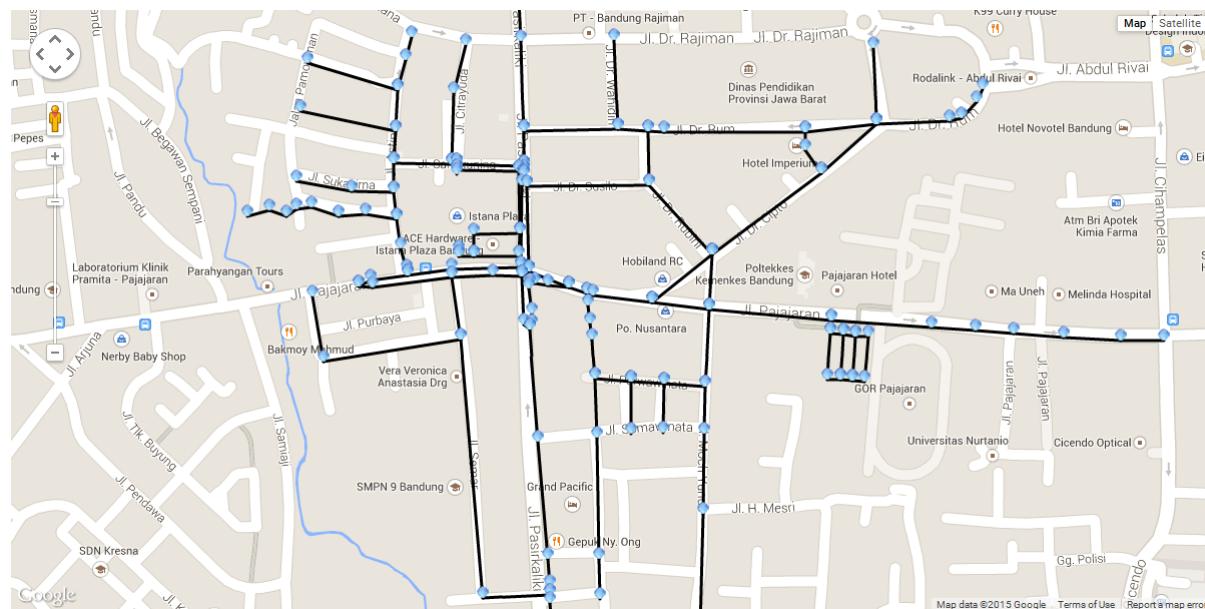
6. Membuat fungsi untuk menambahkan *info window* pada setiap node untuk memberikan informasi id dan index node.

```
function addInfoWindow(marker, message) {
    var infoWindow = new google.maps.InfoWindow({
        content: message
    });

    google.maps.event.addListener(marker, 'click', function () {
        infoWindow.open(map, marker);
    });
}
```

Berikut ini adalah tahap-tahap yang dilakukan:

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

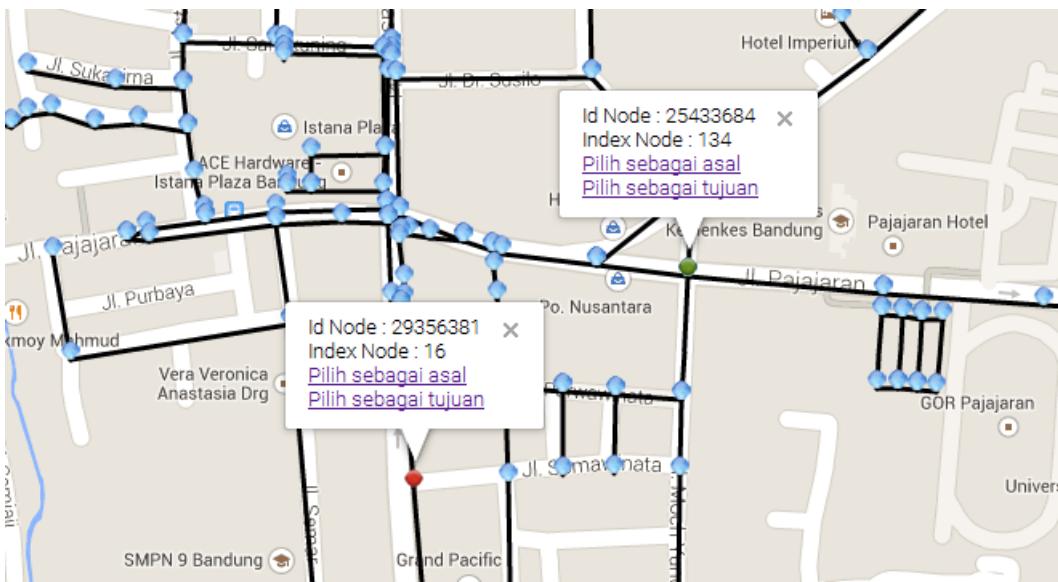


Gambar 3.7: Visualisasi

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. Setiap edge pada graf akan menjadi garis pada peta yang dibuat menggunakan *polyline*.

### 3.6 Algoritma Dijkstra

Untuk pencarian rute terdekat, aplikasi menggunakan algoritma Dijkstra. Algoritma tersebut diimplementasikan pada graf yang sudah dimodelkan sebelumnya. Fungsi dijkstra menerima *input* berupa objek “edge” yang berisi informasi dari graf, selanjutnya fungsi akan mengeluarkan *output* yaitu jalur terpendek dari satu titik ke titik lain dalam bentuk array. Untuk hasil implementasi algoritma Dijkstra dapat dilihat pada Bab 5. Contoh kasus yang digunakan, yaitu mencari rute terdekat dari 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.



Gambar 3.8: Info Window

```

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

## 3.7 Analisis Berorientasi Objek

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

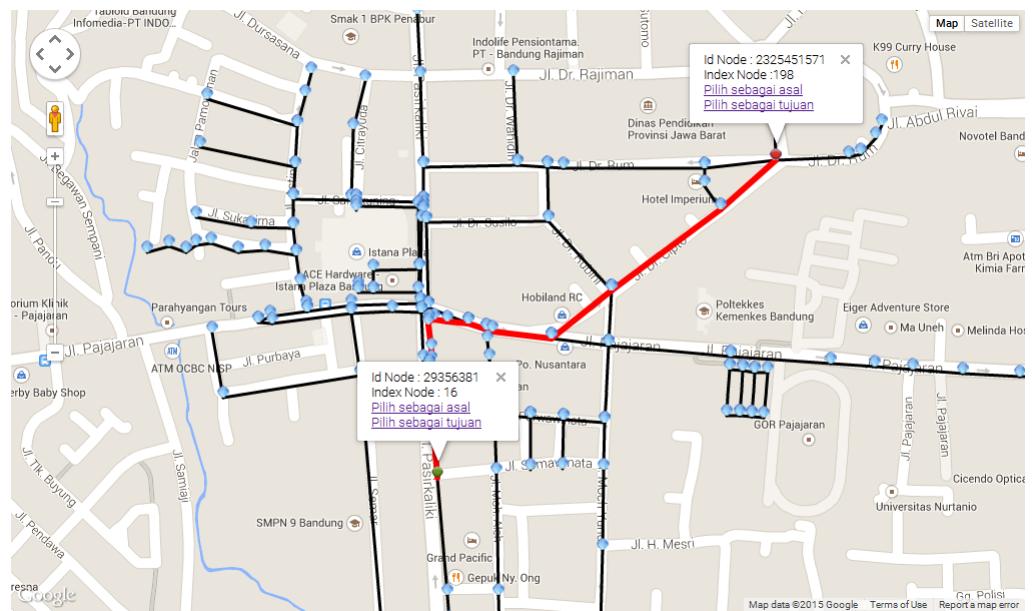
### 3.7.1 Diagram *Use Case*

Diagram *use case* adalah pemodelan yang berfungsi memperjelas interaksi antara aktor atau *user* dengan sistem. Diagram *use case* aplikasi pencarian rute terdekat dapat dilihat pada Gambar 3.11.

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

1. Memilih titik asal.

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



Gambar 3.10: Visualisasi Rute Terdekat

## 2. Memilih titik tujuan.

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.

## 3. Mencari rute terdekat dari titik asal ke titik tujuan.

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

### 3.7.2 Skenario

Berikut ini adalah skenario untuk setiap *use case*:

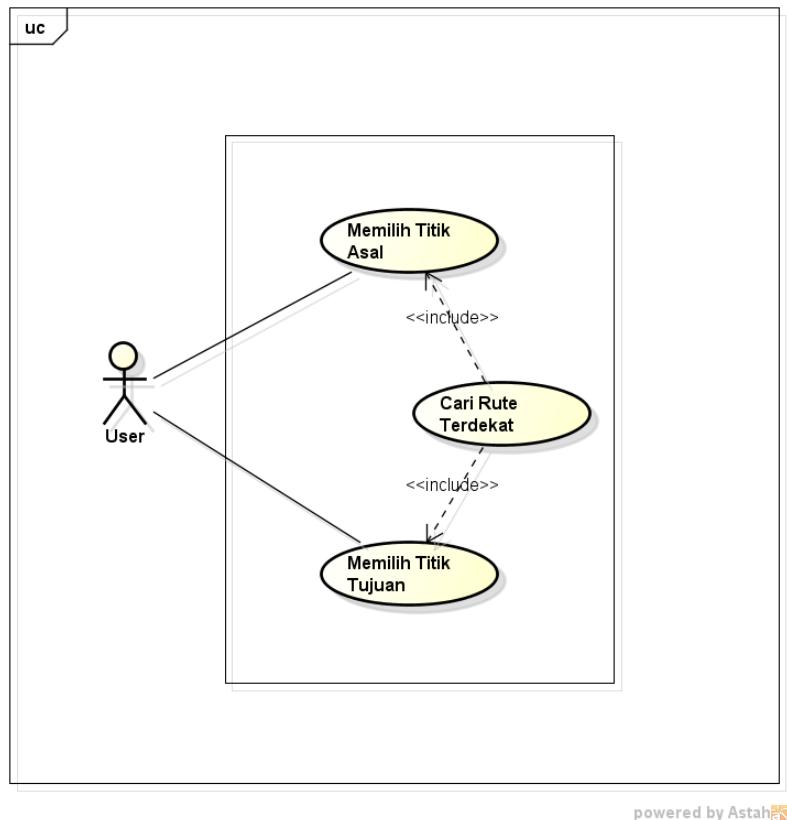
#### 1. Skenario memilih titik asal dapat dilihat pada Tabel 3.1.

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	-

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

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



powered by Astah

Gambar 3.11: Diagram *use case*

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	-

### 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

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

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

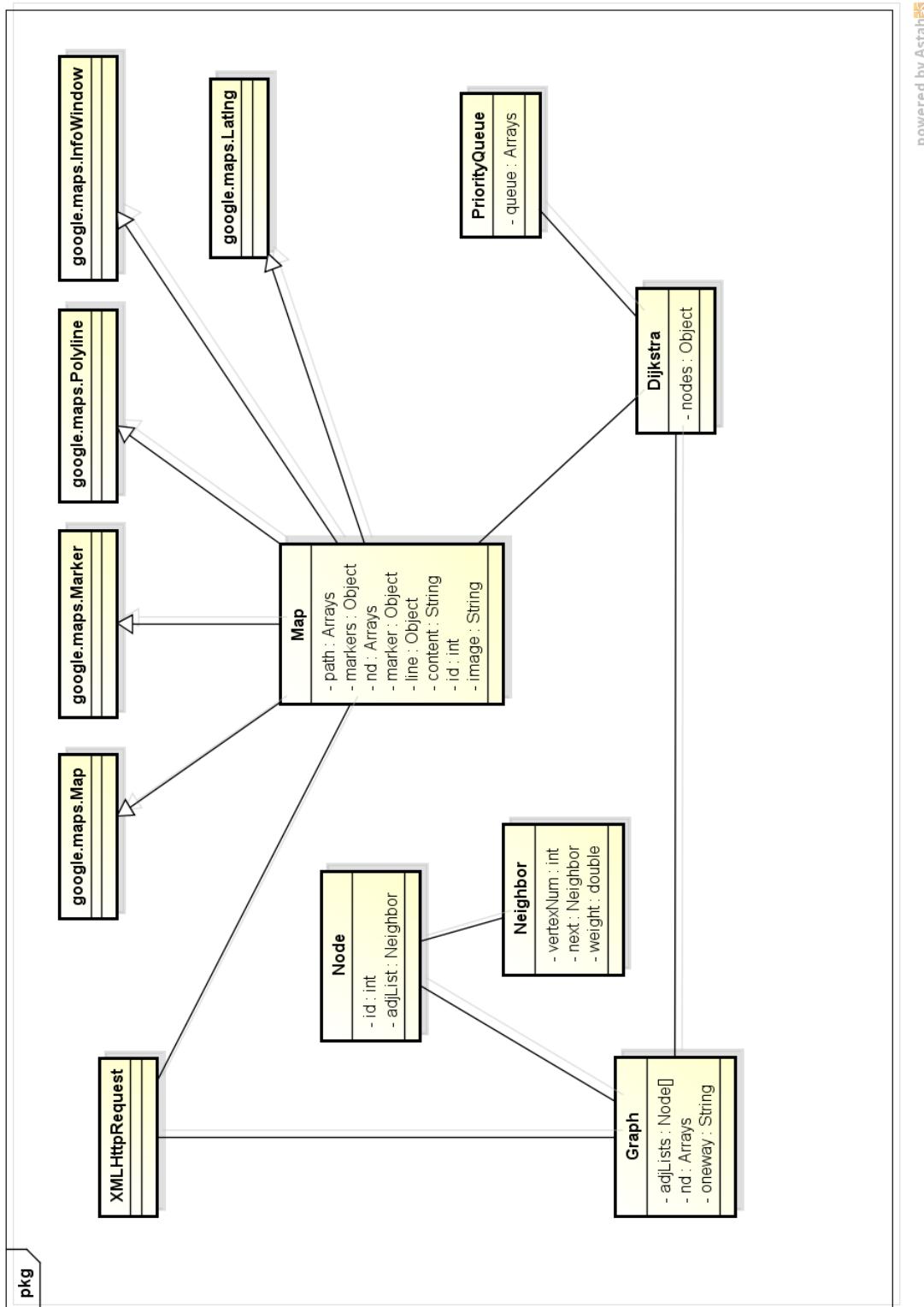
Kelas ini berfungsi untuk membuat objek *Latlng*. *Latlng* merupakan objek yang berisi informasi koordinat (*latitude* dan *longitude*).

- 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



## BAB 4

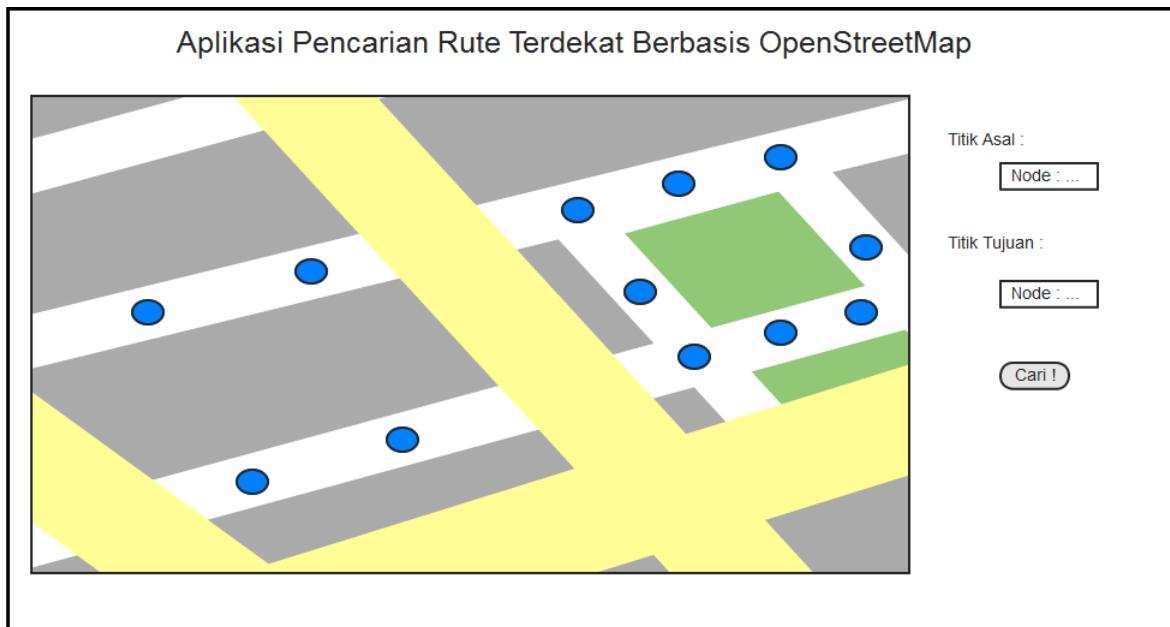
# PERANCANGAN

Pada bab ini dijelaskan perancangan aplikasi pencarian rute terdekat berbasis OpenStreetMap. Pada bagian pertama dijelaskan perancangan antar muka untuk aplikasi yang dibangun. Diawali dengan *user* membuka aplikasi, selanjutnya *user* dapat memilih titik-titik yang terdapat pada peta sebagai titik asal ataupun titik tujuan. Setelah itu, *user* dapat menekan tombol cari dan aplikasi memulai proses pencarian rute terdekat antara kedua titik tersebut.

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

### 4.1 Perancangan Antar Muka

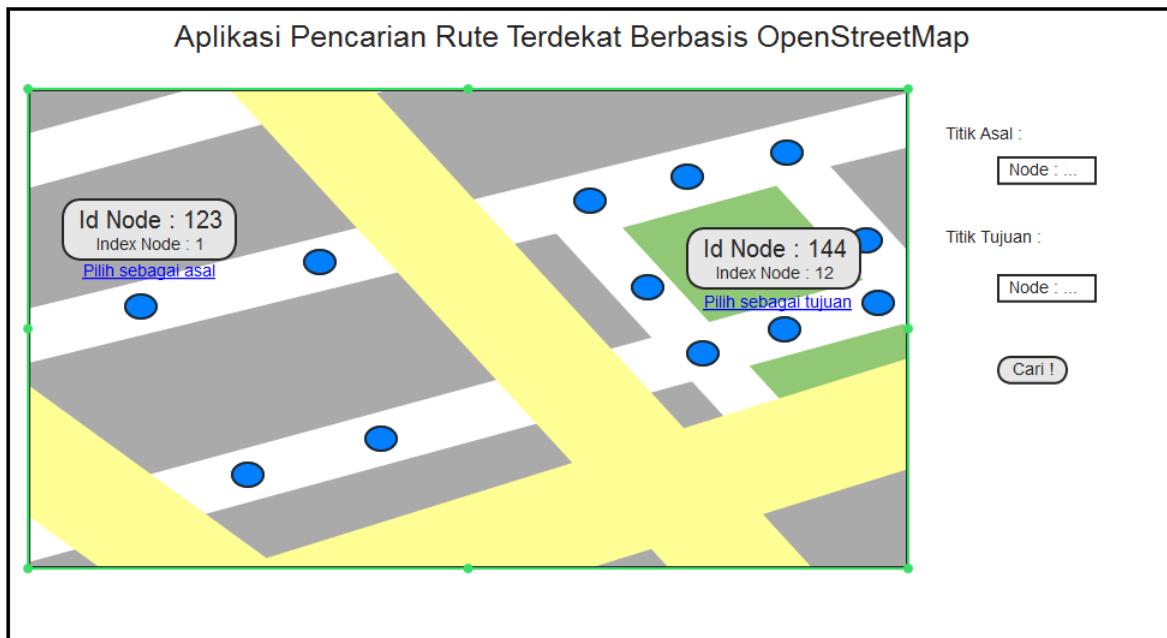
Pada subbab ini akan dijelaskan rancangan antar muka untuk aplikasi pencarian rute terdekat. Aplikasi yang dibangun akan memiliki tampilan awal seperti pada Gambar 4.1.



Gambar 4.1: Rancangan Antar Muka Awal Aplikasi

Setelah halaman awal terbuka, *User* dapat menekan setiap titik tersebut dan akan menampilkan *info window* yang memberikan informasi titik beserta *link*. *User* dapat menekan *link* tersebut untuk

menjadikannya sebagai titik asal ataupun titik tujuan. Rancangan antar muka saat user memilih titik, dapat dilihat pada Gambar 4.2.



Gambar 4.2: Rancangan Pemilihan Titik

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

## 4.2 Perancangan Kelas

Pada bagian ini akan dibahas perancangan kelas yang dilakukan untuk aplikasi yang akan dibangun. Perancangan kelas bertujuan untuk menjelaskan seluruh atribut dan fungsi yang akan diimplementasikan pada setiap kelas yang sudah dibahas pada bab analisis. Perancangan kelas yang dibuat dapat dilihat pada Gambar 4.4. Berikut ini adalah penjelasan dari setiap kelas beserta atribut dan fungsi yang digunakan:

- Kelas XMLHttpRequest

Kelas ini berfungsi untuk melakukan *load* dokumen OSMXML.

– Fungsi

\* *open()*

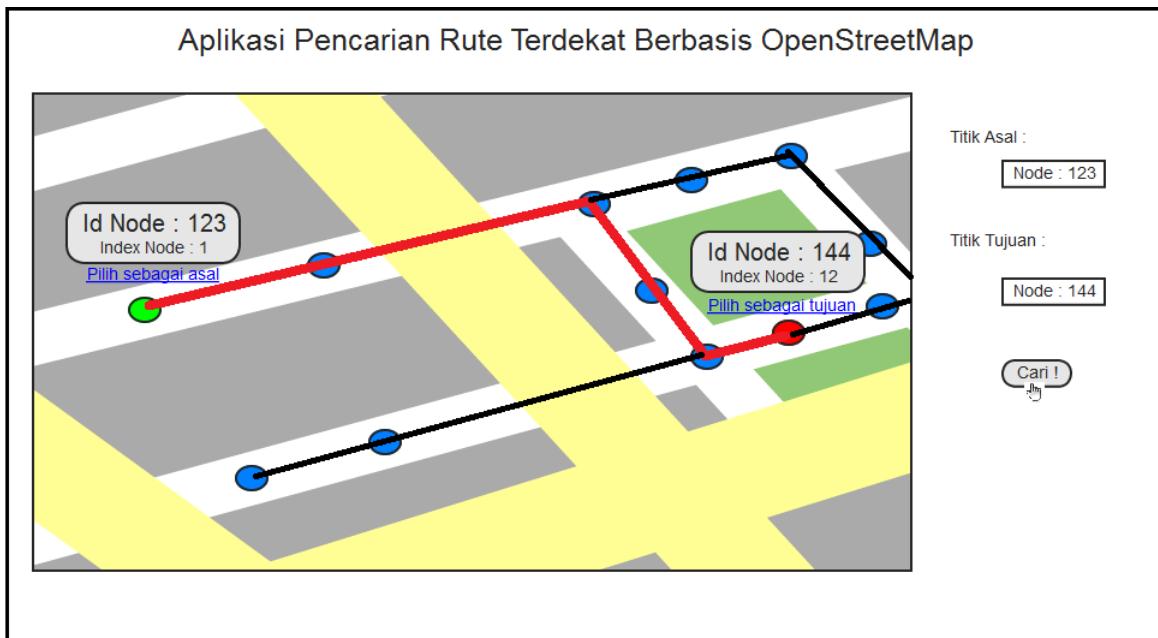
Fungsi *open()* digunakan untuk membuka dokumen OSMXML.

\* *send()*

Fungsi *send()* digunakan untuk mengirimkan dokumen OSMXML yang sudah dibuka.

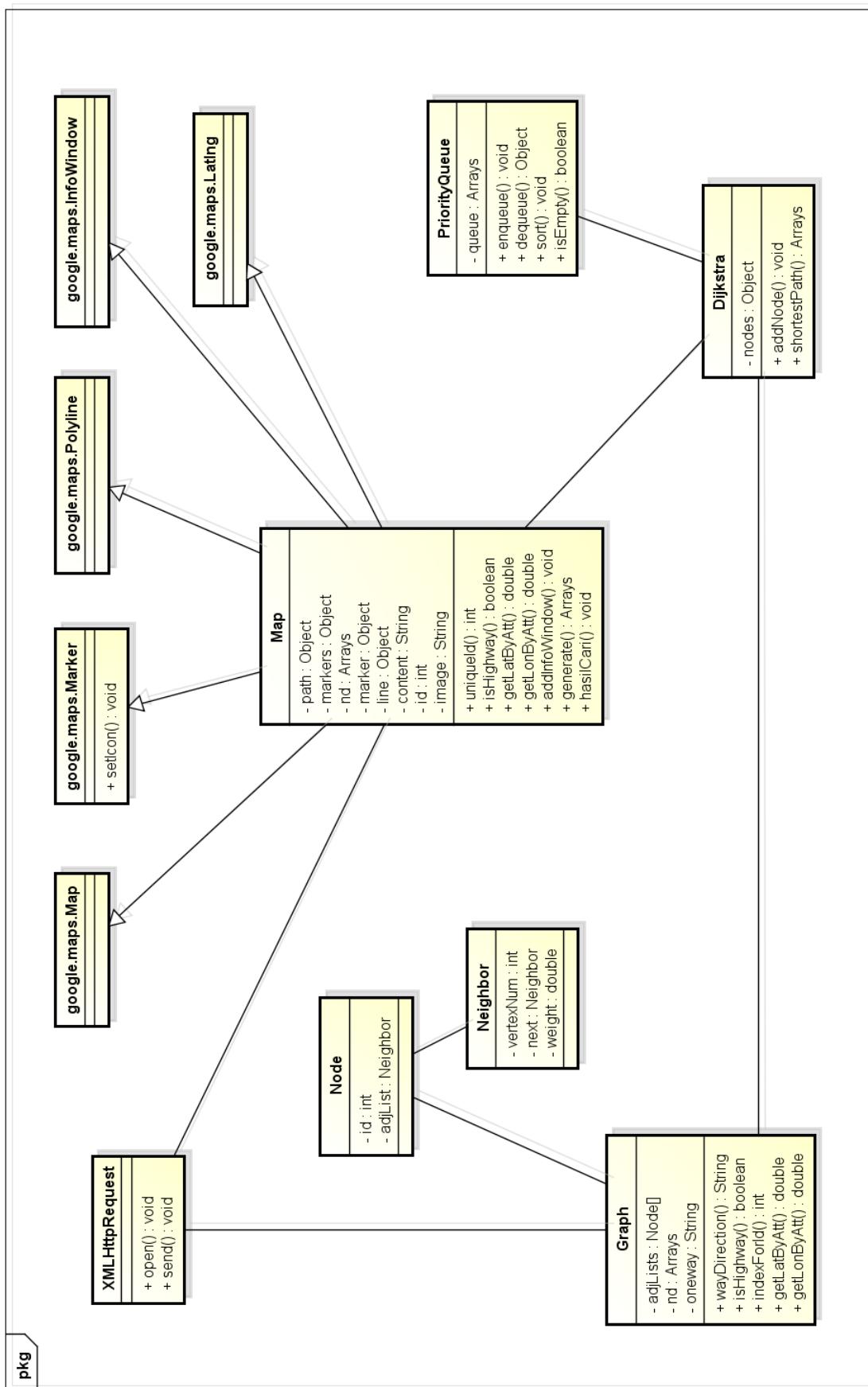
- Kelas Node dan Neighbor

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



Gambar 4.3: Rancangan Cari Rute

- Atribut Kelas Node
  - \* id : int  
Atribut id menyimpan id node yang didapatkan dari dokumen OSMXML.
  - \* adjList : Neighbor  
Atribut adjList menyimpan objek Neighbor.
- Atribut Kelas Neighbor
  - \* vertexNum : int  
Atribut vertexNum menyimpan index dari node.
  - \* next : Neighbor  
Atribut next menyimpan objek Neighbor.
  - \* weight : double  
Atribut weight menyimpan jarak antar node.
- Kelas Graph
  - Kelas ini berfungsi untuk mengubah informasi yang didapatkan dari OSMXML menjadi graf.
  - Atribut
    - \* adjLists : Array of Node  
Atribut adjLists merupakan *array* yang menyimpan objek node.
    - \* nd : Array  
Atribut nd menyimpan informasi berupa id node yang terdapat pada tag way di dalam dokumen OSMXML, atribut nd bertipe array.
    - \* oneway : String  
Atribut nd menyimpan informasi berupa *value* dari *key* oneway yang terdapat pada tag way di dalam dokumen OSMXML, atribut oneway bertipe String.



Gambar 4.4: Diagram Kelas

- Fungsi

- \* wayDirection() : String

Fungsi ini digunakan untuk mengetahui arah pada setiap node berdasarkan *value* yang terdapat pada OSMXML.

- \* isHighway() : boolean

Fungsi ini digunakan untuk melakukan *filter* pada tag way, yaitu hanya way yang bertipe “Highway” saja yang akan digunakan pada pemodelan graf.

- \* indexForId() : int

Fungsi ini digunakan untuk mengetahui index node pada graf berdasarkan id node.

- \* getLatByAtt() : double

Fungsi ini digunakan untuk mendapatkan informasi *latitude* pada sebuah node berdasarkan atributnya, yaitu id node.

- \* getLonByAtt() : double

Fungsi ini digunakan untuk mendapatkan informasi *longitude* pada sebuah node berdasarkan atributnya, yaitu id node.

- Kelas Map

Kelas ini berfungsi untuk melakukan *generate* peta, visualisasi graf, dan visualisasi rute ter-dekat.

- Atribut

- \* path : Object

Atribut path adalah variabel yang menyimpan informasi titik koordinat dalam bentuk objek, atribut path digunakan untuk pembuatan *marker* dan *polyline*.

- \* markers : Object

Atribut ini menyimpan seluruh objek *marker*.

- \* nd : Arrays

Atribut nd menyimpan informasi berupa id node yang terdapat pada tag way di dalam dokumen OSMXML, atribut nd bertipe array.

- \* marker : Object

Atribut ini menyimpan objek *marker* untuk ditampilkan pada peta.

- \* line : Object

Atribut ini menyimpan objek *polyline* untuk ditampilkan pada peta.

- \* content : String

Atribut ini merupakan isi dari *info window* yang disisipkan pada setiap marker.

Atribut ini berisi id node, index node, *link* titik asal, dan *link* titik tujuan.

- \* id : int

Atribut ini merupakan id yang digunakan untuk melakukan *generate* id pada fungsi *uniqueId*.

- \* image : String

Atribut ini menyimpan alamat dari *image* atau gambar yang digunakan oleh *marker*.

- Fungsi

- \* `uniqueId()` : id

Fungsi ini digunakan untuk melakukan *generate* id, id tersebut digunakan oleh *marker*.

- \* `isHighway()` : boolean

Fungsi ini digunakan untuk melakukan *filter* pada tag way, hanya way yang bertipe “Highway” saja yang digunakan. Fungsi ini digunakan pada saat pembuatan objek *marker* pada peta.

- \* `getLatByAtt()` : double

Fungsi ini digunakan untuk mendapatkan informasi *latitude* pada sebuah node berdasarkan atributnya, yaitu id node.

- \* `getLonByAtt()` : double

Fungsi ini digunakan untuk mendapatkan informasi *longitude* pada sebuah node berdasarkan atributnya, yaitu id node.

- \* `addInfoWindow()`

Fungsi ini digunakan untuk menambahkan objek *info window* pada setiap *marker*.

- \* `generate()` : Arrays

Fungsi ini digunakan untuk menampilkan secara keseluruhan *marker* dan *polyline* pada peta. Fungsi akan mengembalikan objek *marker* di dalam bentuk array.

- \* `hasilCari()`

Fungsi ini digunakan untuk melakukan visualisasi rute terdekat menggunakan *polyline*.

- 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.

- Fungsi

- \* `SetIcon()`

Fungsi ini digunakan untuk mengubah icon dari *marker*.

- 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.

- Atribut

- \* nodes : Object

Atribut ini menyimpan informasi node dalam bentuk objek.

- Fungsi

- \* addNode()

Fungsi ini digunakan untuk menambahkan satu node beserta edgenya.

- \* shortestPath() : Arrays

Fungsi ini digunakan untuk pencarian rute terdekat, fungsi akan mengembalikan rute atau jalur terpendek dalam bentuk array.

- Kelas PriorityQueue

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

- Atribut

- \* queue : Array

Atribut ini merupakan *queue* dari kelas PriorityQueue.

- Fungsi

- \* enqueue()

Fungsi ini digunakan untuk menambahkan objek ke dalam *queue*.

- \* dequeue() : Object

Fungsi ini digunakan untuk mengeluarkan objek dari dalam *queue*.

- \* sort()

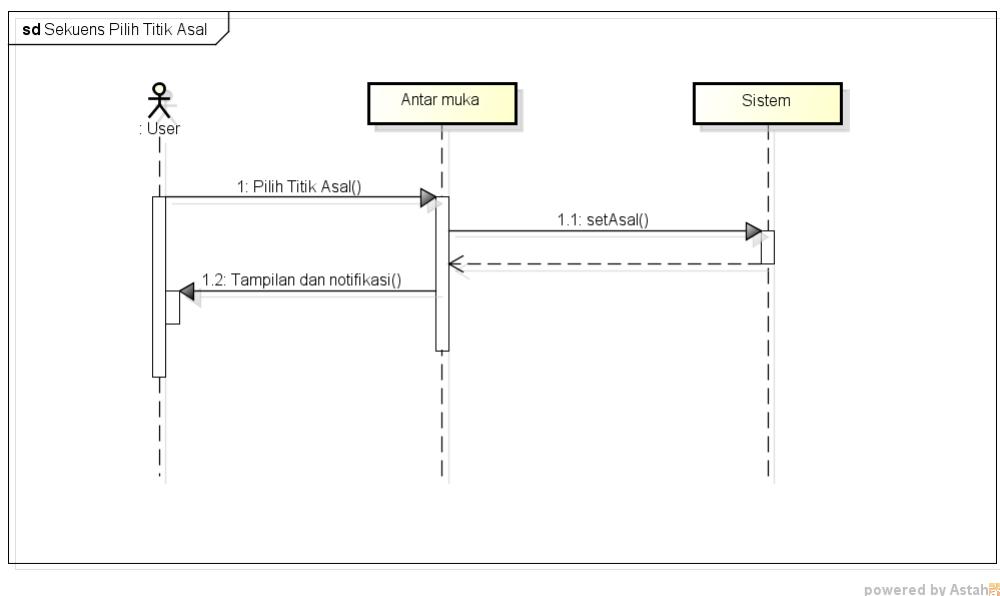
Fungsi ini digunakan untuk menyusun *queue*.

- \* isEmpty() : boolean

Fungsi ini digunakan untuk pengecekan isi dari *queue*. jika *queue* kosong, fungsi akan mengembalikan “true” dan sebaliknya “false” jika *queue* tidak kosong.

## 4.3 Diagram Sekuens

Diagram sekuens adalah diagram yang digunakan untuk menggambarkan interaksi antara objek dengan waktu, interaksi tersebut digambarkan dengan grafik dua dimensi. Kedua dimensi tersebut adalah dimensi horizontal dan dimensi vertikal. Dimensi horizontal menggambarkan objek-objek yang berperan, sedangkan dimensi vertikal menggambarkan waktu. Pesan yang dikirimkan oleh suatu objek digambarkan dengan panah dan panah dengan garis putus-putus menggambarkan pesan balasan. Diagram sekuens pada bagian ini mengacu kepada diagram *use case* yang terdapat pada bab 3, dapat dilihat pada Gambar 3.11. Berikut ini adalah diagram sekuens berdasarkan diagram *use case* tersebut:



powered by Astah

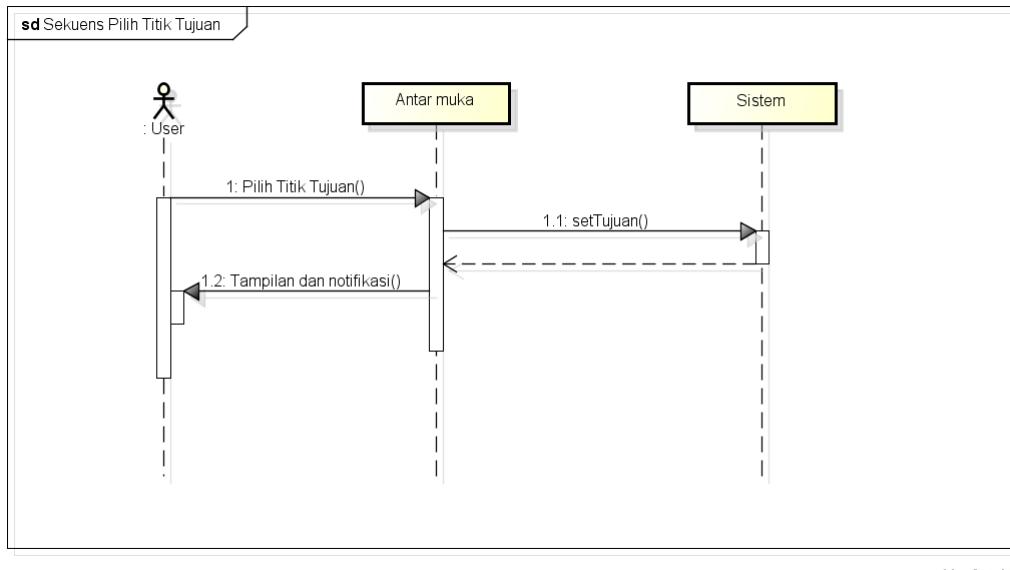
Gambar 4.5: Diagram Sekuens Pemilihan Titik Asal

### 1. Pemilihan Titik Asal

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

### 2. Pemilihan Titik Tujuan

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

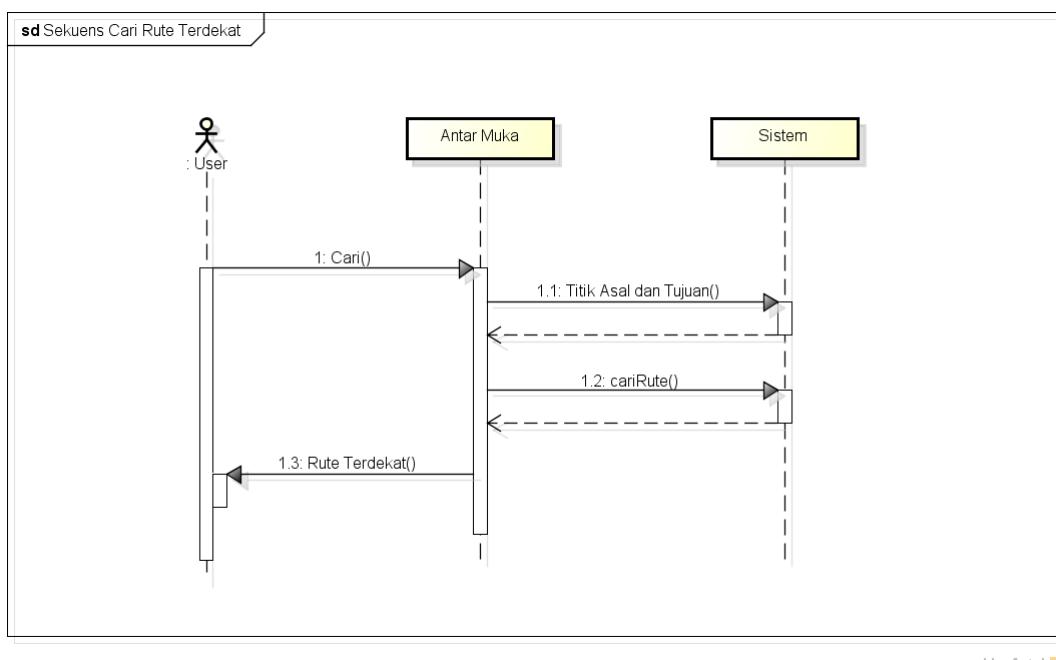


powered by Astah

Gambar 4.6: Diagram Sekuens Pemilihan Titik Tujuan

### 3. Pencarian Rute Terdekat

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



Gambar 4.7: Diagram Sekuens Pencarian Rute Terdekat



# BAB 5

## IMPLEMENTASI DAN PENGUJIAN

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. Pengujian dilakukan untuk mengetahui apakah fungsi-fungsi utama aplikasi sudah berjalan dengan baik dan juga untuk mengetahui kekurangan yang dimiliki aplikasi tersebut.

### 5.1 Implementasi

Implementasi dilakukan menggunakan bahasa pemrograman javascript, hasil implementasi adalah dokumen HTML. Berikut ini adalah beberapa bagian dari hasil implementasi, keseluruhan hasil implementasi dapat dilihat pada Lampiran A:

- Pembuatan Kelas Node dan Kelas Neighbor

Berikut ini adalah hasil implementasi dalam bentuk kode program dari kelas node dan neighbor. Kedua kelas tersebut merupakan komponen utama untuk pembuatan graf.

