

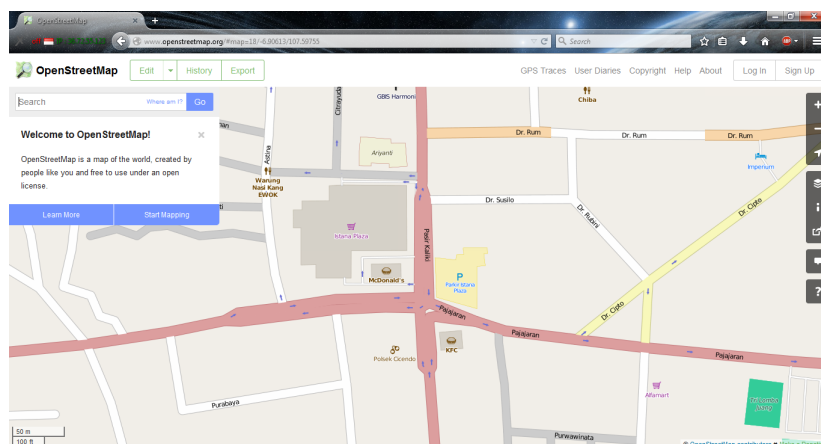
BAB 1

PENDAHULUAN

1.1 Latar Belakang

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

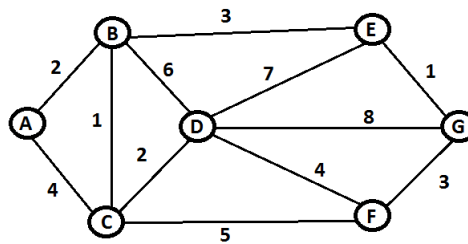
Aplikasi yang dibuat akan berbasis OpenStreetMap dan menggunakan algoritma Dijkstra. OpenStreetMap adalah portal peta terbuka yang menyediakan data dalam bentuk peta maupun XML, pengguna dapat mencari lokasi dan memilih area yang diinginkan. Setelah pengguna memilih area yang diinginkan, pengguna dapat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



Gambar 1.1: Tampilan website OpenStreetMap ¹

¹<http://www.openstreetmap.org>

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



Gambar 1.2: Contoh Graf

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

16 1.2 Rumusan Masalah

17 Berdasarkan latar belakang, maka rumusan masalah berikut:

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

22 1.3 Tujuan

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

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

- 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.

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 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.

5. Bab Implementasi dan Pengujian

Bab 5 berisi 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.

1

2

3

4



13

14

15

16

19

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 menyisipkan 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:

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

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

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

13 1. Tag pada XML

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

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

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

19 2. Nama pada XML

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

23 3. Atribut

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

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

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

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

30 4. Elemen Kosong

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

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

34 Elemen email tidak memiliki nilai atau isi.

35 5. *Nesting of Elements*

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

```

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

```

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:

```

13      <?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:

```

19 <?xml version="1.0" encoding="UTF-8"?>
20 <osm version="0.6" generator="CGImap_0.0.2">
21   <bounds minlat="54.0889580" minlon="12.2487570" maxlat