```
1 // Kelas Node
2 function Node(id , neighbors){
3     this . id = id ;
4     this . adjList = neighbors ;
5 }
6
7 // Kelas Neighbor
8 function Neighbor(vnum, nbr , weight){
9     this . vertexNum = vnum ;
10    this . next = nbr ;
11    this . weight = weight ;
12 }
```

- Pemodelan OSMXML menjadi graf.

Berikut ini adalah hasil implementasi dalam bentuk kode program yang berfungsi untuk memodelkan OSMXML menjadi graf

```
1 for (v=0; v < node . length ; v++){
2     adjLists . push (new Node(node [v] . getAttribute ('id') , null)) ;
3 }
4
5 for ( i=0;i<way . length ; i++){
6     nd = way [ i ] . getElementsByTagName ("nd") ;
```

```

7  if (isHighway(way, i)){
8      oneway = wayDirection(way, i);
9      for (j=0;j<nd.length-1;j++){
10         //cari jarak
11         v1 = indexForId(adjLists, nd[j].getAttribute('ref'));
12         v2 = indexForId(adjLists, nd[j+1].getAttribute('ref'));
13         lat1 = getLatByAtt(nd[j].getAttribute('ref'));
14         lon1 = getLonByAtt(nd[j].getAttribute('ref'));
15         lat2 = getLatByAtt(nd[j+1].getAttribute('ref'));
16         lon2 = getLonByAtt(nd[j+1].getAttribute('ref'));
17
18         point1 = new google.maps.LatLng(lat1,lon1);
19         point2 = new google.maps.LatLng(lat2,lon2);
20         distance = google.maps.geometry.spherical.
21         computeDistanceBetween(point1,point2);
22
23         //memasukkan informasi edge
24         if(oneway == "yes"){
25             adjLists[v1].adjList = new Neighbor(v2,
26             adjLists[v1].adjList,distance);
27         }else if(oneway == "no"){
28             adjLists[v1].adjList = new Neighbor(v2,
29             adjLists[v1].adjList,distance);
30             adjLists[v2].adjList = new Neighbor(v1,
31             adjLists[v2].adjList,distance);
32         }else if(oneway == "-1"){
33             adjLists[v2].adjList = new Neighbor(v1,
34             adjLists[v2].adjList,distance);
35         }else{
36             adjLists[v1].adjList = new Neighbor(v2,
37             adjLists[v1].adjList,distance);
38             adjLists[v2].adjList = new Neighbor(v1,
39             adjLists[v2].adjList,distance);
40         }
41     }
42   }
43 }
```

- Visualisasi Graf

Berikut ini adalah hasil implementasi kode program yang berfungsi untuk menampilkan node dan edge pada peta digital.

```

1 this.generate = function(){
2     for (i=0;i<way.length;i++)
3     {
4         nd = way[i].getElementsByName("nd");
5         if(isHighway(way, i))
6         {
7             for (j=0;j<nd.length;j++)
8             {
9                 id = uniqueId();
10                marker = new google.maps.Marker({
11                    id: id,
12                    position: new google.maps.LatLng(getLatByAtt(nd[j].
13                                         getAttribute('ref')), getLonByAtt(nd[j].getAttribute('ref
14                                         '))),
15                    map: map,
16                    icon: image,
17                });
18                markers[id] = marker;
19            }
20        }
21    }
22}
```

```

17         content = 'Id Node : '+nd[j].getAttribute('ref')+'<br>'+
18         'Index Node : '+indexForId(list,nd[j].getAttribute('ref'))+'<br>'+
19         '<a href="#" onclick="setAsal(\''+id+'\','+indexForId(list,nd[j]).
20             getAttribute('ref')+'\',\''+indexForId(list,nd[j]).
21             getAttribute('ref')+'\')">Pilih sebagai asal</a>'+<br>+
22         '<a href="#" onclick="setTujuan(\''+id+'\','+indexForId(list,nd[j]).
23             getAttribute('ref')+'\',\''+indexForId(list,nd[j]).
24             getAttribute('ref')+'\')">Pilih sebagai tujuan</a>';
25
26         addInfoWindow(marker, content);
27     }
28
29     for (k=0;k<nd.length-1;k++)
30     {
31         line = new google.maps.Polyline({
32             path: [new google.maps.LatLng(getLatByAtt(nd[k].getAttribute(
33                 'ref')), getLonByAtt(nd[k].getAttribute('ref'))),
34             new google.maps.LatLng(getLatByAtt(nd[k+1].getAttribute('ref'))
35                 , getLonByAtt(nd[k+1].getAttribute('ref')))],
36             strokeColor: "#000000",
37             strokeOpacity: 1,
38             strokeWeight: 3,
39             map: map
40         });
41     }
42     return markers;
43 }

```

- Pencarian Rute Terdekat

Berikut ini adalah hasil implementasi algoritma Dijkstra untuk pencarian rute terdekat.

```

1 this.shortestPath = function(asal, tujuan){
2     var PQ = new PriorityQueue();
3     var distances = {};
4     var previous = {};
5     var path = [];
6     var smallest, node, neighbor, alt;
7
8     for(node in this.nodes){
9         if(node === asal){
10             distances[node] = 0;
11             PQ.enqueue(0, node);
12         }
13         else{
14             distances[node] = INFINITY;
15             PQ.enqueue(INFINITY, node);
16         }
17         previous[node] = null;
18     }
19
20     while(!PQ.isEmpty()){
21         smallest = PQ.dequeue();
22         if(smallest === tujuan){
23             while(previous[smallest]){
24                 path.push(smallest);
25                 smallest = previous[smallest];
26             }

```

```

27         break;
28     }
29
30     if( distances [smallest] === INFINITY){
31         break;
32     }
33
34     for(neighbor in this.nodes[smallest]){
35         alt = distances [smallest] + this.nodes[smallest][neighbor];
36         if( alt < distances [neighbor]) {
37             distances [neighbor] = alt;
38             previous [neighbor] = smallest;
39             PQ.enqueue(alt , neighbor);
40         }
41     }
42 }
43 return path;
44 }
```

## 5.2 Pengujian

Pada subbab ini dibahas lingkungan pengujian dan 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 Lingkungan Pengujian

Berikut ini adalah lingkungan pengujian aplikasi:

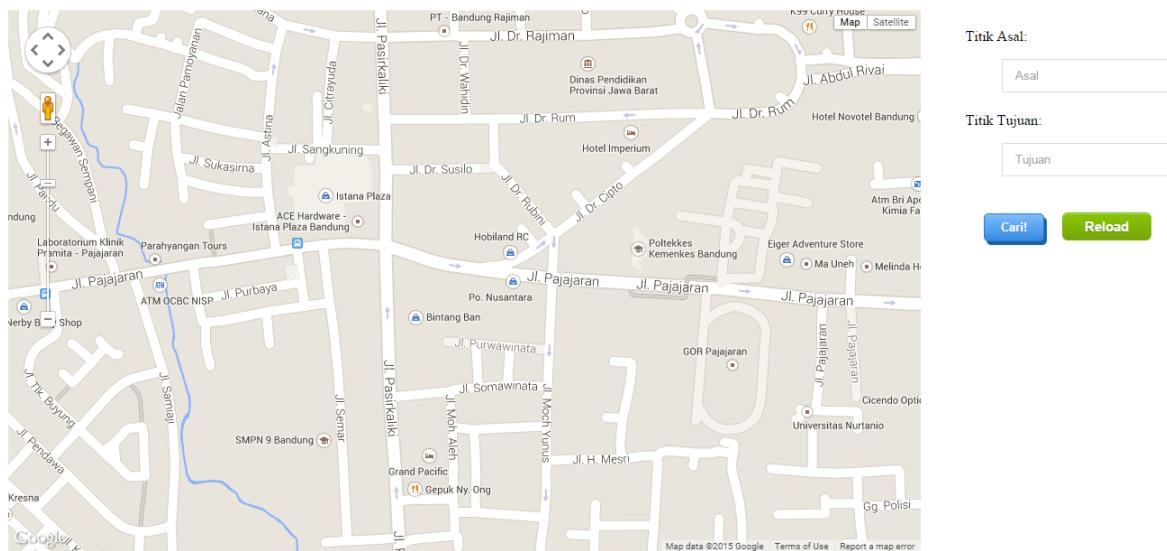
1. Processor  
Intel Core i5-2410M CPU @ 2.30Ghz (4 CPU)
2. Memory  
4096MB RAM
3. Display  
AMD Radeon HD 6470M
4. Operating System  
Windows 7 Ultimate 64-bit
5. Browser  
Google Chrome Version 41.0.2272.118 m
6. Bahasa Pemrograman  
Javascript
7. Teks Editor  
Notepad++

### 5.2.2 Pengujian Fungsional

Pengujian fungsional menguji tampilan antar muka aplikasi beserta method atau fungsi dasar. Seluruh pengujian fungsional menggunakan OSMXML yang bernama “bandung1.xml”, batas area pada dokumen ini memiliki koordinat yaitu <bounds minlat="-6.9076500" minlon="107.5961800" maxlat="-6.9044500" maxlon="107.6016300"/>, cakupan area tersebut berlokasi di Kota Bandung. Untuk dokumen bandung1.xml dapat dilihat pada Lampiran B. 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

#### 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="CGImap 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 bandung1.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>
```

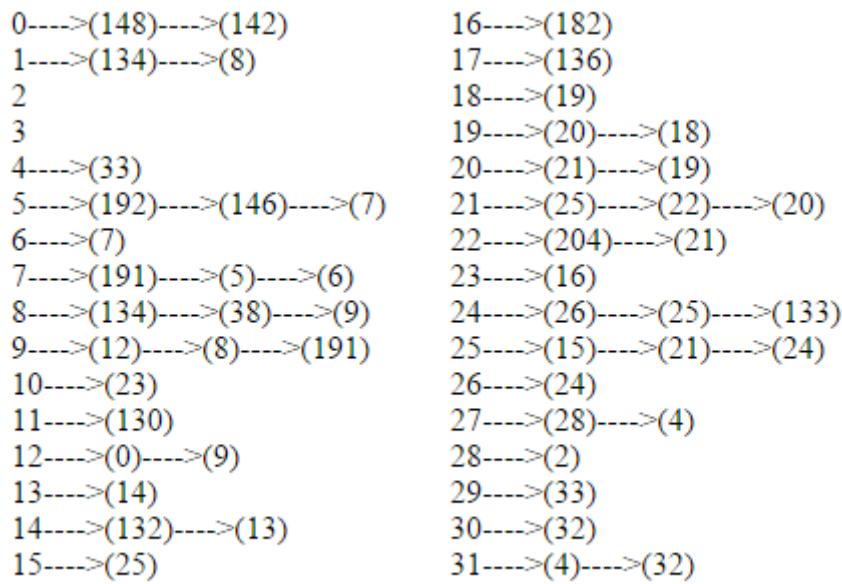
Gambar 5.4: Dokumen bandung1.xml

<b>Id Way</b>	<b>Edge</b>	<b>Id Node 1</b>	<b>Id Node 2</b>	<b>Jarak dalam meter</b>
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

4. 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 208 buah node dan terlalu banyak jika



Gambar 5.6: Pengujian Fungsi OSMXML Menjadi Graf

ditampilkan seluruhnya.

5. Fungsi untuk melakukan visualisasi.

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

6. Fungsi untuk memilih titik asal dan titik tujuan.

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

7. Fungsi untuk mencari rute terdekat antar dua titik menggunakan algoritma dijkstra.

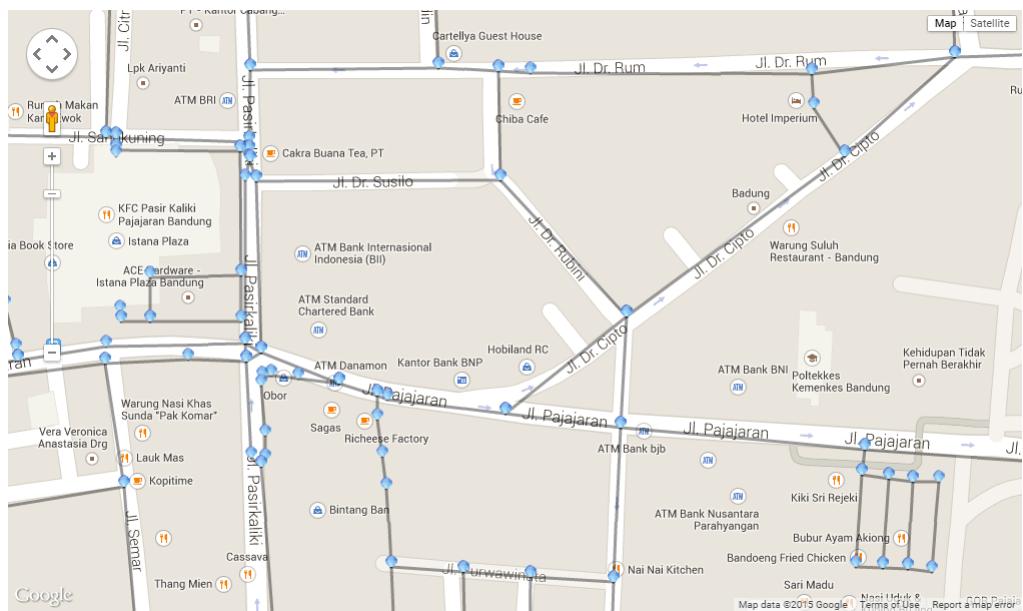
Pengujian dilakukan dengan contoh kasus mencari rute terdekat dari titik asal (id node : 29356381, index node : 16) ke titik tujuan (id node : 25500626, index node : 11), hasil pengujian dapat dilihat pada Gambar 5.9.

8. Fungsi untuk visualisasi rute terdekat.

Pengujian dilakukan dengan contoh kasus yang sama pada fungsi pencari rute terdekat, rute terdekat digambarkan dengan *polyline* berwarna merah. Hasil pengujian dapat dilihat pada Gambar 5.10.

### 5.2.3 Pengujian Eksperimental

Pengujian eksperimental terdiri dari dua tahap, yaitu pengujian pencarian rute dari titik asal hingga tujuan dengan beberapa pilihan rute, sehingga dapat dipastikan bahwa aplikasi benar-benar menunjukkan rute yang terdekat. Tahap kedua, pengujian dilakukan dengan menggunakan tiga



Gambar 5.7: Pengujian Fungsi Visualisasi

dokumen OSMXML yang berbeda, hal ini dilakukan untuk menguji bahwa aplikasi tetap berjalan dengan baik menggunakan dokumen OSMXML yang berbeda-beda. Ketiga dokumen tersebut diberi nama bandung1.xml, bandung2.xml, dan bandung3.xml. Ketiga dokumen tersebut dihitung luasnya berdasarkan informasi *latitude* dan *longitude* secara manual, berikut ini adalah cakupan area dari ketiga OSMXML tersebut:

### 1. Bandung 1

Dokumen bandung1.xml hanya meliputi sebagian kecil Kota Bandung dengan titik pusat di sekitar Jalan.Pajajaran. Dokumen ini memiliki luas sekitar 0,3907 km persegi.

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

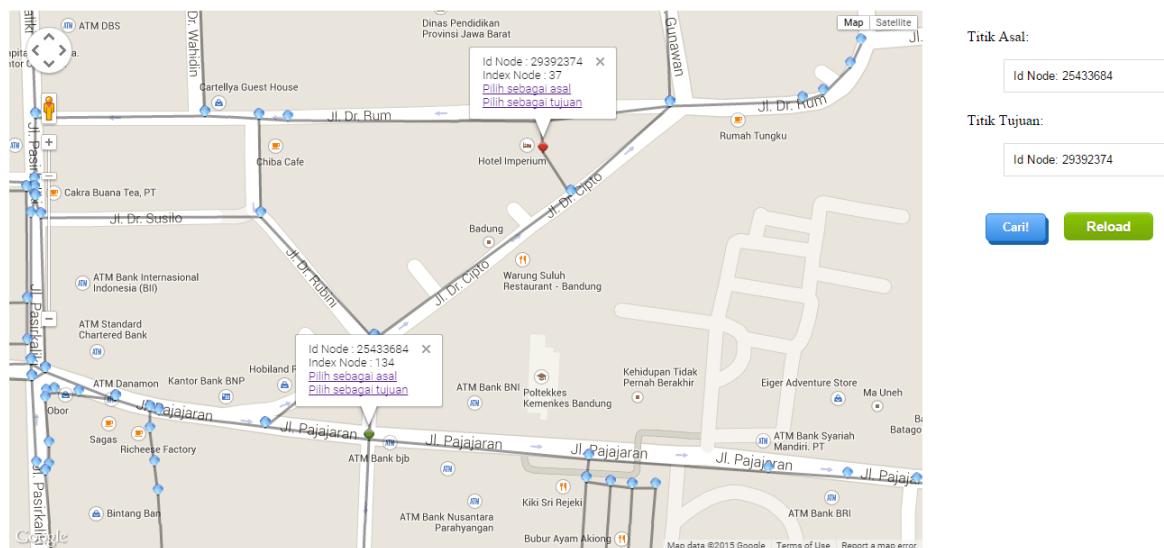
### 2. Bandung 2

Dokumen bandung2.xml memiliki ukuran yang lebih besar dibandingkan dengan bandung1.xml, yaitu dengan luas sekitar 3,423 km persegi.

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

### 3. Bandung 3

Dokumen bandung3.xml memiliki ukuran yang paling besar dan mencakup seluruh wilayah Kota Bandung dengan luas sekitar 415,8616 km persegi.



Gambar 5.8: Pengujian Titik Asal dan Tujuan

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

### 5.2.3.1 Pengujian Rute Terdekat

Pada bagian ini dijelaskan pengujian yang dilakukan untuk memastikan bahwa rute yang ditunjukkan oleh aplikasi adalah rute yang paling dekat. Pengujian dilakukan dengan memilih titik asal yang memiliki beberapa rute untuk dapat sampai ke titik tujuan. Dipilih titik asal pada sekitar Jl. Pasirkaliki dan titik tujuan di sekitar Jl. Dr Rum. Untuk mencapai titik tujuan terdapat beberapa rute, di antaranya melalui Jl. Dr Rajiman, Jl. Dr Rum dan Jl. Dr Cipto. Pada peta, dapat dilihat bahwa rute yang terdekat adalah melalui Jl. Dr Cipto dan aplikasi juga menunjukkan rute yang sama, hasil pengujian yang dilakukan dapat dilihat pada Gambar 5.11.

### 5.2.3.2 Pengujian Eksperimental Bandung 1

Pengujian dilakukan dengan melihat waktu *load* “bandung1.xml” (ukuran *file* sebesar 98Kb), hasil pengujian dapat dilihat pada Gambar 5.12. Pada Gambar 5.12 menunjukkan total waktu yang dibutuhkan untuk melakukan *load* “bandung1.xml” adalah 7,710ms. Berikut ini adalah aplikasi yang menggunakan dokumen “bandung1.xml” dapat dilihat pada Gambar 5.13.

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 5.14 ditemukan kesalahan 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 ditemukan pada dokumen OSMXML



The screenshot shows a browser's developer tools console with the 'Console' tab selected. At the top, there are tabs for Elements, Network, Sources, Timeline, Profiles, Resources, Audits, and Console. A warning message is displayed: "⚠ 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/.". Below this, the search results are shown:

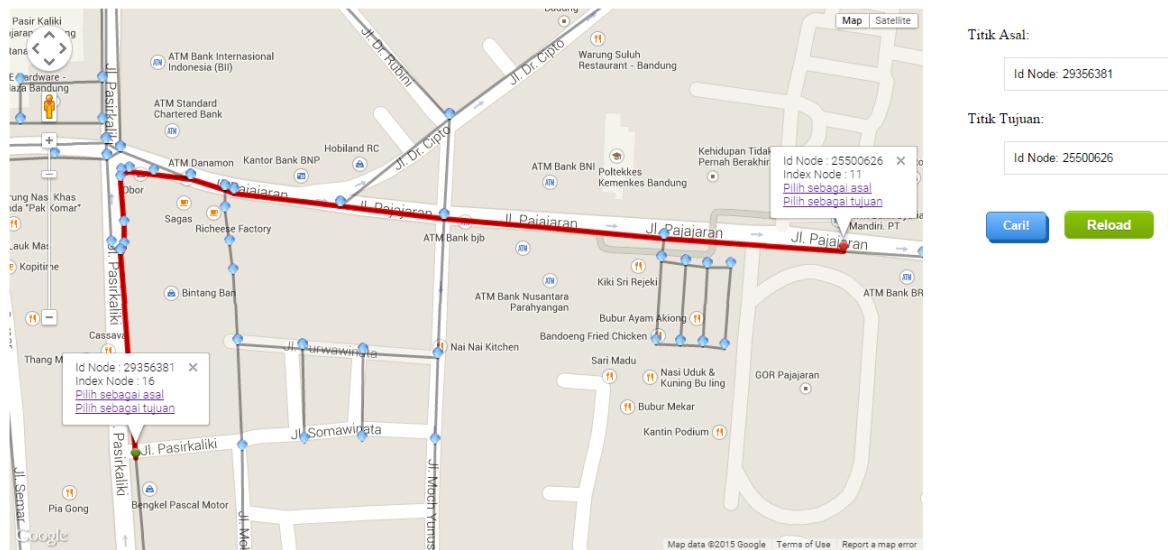
```
Titik Asal: 16
Titik Tujuan: 11
▼ Array[15] ⓘ
  0: "16"
  1: "182"
  2: "145"
  3: "45"
  4: "141"
  5: "144"
  6: "143"
  7: "138"
  8: "142"
  9: "206"
  10: "147"
  11: "1"
  12: "134"
  13: "82"
  14: "11"
  length: 15
```

Gambar 5.9: Pengujian Rute Terdekat

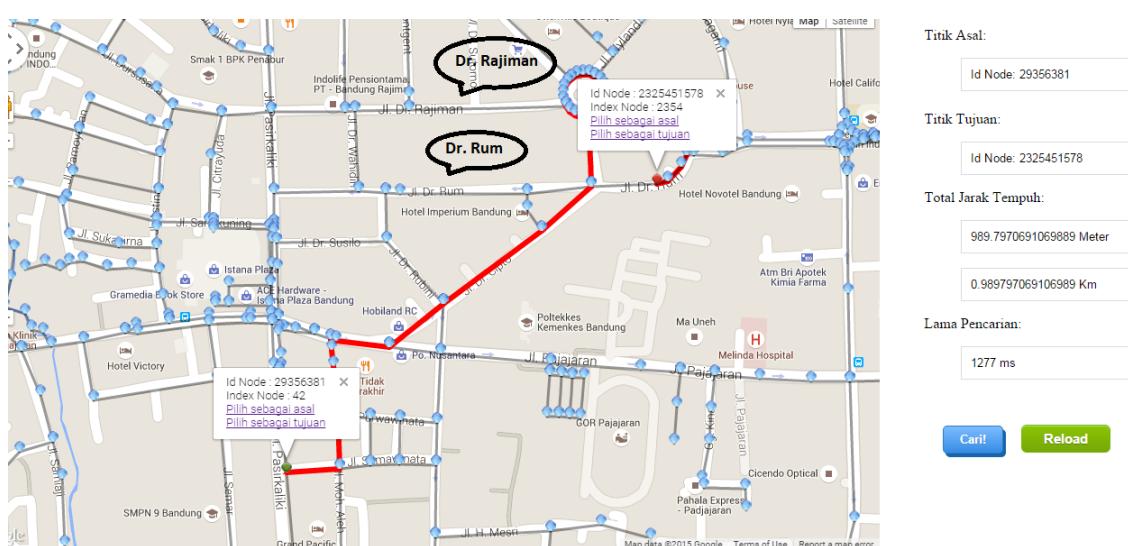
yaitu “bandung1.xml” yang tidak memberikan informasi “oneway”, sehingga aplikasi memasukkan informasi yang salah tersebut ke dalam graf sebagai jalan dua arah. Berikut ini adalah potongan dokumen “bandung1.xml” yang memberikan informasi Jl. Dr Rum, dapat dilihat pada Gambar 5.15

### 5.2.3.3 Pengujian Eksperimental Bandung 2

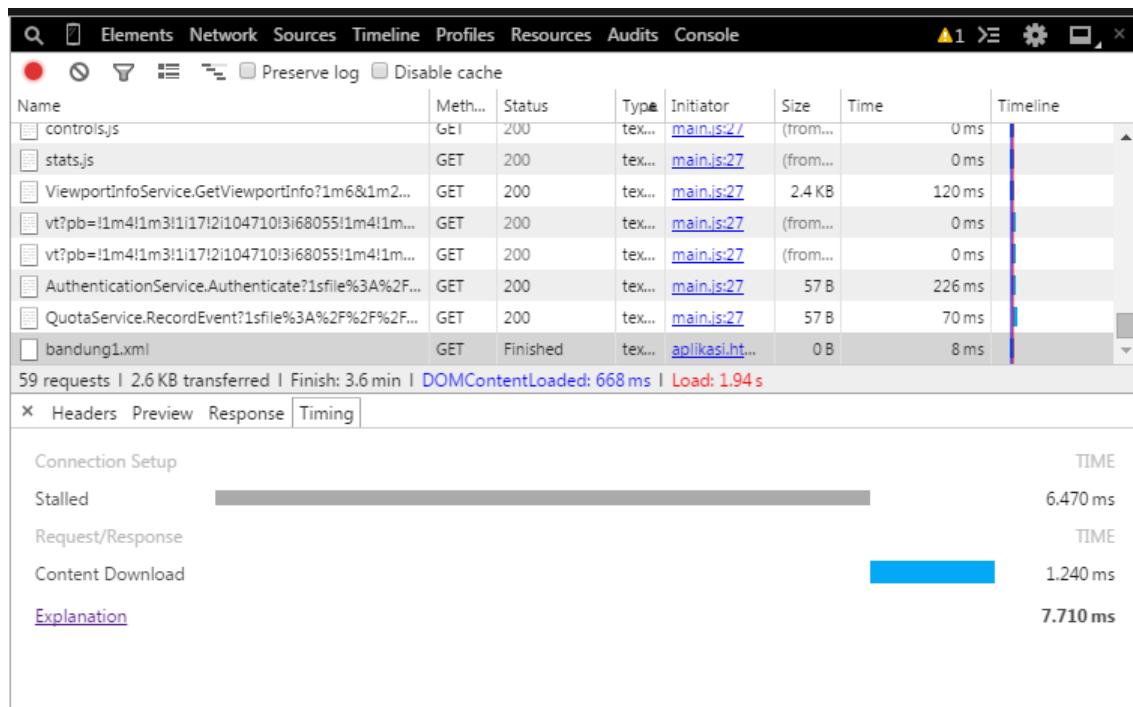
Pengujian dilakukan dengan melihat waktu *load* “bandung2.xml” (ukuran file sebesar 821Kb), hasil pengujian dapat dilihat pada Gambar 5.16. Pada Gambar 5.16 menunjukkan total waktu yang dibutuhkan untuk melakukan *load* “bandung2.xml” adalah 70,460ms. Berikut ini adalah aplikasi yang menggunakan dokumen “bandung2.xml” dapat dilihat pada Gambar 5.17. Setelah melakukan *load* aplikasi menggunakan “bandung2.xml”, pengujian dilanjutkan dengan mencari rute terdekat dari titik asal (Id Node : 29356503, Index Node : 47) ke titik tujuan (Id Node : 1700554920, Index Node : 1163). Hasil pengujian dapat dilihat pada Gambar 5.18.



Gambar 5.10: Pengujian Visualisasi Rute Terdekat



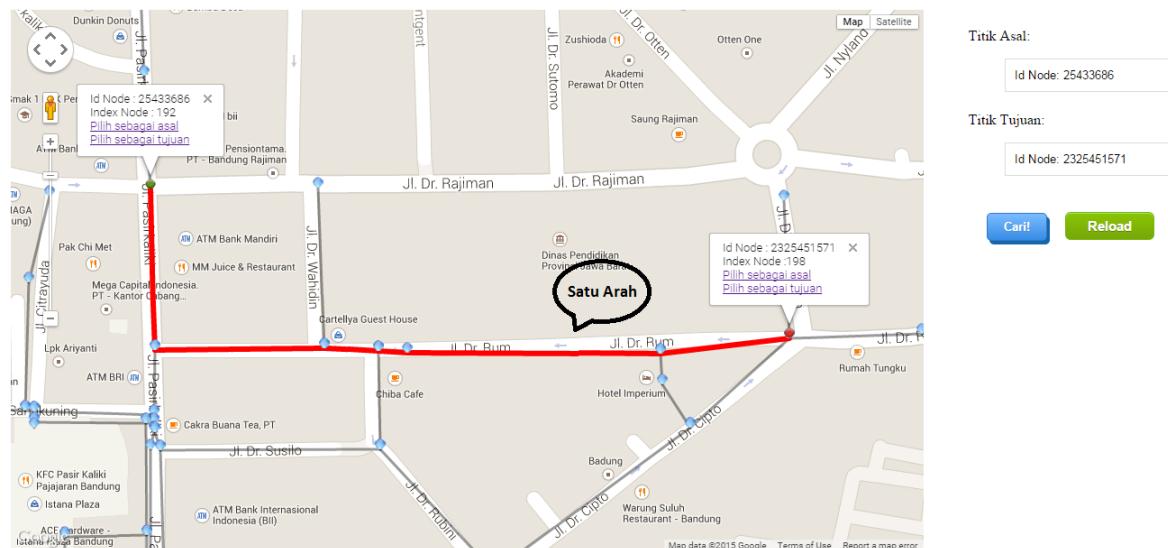
Gambar 5.11: Pengujian Rute Terdekat



Gambar 5.12: Pengujian Bandung 1



Gambar 5.13: Pengujian Bandung 1



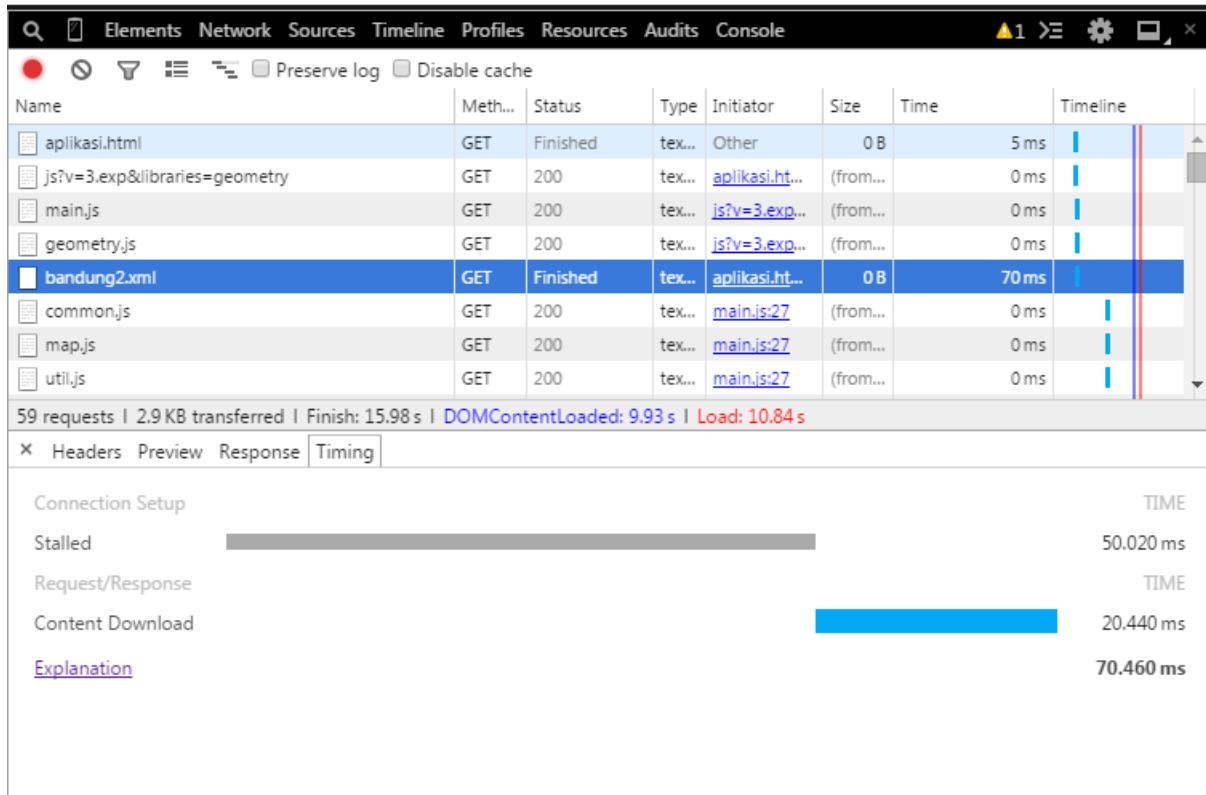
Gambar 5.14: Pengujian Bandung 1

```

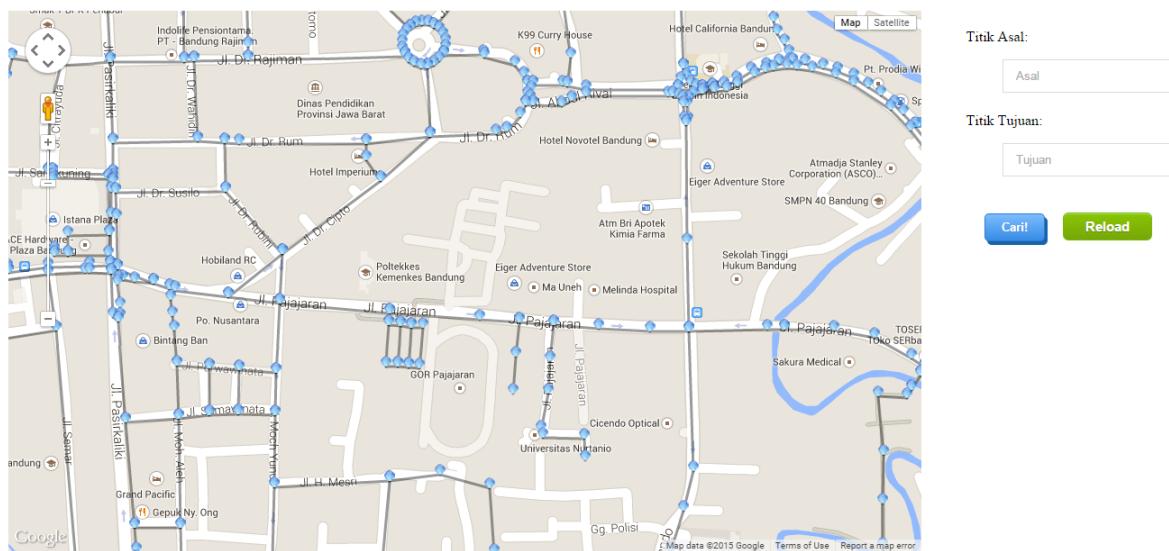
▼<way id="4567503" visible="true" version="6" changeset="27916265" timestamp="2015-01-04T18:11:06Z" user="isonpurba" uid="2552445">
  <nd ref="25433685"/>
  <nd ref="25433679"/>
  <nd ref="25433682"/>
  <nd ref="32041794"/>
  <nd ref="29392373"/>
  <nd ref="2325451571"/>
  <tag k="avgspeed" v="30"/>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

```

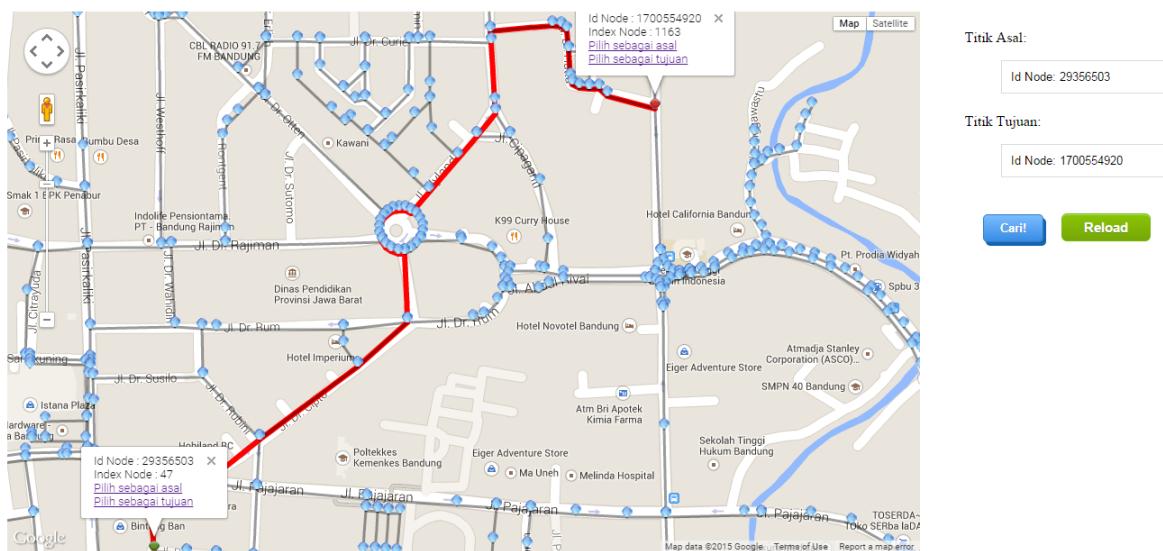
Gambar 5.15: bandung1.xml



Gambar 5.16: Pengujian Bandung 2



Gambar 5.17: Pengujian Bandung 2



Gambar 5.18: Pengujian Bandung 2

#### 5.2.3.4 Pengujian Eksperimental Bandung 3

Pengujian dilakukan dengan melihat waktu *load* "bandung3.xml" (ukuran file sebesar 17.631Kb). Dokumen "bandung3.xml" adalah OSMXML yang mencakup seluruh wilayah Kota Bandung, berdasarkan pengujian yang telah dilakukan, aplikasi tidak berhasil untuk menggunakan "bandung3.xml". Hal ini disebabkan oleh dokumen "bandung3.xml" yang terlalu besar dan waktu *load* yang terlalu lama.

#### 5.2.3.5 Hasil Pengujian

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

1. Aplikasi telah menunjukkan rute yang benar-benar terdekat.

2. *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	-	-	-

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

3. Kesalahan

Kesalahan ditemukan ketika pengujian berlangsung, yaitu rute terdekat yang ditunjukkan

oleh aplikasi melawan arus jalan, hal ini disebabkan oleh dokumen OSMXML yang tidak memberikan informasi “oneway”.



# BAB 6

## KESIMPULAN DAN SARAN

### 6.1 Kesimpulan

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

### 6.2 Saran

1. 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.
2. Informasi yang didapatkan dari OSMXML dapat disimpan terlebih dahulu pada basis data atau file teks, sehingga kecepatan proses lebih cepat.



## DAFTAR REFERENSI

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



# LAMPIRAN A

## KODE PROGRAM

Listing A.1: aplikasi.html

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

```

68     background:-moz-linear-gradient(top, #89c403 5%, #77a809 100%);  

69     background:-webkit-linear-gradient(top, #89c403 5%, #77a809 100%);  

70     background:-o-linear-gradient(top, #89c403 5%, #77a809 100%);  

71     background:-ms-linear-gradient(top, #89c403 5%, #77a809 100%);  

72     background:linear-gradient(to bottom, #89c403 5%, #77a809 100%);  

73     filter:progid:DXImageTransform.Microsoft.gradient(startColorstr='#89c403', endColorstr='#77a809', GradientType=0);  

74     background-color:#89c403;  

75     -moz-border-radius:6px;  

76     -webkit-border-radius:6px;  

77     border-radius:6px;  

78     border:1px solid #74b807;  

79     display:inline-block;  

80     cursor:pointer;  

81     color:#ffffff;  

82     font-family:arial;  

83     font-size:15px;  

84     font-weight:bold;  

85     padding:6px 24px;  

86     text-decoration:none;  

87     text-shadow:0px 1px 0px #528009;  

88 }  

89 .reload:hover {  

90     background:-webkit-gradient(linear, left top, left bottom, color-stop(0.05, #77a809), color-stop(1, #89c403));  

91     background:-moz-linear-gradient(top, #77a809 5%, #89c403 100%);  

92     background:-o-linear-gradient(top, #77a809 5%, #89c403 100%);  

93     background:-ms-linear-gradient(top, #77a809 5%, #89c403 100%);  

94     background:linear-gradient(to bottom, #77a809 5%, #89c403 100%);  

95     filter:progid:DXImageTransform.Microsoft.gradient(startColorstr='#77a809', endColorstr='#89c403', GradientType=0);  

96     background-color:#77a809;  

97 }  

98 }  

99 }  

100 .reload:active {  

101     position:relative;  

102     top:1px;  

103 }  

104 </style>  

105 <body>  

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

107 <div id="cari" style="width:25%;height:60px;float:left">  

108 <div style="margin-left:50px;margin-top:20px">  

109     <table style="border:1">  

110         <tr>  

111             <label>Titik Asal:</label>  

112         </tr>  

113         <tr>  

114             <ul class="input-list-style-1 clearfix">  

115                 <input type="text" id="asal" placeholder="Asal">  

116             </ul>  

117         </tr>  

118     </table>  

119 </div>  

120 <div style="margin-left:50px">  

121 <table>  

122     <tr>  

123         <label>Titik Tujuan:</label>  

124     </tr>  

125     <tr>  

126         <ul class="input-list-style-1 clearfix">  

127             <input type="text" id="tujuan" placeholder="Tujuan">  

128         </ul>  

129     </tr>  

130 </table>  

131 </div>  

132 <br/>  

133 <div style="margin-left:70px">  

134     <a href="#" onclick="result()" class="cari">Cari!</a> &nbsp &nbsp  

135     <a href="javascript:location.reload(true)" class="reload">Reload</a>  

136 </div>  

137 </div>  

138 </div>  

139 <script>  

140 //Load XML  

141 xmlhttp=new XMLHttpRequest();  

142 xmlhttp.open("GET","bandung2.xml",false);  

143 xmlhttp.send();  

144 xmlDoc=xmlhttp.responseXML;  

145  

146 // Adjacency List  

147 //Kelas Neighbor  

148 function Neighbor(vnum, nbr, weight){  

149     this.vertexNum = vnum;  

150     this.next = nbr;  

151     this.weight = weight;  

152 }  

153  

154 //Kelas Node  

155 function Node(id, neighbors){  

156     this.id = id;  

157     this.adjList = neighbors;  

158 }  

159  

160 //Kelas Graph  

161 function Graph(node,way){  

162     var adjLists = [];  

163     var nd;

```

```

164|     var oneway;
165|
166|     function wayDirection(way, index){
167|         var tag = way[index].getElementsByTagName("tag");
168|         for (hg=0;hg<tag.length;hg++)
169|         {
170|             if (tag[hg].getAttribute('k') == "oneway"){
171|                 return tag[hg].getAttribute('v');
172|             }
173|         }
174|         return false;
175|     }
176|
177|     function isHighway(way, index){
178|         var tag = way[index].getElementsByTagName("tag");
179|         for (hg=0;hg<tag.length;hg++)
180|         {
181|             if (tag[hg].getAttribute('k') == "highway"){
182|                 return true;
183|             }
184|         }
185|         return false;
186|     }
187|
188|     function indexForId(adjLists, id) {
189|         for (var i=0; i < adjLists.length; i++){
190|             if (adjLists[i].id == id){
191|                 return i;
192|             }
193|         }
194|         return -1;
195|     }
196|
197|     function getLatByAtt(str)
198|     {
199|         for (n=0;n<node.length;n++)
200|         {
201|             if (node[n].getAttribute('id') == str)
202|             {
203|                 return node[n].getAttribute('lat');
204|             }
205|         }
206|     }
207|
208|     function getLonByAtt(str)
209|     {
210|         for (m=0;m<node.length;m++)
211|         {
212|             if (node[m].getAttribute('id') == str)
213|             {
214|                 return node[m].getAttribute('lon');
215|             }
216|         }
217|     }
218|
219|     for(v=0; v < node.length; v++){
220|         adjLists.push(new Node(node[v].getAttribute('id'), null));
221|     }
222|
223|     var lat1, lon1, lat2, lon2;
224|     var point1, point2, distance;
225|
226|     for (i=0;i<way.length; i++)
227|     {
228|         nd = way[i].getElementsByTagName("nd");
229|         if (isHighway(way, i)){
230|             oneway = wayDirection(way, i);
231|             for (j=0;j<nd.length-1;j++)
232|             {
233|                 v1 = indexForId(adjLists, nd[j].getAttribute('ref'));
234|                 v2 = indexForId(adjLists, nd[j+1].getAttribute('ref'));
235|
236|                 lat1 = getLatByAtt(nd[j].getAttribute('ref'));
237|                 lon1 = getLonByAtt(nd[j].getAttribute('ref'));
238|                 lat2 = getLatByAtt(nd[j+1].getAttribute('ref'));
239|                 lon2 = getLonByAtt(nd[j+1].getAttribute('ref'));
240|
241|                 point1 = new google.maps.LatLng(lat1, lon1);
242|                 point2 = new google.maps.LatLng(lat2, lon2);
243|                 distance = google.maps.geometry.spherical.computeDistanceBetween(point1, point2);
244|
245|                 if (oneway == "yes"){
246|                     adjLists[v1].adjList = new Neighbor(v2, adjLists[v1].adjList, distance);
247|                 } else if (oneway == "no"){
248|                     adjLists[v1].adjList = new Neighbor(v2, adjLists[v1].adjList, distance);
249|                     adjLists[v2].adjList = new Neighbor(v1, adjLists[v2].adjList, distance);
250|                 } else if (oneway == "-1"){
251|                     adjLists[v2].adjList = new Neighbor(v1, adjLists[v2].adjList, distance);
252|                 } else{
253|                     adjLists[v1].adjList = new Neighbor(v2, adjLists[v1].adjList, distance);
254|                     adjLists[v2].adjList = new Neighbor(v1, adjLists[v2].adjList, distance);
255|                 }
256|             }
257|         }
258|     }
259|     return adjLists;
260| }
261| // Kelas Map

```





```

453
454     for(neighbor in this.nodes[smallest]){
455         alt = distances[smallest] + this.nodes[smallest][neighbor];
456         if(alt < distances[neighbor]) {
457             distances[neighbor] = alt;
458             previous[neighbor] = smallest;
459             PQ.enqueue(alt , neighbor);
460         }
461     }
462 }
463 return path;
464 }
465 }
466
467
468 var node = xmlDoc.getElementsByTagName("node");
469 var way = xmlDoc.getElementsByTagName("way");
470
471 var list = new Graph(node,way);
472 var peta = new Map();
473 var mark = peta.generate();
474
475 // Fungsi SetAsal dan Tujuan
476 var isAsalClicked = false;
477 var isTujuanClicked = false;
478 var idAsal, idTujuan, indexAsal, indexTujuan;
479
480 function setAsal(id,idnode,indexNode){
481     if(!isAsalClicked){
482         mark[id].setIcon('icon/dot_green.png');
483         isAsalClicked = true;
484         idAsal = id;
485         indexAsal = indexNode;
486     } else{
487         mark[idAsal].setIcon('icon/dot_blue.png');
488         mark[id].setIcon('icon/dot_green.png');
489         idAsal = id;
490         indexAsal = indexNode;
491     }
492     document.getElementById('asal').value = "Id_Node:"+ idnode;
493 }
494
495 function setTujuan(id,idnode,indexNode){
496     if(!isTujuanClicked){
497         mark[id].setIcon('icon/dot_red.png');
498         isTujuanClicked = true;
499         idTujuan = id;
500         indexTujuan = indexNode;
501     } else{
502         mark[idTujuan].setIcon('icon/dot_blue.png');
503         mark[id].setIcon('icon/dot_red.png');
504         idTujuan = id;
505         indexTujuan = indexNode;
506     }
507     document.getElementById('tujuan').value = "Id_Node:"+ idnode;
508 }
509
510 function result(){
511     var cari = new Dijkstra();
512     var rute;
513     var edge;
514     for(j=0; j < list.length; j++){
515         edge = new Object();
516         for(nbr=list[j].adjList; nbr != null; nbr=nbr.next) {
517             edge[nbr.vertexNum] = nbr.weight;
518         }
519         cari.addNode(j,edge);
520     }
521
522     console.log("Titik_Asal:"+indexAsal);
523     console.log("Titik_Tujuan:"+indexTujuan);
524     rute = cari.shortestPath(indexAsal,indexTujuan).concat([indexAsal]).reverse();
525     console.log(rute);
526     if(rute.length == 1){
527         if(indexAsal == null){
528             alert("Anda belum memilih titik_Asal");
529         } else if(indexTujuan == null){
530             alert("Anda belum memilih titik_Tujuan");
531         } else if(indexAsal != null && indexTujuan != null){
532             alert("Rute Tidak Ditemukan");
533         }
534     } else{
535         peta.hasilCari(rute);
536     }
537 }
538
539 </script>
540 </body>
541 </html>

```

Listing A.2: list.html

```

1<!DOCTYPE html>
2<html>
3<head>
4<script src="https://maps.googleapis.com/maps/api/js?v=3.exp&libraries=geometry"></script>
5</head>
6<body>

```

```

7<script>
8 xmlhttp=new XMLHttpRequest();
9 xmlhttp.open("GET","test.xml",false);
10 xmlhttp.send();
11 xmlDoc=xmlhttp.responseXML;
12
13 function Neighbor(vnum, nbr, weight){
14     this.vertexNum = vnum;
15     this.weight = weight;
16     this.next = nbr;
17 }
18
19 function Node(id, neighbors){
20     this.id = id;
21     this.adjList = neighbors;
22 }
23
24 function isOneway(way, index){
25     var tag = way[index].getElementsByTagName("tag");
26     for (hg=0;hg<tag.length;hg++)
27     {
28         if (tag[hg].getAttribute('k') == "oneway"){
29             return tag[hg].getAttribute('v');
30         }
31     }
32     return false;
33 }
34
35 function isHighway(way, index){
36     var tag = way[index].getElementsByTagName("tag");
37     for (hg=0;hg<tag.length;hg++)
38     {
39         if (tag[hg].getAttribute('k') == "highway"){
40             return true;
41         }
42     }
43     return false;
44 }
45
46 function indexForId(adjLists ,id) {
47     for (var i=0; i < adjLists.length; i++){
48         if (adjLists[i].id == id){
49             return i;
50         }
51     }
52     return -1;
53 }
54
55 function getLatByAtt(str)
56 {
57     for (n=0;n<node.length;n++)
58     {
59         if (node[n].getAttribute('id') == str)
60         {
61             return node[n].getAttribute('lat');
62         }
63     }
64 }
65 function getLonByAtt(str)
66 {
67     for (m=0;m<node.length;m++)
68     {
69         if (node[m].getAttribute('id') == str)
70         {
71             return node[m].getAttribute('lon');
72         }
73     }
74 }
75
76 function Graph(node,way){
77     var adjLists = [];
78     var nd;
79     var oneway;
80     var lat1,lon1,lat2,lon2;
81     var point1,point2,distance;
82
83     for(v=0; v < node.length; v++){
84         adjLists.push(new Node(node[v].getAttribute('id'),null));
85     }
86
87     for (i=0;i<way.length ;i++)
88     {
89         nd = way[i].getElementsByTagName("nd");
90         if (isHighway(way,i)){
91             oneway = isOneway(way,i);
92             for (j=0;j<nd.length -1;j++)
93             {
94                 v1 = indexForId(adjLists,nd[j].getAttribute('ref'));
95                 v2 = indexForId(adjLists,nd[j+1].getAttribute('ref'));
96
97                 lat1 = getLatByAtt(nd[j].getAttribute('ref'));
98                 lon1 = getLonByAtt(nd[j].getAttribute('ref'));
99                 lat2 = getLatByAtt(nd[j+1].getAttribute('ref'));
100                lon2 = getLonByAtt(nd[j+1].getAttribute('ref'));
101
102                point1 = new google.maps.LatLng(lat1,lon1);
103                point2 = new google.maps.LatLng(lat2,lon2);
104                distance = google.maps.geometry.spherical.computeDistanceBetween(point1, point2);
105            }
106        }
107    }
108 }
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2158
2159
2160
2161
2162
2163
2164
2165
2166

```

```

106     if (oneway == "yes"){
107         adjLists[v1].adjList = new Neighbor(v2, adjLists[v1].adjList, distance);
108     } else if (oneway == "no"){
109         adjLists[v1].adjList = new Neighbor(v2, adjLists[v1].adjList, distance);
110         adjLists[v2].adjList = new Neighbor(v1, adjLists[v2].adjList, distance);
111     } else if (oneway == "-1"){
112         adjLists[v2].adjList = new Neighbor(v1, adjLists[v2].adjList, distance);
113     } else{
114         adjLists[v1].adjList = new Neighbor(v2, adjLists[v1].adjList, distance);
115         adjLists[v2].adjList = new Neighbor(v1, adjLists[v2].adjList, distance);
116     }
117 }
118 }
119 return adjLists;
120 }
121 }
122
123 function print(list){
124     for (j=0; j < list.length; j++){
125         document.write(j);
126         for (nbr=list[j].adjList; nbr != null; nbr=nbr.next) {
127             document.write("---->");
128             document.write('(' + nbr.vertexNum + ')');
129         }
130         document.write("<br>");
131     }
132 }
133
134 function print2(list){
135     for (j=0; j < list.length; j++){
136         document.write(j);
137         document.write("---->");
138         document.write(list[j].id);
139         for (nbr=list[j].adjList; nbr != null; nbr=nbr.next) {
140             document.write("---->");
141             document.write(list[nbr.vertexNum].id);
142         }
143         document.write("<br>");
144     }
145 }
146
147 var node = xmlDoc.getElementsByTagName("node");
148 var way = xmlDoc.getElementsByTagName("way");
149
150 var list = new Graph(node, way);
151 print(list);
152 //print2(list);
153
154
155</script>
156</body>
157</html>
```

Listing A.3: xml\_parsing.html

```

1<html>
2<head>
3<script src="https://maps.googleapis.com/maps/api/js?v=3.exp&libraries=geometry"></script>
4<style>
5table, th, td {
6    border: 1px solid black;
7    border-collapse: collapse;
8}
9th, td {
10    padding: 5px;
11}
12</style>
13</head>
14<body>
15<script>
16xmlhttp=new XMLHttpRequest();
17xmlhttp.open("GET","test.xml",false);
18xmlhttp.send();
19xmlDoc=xmlhttp.responseXML;
20
21function getDistance(lat1,lon1,lat2,lon2) {
22    var R = 6371; // Radius of the earth in km
23    var dLat = deg2rad(lat2-lat1); // deg2rad below
24    var dLon = deg2rad(lon2-lon1);
25    var a =
26        Math.sin(dLat/2) * Math.sin(dLat/2) +
27        Math.cos(deg2rad(lat1)) * Math.cos(deg2rad(lat2)) *
28        Math.sin(dLon/2) * Math.sin(dLon/2)
29    ;
30    var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
31    var d = R * c; // Distance in km
32    return d;
33}
34
35function deg2rad(deg) {
36    return deg * (Math.PI/180)
37}
38
39document.write("<div style='float:left'>");
40document.write("<table><tr><th>Node</th><th>Id</th><th>Latitude</th><th>Longitude</th></tr>");
41document.write("<caption>Node</caption>");
42var node=xmlDoc.getElementsByTagName("node");
43for (ct=0;ct<node.length;ct++)
```

```

44| {
45|   document.write("<tr><td>");
46|   document.write(ct);
47|   document.write("</td><td>");
48|   document.write(node[ct].getAttribute('id'));
49|   document.write("</td><td>");
50|   document.write(node[ct].getAttribute('lat'));
51|   document.write("</td><td>");
52|   document.write(node[ct].getAttribute('lon'));
53|   document.write("</td></tr>");
54| }
55| document.write("</table>");
56| document.write("</div>");
57|
58| function isHighway(way, index){
59|   var tag = way[index].getElementsByTagName("tag");
60|   for (hg=0;hg<tag.length;hg++)
61|   {
62|     if (tag[hg].getAttribute('k') == "highway"){
63|       return true;
64|     }
65|   }
66|   return false;
67| }
68|
69| function getLatByAtt(str)
70| {
71|   for (n=0;n<node.length;n++)
72|   {
73|     if (node[n].getAttribute('id') == str)
74|     {
75|       return node[n].getAttribute('lat');
76|     }
77|   }
78|   return -1;
79| }
80| function getLonByAtt(str)
81| {
82|   for (m=0;m<node.length;m++)
83|   {
84|     if (node[m].getAttribute('id') == str)
85|     {
86|       return node[m].getAttribute('lon');
87|     }
88|   }
89|   return -1;
90| }
91|
92| document.write("<div style='margin-left: 20px; float: left;'>");
93| 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>");
94| document.write("<caption>Edge</caption>");
95| var way = xmlDoc.getElementsByTagName("way");
96| var nd;
97| var lat1;
98| var lon1;
99| var lat2;
100| var lon2;
101| for (i=0;i<way.length;i++)
102| {
103|   nd = way[i].getElementsByTagName("nd");
104|   if (isHighway(way, i)){
105|     for (j=0;j<nd.length-1;j++)
106|     {
107|       document.write("<tr><td>");
108|       document.write(i);
109|       document.write("</td><td>");
110|       document.write(way[i].getAttribute('id'));
111|       document.write("</td><td>");
112|       document.write(j);
113|       document.write("</td><td>");
114|       document.write(nd[j].getAttribute('ref'));
115|       document.write("</td><td>");
116|       document.write(nd[j+1].getAttribute('ref'));
117|       document.write("</td><td>");
118|       lat1 = getLatByAtt(nd[j].getAttribute('ref'));
119|       lon1 = getLonByAtt(nd[j].getAttribute('ref'));
120|       lat2 = getLatByAtt(nd[j+1].getAttribute('ref'));
121|       lon2 = getLonByAtt(nd[j+1].getAttribute('ref'));
122|
123|       point1 = new google.maps.LatLng(lat1,lon1);
124|       point2 = new google.maps.LatLng(lat2,lon2);
125|       distance = google.maps.geometry.spherical.computeDistanceBetween(point1, point2);
126|       //document.write(getDistance(lat1,lon1,lat2,lon2)*1000);
127|       document.write(distance);
128|       document.write("</td></tr>");
129|     }
130|   }
131| }
132| document.write("</div>");
133|
134| </script>
135| </body>
136| </html>
```



# LAMPIRAN B

## BANDUNG 1 OSMXML

Listing B.1: bandung1.xml

```

1<?xml version="1.0" encoding="UTF-8"?>
2<osm version="0.6" generator="CGImap_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/"/>
3<bounds minlat="-6.9076500" minlon="107.5961800" maxlat="-6.9044500" maxlon="107.6016300"/>
4<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"/>
5<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"/>
6<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"/>
7<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"/>
8<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"/>
9<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"/>
10<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"/>
11<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"/>
12<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"/>
13<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"/>
14<node id="25434115" visible="true" version="2" changeset="839915" timestamp="2009-03-21T14:19:29Z" user=
  ="adhitya" uid="7748" lat="-6.9097812" lon="107.5975805"/>
15<node id="25500626" visible="true" version="3" changeset="839915" timestamp="2009-03-21T14:22:17Z" user=
  ="adhitya" uid="7748" lat="-6.9070329" lon="107.6019401"/>
16<node id="28802396" visible="true" version="2" changeset="839915" timestamp="2009-03-21T14:23:18Z" user=
  ="adhitya" uid="7748" lat="-6.9055299" lon="107.5976092">
17<tag k="created_by" v="YahooApplet_1.0"/>
18</node>
19<node id="29356177" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:47:33Z" user=
  ="adhitya" uid="7748" lat="-6.9102898" lon="107.5994584"/>
20<node id="29356178" visible="true" version="2" changeset="11905099" timestamp="2012-06-15T14:05:57Z" user=
  ="andryono" uid="643030" lat="-6.9090157" lon="107.5994925"/>
21<node id="29356374" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:50:18Z" user=
  ="adhitya" uid="7748" lat="-6.9081596" lon="107.5987289"/>
22<node id="29356381" visible="true" version="2" changeset="27915880" timestamp="2015-01-04T17:57:25Z" user=
  ="isonpurba" uid="2552445" lat="-6.9082496" lon="107.5977288"/>
23<node id="29356499" visible="true" version="2" changeset="839915" timestamp="2009-03-21T14:20:07Z" user=
  ="adhitya" uid="7748" lat="-6.9099650" lon="107.5978505"/>
24<node id="29356500" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:49:27Z" user=
  ="adhitya" uid="7748" lat="-6.9099148" lon="107.5983941"/>
25<node id="29356501" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:49:27Z" user=
  ="adhitya" uid="7748" lat="-6.9094973" lon="107.5983770"/>
26<node id="29356502" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:50:18Z" user=
  ="adhitya" uid="7748" lat="-6.9082022" lon="107.5983598"/>
27<node id="29356503" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:49:27Z" user=
  ="adhitya" uid="7748" lat="-6.9075802" lon="107.5983340"/>
28<node id="29356504" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:49:27Z" user=
  ="adhitya" uid="7748" lat="-6.9071626" lon="107.5983083"/>
29<node id="29356528" visible="true" version="2" changeset="839915" timestamp="2009-03-21T14:20:07Z" user=
  ="adhitya" uid="7748" lat="-6.9094946" lon="107.5978334"/>
30<node id="29356534" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:49:55Z" user=
  ="adhitya" uid="7748" lat="-6.9076313" lon="107.5990808"/>
31<node id="29356535" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:49:55Z" user=
  ="adhitya" uid="7748" lat="-6.9076057" lon="107.5987203"/>
32<node id="29356540" visible="true" version="1" changeset="57478" timestamp="2007-05-19T17:50:15Z" user=
  ="adhitya" uid="7748" lat="-6.9081511" lon="107.5990722"/>
33<node id="29376827" visible="true" version="1" changeset="58597" timestamp="2007-05-20T11:57:31Z" user=
  ="adhitya" uid="7748" lat="-6.9052966" lon="107.5968063"/>
34<node id="29376829" visible="true" version="1" changeset="58597" timestamp="2007-05-20T11:57:31Z" user=
  ="adhitya" uid="7748" lat="-6.9045382" lon="107.5968234"/>
35<node id="29376832" visible="true" version="1" changeset="58597" timestamp="2007-05-20T11:57:15Z" user=
  ="adhitya" uid="7748" lat="-6.9047342" lon="107.5951840"/>
36<node id="29376835" visible="true" version="1" changeset="58597" timestamp="2007-05-20T11:57:15Z" user=
  ="adhitya" uid="7748" lat="-6.9054755" lon="107.5951411"/>
37<node id="29376836" visible="true" version="2" changeset="839915" timestamp="2009-03-21T14:21:11Z" user=
  ="adhitya" uid="7748" lat="-6.9055948" lon="107.5961854"/>
38<node id="29376837" visible="true" version="1" changeset="58597" timestamp="2007-05-20T11:57:12Z" user=
  ="adhitya" uid="7748" lat="-6.9055863" lon="107.5957334"/>
```

```

39 <node id="29376839" visible="true" version="1" changeset="58597" timestamp="2007-05-20T11:56:55Z" user="adhitya" uid="7748" lat="-6.9049472" lon="107.5962054"/>
40 <node id="29376842" visible="true" version="1" changeset="58597" timestamp="2007-05-20T11:57:47Z" user="adhitya" uid="7748" lat="-6.9073927" lon="107.5954330"/>
41 <node id="29376841" visible="true" version="2" changeset="12686463" timestamp="2012-08-11T00:36:04Z" user="ferdyodin" uid="796044" lat="-6.9071561" lon="107.5968980"/>
42 <node id="29392373" visible="true" version="2" changeset="839915" timestamp="2009-03-21T14:19:03Z" user="adhitya" uid="7748" lat="-6.9049590" lon="107.6005861"/>
43 <node id="29392374" visible="true" version="1" changeset="147858" timestamp="2007-07-16T21:40:29Z" user="adhitya" uid="7748" lat="-6.9051439" lon="107.6005958">
44 <tag k="name" v="Imperium"/>
45 <tag k="tourism" v="hotel"/>
46 </node>
47 <node id="29392377" visible="true" version="2" changeset="839915" timestamp="2009-03-21T14:19:03Z" user="adhitya" uid="7748" lat="-6.9053946" lon="107.6007639"/>
48 <node id="29392413" visible="true" version="3" changeset="839915" timestamp="2009-03-21T14:17:04Z" user="adhitya" uid="7748" lat="-6.9071747" lon="107.6039686"/>
49 <node id="32041785" visible="true" version="1" changeset="147858" timestamp="2007-07-16T21:20:58Z" user="adhitya" uid="7748" lat="-6.9045608" lon="107.5990459">
50 <tag k="amenity" v="restaurant"/>
51 <tag k="created_by" v="JOSM"/>
52 <tag k="name" v="Chiba"/>
53 </node>
54 <node id="32041794" visible="true" version="2" changeset="839915" timestamp="2009-03-21T14:20:08Z" user="adhitya" uid="7748" lat="-6.9049540" lon="107.5990764">
55 <tag k="created_by" v="JOSM"/>
56 </node>
57 <node id="32041828" visible="true" version="2" changeset="13856395" timestamp="2012-11-13T08:27:46Z" user="yudiwbs" uid="268765" lat="-6.9028198" lon="107.5974882"/>
58 <node id="32042055" visible="true" version="1" changeset="147858" timestamp="2007-07-16T21:25:13Z" user="adhitya" uid="7748" lat="-6.9045275" lon="107.5973424">
59 <tag k="amenity" v="place_of_worship"/>
60 <tag k="created_by" v="JOSM"/>
61 <tag k="name" v="GBIS_Harmoni"/>
62 <tag k="religion" v="christian"/>
63 </node>
64 <node id="32042131" visible="true" version="1" changeset="147858" timestamp="2007-07-16T21:25:27Z" user="adhitya" uid="7748" lat="-6.9050969" lon="107.5973817">
65 <tag k="amenity" v="university"/>
66 <tag k="created_by" v="JOSM"/>
67 <tag k="name" v="Ariyanti"/>
68 </node>
69 <node id="32520365" visible="true" version="3" changeset="13860930" timestamp="2012-11-13T15:42:42Z" user="yudiwbs" uid="268765" lat="-6.9068829" lon="107.5976578"/>
70 <node id="32529146" visible="true" version="3" changeset="839915" timestamp="2009-03-21T14:16:03Z" user="adhitya" uid="7748" lat="-6.9071333" lon="107.6033567">
71 <tag k="created_by" v="JOSM"/>
72 </node>
73 <node id="364363837" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:02:46Z" user="adhitya" uid="7748" lat="-6.9109164" lon="107.5979263"/>
74 <node id="364363838" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:02:48Z" user="adhitya" uid="7748" lat="-6.9103801" lon="107.5978906"/>
75 <node id="364363888" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:09Z" user="adhitya" uid="7748" lat="-6.9055397" lon="107.5965000"/>
76 <node id="364363889" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:10Z" user="adhitya" uid="7748" lat="-6.9056185" lon="107.5965003"/>
77 <node id="364363890" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:10Z" user="adhitya" uid="7748" lat="-6.9056188" lon="107.5964229"/>
78 <node id="364363895" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:14Z" user="adhitya" uid="7748" lat="-6.9057778" lon="107.5964236"/>
79 <node id="364363896" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:15Z" user="adhitya" uid="7748" lat="-6.9057775" lon="107.5964918"/>
80 <node id="364363897" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:17Z" user="adhitya" uid="7748" lat="-6.9061063" lon="107.5964919"/>
81 <node id="364363898" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:18Z" user="adhitya" uid="7748" lat="-6.9061031" lon="107.5966256"/>
82 <node id="364363900" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:21Z" user="adhitya" uid="7748" lat="-6.9062220" lon="107.5966261"/>
83 <node id="364363901" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:23Z" user="adhitya" uid="7748" lat="-6.9062225" lon="107.5969390"/>
84 <node id="364363903" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:27Z" user="adhitya" uid="7748" lat="-6.9059809" lon="107.5969397"/>
85 <node id="364363948" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:31Z" user="adhitya" uid="7748" lat="-6.9059787" lon="107.5974369"/>
86 <node id="364363954" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:32Z" user="adhitya" uid="7748" lat="-6.9054018" lon="107.5974346"/>
87 <node id="364363962" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:33Z" user="adhitya" uid="7748" lat="-6.9054047" lon="107.5967435"/>
88 <node id="364363968" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:03:34Z" user="adhitya" uid="7748" lat="-6.9055386" lon="107.5967441"/>
89 <node id="364364075" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:04:26Z" user="adhitya" uid="7748" lat="-6.9060974" lon="107.5971201"/>
90 <node id="364364076" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:04:28Z" user="adhitya" uid="7748" lat="-6.9062244" lon="107.5971218"/>
91 <node id="364364077" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:04:30Z" user="adhitya" uid="7748" lat="-6.9062210" lon="107.5973845"/>
92 <node id="364364078" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:04:30Z" user="adhitya" uid="7748" lat="-6.9060940" lon="107.5973828"/>
93 <node id="364364087" visible="true" version="2" changeset="18997770" timestamp="2013-11-19T17:08:36Z" user="ubanovic" uid="1784103" lat="-6.9048017" lon="107.6022644"/>
94 <node id="364364086" visible="true" version="2" changeset="12593148" timestamp="2012-08-03T02:44:39Z" user="ivan_elangrawa" uid="771585" lat="-6.9046383" lon="107.6024234"/>
95 <node id="364364182" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:05:38Z" user="adhitya" uid="7748" lat="-6.9053722" lon="107.5975201"/>
96 <node id="364364183" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:05:39Z" user="adhitya" uid="7748" lat="-6.9062774" lon="107.5975276"/>
97 <node id="364364184" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:05:43Z" user="adhitya" uid="7748" lat="-6.9062749" lon="107.5970425"/>
```

```

98 <node id="364364192" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:05:48Z" user=
99   "adhitya" uid="7748" lat="-6.9062745" lon="107.5968821"/>
100 <node id="364364194" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:05:56Z" user=
101   "adhitya" uid="7748" lat="-6.9060423" lon="107.5970375"/>
102 <node id="364364195" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:05:57Z" user=
103   "adhitya" uid="7748" lat="-6.9060323" lon="107.5975301"/>
104 <node id="364364202" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:05:59Z" user=
105   "adhitya" uid="7748" lat="-6.9053647" lon="107.5968575"/>
106 <node id="364364220" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:06:06Z" user=
107   "adhitya" uid="7748" lat="-6.9053972" lon="107.5968575"/>
108 <node id="364364237" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:06:10Z" user=
109   "adhitya" uid="7748" lat="-6.9062223" lon="107.5968807"/>
110 <node id="364364242" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:06:11Z" user=
111   "adhitya" uid="7748" lat="-6.9053002" lon="107.5968551"/>
112 <node id="364364247" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:06:11Z" user=
113   "adhitya" uid="7748" lat="-6.9061909" lon="107.5962609"/>
114 <node id="364364251" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:06:12Z" user=
115   "adhitya" uid="7748" lat="-6.9058764" lon="107.5962152"/>
116 <node id="364364179" visible="true" version="2" changeset="13860930" timestamp="2012-11-13T15:42:43Z"
117   user="yudiwbz" uid="268765" lat="-6.9053684" lon="107.5975736"/>
118 <node id="364365012" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:19Z" user=
119   "adhitya" uid="7748" lat="-6.9069591" lon="107.6008710"/>
120 <node id="364365018" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:20Z" user=
121   "adhitya" uid="7748" lat="-6.9075810" lon="107.6008234"/>
122 <node id="364365025" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:21Z" user=
123   "adhitya" uid="7748" lat="-6.9076078" lon="107.6012281"/>
124 <node id="364365061" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:25Z" user=
125   "adhitya" uid="7748" lat="-6.9071287" lon="107.6012668"/>
126 <node id="364365069" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:25Z" user=
127   "adhitya" uid="7748" lat="-6.9070930" lon="107.6008561"/>
128 <node id="364365077" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:26Z" user=
129   "adhitya" uid="7748" lat="-6.9075869" lon="107.6009692"/>
130 <node id="364365090" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:34Z" user=
131   "adhitya" uid="7748" lat="-6.9071108" lon="107.6009990"/>
132 <node id="364365091" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:34Z" user=
133   "adhitya" uid="7748" lat="-6.9075929" lon="107.6011001"/>
134 <node id="364365096" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:35Z" user=
135   "adhitya" uid="7748" lat="-6.9071257" lon="107.6011269"/>
136 <node id="364365097" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:37Z" user=
137   "adhitya" uid="7748" lat="-6.9071132" lon="107.6004979"/>
138 <node id="364365098" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:41Z" user=
139   "adhitya" uid="7748" lat="-6.9071511" lon="107.6007965"/>
140 <node id="364365099" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:42Z" user=
141   "adhitya" uid="7748" lat="-6.9075492" lon="107.6007585"/>
142 <node id="364365133" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:49Z" user=
143   "adhitya" uid="7748" lat="-6.9075161" lon="107.6004505"/>
144 <node id="364365134" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:51Z" user=
145   "adhitya" uid="7748" lat="-6.9076448" lon="107.6007866"/>
146 <node id="364365135" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:55Z" user=
147   "adhitya" uid="7748" lat="-6.9080087" lon="107.6007623"/>
148 <node id="364365136" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:56Z" user=
149   "adhitya" uid="7748" lat="-6.9080421" lon="107.6012506"/>
150 <node id="364365137" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:07:58Z" user=
151   "adhitya" uid="7748" lat="-6.9076812" lon="107.6012810"/>
152 <node id="364365142" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:02Z" user=
153   "adhitya" uid="7748" lat="-6.9071929" lon="107.6017996"/>
154 <node id="364365143" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:02Z" user=
155   "adhitya" uid="7748" lat="-6.9073415" lon="107.6014084"/>
156 <node id="364365144" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:04Z" user=
157   "adhitya" uid="7748" lat="-6.9083514" lon="107.6013295"/>
158 <node id="364365145" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:05Z" user=
159   "adhitya" uid="7748" lat="-6.9086244" lon="107.6016722"/>
160 <node id="364365146" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:12Z" user=
161   "adhitya" uid="7748" lat="-6.9084728" lon="107.6020513"/>
162 <node id="364365151" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:16Z" user=
163   "adhitya" uid="7748" lat="-6.9074021" lon="107.6021272"/>
164 <node id="364365155" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:17Z" user=
165   "adhitya" uid="7748" lat="-6.9072323" lon="107.6015873"/>
166 <node id="364365159" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:18Z" user=
167   "adhitya" uid="7748" lat="-6.9072656" lon="107.6020028"/>
168 <node id="364365161" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:22Z" user=
169   "adhitya" uid="7748" lat="-6.9085729" lon="107.6018997"/>
170 <node id="364365162" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:23Z" user=
171   "adhitya" uid="7748" lat="-6.9085334" lon="107.6014690"/>
172 <node id="364365164" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:24Z" user=
173   "adhitya" uid="7748" lat="-6.9071929" lon="107.6016935"/>
174 <node id="364365165" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:26Z" user=
175   "adhitya" uid="7748" lat="-6.9072687" lon="107.6014872"/>
176 <node id="364365166" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:26Z" user=
177   "adhitya" uid="7748" lat="-6.9072201" lon="107.6019088"/>
178 <node id="364365167" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:27Z" user=
179   "adhitya" uid="7748" lat="-6.9073202" lon="107.6020726"/>
180 <node id="364365168" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:28Z" user=
181   "adhitya" uid="7748" lat="-6.9085365" lon="107.6019786"/>
182 <node id="364365169" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:30Z" user=
183   "adhitya" uid="7748" lat="-6.9086123" lon="107.6017875"/>
184 <node id="364365170" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:31Z" user=
185   "adhitya" uid="7748" lat="-6.9085911" lon="107.6015600"/>
186 <node id="364365171" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:08:33Z" user=
187   "adhitya" uid="7748" lat="-6.9084788" lon="107.6013811"/>
188 <node id="1666615973" visible="true" version="1" changeset="10914639" timestamp="2012-03-08T22:37:16Z"
189   user="juey" uid="617650" lat="-6.9057580" lon="107.5953036"/>
190 <node id="1666616129" visible="true" version="1" changeset="10914639" timestamp="2012-03-08T22:37:19Z"
191   user="juey" uid="617650" lat="-6.9058491" lon="107.5955873"/>
192 <node id="1666616062" visible="true" version="1" changeset="10914639" timestamp="2012-03-08T22:37:18Z"
193   user="juey" uid="617650" lat="-6.9058491" lon="107.5946111"/>
194 <node id="1666615284" visible="true" version="1" changeset="10914639" timestamp="2012-03-08T22:36:51Z"
195   user="juey" uid="617650" lat="-6.9041841" lon="107.5963120"/>
```

```

147 <node id="1666616021" visible="true" version="1" changeset="10914639" timestamp="2012-03-08T22:37:17Z"
148   user="juey" uid="617650" lat="-6.9057870" lon="107.5948489"/>
149 <node id="1666616044" visible="true" version="1" changeset="10914639" timestamp="2012-03-08T22:37:17Z"
150   user="juey" uid="617650" lat="-6.9058242" lon="107.5958834"/>
151 <node id="1666616103" visible="true" version="1" changeset="10914639" timestamp="2012-03-08T22:37:19Z"
152   user="juey" uid="617650" lat="-6.9058491" lon="107.5950283"/>
153 <node id="1666616003" visible="true" version="1" changeset="10914639" timestamp="2012-03-08T22:37:16Z"
154   user="juey" uid="617650" lat="-6.9057745" lon="107.5951367"/>
155 <node id="1700554868" visible="true" version="1" changeset="11172636" timestamp="2012-04-01T03:16:26Z"
156   user="andryono" uid="643030" lat="-6.9068452" lon="107.5979369"/>
157 <node id="1700554884" visible="true" version="1" changeset="11172636" timestamp="2012-04-01T03:16:27Z"
158   user="andryono" uid="643030" lat="-6.9068613" lon="107.5977407"/>
159 <node id="1700554892" visible="true" version="1" changeset="11172636" timestamp="2012-04-01T03:16:27Z"
160   user="andryono" uid="643030" lat="-6.9067023" lon="107.5977275"/>
161 <node id="1700554901" visible="true" version="1" changeset="11172636" timestamp="2012-04-01T03:16:27Z"
162   user="andryono" uid="643030" lat="-6.9066862" lon="107.5979237"/>
163 <node id="1706564155" visible="true" version="1" changeset="11220464" timestamp="2012-04-08T02:49:00Z"
164   user="andryono" uid="643030" lat="-6.9070955" lon="107.6028239"/>
165 <node id="1706564168" visible="true" version="1" changeset="11220464" timestamp="2012-04-08T02:49:01Z"
166   user="andryono" uid="643030" lat="-6.9070663" lon="107.6024111"/>
167 <node id="25447003" visible="true" version="5" changeset="27916246" timestamp="2015-01-04T18:10:18Z" user
168   ="isonpurba" uid="252445" lat="-6.9040554" lon="107.6013227"/>
169 <node id="29356179" visible="true" version="2" changeset="11905099" timestamp="2012-06-15T14:05:57Z" user
170   ="andryono" uid="643030" lat="-6.9081649" lon="107.5995085"/>
171 <node id="29356180" visible="true" version="2" changeset="11905099" timestamp="2012-06-15T14:05:57Z" user
172   ="andryono" uid="643030" lat="-6.9076627" lon="107.5995236"/>
173 <node id="25433684" visible="true" version="4" changeset="11905099" timestamp="2012-06-15T14:05:57Z" user
174   ="andryono" uid="643030" lat="-6.9068381" lon="107.5995616"/>
175 <node id="1860932973" visible="true" version="1" changeset="12686463" timestamp="2012-08-11T00:36:04Z"
176   user="ferdyodin" uid="796044" lat="-6.9099128" lon="107.5971365"/>
177 <node id="1860932975" visible="true" version="1" changeset="12686463" timestamp="2012-08-11T00:36:04Z"
178   user="ferdyodin" uid="796044" lat="-6.9098721" lon="107.5978507"/>
179 <node id="1860932974" visible="true" version="3" changeset="14069373" timestamp="2012-11-28T07:04:18Z"
180   user="dadan_dany_dipoera" uid="922816" lat="-6.9065003" lon="107.5968009"/>
181 <node id="2012498678" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:42:41Z"
182   user="yudiwbs" uid="268765" lat="-6.9065807" lon="107.5978333"/>
183 <node id="2012498680" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:42:41Z"
184   user="yudiwbs" uid="268765" lat="-6.9054261" lon="107.5975749"/>
185 <node id="2012498682" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:42:41Z"
186   user="yudiwbs" uid="268765" lat="-6.9064253" lon="107.5963227"/>
187 <node id="2012498684" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:42:41Z"
188   user="yudiwbs" uid="268765" lat="-6.9066150" lon="107.5976332"/>
189 <node id="2012498693" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:42:41Z"
190   user="yudiwbs" uid="268765" lat="-6.9066061" lon="107.5980546"/>
191 <node id="2012498699" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:42:42Z"
192   user="yudiwbs" uid="268765" lat="-6.9065619" lon="107.5976909"/>
193 <node id="2012498701" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:42:42Z"
194   user="yudiwbs" uid="268765" lat="-6.9065713" lon="107.5976410"/>
195 <node id="2012498709" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:42:42Z"
196   user="yudiwbs" uid="268765" lat="-6.9070083" lon="107.5976580"/>
197 <node id="29376826" visible="true" version="2" changeset="13860930" timestamp="2012-11-13T15:42:42Z" user
198   ="yudiwbs" uid="268765" lat="-6.9053204" lon="107.5975733"/>
199 <node id="25500712" visible="true" version="4" changeset="13860930" timestamp="2012-11-13T15:42:42Z" user
200   ="yudiwbs" uid="268765" lat="-6.9066888" lon="107.5983123"/>
201 <node id="2012498673" visible="true" version="4" changeset="27915808" timestamp="2015-01-04T17:54:58Z"
202   user="isonpurba" uid="2552445" lat="-6.9064863" lon="107.5975537"/>
203 <node id="2012502549" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z"
204   user="yudiwbs" uid="268765" lat="-6.9061123" lon="107.5981368"/>
205 <node id="2012502551" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z"
206   user="yudiwbs" uid="268765" lat="-6.9059552" lon="107.5979383"/>
207 <node id="2012502553" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z"
208   user="yudiwbs" uid="268765" lat="-6.9060146" lon="107.5978821"/>
209 <node id="2012502555" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z"
210   user="yudiwbs" uid="268765" lat="-6.9069030" lon="107.5974055"/>
211 <node id="2012502557" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z"
212   user="yudiwbs" uid="268765" lat="-6.9060119" lon="107.5979465"/>
213 <node id="2012502559" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z"
214   user="yudiwbs" uid="268765" lat="-6.9069128" lon="107.5972456"/>
215 <node id="2012502561" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z"
216   user="yudiwbs" uid="268765" lat="-6.9060617" lon="107.5978739"/>
217 <node id="2012502563" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z"
218   user="yudiwbs" uid="268765" lat="-6.9067851" lon="107.5973982"/>
219 <node id="2012502565" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z"
220   user="yudiwbs" uid="268765" lat="-6.9060412" lon="107.5976890"/>
221 <node id="2012502566" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z"
222   user="yudiwbs" uid="268765" lat="-6.9063927" lon="107.5977024"/>
223 <node id="2012502567" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z"
224   user="yudiwbs" uid="268765" lat="-6.9060039" lon="107.5981584"/>
225 <node id="2012502568" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z"
226   user="yudiwbs" uid="268765" lat="-6.9059533" lon="107.5981530"/>
227 <node id="2012502569" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z"
228   user="yudiwbs" uid="268765" lat="-6.9068199" lon="107.5973263"/>
229 <node id="2012502570" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z"
230   user="yudiwbs" uid="268765" lat="-6.9061184" lon="107.5980645"/>
231 <node id="2012502572" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z"
232   user="yudiwbs" uid="268765" lat="-6.9064931" lon="107.5980644"/>
233 <node id="2012502574" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z"
234   user="yudiwbs" uid="268765" lat="-6.9068252" lon="107.5972402"/>
235 <node id="2012502576" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z"
236   user="yudiwbs" uid="268765" lat="-6.9067896" lon="107.5973244"/>
237 <node id="2012502577" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z"
238   user="yudiwbs" uid="268765" lat="-6.9060066" lon="107.5981235"/>
239 <node id="1700556141" visible="true" version="2" changeset="13860930" timestamp="2012-11-13T15:47:32Z"
240   user="yudiwbs" uid="268765" lat="-6.9067781" lon="107.5978365"/>
241   <tag k="amenity" v="fast_food"/>
242   <tag k="cuisine" v="american"/>
243   <tag k="name" v="KFC"/>
244   </node>

```

```

198 <node id="2012512606" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:51:18Z"
199   user="yudiwbs" uid="268765" lat="-6.9052099" lon="107.5972778"/>
200 <node id="2012512608" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:51:18Z"
201   user="yudiwbs" uid="268765" lat="-6.9050292" lon="107.5972977"/>
202 <node id="2012512610" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:51:18Z"
203   user="yudiwbs" uid="268765" lat="-6.9052257" lon="107.5970093"/>
204 <node id="2012512612" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:51:18Z"
205   user="yudiwbs" uid="268765" lat="-6.9052321" lon="107.5974396"/>
206 <node id="2012512614" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:51:18Z"
207   user="yudiwbs" uid="268765" lat="-6.9051312" lon="107.5970096"/>
208 <node id="2012512616" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:51:18Z"
209   user="yudiwbs" uid="268765" lat="-6.9049922" lon="107.5972962"/>
210 <node id="2012512618" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:51:18Z"
211   user="yudiwbs" uid="268765" lat="-6.9050263" lon="107.5974231"/>
212 <node id="2012512620" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:51:18Z"
213   user="yudiwbs" uid="268765" lat="-6.9052440" lon="107.5972790"/>
214 <node id="2012512622" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:51:18Z"
215   user="yudiwbs" uid="268765" lat="-6.9052298" lon="107.5971181"/>
216 <node id="2012512623" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:51:18Z"
217   user="yudiwbs" uid="268765" lat="-6.9052039" lon="107.5971181"/>
218 <node id="2012512624" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:51:18Z"
219   user="yudiwbs" uid="268765" lat="-6.9049711" lon="107.5970233"/>
220 <node id="1837514693" visible="true" version="3" changeset="14069373" timestamp="2012-11-28T07:04:08Z"
221   user="dadan_dany_dipoera" uid="922816" lat="-6.9071711" lon="107.6044350"/>
222 <node id="2025969631" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07:04:29Z"
223   user="dadan_dany_dipoera" uid="922816" lat="-6.9064106" lon="107.5968064"/>
224 <node id="1666615070" visible="true" version="3" changeset="14069373" timestamp="2012-11-28T07:03:54Z"
225   user="dadan_dany_dipoera" uid="922816" lat="-6.9033176" lon="107.5975100"/>
226 <node id="2012498676" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07:04:29Z"
227   user="dadan_dany_dipoera" uid="922816" lat="-6.9070495" lon="107.5976364"/>
228 <node id="2012498686" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07:04:29Z"
229   user="dadan_dany_dipoera" uid="922816" lat="-6.9065298" lon="107.5959311"/>
230 <node id="2012498695" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07:04:29Z"
231   user="dadan_dany_dipoera" uid="922816" lat="-6.9055222" lon="107.5975513"/>
232 <node id="2012498697" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07:04:29Z"
233   user="dadan_dany_dipoera" uid="922816" lat="-6.9054179" lon="107.5975701"/>
234 <node id="2012498703" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07:04:29Z"
235   user="dadan_dany_dipoera" uid="922816" lat="-6.9065988" lon="107.5958098"/>
236 <node id="2012498705" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07:04:29Z"
237   user="dadan_dany_dipoera" uid="922816" lat="-6.9064822" lon="107.5963313"/>
238 <node id="2012498707" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07:04:29Z"
239   user="dadan_dany_dipoera" uid="922816" lat="-6.9066042" lon="107.5959524"/>
240 <node id="2012498711" visible="true" version="3" changeset="14069373" timestamp="2012-11-28T07:04:29Z"
241   user="dadan_dany_dipoera" uid="922816" lat="-6.9063931" lon="107.5975475"/>
242 <node id="2012498713" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07:04:29Z"
243   user="dadan_dany_dipoera" uid="922816" lat="-6.9069982" lon="107.5975809"/>
244 <node id="25433682" visible="true" version="3" changeset="14069373" timestamp="2012-11-28T07:04:32Z" user
245   ="dadan_dany_dipoera" uid="922816" lat="-6.9049424" lon="107.5989062"/>
246 <node id="25433686" visible="true" version="4" changeset="14069373" timestamp="2012-11-28T07:04:32Z" user
247   ="dadan_dany_dipoera" uid="922816" lat="-6.9039897" lon="107.5975474"/>
248 <node id="25433689" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07:04:32Z" user
249   ="dadan_dany_dipoera" uid="922816" lat="-6.9045058" lon="107.5962277"/>
250 <node id="29376831" visible="true" version="2" changeset="14069373" timestamp="2012-11-28T07:04:34Z" user
251   ="dadan_dany_dipoera" uid="922816" lat="-6.9042246" lon="107.5952578"/>
252 <node id="29376843" visible="true" version="3" changeset="14069373" timestamp="2012-11-28T07:04:34Z" user
253   ="dadan_dany_dipoera" uid="922816" lat="-6.9066987" lon="107.5953179"/>
254 <node id="32520364" visible="true" version="4" changeset="27915808" timestamp="2015-01-04T17:54:58Z" user
255   ="isonpurba" uid="2552445" lat="-6.9064793" lon="107.5972458"/>
256 <node id="2325451578" visible="true" version="2" changeset="18997770" timestamp="2013-11-19T17:08:36Z"
257   user="urbanovic" uid="1784103" lat="-6.9048417" lon="107.6021410"/>
258 <node id="2325451571" visible="true" version="3" changeset="24131990" timestamp="2014-07-14T03:03:32Z"
259   user="brambanan" uid="2092576" lat="-6.9048702" lon="107.6013524"/>
260 <node id="2325451286" visible="true" version="2" changeset="18998018" timestamp="2013-11-19T17:23:34Z"
261   user="urbanovic" uid="1784103" lat="-6.9012529" lon="107.5974289"/>
262 <node id="2490686820" visible="true" version="1" changeset="18276242" timestamp="2013-10-10T07:28:52Z"
263   user="ArjanO" uid="38066" lat="-6.9071703" lon="107.5996685"/>
264 <tag k="name" v="Alfamart"/>
265 <tag k="shop" v="supermarket"/>
266 </node>
267 <node id="2628264132" visible="true" version="1" changeset="20081263" timestamp="2014-01-19T09:38:29Z"
268   user="Irfan_Muhammad" uid="646006" lat="-6.9116314" lon="107.5979581"/>
269 <node id="29376968" visible="true" version="3" changeset="20081263" timestamp="2014-01-19T09:38:32Z" user
270   ="Irfan_Muhammad" uid="646006" lat="-6.9117870" lon="107.5979651"/>
271 <node id="2725732963" visible="true" version="1" changeset="21180806" timestamp="2014-03-18T19:57:06Z"
272   user="Warung_Nasi_Kang_EWOK" uid="1991592" lat="-6.9052749" lon="107.5961955"/>
273 <tag k="addr:city" v="Bandung"/>
274 <tag k="addr:housenumber" v="16"/>
275 <tag k="addr:street" v="Astina"/>
276 <tag k="amenity" v="restaurant"/>
277 <tag k="capacity" v="50"/>
278 <tag k="cuisine" v="sundanese"/>
279 <tag k="name" v="Warung_Nasi_Kang_EWOK"/>
280 <tag k="phone" v="0226038783_02292035359"/>
281 </node>
282 <node id="3030289969" visible="true" version="1" changeset="24892866" timestamp="2014-08-20T18:40:31Z"
283   user="albahrimaraxsa" uid="2162153" lat="-6.9069924" lon="107.5982831"/>
284 <node id="3030289970" visible="true" version="1" changeset="24892866" timestamp="2014-08-20T18:40:31Z"
285   user="albahrimaraxsa" uid="2162153" lat="-6.9067987" lon="107.5982587"/>
286 <node id="3030289971" visible="true" version="1" changeset="24892866" timestamp="2014-08-20T18:40:31Z"
287   user="albahrimaraxsa" uid="2162153" lat="-6.9066710" lon="107.5982569"/>
288 <node id="2325451442" visible="true" version="4" changeset="27916144" timestamp="2015-01-04T18:06:33Z"
289   user="isonpurba" uid="2552445" lat="-6.9045011" lon="107.6024922"/>
290 <way id="4567626" visible="true" version="4" changeset="15861148" timestamp="2013-04-25T13:56:12Z" user=
291   "mrdoggie94" uid="1331966">
292   <nd ref="25433682"/>
293   <nd ref="25433681"/>
294   <nd ref="25433680"/>
295   <tag k="avgspeed" v="15"/>
296   <tag k="highway" v="residential"/>

```

```

256 <tag k="name" v="Dr. J. Rubini" />
257 </way>
258 <way id="4567634" visible="true" version="2" changeset="7821743" timestamp="2011-04-10T11:15:30Z" user=
259   evo2mind" uid="234610">
260   <nd ref="25433681"/>
261   <nd ref="28802396"/>
262   <tag k="avgsped" v="15" />
263   <tag k="highway" v="residential" />
264   <tag k="name" v="Dr. J. Susilo" />
265 </way>
266 <way id="4625182" visible="true" version="5" changeset="11905099" timestamp="2012-06-15T14:05:57Z" user=
267   andryono" uid="643030">
268   <nd ref="29376826"/>
269   <nd ref="364364242"/>
270   <nd ref="29376827"/>
271   <nd ref="25433690"/>
272   <tag k="avgsped" v="20" />
273   <tag k="highway" v="living_street" />
274   <tag k="oneway" v="yes" />
275 </way>
276 <way id="4627111" visible="true" version="2" changeset="7821743" timestamp="2011-04-10T11:15:36Z" user=
277   evo2mind" uid="234610">
278   <nd ref="29356177"/>
279   <nd ref="29356178"/>
280   <nd ref="29356179"/>
281   <nd ref="29356180"/>
282   <nd ref="25433684"/>
283   <tag k="avgsped" v="15" />
284   <tag k="highway" v="residential" />
285   <tag k="name" v="Muhammad Yunus" />
286 </way>
287 <way id="4628057" visible="true" version="3" changeset="11905099" timestamp="2012-06-15T14:05:58Z" user=
288   andryono" uid="643030">
289   <nd ref="29392373"/>
290   <nd ref="29392374"/>
291   <nd ref="29392377"/>
292   <tag k="avgsped" v="10" />
293   <tag k="highway" v="service" />
294   <tag k="service" v="parking_aisle" />
295 </way>
296 <way id="4907167" visible="true" version="2" changeset="7821743" timestamp="2011-04-10T11:12:42Z" user=
297   evo2mind" uid="234610">
298   <nd ref="32041794"/>
299   <nd ref="32041785"/>
300   <tag k="avgsped" v="10" />
301 </way>
302 <way id="247058985" visible="true" version="2" changeset="24131990" timestamp="2014-07-14T03:03:33Z" user=
303   "brambanan" uid="2092576">
304   <nd ref="2325451442"/>
305   <nd ref="364364086"/>
306   <nd ref="364364087"/>
307   <nd ref="2325451578"/>
308   <nd ref="2325451571"/>
309   <tag k="avgsped" v="30" />
310   <tag k="highway" v="secondary" />
311   <tag k="name" v="Dr. J. Rum" />
312   <tag k="oneway" v="yes" />
313 </way>
314 <way id="32388775" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:15:09Z" user=
315   adhitya" uid="7748">
316   <nd ref="364363888"/>
317   <nd ref="364363889"/>
318   <nd ref="364363890"/>
319   <nd ref="364363895"/>
320   <nd ref="364363896"/>
321   <nd ref="364363897"/>
322   <nd ref="364363898"/>
323   <nd ref="364363900"/>
324   <nd ref="364364237"/>
325   <nd ref="364364238"/>
326   <nd ref="364364239"/>
327   <nd ref="364364240"/>
328   <nd ref="364364241"/>
329   <nd ref="364364242"/>
330   <nd ref="364364243"/>
331 </way>
332 <way id="32388776" visible="true" version="2" changeset="943187" timestamp="2009-04-25T10:24:30Z" user=
333   Andre68" uid="31231">
334   <nd ref="364364075"/>
335   <nd ref="364364076"/>
336   <nd ref="364364077"/>
337   <nd ref="364364078"/>
338   <nd ref="364364075"/>
339   <tag k="amenity" v="fast_food , restaurant" />
340   <tag k="atm" v="yes" />
341   <tag k="building" v="yes" />
342   <tag k="name" v="Istana Plaza" />
343   <tag k="shop" v="supermarket" />
344 </way>
345 <way id="32388779" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:15:19Z" user=
346   adhitya" uid="7748">
347   <nd ref="364364184"/>
348   <nd ref="364364194"/>

```

```

346  <nd ref="364364195"/>
347  <tag k="highway" v="service"/>
348  <tag k="oneway" v="yes"/>
349  <tag k="service" v="parking_aisle"/>
350 </way>
351 <way id="32388780" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:15:20Z" user=
352   adhitya" uid="7748">
353  <nd ref="364364182"/>
354  <nd ref="364364202"/>
355  <nd ref="364364220"/>
356  <tag k="highway" v="service"/>
357  <tag k="oneway" v="yes"/>
358  <tag k="service" v="parking_aisle"/>
359 </way>
360 <way id="32388782" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:15:21Z" user=
361   adhitya" uid="7748">
362  <nd ref="364364202"/>
363  <nd ref="364364242"/>
364  <tag k="highway" v="service"/>
365  <tag k="oneway" v="yes"/>
366  <tag k="service" v="parking_aisle"/>
367 </way>
368 <way id="32388783" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:15:26Z" user=
369   adhitya" uid="7748">
370  <nd ref="364365012"/>
371  <nd ref="364365069"/>
372  <nd ref="364365018"/>
373  <nd ref="364365077"/>
374  <nd ref="364365091"/>
375  <nd ref="364365025"/>
376  <nd ref="364365061"/>
377  <nd ref="364365096"/>
378  <nd ref="364365090"/>
379  <nd ref="364365069"/>
380  <tag k="highway" v="service"/>
381  <tag k="service" v="parking_aisle"/>
382 </way>
383 <way id="32388784" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:15:27Z" user=
384   adhitya" uid="7748">
385  <nd ref="364365077"/>
386  <nd ref="364365090"/>
387  <tag k="highway" v="service"/>
388  <tag k="service" v="parking_aisle"/>
389 </way>
390 <way id="32388785" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:15:29Z" user=
391   adhitya" uid="7748">
392  <nd ref="364365091"/>
393  <nd ref="364365096"/>
394  <tag k="highway" v="service"/>
395  <tag k="service" v="parking_aisle"/>
396 </way>
397 <way id="32388787" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:15:30Z" user=
398   adhitya" uid="7748">
399  <nd ref="364365134"/>
400  <nd ref="364365135"/>
401  <nd ref="364365136"/>
402  <nd ref="364365137"/>
403  <nd ref="364365134"/>
404  <tag k="leisure" v="stadium"/>
405  <tag k="name" v="GOR_Padjajaran"/>
406  <tag k="sport" v="basketball"/>
407 </way>
408 <way id="32388788" visible="true" version="1" changeset="839915" timestamp="2009-03-21T14:15:38Z" user=
409   adhitya" uid="7748">
410  <nd ref="364365142"/>
411  <nd ref="364365164"/>
412  <nd ref="364365155"/>
413  <nd ref="364365165"/>
414  <nd ref="364365143"/>
415  <nd ref="364365144"/>
416  <nd ref="364365171"/>
417  <nd ref="364365162"/>
418  <nd ref="364365170"/>
419  <nd ref="364365145"/>
420  <nd ref="364365169"/>
421  <nd ref="364365161"/>
422  <nd ref="364365168"/>
423  <nd ref="364365146"/>
424  <nd ref="364365151"/>
425 </way>
426 <way id="32388786" visible="true" version="2" changeset="11220464" timestamp="2012-04-08T02:49:11Z" user=
427   "andryono" uid="643030">
428  <nd ref="364365097"/>
429  <nd ref="364365098"/>
430  <nd ref="364365099"/>
431  <nd ref="364365133"/>
432  <nd ref="364365097"/>
433  <tag k="leisure" v="sports_centre"/>
434  <tag k="name" v="Tri_Lomba_Juang"/>
435 </way>
```

```

436 | <way id="32388777" visible="true" version="2" changeset="11905099" timestamp="2012-06-15T14:05:59Z" user=
437 |   "andryono" uid="643030">
438 |   <nd ref="364364179"/>
439 |   <nd ref="364364182"/>
440 |   <nd ref="364364195"/>
441 |   <nd ref="364364183"/>
442 |   <nd ref="364364184"/>
443 |   <nd ref="364364192"/>
444 |   <tag k="highway" v="service"/>
444 |   <tag k="oneway" v="yes"/>
445 |   <tag k="service" v="parking_aisle"/>
446 | </way>
447 | <way id="4625197" visible="true" version="2" changeset="7821743" timestamp="2011-04-10T11:14:24Z" user="evo2mind" uid="234610">
448 |   <nd ref="29376841"/>
449 |   <nd ref="29376842"/>
450 |   <nd ref="29376843"/>
451 |   <tag k="avgsped" v="15"/>
452 |   <tag k="highway" v="residential"/>
453 |   <tag k="name" v="Purabaya"/>
454 | </way>
455 | <way id="4567527" visible="true" version="4" changeset="18997770" timestamp="2013-11-19T17:08:37Z" user="ubanovic" uid="1784103">
456 |   <nd ref="2325451571"/>
457 |   <nd ref="29392377"/>
458 |   <nd ref="25433680"/>
459 |   <tag k="avgsped" v="15"/>
460 |   <tag k="highway" v="tertiary"/>
461 |   <tag k="name" v="Dr. Cipto"/>
462 |   <tag k="oneway" v="-1"/>
463 | </way>
464 | <way id="4625184" visible="true" version="4" changeset="11905099" timestamp="2012-06-15T14:05:58Z" user="andryono" uid="643030">
465 |   <nd ref="25433687"/>
466 |   <nd ref="29376829"/>
467 |   <nd ref="29376827"/>
468 |   <tag k="avgsped" v="15"/>
469 |   <tag k="highway" v="living_street"/>
470 |   <tag k="name" v="Citrayuda"/>
471 |   <tag k="oneway" v="-1"/>
472 | </way>
473 | <way id="4625188" visible="true" version="2" changeset="7821743" timestamp="2011-04-10T11:12:21Z" user="evo2mind" uid="234610">
474 |   <nd ref="25433689"/>
475 |   <nd ref="29376831"/>
476 |   <tag k="avgsped" v="15"/>
477 |   <tag k="highway" v="residential"/>
478 | </way>
479 | <way id="154066530" visible="true" version="2" changeset="11884346" timestamp="2012-06-13T12:22:04Z" user="andryono" uid="643030">
480 |   <nd ref="364364251"/>
481 |   <nd ref="1666616044"/>
482 |   <nd ref="1666616129"/>
483 |   <nd ref="1666615973"/>
484 |   <nd ref="1666616003"/>
485 |   <nd ref="1666616103"/>
486 |   <nd ref="1666616021"/>
487 |   <nd ref="1666616062"/>
488 |   <tag k="highway" v="living_street"/>
489 | </way>
490 | <way id="4627113" visible="true" version="4" changeset="24892866" timestamp="2014-08-20T18:40:32Z" user="albahrimaraxsa" uid="2162153">
491 |   <nd ref="29356500"/>
492 |   <nd ref="29356501"/>
493 |   <nd ref="29356502"/>
494 |   <nd ref="29356503"/>
495 |   <nd ref="29356504"/>
496 |   <nd ref="3030289969"/>
497 |   <nd ref="3030289970"/>
498 |   <nd ref="3030289971"/>
499 |   <tag k="avgsped" v="15"/>
500 |   <tag k="highway" v="living_street"/>
501 |   <tag k="name" v="Muhammad_Aleh"/>
502 | </way>
503 | <way id="4567530" visible="true" version="3" changeset="11220464" timestamp="2012-04-08T02:49:10Z" user="andryono" uid="643030">
504 |   <nd ref="25433678"/>
505 |   <nd ref="25433679"/>
506 |   <tag k="avgsped" v="15"/>
507 |   <tag k="highway" v="residential"/>
508 | </way>
509 | <way id="4627112" visible="true" version="3" changeset="11220464" timestamp="2012-04-08T02:49:12Z" user="andryono" uid="643030">
510 |   <nd ref="29356180"/>
511 |   <nd ref="29356534"/>
512 |   <nd ref="29356535"/>
513 |   <nd ref="29356503"/>
514 |   <tag k="avgsped" v="15"/>
515 |   <tag k="highway" v="living_street"/>
516 |   <tag k="name" v="Purwawinata"/>
517 | </way>
518 | <way id="4627146" visible="true" version="3" changeset="11220464" timestamp="2012-04-08T02:49:12Z" user="andryono" uid="643030">
519 |   <nd ref="29356540"/>
520 |   <nd ref="29356534"/>
521 |   <tag k="avgsped" v="15"/>
522 |   <tag k="highway" v="living_street"/>
523 | </way>

```

```

524 <way id="4627117" visible="true" version="3" changeset="11220464" timestamp="2012-04-08T02:49:13Z" user="andryono" uid="643030">
525 <nd ref="29356374"/>
526 <nd ref="29356535"/>
527 <tag k="avgspeed" v="15"/>
528 <tag k="highway" v="living_street"/>
529 </way>
530 <way id="4625190" visible="true" version="3" changeset="11220620" timestamp="2012-04-08T04:00:41Z" user="andryono" uid="643030">
531 <nd ref="29376836"/>
532 <nd ref="29376837"/>
533 <nd ref="29376835"/>
534 <tag k="avgspeed" v="15"/>
535 <tag k="highway" v="living_street"/>
536 <tag k="name" v="Suka_Bakti"/>
537 </way>
538 <way id="4625194" visible="true" version="3" changeset="11220620" timestamp="2012-04-08T04:00:42Z" user="andryono" uid="643030">
539 <nd ref="29376839"/>
540 <nd ref="29376832"/>
541 <tag k="avgspeed" v="15"/>
542 <tag k="highway" v="living_street"/>
543 <tag k="name" v="Pamoyanan"/>
544 </way>
545 <way id="4567503" visible="true" version="6" changeset="27916265" timestamp="2015-01-04T18:11:06Z" user="isonpurba" uid="2552445">
546 <nd ref="25433685"/>
547 <nd ref="25433679"/>
548 <nd ref="25433682"/>
549 <nd ref="32041794"/>
550 <nd ref="29392373"/>
551 <nd ref="2325451571"/>
552 <tag k="avgspeed" v="30"/>
553 <tag k="bicycle" v="yes"/>
554 <tag k="foot" v="yes"/>
555 <tag k="highway" v="secondary"/>
556 <tag k="name" v="Dr._Rum"/>
557 </way>
558 <way id="4567635" visible="true" version="8" changeset="13860930" timestamp="2012-11-13T15:42:43Z" user="yudiwbs" uid="268765">
559 <nd ref="25418868"/>
560 <nd ref="28802396"/>
561 <nd ref="2012498680"/>
562 <tag k="avgspeed" v="30"/>
563 <tag k="bicycle" v="yes"/>
564 <tag k="foot" v="yes"/>
565 <tag k="highway" v="primary"/>
566 <tag k="name" v="Pasir_Kaliki"/>
567 <tag k="oneway" v="-1"/>
568 </way>
569 <way id="167186330" visible="true" version="1" changeset="11884346" timestamp="2012-06-13T12:22:01Z" user="andryono" uid="643030">
570 <nd ref="25433680"/>
571 <nd ref="25433684"/>
572 <tag k="avgspeed" v="15"/>
573 <tag k="highway" v="tertiary_link"/>
574 <tag k="oneway" v="yes"/>
575 </way>
576 <way id="4567529" visible="true" version="3" changeset="11884346" timestamp="2012-06-13T12:22:06Z" user="andryono" uid="643030">
577 <nd ref="25433683"/>
578 <nd ref="25433680"/>
579 <tag k="avgspeed" v="15"/>
580 <tag k="highway" v="tertiary"/>
581 <tag k="name" v="Dr._Cipto"/>
582 <tag k="oneway" v="yes"/>
583 </way>
584 <way id="157803724" visible="true" version="3" changeset="13860930" timestamp="2012-11-13T15:47:32Z" user="yudiwbs" uid="268765">
585 <nd ref="1700554884"/>
586 <nd ref="1700554892"/>
587 <nd ref="1700554901"/>
588 <nd ref="1700554868"/>
589 <nd ref="1700554884"/>
590 <tag k="building" v="retail"/>
591 <tag k="name" v="KFC"/>
592 </way>
593 <way id="167462800" visible="true" version="2" changeset="12389881" timestamp="2012-07-20T20:14:38Z" user="OSMF_Redaction_Account" uid="722137">
594 <nd ref="364364237"/>
595 <nd ref="364364192"/>
596 <tag k="highway" v="service"/>
597 <tag k="oneway" v="no"/>
598 <tag k="service" v="parking_aisle"/>
599 </way>
600 <way id="4256037" visible="true" version="10" changeset="27611233" timestamp="2014-12-21T15:25:04Z" user="gnocin" uid="2526082">
601 <nd ref="25418868"/>
602 <nd ref="2012498693"/>
603 <tag k="avgspeed" v="30"/>
604 <tag k="bicycle" v="yes"/>
605 <tag k="foot" v="yes"/>
606 <tag k="highway" v="primary"/>
607 <tag k="name" v="Pajajaran"/>
608 <tag k="oneway" v="yes"/>
609 </way>
610 <way id="175517312" visible="true" version="1" changeset="12686463" timestamp="2012-08-11T00:36:04Z" user="ferdyodin" uid="796044">
611 <nd ref="1860932974"/>

```

```

612 | <nd ref="29376841"/>
613 | <nd ref="1860932973"/>
614 | <nd ref="1860932975"/>
615 | <tag k="highway" v="unclassified"/>
616 | <tag k="name" v="Semar"/>
617 | </way>
618 | <way id="190605660" visible="true" version="2" changeset="18998018" timestamp="2013-11-19T17:23:36Z" user
619 |   = "ubanovic" uid="1784103">
620 |     <nd ref="2012498680"/>
621 |     <nd ref="2012498697"/>
622 |     <nd ref="364364179"/>
623 |     <nd ref="29376826"/>
624 |     <nd ref="25433685"/>
625 |     <nd ref="25433686"/>
626 |     <nd ref="1666615070"/>
627 |     <nd ref="32041828"/>
628 |     <nd ref="2325451286"/>
629 |     <tag k="avgsped" v="30"/>
630 |     <tag k="bicycle" v="yes"/>
631 |     <tag k="foot" v="yes"/>
632 |     <tag k="highway" v="primary"/>
633 |     <tag k="name" v="Pasir_Kaliki"/>
634 |     <tag k="oneway" v="no"/>
635 |   </way>
636 |   <way id="190605655" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:42:42Z" user
637 |     = "yudiwbs" uid="268765">
638 |       <nd ref="2012498697"/>
639 |       <nd ref="2012498695"/>
640 |       <nd ref="2012498711"/>
641 |       <tag k="highway" v="primary"/>
642 |       <tag k="oneway" v="-1"/>
643 |     </way>
644 |     <way id="190605657" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:42:42Z" user
645 |       = "yudiwbs" uid="268765">
646 |         <nd ref="2012498705"/>
647 |         <nd ref="2012498682"/>
648 |         <nd ref="364364247"/>
649 |         <nd ref="364364251"/>
650 |         <nd ref="29376836"/>
651 |         <nd ref="25433690"/>
652 |         <nd ref="29376839"/>
653 |         <nd ref="25433689"/>
654 |         <nd ref="1666615284"/>
655 |         <nd ref="25433688"/>
656 |         <tag k="avgsped" v="15"/>
657 |         <tag k="highway" v="unclassified"/>
658 |         <tag k="name" v="Astina"/>
659 |         <tag k="oneway" v="yes"/>
660 |       </way>
661 |       <way id="190605653" visible="true" version="2" changeset="13981717" timestamp="2012-11-22T06:24:04Z" user
662 |         = "yudiwbs" uid="268765">
663 |           <nd ref="2012498676"/>
664 |           <nd ref="2012498713"/>
665 |           <nd ref="2012498673"/>
666 |           <nd ref="2012498711"/>
667 |           <tag k="highway" v="primary"/>
668 |           <tag k="oneway" v="yes"/>
669 |         </way>
670 |         <way id="190606151" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z" user
671 |           = "yudiwbs" uid="268765">
672 |             <nd ref="2012502574"/>
673 |             <nd ref="2012502569"/>
674 |             <nd ref="2012502576"/>
675 |             <nd ref="2012502563"/>
676 |             <nd ref="2012502555"/>
677 |             <nd ref="2012502559"/>
678 |             <nd ref="2012502574"/>
679 |             <tag k="amenity" v="police"/>
680 |             <tag k="name" v="Polsek_Cicendo"/>
681 |           </way>
682 |           <way id="190606152" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:47:31Z" user
683 |             = "yudiwbs" uid="268765">
684 |               <nd ref="2012502565"/>
685 |               <nd ref="2012502577"/>
686 |               <nd ref="2012502549"/>
687 |               <nd ref="2012502570"/>
688 |               <nd ref="2012502572"/>
689 |               <nd ref="2012502566"/>
690 |               <nd ref="2012502565"/>
691 |               <tag k="amenity" v="parking"/>
692 |               <tag k="name" v="Parkir_Istana_Plaza"/>
693 |               <tag k="parking" v="surface"/>
694 |             </way>
695 |             <way id="190606798" visible="true" version="1" changeset="13860930" timestamp="2012-11-13T15:51:18Z" user
696 |               = "yudiwbs" uid="268765">
697 |                 <nd ref="2012512624"/>
698 |                 <nd ref="2012512616"/>
699 |                 <nd ref="2012512608"/>
700 |                 <nd ref="2012512618"/>
701 |                 <nd ref="2012512612"/>
702 |                 <nd ref="2012512620"/>
703 |                 <nd ref="2012512606"/>
704 |                 <nd ref="2012512623"/>

```

```

704 <nd ref="2012512622"/>
705 <nd ref="2012512610"/>
706 <nd ref="2012512614"/>
707 <nd ref="2012512624"/>
708 <tag k="amenity" v="college"/>
709 <tag k="name" v="Ariyanti"/>
710 </way>
711 <way id="190605650" visible="true" version="3" changeset="13981717" timestamp="2012-11-22T06:24:04Z" user
    ="yudiwbs" uid="268765">
712 <nd ref="2012498673"/>
713 <nd ref="32520364"/>
714 <nd ref="1860932974"/>
715 <nd ref="2012498707"/>
716 <nd ref="2012498703"/>
717 <tag k="highway" v="primary"/>
718 <tag k="oneway" v="yes"/>
719 </way>
720 <way id="190605663" visible="true" version="3" changeset="27592401" timestamp="2014-12-20T16:49:58Z" user
    ="gnocin" uid="2526082">
721 <nd ref="2012498703"/>
722 <nd ref="2012498686"/>
723 <nd ref="2012498705"/>
724 <tag k="highway" v="primary"/>
725 <tag k="oneway" v="yes"/>
726 </way>
727 <way id="257227478" visible="true" version="4" changeset="27915834" timestamp="2015-01-04T17:55:47Z" user
    ="isonpurba" uid="2552445">
728 <nd ref="29376968"/>
729 <nd ref="2628264132"/>
730 <nd ref="364363837"/>
731 <nd ref="364363838"/>
732 <nd ref="29356499"/>
733 <nd ref="1860932975"/>
734 <nd ref="25434115"/>
735 <nd ref="29356528"/>
736 <nd ref="29356381"/>
737 <nd ref="2012498676"/>
738 <tag k="avgspeed" v="30"/>
739 <tag k="bicycle" v="yes"/>
740 <tag k="foot" v="yes"/>
741 <tag k="highway" v="primary"/>
742 <tag k="name" v="Jln .HOS. Tjokroaminoto"/>
743 <tag k="oneway" v="yes"/>
744 </way>
745 <way id="318100678" visible="true" version="1" changeset="27592401" timestamp="2014-12-20T16:49:54Z" user
    ="gnocin" uid="2526082">
746 <nd ref="2012498705"/>
747 <nd ref="2025969631"/>
748 <nd ref="2012498711"/>
749 <nd ref="25418868"/>
750 <tag k="highway" v="primary"/>
751 <tag k="oneway" v="yes"/>
752 </way>
753 <way id="318222725" visible="true" version="2" changeset="27726149" timestamp="2014-12-27T09:47:32Z" user
    ="gnocin" uid="2526082">
754 <nd ref="2012498693"/>
755 <nd ref="3030289971"/>
756 <nd ref="25500712"/>
757 <nd ref="25433683"/>
758 <tag k="avgspeed" v="30"/>
759 <tag k="bicycle" v="yes"/>
760 <tag k="foot" v="yes"/>
761 <tag k="highway" v="primary"/>
762 <tag k="name" v="Pajajaran"/>
763 <tag k="oneway" v="yes"/>
764 </way>
765 <way id="318892821" visible="true" version="1" changeset="27700068" timestamp="2014-12-26T00:34:22Z" user
    ="gnocin" uid="2526082">
766 <nd ref="25418868"/>
767 <nd ref="2012498673"/>
768 <tag k="highway" v="primary"/>
769 <tag k="oneway" v="yes"/>
770 </way>
771 <way id="319068174" visible="true" version="1" changeset="27726149" timestamp="2014-12-27T09:47:29Z" user
    ="gnocin" uid="2526082">
772 <nd ref="25433683"/>
773 <nd ref="25433684"/>
774 <nd ref="364365012"/>
775 <nd ref="25500626"/>
776 <nd ref="1706564168"/>
777 <nd ref="1706564155"/>
778 <nd ref="32529146"/>
779 <nd ref="29392413"/>
780 <nd ref="1837514693"/>
781 <tag k="avgspeed" v="30"/>
782 <tag k="bicycle" v="yes"/>
783 <tag k="foot" v="yes"/>
784 <tag k="highway" v="primary"/>
785 <tag k="name" v="Pajajaran"/>
786 <tag k="oneway" v="yes"/>
787 </way>
788 <way id="318670708" visible="true" version="3" changeset="27915808" timestamp="2015-01-04T17:54:58Z" user
    ="isonpurba" uid="2552445">
789 <nd ref="2012498676"/>
790 <nd ref="2012498709"/>
791 <nd ref="32520365"/>
792 <nd ref="2012498684"/>
793 <nd ref="2012498701"/>
794 <nd ref="2012498699"/>
```

```

795 <nd ref="2012498678"/>
796 <nd ref="2012498693"/>
797 <tag k="avgsped" v="30"/>
798 <tag k="bicycle" v="yes"/>
799 <tag k="foot" v="yes"/>
800 <tag k="highway" v="primary_link"/>
801 <tag k="name" v="Jln.HOS._Tjokroaminoto"/>
802 <tag k="oneway" v="yes"/>
803 <tag k="surface" v="asphalt"/>
804 </way>
805 <way id="320417107" visible="true" version="1" changeset="27916246" timestamp="2015-01-04T18:10:18Z" user="isonpurba" uid="2552445">
806 <nd ref="2325451571"/>
807 <nd ref="25447003"/>
808 <tag k="highway" v="secondary"/>
809 </way>
810 <relation id="4294473" visible="true" version="18" changeset="28004961" timestamp="2015-01-08T20:02:57Z" user="isonpurba" uid="2552445">
811 <member type="way" ref="4627204" role="" />
812 <member type="way" ref="318756518" role="" />
813 <member type="way" ref="4760087" role="" />
814 <member type="way" ref="168942784" role="" />
815 <member type="way" ref="318231943" role="" />
816 <member type="way" ref="153586311" role="" />
817 <member type="way" ref="318756514" role="" />
818 <member type="way" ref="318756516" role="" />
819 <member type="way" ref="318231944" role="" />
820 <member type="way" ref="4936382" role="" />
821 <member type="way" ref="318231946" role="" />
822 <member type="way" ref="318729558" role="" />
823 <member type="way" ref="153586324" role="" />
824 <member type="way" ref="192917039" role="" />
825 <member type="way" ref="318054780" role="" />
826 <member type="way" ref="4623278" role="" />
827 <member type="way" ref="318729560" role="" />
828 <member type="way" ref="153315566" role="" />
829 <member type="way" ref="153315568" role="" />
830 <member type="node" ref="3246270247" role="" />
831 <member type="way" ref="4623279" role="" />
832 <member type="way" ref="186246670" role="" />
833 <member type="way" ref="174270181" role="" />
834 <member type="way" ref="318222726" role="" />
835 <member type="way" ref="318729562" role="" />
836 <member type="way" ref="318729561" role="" />
837 <member type="way" ref="4255951" role="" />
838 <member type="way" ref="174270180" role="" />
839 <member type="way" ref="299081906" role="" />
840 <member type="way" ref="174270184" role="" />
841 <member type="way" ref="174270182" role="" />
842 <member type="way" ref="318478990" role="" />
843 <member type="way" ref="4255981" role="" />
844 <member type="way" ref="4255990" role="" />
845 <member type="way" ref="32388792" role="" />
846 <member type="node" ref="3246270248" role="" />
847 <member type="way" ref="318233616" role="" />
848 <member type="way" ref="4256015" role="" />
849 <member type="way" ref="168569217" role="" />
850 <member type="way" ref="4251274" role="" />
851 <member type="way" ref="4567655" role="" />
852 <member type="node" ref="3246190446" role="stop" />
853 <member type="way" ref="257227459" role="" />
854 <member type="way" ref="4567652" role="" />
855 <member type="way" ref="4625235" role="" />
856 <member type="way" ref="4625237" role="" />
857 <member type="way" ref="4569060" role="" />
858 <member type="way" ref="190605663" role="" />
859 <member type="way" ref="190605657" role="" />
860 <member type="way" ref="4567636" role="" />
861 <member type="way" ref="318231955" role="" />
862 <member type="way" ref="318892822" role="" />
863 <member type="way" ref="257232273" role="" />
864 <member type="way" ref="318100679" role="" />
865 <member type="way" ref="318891912" role="" />
866 <member type="way" ref="257232272" role="" />
867 <member type="way" ref="257232274" role="" />
868 <member type="way" ref="318804217" role="" />
869 <member type="way" ref="192031850" role="" />
870 <member type="way" ref="4567524" role="" />
871 <member type="way" ref="247058986" role="" />
872 <member type="way" ref="318478989" role="" />
873 <member type="way" ref="292668836" role="" />
874 <member type="way" ref="292668641" role="" />
875 <member type="way" ref="321030145" role="" />
876 <member type="way" ref="235830734" role="" />
877 <member type="way" ref="257232271" role="" />
878 <member type="way" ref="318231951" role="" />
879 <member type="way" ref="4627108" role="" />
880 <member type="way" ref="4567685" role="" />
881 <member type="way" ref="318193812" role="" />
882 <member type="way" ref="318231952" role="" />
883 <member type="way" ref="4567680" role="" />
884 <member type="node" ref="2352383137" role="stop" />
885 <member type="way" ref="306556225" role="" />
886 <member type="way" ref="306586869" role="" />
887 <member type="way" ref="4625951" role="" />
888 <member type="way" ref="166752446" role="" />
889 <member type="way" ref="166752447" role="" />
890 <member type="way" ref="4625948" role="" />
891 <member type="way" ref="318191421" role="" />
```

```

892 <member type="way" ref="318193811" role="" />
893 <member type="way" ref="102775972" role="" />
894 <member type="way" ref="255736436" role="" />
895 <member type="way" ref="4637816" role="" />
896 <member type="way" ref="306586886" role="" />
897 <member type="way" ref="318231958" role="" />
898 <member type="node" ref="3246270249" role="" />
899 <member type="node" ref="3240012631" role="" />
900 <member type="node" ref="3249368131" role="" />
901 <member type="node" ref="3252877763" role="stop" />
902 <member type="way" ref="239929577" role="" />
903 <member type="way" ref="4569058" role="" />
904 <member type="node" ref="3252892639" role="stop" />
905 <member type="node" ref="3252929646" role="stop" />
906 <tag k="name" v="Rute_Angkot_34._Sadang_Serang_--Caringin" />
907 <tag k="network" v="Angkot_Kota_Bandung" />
908 <tag k="ref" v="34" />
909 <tag k="route" v="bus" />
910 <tag k="type" v="route" />
911 <tag k="wikipedia" v="id : Daftar_Angkutan_Umum_di_Kota_Bandung" />
912 </relation>
913 <relation id="4308299" visible="true" version="8" changeset="27739817" timestamp="2014-12-27T20:32:33Z"
  user="gnocin" uid="2526082">
914   <member type="way" ref="182987706" role="" />
915   <member type="way" ref="182628294" role="" />
916   <member type="way" ref="182628297" role="" />
917   <member type="way" ref="162762593" role="" />
918   <member type="way" ref="255739144" role="" />
919   <member type="node" ref="3248857676" role="" />
920   <member type="way" ref="172913717" role="" />
921   <member type="way" ref="318678696" role="" />
922   <member type="way" ref="167539549" role="" />
923   <member type="way" ref="318054785" role="" />
924   <member type="way" ref="223676585" role="" />
925   <member type="way" ref="153284261" role="" />
926   <member type="way" ref="4698221" role="" />
927   <member type="way" ref="318102560" role="" />
928   <member type="way" ref="318054784" role="" />
929   <member type="way" ref="4567624" role="" />
930   <member type="way" ref="318054782" role="" />
931   <member type="way" ref="319083572" role="" />
932   <member type="way" ref="319083569" role="" />
933   <member type="way" ref="292274154" role="" />
934   <member type="way" ref="153175500" role="" />
935   <member type="way" ref="292668299" role="" />
936   <member type="way" ref="318670704" role="" />
937   <member type="way" ref="153181146" role="" />
938   <member type="way" ref="153180024" role="" />
939   <member type="way" ref="318670705" role="" />
940   <member type="way" ref="318478988" role="" />
941   <member type="node" ref="3244264888" role="" />
942   <member type="way" ref="152949024" role="" />
943   <member type="way" ref="318538778" role="" />
944   <member type="way" ref="318478986" role="" />
945   <member type="way" ref="4623261" role="" />
946   <member type="way" ref="4623260" role="" />
947   <member type="way" ref="247064779" role="" />
948   <member type="way" ref="318670701" role="" />
949   <member type="way" ref="223676540" role="" />
950   <member type="way" ref="95608890" role="" />
951   <member type="way" ref="4623264" role="" />
952   <member type="way" ref="4256037" role="" />
953   <member type="way" ref="318222725" role="" />
954   <member type="way" ref="319068174" role="" />
955   <member type="way" ref="4251274" role="" />
956   <member type="way" ref="4567655" role="" />
957   <member type="way" ref="4567635" role="" />
958   <member type="way" ref="190605660" role="" />
959   <member type="way" ref="4567652" role="" />
960   <member type="way" ref="257227478" role="" />
961   <member type="way" ref="190605653" role="" />
962   <member type="way" ref="32360540" role="" />
963   <member type="way" ref="32360546" role="" />
964   <member type="way" ref="4567653" role="" />
965   <member type="way" ref="4567659" role="" />
966   <member type="way" ref="318222717" role="" />
967   <member type="way" ref="87148402" role="" />
968   <member type="way" ref="257227462" role="" />
969   <member type="way" ref="257227468" role="" />
970   <member type="way" ref="4627123" role="" />
971   <member type="way" ref="4627122" role="" />
972   <member type="way" ref="32388789" role="" />
973   <member type="way" ref="193626591" role="" />
974   <member type="way" ref="318670707" role="" />
975   <member type="way" ref="4623313" role="" />
976   <member type="node" ref="3246190446" role="" />
977   <member type="node" ref="1013407012" role="" />
978   <member type="way" ref="318670708" role="" />
979   <member type="node" ref="29391345" role="stop" />
980   <tag k="name" v="Rute_Angkot_Stasiun_Hall_--Lembang" />
981   <tag k="network" v="Angkot_Lintas_Kota_Bandung" />
982   <tag k="ref" v="St.Hall-Lembang" />
983   <tag k="route" v="bus" />
984   <tag k="type" v="route" />
985   <tag k="wikipedia" v="id : Daftar_Angkutan_Umum_di_Kota_Bandung" />
986 </relation>
987 <relation id="4420962" visible="true" version="5" changeset="27744522" timestamp="2014-12-28T01:11:33Z"
  user="gnocin" uid="2526082">
<member type="way" ref="153106229" role="" />

```

```

989 <member type="way" ref="4623276" role="" />
990 <member type="way" ref="153106230" role="" />
991 <member type="way" ref="318670697" role="" />
992 <member type="way" ref="292276222" role="" />
993 <member type="way" ref="318531181" role="" />
994 <member type="way" ref="318670699" role="" />
995 <member type="way" ref="318478988" role="" />
996 <member type="way" ref="223676540" role="" />
997 <member type="way" ref="292274152" role="" />
998 <member type="way" ref="318670698" role="" />
999 <member type="way" ref="318531182" role="" />
1000 <member type="way" ref="292276221" role="" />
1001 <member type="way" ref="318531184" role="" />
1002 <member type="way" ref="318531179" role="" />
1003 <member type="way" ref="318670695" role="" />
1004 <member type="way" ref="292276706" role="" />
1005 <member type="way" ref="4623249" role="" />
1006 <member type="way" ref="318531180" role="" />
1007 <member type="way" ref="4623250" role="" />
1008 <member type="way" ref="4256037" role="" />
1009 <member type="way" ref="318222725" role="" />
1010 <member type="way" ref="319068174" role="" />
1011 <member type="way" ref="4251274" role="" />
1012 <member type="way" ref="4567655" role="" />
1013 <member type="way" ref="4567635" role="" />
1014 <member type="way" ref="190605660" role="" />
1015 <member type="node" ref="3246190446" role="" />
1016 <member type="way" ref="32360540" role="" />
1017 <member type="way" ref="32360546" role="" />
1018 <member type="way" ref="4567653" role="" />
1019 <member type="way" ref="4567659" role="" />
1020 <member type="way" ref="318222717" role="" />
1021 <member type="way" ref="87148402" role="" />
1022 <member type="node" ref="1013407012" role="" />
1023 <member type="way" ref="257227462" role="" />
1024 <member type="way" ref="318104555" role="" />
1025 <member type="way" ref="319194428" role="" />
1026 <member type="way" ref="4627124" role="" />
1027 <member type="way" ref="4627123" role="" />
1028 <member type="way" ref="318670707" role="" />
1029 <member type="way" ref="4623313" role="" />
1030 <member type="way" ref="193626591" role="" />
1031 <member type="way" ref="32388789" role="" />
1032 <member type="way" ref="4567652" role="" />
1033 <member type="way" ref="257227478" role="" />
1034 <member type="way" ref="190605653" role="" />
1035 <member type="way" ref="190605655" role="" />
1036 <member type="way" ref="55780163" role="" />
1037 <member type="way" ref="55783893" role="" />
1038 <member type="way" ref="143954415" role="" />
1039 <member type="way" ref="95507069" role="" />
1040 <member type="node" ref="3249368133" role="" />
1041 <member type="node" ref="3250415165" role="" />
1042 <tag k="name" v="Rute_Angkot_13._Stasiun_Hall_--_Sarijadi" />
1043 <tag k="network" v="Angkot_Kota_Bandung" />
1044 <tag k="ref" v="13" />
1045 <tag k="route" v="bus" />
1046 <tag k="type" v="route" />
1047 <tag k="wikipedia" v="id:Daftar_Angkutan_Umum_di_Kota_Bandung" />
1048 </relation>
<relation id="4420961" visible="true" version="7" changeset="28001835" timestamp="2015-01-08T17:46:26Z"
  user="isonpurba" uid="2552445">
1049 <member type="node" ref="3250402136" role="" />
1050 <member type="way" ref="318670693" role="" />
1051 <member type="way" ref="125813777" role="" />
1052 <member type="way" ref="125813784" role="" />
1053 <member type="way" ref="125813781" role="" />
1054 <member type="way" ref="125813778" role="" />
1055 <member type="way" ref="4625407" role="" />
1056 <member type="way" ref="318531173" role="" />
1057 <member type="way" ref="292276703" role="" />
1058 <member type="way" ref="292276219" role="" />
1059 <member type="way" ref="318670697" role="" />
1060 <member type="way" ref="292276222" role="" />
1061 <member type="way" ref="318531181" role="" />
1062 <member type="way" ref="318670700" role="" />
1063 <member type="way" ref="318670702" role="" />
1064 <member type="way" ref="318670704" role="" />
1065 <member type="way" ref="179614119" role="" />
1066 <member type="way" ref="153180024" role="" />
1067 <member type="way" ref="292276224" role="" />
1068 <member type="way" ref="153134040" role="" />
1069 <member type="way" ref="165804161" role="" />
1070 <member type="way" ref="318531179" role="" />
1071 <member type="way" ref="318531184" role="" />
1072 <member type="way" ref="292276221" role="" />
1073 <member type="way" ref="318531182" role="" />
1074 <member type="way" ref="318670698" role="" />
1075 <member type="way" ref="186246671" role="" />
1076 <member type="way" ref="192575395" role="" />
1077 <member type="way" ref="192575392" role="" />
1078 <member type="way" ref="4567534" role="" />
1079 <member type="way" ref="179614116" role="" />
1080 <member type="way" ref="318670706" role="" />
1081 <member type="way" ref="4567524" role="" />
1082 <member type="way" ref="247058986" role="" />
1083 <member type="way" ref="318478989" role="" />
1084 <member type="way" ref="292668836" role="" />
1085 <member type="way" ref="4567490" role="" />
1086

```

```

1087 <member type="way" ref="168569217" role="" />
1088 <member type="way" ref="292668468" role="" />
1089 <member type="way" ref="247058988" role="" />
1090 <member type="way" ref="247058985" role="" />
1091 <member type="way" ref="4567491" role="" />
1092 <member type="way" ref="168569234" role="" />
1093 <member type="way" ref="4256015" role="" />
1094 <member type="way" ref="4251274" role="" />
1095 <member type="way" ref="4567635" role="" />
1096 <member type="node" ref="3246190446" role="" />
1097 <member type="way" ref="32360540" role="" />
1098 <member type="way" ref="32360546" role="" />
1099 <member type="way" ref="4567633" role="" />
1100 <member type="way" ref="4567659" role="" />
1101 <member type="way" ref="318222717" role="" />
1102 <member type="way" ref="87148402" role="" />
1103 <member type="node" ref="1013407012" role="" />
1104 <member type="way" ref="257227462" role="" />
1105 <member type="way" ref="318104555" role="" />
1106 <member type="way" ref="319194428" role="" />
1107 <member type="way" ref="4627124" role="" />
1108 <member type="way" ref="4627123" role="" />
1109 <member type="way" ref="318670707" role="" />
1110 <member type="way" ref="4623313" role="" />
1111 <member type="way" ref="32388790" role="" />
1112 <member type="way" ref="4623315" role="" />
1113 <member type="way" ref="318896304" role="" />
1114 <member type="way" ref="318479413" role="" />
1115 <member type="way" ref="235830733" role="" />
1116 <member type="way" ref="292668884" role="" />
1117 <member type="way" ref="318233616" role="" />
1118 <member type="node" ref="3249368133" role="" />
1119 <tag k="name" v="Rute_Angkot_14._Stasiun_Hall_Gunung_Batu" />
1120 <tag k="network" v="Angkot_Kota_Bandung" />
1121 <tag k="ref" v="14" />
1122 <tag k="route" v="bus" />
1123 <tag k="type" v="route" />
1124 <tag k="wikipedia" v="id : Daftar_Angkutan_Umum_di_Kota_Bandung" />
1125 </relation>
1126 <relation id="4294474" visible="true" version="12" changeset="28005713" timestamp="2015-01-08T20:32:47Z"
    user="isonpurba" uid="2552445">
1127   <member type="node" ref="3245971834" role="" />
1128   <member type="way" ref="158851159" role="" />
1129   <member type="way" ref="244646404" role="" />
1130   <member type="way" ref="244646402" role="" />
1131   <member type="way" ref="127343945" role="" />
1132   <member type="way" ref="192917221" role="" />
1133   <member type="way" ref="28410794" role="" />
1134   <member type="way" ref="4910295" role="" />
1135   <member type="way" ref="153340105" role="" />
1136   <member type="way" ref="168549935" role="" />
1137   <member type="way" ref="168549936" role="" />
1138   <member type="way" ref="318222714" role="" />
1139   <member type="way" ref="192917040" role="" />
1140   <member type="way" ref="318054780" role="" />
1141   <member type="way" ref="192917222" role="" />
1142   <member type="way" ref="318222716" role="" />
1143   <member type="way" ref="119439452" role="" />
1144   <member type="node" ref="3240012631" role="" />
1145   <member type="way" ref="4698220" role="" />
1146   <member type="way" ref="166687310" role="" />
1147   <member type="way" ref="4567624" role="" />
1148   <member type="way" ref="318054782" role="" />
1149   <member type="way" ref="319083572" role="" />
1150   <member type="way" ref="319083569" role="" />
1151   <member type="way" ref="292274154" role="" />
1152   <member type="way" ref="247060527" role="" />
1153   <member type="way" ref="321034670" role="" />
1154   <member type="way" ref="292668612" role="" />
1155   <member type="way" ref="292668641" role="" />
1156   <member type="way" ref="321030145" role="" />
1157   <member type="way" ref="292668470" role="" />
1158   <member type="way" ref="4567491" role="" />
1159   <member type="way" ref="168569234" role="" />
1160   <member type="way" ref="4256015" role="" />
1161   <member type="way" ref="168569217" role="" />
1162   <member type="way" ref="318222725" role="" />
1163   <member type="way" ref="319068174" role="" />
1164   <member type="way" ref="318670708" role="" />
1165   <member type="way" ref="4567655" role="" />
1166   <member type="way" ref="193626591" role="" />
1167   <member type="way" ref="32360540" role="" />
1168   <member type="way" ref="32360546" role="" />
1169   <member type="way" ref="4567633" role="" />
1170   <member type="way" ref="4567659" role="" />
1171   <member type="way" ref="4627139" role="backward" />
1172   <member type="way" ref="318222718" role="" />
1173   <member type="way" ref="318222719" role="" />
1174   <member type="way" ref="215583979" role="" />
1175   <member type="way" ref="318191410" role="" />
1176   <member type="way" ref="318191412" role="" />
1177   <member type="way" ref="318222722" role="" />
1178   <member type="node" ref="3246190446" role="" />
1179   <member type="node" ref="3246190447" role="" />
1180   <member type="way" ref="306163524" role="" />
1181   <member type="way" ref="318222723" role="" />
1182   <member type="way" ref="318191414" role="" />
1183   <member type="way" ref="318191415" role="" />
1184   <member type="way" ref="318191416" role="" />

```

```

1185 <member type="way" ref="4567660" role="" />
1186 <member type="way" ref="4567705" role="" />
1187 <member type="way" ref="318222717" role="" />
1188 <member type="way" ref="318104555" role="" />
1189 <member type="way" ref="319194428" role="" />
1190 <member type="way" ref="4627124" role="" />
1191 <member type="way" ref="4627123" role="" />
1192 <member type="way" ref="318670707" role="" />
1193 <member type="way" ref="257227459" role="" />
1194 <member type="way" ref="257227462" role="" />
1195 <member type="way" ref="4623313" role="" />
1196 <member type="node" ref="364363784" role="" />
1197 <member type="way" ref="32388789" role="" />
1198 <member type="way" ref="4567652" role="" />
1199 <member type="way" ref="257227478" role="" />
1200 <member type="way" ref="4251274" role="" />
1201 <member type="way" ref="4567490" role="" />
1202 <member type="way" ref="235830734" role="" />
1203 <member type="way" ref="318222726" role="backward" />
1204 <member type="way" ref="318729562" role="" />
1205 <member type="way" ref="318729561" role="" />
1206 <member type="way" ref="174270181" role="" />
1207 <member type="way" ref="186246670" role="" />
1208 <member type="way" ref="4623279" role="" />
1209 <member type="way" ref="153315568" role="" />
1210 <member type="way" ref="153315566" role="" />
1211 <member type="way" ref="318729560" role="" />
1212 <member type="way" ref="4623278" role="" />
1213 <member type="node" ref="1013407012" role="" />
1214 <tag k="name" v="Rute_Angkot_26._Cisitu_-Tegalega" />
1215 <tag k="network" v="Angkot_Kota_Bandung" />
1216 <tag k="ref" v="26" />
1217 <tag k="route" v="bus" />
1218 <tag k="type" v="route" />
1219 <tag k="wikipedia" v="id : Daftar_Angkutan_Umum_di_Kota_Bandung" />
1220 </relation>
1221 <relation id="4423238" visible="true" version="12" changeset="28004961" timestamp="2015-01-08T20:02:58Z"
    user="isonpurba" uid="2552445">
1222   <member type="node" ref="3240012632" role="stop" />
1223   <member type="way" ref="168942781" role="" />
1224   <member type="way" ref="318756527" role="" />
1225   <member type="way" ref="28446157" role="" />
1226   <member type="way" ref="242633052" role="" />
1227   <member type="way" ref="242633051" role="" />
1228   <member type="way" ref="28446283" role="" />
1229   <member type="way" ref="318756523" role="" />
1230   <member type="way" ref="319191948" role="" />
1231   <member type="way" ref="162812231" role="" />
1232   <member type="way" ref="318760037" role="" />
1233   <member type="node" ref="3248882521" role="stop" />
1234   <member type="way" ref="153315564" role="" />
1235   <member type="way" ref="153517385" role="" />
1236   <member type="way" ref="318054779" role="" />
1237   <member type="way" ref="319191949" role="" />
1238   <member type="way" ref="292274159" role="" />
1239   <member type="way" ref="4623279" role="" />
1240   <member type="way" ref="241791694" role="" />
1241   <member type="way" ref="153483321" role="" />
1242   <member type="way" ref="186246670" role="" />
1243   <member type="way" ref="174270181" role="" />
1244   <member type="way" ref="174270180" role="" />
1245   <member type="way" ref="299081906" role="" />
1246   <member type="way" ref="174270184" role="" />
1247   <member type="way" ref="174270182" role="" />
1248   <member type="way" ref="318729562" role="" />
1249   <member type="way" ref="318231948" role="" />
1250   <member type="way" ref="174270183" role="" />
1251   <member type="way" ref="4255986" role="" />
1252   <member type="way" ref="4255981" role="" />
1253   <member type="way" ref="318222726" role="" />
1254   <member type="way" ref="4255951" role="" />
1255   <member type="way" ref="318478990" role="" />
1256   <member type="node" ref="3246270248" role="" />
1257   <member type="way" ref="4255990" role="" />
1258   <member type="way" ref="32388792" role="" />
1259   <member type="way" ref="318233616" role="" />
1260   <member type="way" ref="4256015" role="" />
1261   <member type="way" ref="168569217" role="" />
1262   <member type="way" ref="4567490" role="" />
1263   <member type="way" ref="4567655" role="" />
1264   <member type="way" ref="4251274" role="" />
1265   <member type="way" ref="319068174" role="" />
1266   <member type="way" ref="292668470" role="" />
1267   <member type="way" ref="292668641" role="" />
1268   <member type="way" ref="321030145" role="" />
1269   <member type="way" ref="292668837" role="" />
1270   <member type="way" ref="318222725" role="" />
1271   <member type="way" ref="318670708" role="" />
1272   <member type="way" ref="4567652" role="" />
1273   <member type="way" ref="257227478" role="" />
1274   <member type="way" ref="190605653" role="" />
1275   <member type="way" ref="190605650" role="" />
1276   <member type="way" ref="192031850" role="" />
1277   <member type="way" ref="318804216" role="" />
1278   <member type="way" ref="4569062" role="" />
1279   <member type="way" ref="4569064" role="" />
1280   <member type="way" ref="318891912" role="" />
1281   <member type="way" ref="318891913" role="" />
1282   <member type="way" ref="4625292" role="" />

```

```

1283 <member type="way" ref="318891914" role="" />
1284 <member type="way" ref="318891916" role="" />
1285 <member type="way" ref="318891920" role="" />
1286 <member type="way" ref="318891919" role="" />
1287 <member type="way" ref="318891915" role="" />
1288 <member type="way" ref="4567662" role="" />
1289 <member type="way" ref="257229251" role="" />
1290 <member type="way" ref="257229252" role="" />
1291 <member type="node" ref="3246190446" role="" />
1292 <member type="node" ref="3249368131" role="stop" />
1293 <member type="way" ref="257232271" role="" />
1294 <member type="way" ref="318891917" role="" />
1295 <member type="way" ref="318891918" role="" />
1296 <member type="way" ref="180710757" role="" />
1297 <member type="way" ref="318891933" role="" />
1298 <member type="way" ref="318891935" role="" />
1299 <member type="way" ref="318891928" role="" />
1300 <member type="way" ref="318891925" role="" />
1301 <member type="way" ref="223676554" role="" />
1302 <member type="way" ref="4625286" role="" />
1303 <member type="way" ref="4625283" role="" />
1304 <member type="way" ref="4625281" role="" />
1305 <member type="way" ref="4625256" role="" />
1306 <member type="node" ref="3244745033" role="stop" />
1307 <member type="way" ref="223676441" role="" />
1308 <member type="way" ref="223676380" role="" />
1309 <member type="way" ref="318891923" role="" />
1310 <member type="way" ref="318891932" role="" />
1311 <member type="way" ref="318891930" role="" />
1312 <member type="way" ref="318891934" role="" />
1313 <member type="node" ref="3252877763" role="stop" />
1314 <member type="way" ref="172898300" role="" />
1315 <member type="node" ref="3248969441" role="stop" />
1316 <member type="node" ref="1662376687" role="stop" />
1317 <member type="node" ref="3262619802" role="stop" />
1318 <tag k="name" v="Rute_Angkot_17._Dago_-Pasar_Induk_Caringin" />
1319 <tag k="network" v="Angkot_Kota_Bandung" />
1320 <tag k="ref" v="17" />
1321 <tag k="route" v="bus" />
1322 <tag k="type" v="route" />
1323 <tag k="wikipedia" v="id:Daftar_Angkutan_Umum_di_Kota_Bandung" />
1324 </relation>
1325 <relation id="4425899" visible="true" version="3" changeset="27740065" timestamp="2014-12-27T20:46:39Z"
    user="gnocin" uid="2526082">
1326   <member type="node" ref="3249461527" role="stop" />
1327   <member type="way" ref="4626653" role="" />
1328   <member type="way" ref="319083571" role="" />
1329   <member type="way" ref="4716961" role="backward" />
1330   <member type="way" ref="4716962" role="" />
1331   <member type="way" ref="4623260" role="" />
1332   <member type="way" ref="247064779" role="" />
1333   <member type="way" ref="223676546" role="" />
1334   <member type="way" ref="318670699" role="" />
1335   <member type="way" ref="318478988" role="" />
1336   <member type="way" ref="223676540" role="" />
1337   <member type="way" ref="318670701" role="" />
1338   <member type="way" ref="4567635" role="" />
1339   <member type="way" ref="318892821" role="" />
1340   <member type="way" ref="190605660" role="" />
1341   <member type="way" ref="190605650" role="" />
1342   <member type="way" ref="190605650" role="" />
1343   <member type="way" ref="192031850" role="" />
1344   <member type="way" ref="318804217" role="" />
1345   <member type="way" ref="257232274" role="" />
1346   <member type="way" ref="257232272" role="" />
1347   <member type="way" ref="318891912" role="" />
1348   <member type="way" ref="318100679" role="" />
1349   <member type="way" ref="190605663" role="" />
1350   <member type="way" ref="190605657" role="" />
1351   <member type="way" ref="4567636" role="" />
1352   <member type="way" ref="239929577" role="" />
1353   <member type="way" ref="4569058" role="" />
1354   <member type="node" ref="3249368131" role="" />
1355   <member type="way" ref="257232271" role="" />
1356   <member type="way" ref="257232273" role="" />
1357   <member type="way" ref="318231955" role="" />
1358   <member type="way" ref="318892822" role="" />
1359   <member type="way" ref="318231951" role="" />
1360   <member type="way" ref="4627108" role="" />
1361   <member type="way" ref="4567685" role="" />
1362   <member type="way" ref="318193811" role="" />
1363   <member type="way" ref="318191421" role="" />
1364   <member type="way" ref="318193812" role="" />
1365   <member type="way" ref="318231952" role="" />
1366   <member type="way" ref="4567680" role="" />
1367   <member type="node" ref="2352383137" role="stop" />
1368   <member type="node" ref="3252892639" role="stop" />
1369   <member type="node" ref="29392015" role="stop" />
1370   <tag k="name" v="Rute_Angkot_23._Sederhana_-Cijerah" />
1371   <tag k="network" v="Angkot_Kota_Bandung" />
1372   <tag k="ref" v="23" />
1373   <tag k="route" v="bus" />
1374   <tag k="type" v="route" />
1375   <tag k="wikipedia" v="id:Daftar_Angkutan_Umum_di_Kota_Bandung" />
1376 </relation>
1377 <relation id="4291189" visible="true" version="22" changeset="28769426" timestamp="2015-02-11T10:05:22Z"
    user="shravan91" uid="1051550">
1378   <member type="way" ref="4623277" role="" />
1379   <member type="way" ref="318222716" role="" />

```

```

1380 <member type="way" ref="318222715" role="" />
1381 <member type="way" ref="318222714" role="" />
1382 <member type="way" ref="318054781" role="" />
1383 <member type="way" ref="318756515" role="" />
1384 <member type="way" ref="175071635" role="" />
1385 <member type="way" ref="153340105" role="" />
1386 <member type="way" ref="168549935" role="" />
1387 <member type="way" ref="168549936" role="" />
1388 <member type="way" ref="4698220" role="" />
1389 <member type="way" ref="166687310" role="" />
1390 <member type="way" ref="4567624" role="" />
1391 <member type="way" ref="318054782" role="" />
1392 <member type="way" ref="319083572" role="" />
1393 <member type="way" ref="319083569" role="" />
1394 <member type="way" ref="292274154" role="" />
1395 <member type="way" ref="153175500" role="" />
1396 <member type="way" ref="292668298" role="" />
1397 <member type="way" ref="292668301" role="" />
1398 <member type="way" ref="292668297" role="" />
1399 <member type="way" ref="318054783" role="" />
1400 <member type="way" ref="318478987" role="" />
1401 <member type="way" ref="318102560" role="" />
1402 <member type="way" ref="153305903" role="" />
1403 <member type="way" ref="192917039" role="" />
1404 <member type="way" ref="318054780" role="" />
1405 <member type="way" ref="4910295" role="" />
1406 <member type="way" ref="192917041" role="" />
1407 <member type="way" ref="318729557" role="" />
1408 <member type="way" ref="4623302" role="" />
1409 <member type="way" ref="175071632" role="" />
1410 <member type="way" ref="318729556" role="" />
1411 <member type="way" ref="175071643" role="" />
1412 <member type="node" ref="3240012631" role="" />
1413 <member type="way" ref="175071646" role="" />
1414 <member type="way" ref="318100671" role="" />
1415 <member type="way" ref="318102033" role="" />
1416 <member type="way" ref="318051978" role="" />
1417 <member type="way" ref="153315564" role="" />
1418 <member type="way" ref="153517385" role="" />
1419 <member type="way" ref="241791694" role="" />
1420 <member type="way" ref="153483321" role="" />
1421 <member type="way" ref="4627223" role="" />
1422 <member type="way" ref="318481954" role="" />
1423 <member type="way" ref="32360437" role="" />
1424 <member type="way" ref="239944112" role="" />
1425 <member type="way" ref="32360440" role="" />
1426 <member type="way" ref="320405968" role="" />
1427 <member type="way" ref="320605375" role="" />
1428 <member type="node" ref="2352378157" role="" />
1429 <member type="way" ref="318054779" role="" />
1430 <member type="way" ref="319191949" role="" />
1431 <member type="way" ref="318053623" role="" />
1432 <member type="way" ref="318053626" role="" />
1433 <member type="way" ref="4632671" role="" />
1434 <member type="way" ref="318053625" role="" />
1435 <member type="way" ref="318100669" role="" />
1436 <member type="way" ref="175071657" role="" />
1437 <member type="way" ref="292668296" role="" />
1438 <member type="way" ref="318100677" role="" />
1439 <member type="way" ref="4625730" role="" />
1440 <member type="way" ref="4625729" role="" />
1441 <member type="way" ref="4716962" role="" />
1442 <member type="way" ref="4623260" role="" />
1443 <member type="way" ref="247064779" role="" />
1444 <member type="way" ref="223676546" role="" />
1445 <member type="way" ref="318670699" role="" />
1446 <member type="way" ref="318478988" role="" />
1447 <member type="way" ref="190605660" role="" />
1448 <member type="way" ref="4567635" role="" />
1449 <member type="way" ref="318892821" role="" />
1450 <member type="way" ref="4716961" role="" />
1451 <member type="way" ref="4716963" role="" />
1452 <member type="way" ref="190605650" role="" />
1453 <member type="way" ref="190605663" role="" />
1454 <member type="way" ref="190605657" role="" />
1455 <member type="way" ref="4567636" role="" />
1456 <member type="way" ref="192031850" role="" />
1457 <member type="way" ref="318804217" role="" />
1458 <member type="way" ref="257232274" role="" />
1459 <member type="way" ref="257232272" role="" />
1460 <member type="way" ref="318891912" role="" />
1461 <member type="way" ref="318891913" role="" />
1462 <member type="way" ref="318891922" role="" />
1463 <member type="way" ref="4625281" role="" />
1464 <member type="way" ref="4625254" role="" />
1465 <member type="way" ref="4569057" role="" />
1466 <member type="way" ref="257232271" role="" />
1467 <member type="node" ref="3244745033" role="stop" />
1468 <member type="way" ref="257229926" role="stop" />
1469 <member type="node" ref="3244754744" role="" />
1470 <member type="node" ref="3248969441" role="stop" />
1471 <member type="node" ref="3249368131" role="" />
1472 <member type="node" ref="3251007172" role="stop" />
1473 <tag k="name" v="Rute_Angkot_06._Cicaheum-Ciroyom" />
1474 <tag k="network" v="Angkot_Kota_Bandung" />
1475 <tag k="ref" v="06" />
1476 <tag k="route" v="bus" />
1477 <tag k="type" v="route" />
1478 <tag k=".wikipedia" v="id : Daftar_Angkutan_Umum_di_Kota_Bandung" />

```

```

1479 </relation>
1480 <relation id="4430413" visible="true" version="4" changeset="28005713" timestamp="2015-01-08T20:32:48Z"
1481   user="isonpurba" uid="2552445">
1482   <member type="node" ref="3254870563" role="stop"/>
1483   <member type="way" ref="319068173" role="stop"/>
1484   <member type="way" ref="222733920" role="" />
1485   <member type="way" ref="29064646" role="" />
1486   <member type="way" ref="222733921" role="" />
1487   <member type="way" ref="222733919" role="" />
1488   <member type="way" ref="319083573" role="" />
1489   <member type="way" ref="153306178" role="" />
1490   <member type="way" ref="153305903" role="" />
1491   <member type="way" ref="318102560" role="" />
1492   <member type="way" ref="318478987" role="" />
1493   <member type="way" ref="318054783" role="" />
1494   <member type="way" ref="292668297" role="" />
1495   <member type="way" ref="292668301" role="" />
1496   <member type="way" ref="292668298" role="" />
1497   <member type="way" ref="4567617" role="" />
1498   <member type="way" ref="247058986" role="" />
1499   <member type="way" ref="166687310" role="" />
1500   <member type="way" ref="4567624" role="" />
1501   <member type="way" ref="318054782" role="" />
1502   <member type="way" ref="319083572" role="" />
1503   <member type="way" ref="319083569" role="" />
1504   <member type="way" ref="292274154" role="" />
1505   <member type="way" ref="247060527" role="" />
1506   <member type="way" ref="321034670" role="" />
1507   <member type="way" ref="4567623" role="" />
1508   <member type="way" ref="247058988" role="" />
1509   <member type="way" ref="247058985" role="" />
1510   <member type="way" ref="4567527" role="backward"/>
1511   <member type="way" ref="4567529" role="backward"/>
1512   <member type="way" ref="319068174" role="" />
1513   <member type="way" ref="4251274" role="" />
1514   <member type="way" ref="4567655" role="" />
1515   <member type="way" ref="4567490" role="" />
1516   <member type="way" ref="292668468" role="" />
1517   <member type="way" ref="4567502" role="" />
1518   <member type="way" ref="4567652" role="" />
1519   <member type="way" ref="257227478" role="" />
1520   <member type="way" ref="318670708" role="" />
1521   <member type="way" ref="318222725" role="" />
1522   <member type="way" ref="193626591" role="" />
1523   <member type="way" ref="32388789" role="" />
1524   <member type="way" ref="4623313" role="" />
1525   <member type="node" ref="3246190446" role="stop"/>
1526   <member type="way" ref="32360540" role="" />
1527   <member type="way" ref="32360546" role="" />
1528   <member type="way" ref="4567653" role="" />
1529   <member type="way" ref="4567659" role="" />
1530   <member type="way" ref="318222717" role="" />
1531   <member type="way" ref="87148402" role="" />
1532   <member type="way" ref="257227468" role="stop"/>
1533   <member type="way" ref="257227462" role="stop"/>
1534   <member type="node" ref="1228645006" role="stop"/>
1535   <member type="way" ref="4627123" role="" />
1536   <member type="way" ref="318670707" role="" />
1537   <member type="way" ref="4627122" role="" />
1538   <member type="node" ref="1013407012" role="stop"/>
1539   <tag k="name" v="Rute_Angkot_11B,_Stasiun_Hall,_Ciumbeuleuit_via_Cihampelas,&quot;LURUS&quot; ; "/>
1540   <tag k="network" v="Angkot_Kota_Bandung"/>
1541   <tag k="ref" v="11B"/>
1542   <tag k="route" v="bus"/>
1543   <tag k="type" v="route"/>
1544   <tag k="wikipedia" v="id : Daftar_Angkutan_Umum_di_Kota_Bandung"/>
1545 </relation>
<relation id="4430737" visible="true" version="3" changeset="28005713" timestamp="2015-01-08T20:32:48Z"
1546   user="isonpurba" uid="2552445">
1547   <member type="way" ref="32360540" role="" />
1548   <member type="way" ref="32360546" role="" />
1549   <member type="way" ref="4567653" role="" />
1550   <member type="way" ref="4567659" role="" />
1551   <member type="way" ref="318222717" role="" />
1552   <member type="way" ref="87148402" role="" />
1553   <member type="way" ref="4627123" role="" />
1554   <member type="way" ref="318670707" role="" />
1555   <member type="way" ref="4623313" role="" />
1556   <member type="way" ref="4627122" role="" />
1557   <member type="node" ref="1013407012" role="stop"/>
1558   <member type="way" ref="257227462" role="stop"/>
1559   <member type="way" ref="4251274" role="" />
1560   <member type="way" ref="319068174" role="" />
1561   <member type="way" ref="318222725" role="" />
1562   <member type="way" ref="4567655" role="" />
1563   <member type="way" ref="193626591" role="" />
1564   <member type="way" ref="32388789" role="" />
1565   <member type="way" ref="4567652" role="" />
1566   <member type="way" ref="257227478" role="" />
1567   <member type="way" ref="318670708" role="" />
1568   <member type="way" ref="4567490" role="" />
1569   <member type="way" ref="292668468" role="" />
1570   <member type="way" ref="247058988" role="" />
1571   <member type="way" ref="4567502" role="" />
1572   <member type="way" ref="247058986" role="" />
1573   <member type="way" ref="4567617" role="" />
1574   <member type="way" ref="292668301" role="" />
1575   <member type="way" ref="292668297" role="" />
```

```

1576 <member type="way" ref="318054783" role="" />
1577 <member type="way" ref="318478987" role="" />
1578 <member type="way" ref="318102560" role="" />
1579 <member type="way" ref="153305903" role="" />
1580 <member type="way" ref="319083573" role="" />
1581 <member type="way" ref="153306178" role="" />
1582 <member type="way" ref="247058985" role="" />
1583 <member type="way" ref="4567527" role="backward" />
1584 <member type="way" ref="4567529" role="backward" />
1585 <member type="way" ref="4625729" role="" />
1586 <member type="way" ref="319083571" role="" />
1587 <member type="way" ref="4625722" role="" />
1588 <member type="way" ref="4627197" role="" />
1589 <member type="way" ref="318054782" role="" />
1590 <member type="way" ref="4567624" role="" />
1591 <member type="way" ref="166687310" role="" />
1592 <member type="way" ref="4716963" role="" />
1593 <member type="way" ref="318100677" role="" />
1594 <member type="way" ref="179613937" role="" />
1595 <member type="way" ref="319083569" role="" />
1596 <member type="way" ref="292274154" role="" />
1597 <member type="way" ref="247060527" role="" />
1598 <member type="way" ref="321034670" role="" />
1599 <member type="way" ref="4567623" role="" />
1600 <member type="node" ref="3249461527" role="stop" />
1601 <member type="way" ref="222733919" role="" />
1602 <member type="way" ref="222733920" role="" />
1603 <member type="way" ref="29064646" role="" />
1604 <member type="node" ref="3254870563" role="stop" />
1605 <member type="node" ref="29392015" role="stop" />
1606 <tag k="name" v="Rute_Angkot_11A,_Stasiun_Hall,_Ciumbeuleuit_via_Eyckman," ;BELOK"" />
1607 <tag k="network" v="Angkot_Kota_Bandung" />
1608 <tag k="ref" v="11A" />
1609 <tag k="route" v="bus" />
1610 <tag k="type" v="route" />
1611 <tag k="wikipedia" v="id:Daftar_Angkutan_Umum_di_Kota_Bandung" />
1612 </relation>
1613 <relation id="4432958" visible="true" version="4" changeset="28004961" timestamp="2015-01-08T20:02:59Z"
    user="isonpurba" uid="2552445">
1614 <member type="node" ref="3256049561" role="stop" />
1615 <member type="way" ref="319191947" role="" />
1616 <member type="way" ref="319191948" role="" />
1617 <member type="way" ref="162812231" role="" />
1618 <member type="way" ref="318760037" role="" />
1619 <member type="way" ref="159457434" role="" />
1620 <member type="way" ref="318231944" role="" />
1621 <member type="way" ref="175071635" role="" />
1622 <member type="way" ref="192917041" role="" />
1623 <member type="way" ref="318729557" role="" />
1624 <member type="way" ref="4623302" role="" />
1625 <member type="way" ref="175071632" role="" />
1626 <member type="way" ref="4623282" role="" />
1627 <member type="way" ref="318732962" role="" />
1628 <member type="way" ref="318732963" role="" />
1629 <member type="way" ref="319191949" role="" />
1630 <member type="way" ref="292274159" role="" />
1631 <member type="way" ref="4623279" role="" />
1632 <member type="node" ref="3240012632" role="stop" />
1633 <member type="node" ref="1662376687" role="stop" />
1634 <member type="way" ref="318233618" role="" />
1635 <member type="node" ref="3240012631" role="stop" />
1636 <member type="node" ref="3251007172" role="stop" />
1637 <member type="way" ref="153315568" role="" />
1638 <member type="way" ref="4633273" role="" />
1639 <member type="way" ref="186246670" role="" />
1640 <member type="way" ref="174270181" role="" />
1641 <member type="way" ref="174270180" role="" />
1642 <member type="way" ref="299081906" role="" />
1643 <member type="way" ref="174270184" role="" />
1644 <member type="way" ref="174270182" role="" />
1645 <member type="way" ref="318729561" role="" />
1646 <member type="way" ref="318729562" role="" />
1647 <member type="way" ref="318222726" role="" />
1648 <member type="way" ref="4255951" role="" />
1649 <member type="way" ref="318478990" role="" />
1650 <member type="way" ref="4255981" role="" />
1651 <member type="way" ref="4255990" role="" />
1652 <member type="way" ref="32388792" role="" />
1653 <member type="way" ref="318233616" role="" />
1654 <member type="way" ref="4256015" role="" />
1655 <member type="way" ref="168569217" role="" />
1656 <member type="way" ref="4251274" role="" />
1657 <member type="way" ref="4567655" role="" />
1658 <member type="node" ref="3246270248" role="stop" />
1659 <member type="way" ref="318100678" role="" />
1660 <member type="way" ref="4256037" role="" />
1661 <member type="way" ref="318222725" role="" />
1662 <member type="way" ref="319068174" role="" />
1663 <member type="way" ref="4567490" role="" />
1664 <member type="way" ref="292668641" role="" />
1665 <member type="way" ref="321030145" role="" />
1666 <member type="way" ref="292668470" role="" />
1667 <member type="way" ref="235830734" role="" />
1668 <member type="way" ref="4567652" role="" />
1669 <member type="way" ref="257227478" role="" />
1670 <member type="way" ref="190605653" role="" />
1671 <member type="node" ref="3246190446" role="stop" />
1672 <member type="way" ref="190605650" role="" />
1673 <member type="way" ref="192031850" role="" />
```

```
1674 | <member type="way" ref="318804217" role="" />
1675 | <member type="way" ref="257232274" role="" />
1676 | <member type="way" ref="257232272" role="" />
1677 | <member type="way" ref="318891912" role="" />
1678 | <member type="way" ref="318891913" role="" />
1679 | <member type="way" ref="318891922" role="" />
1680 | <member type="way" ref="4625281" role="" />
1681 | <member type="way" ref="4625254" role="" />
1682 | <member type="way" ref="4569057" role="" />
1683 | <member type="node" ref="3244745033" role="stop" />
1684 | <member type="way" ref="257229926" role="stop" />
1685 | <member type="node" ref="3262619802" role="stop" />
1686 | <tag k="name" v="Rute_Angkot_Ciburial_(Dago_Atas)---_Ciroyom" />
1687 | <tag k="network" v="Angkot_Lintas_Kota_Bandung" />
1688 | <tag k="ref" v="Ciburial-Ciroyom" />
1689 | <tag k="route" v="bus" />
1690 | <tag k="source" v="http://www.transportasiumum.com/content/rute-angkot-bandung" />
1691 | <tag k="type" v="route" />
1692 | </relation>
1693 | </osm>
```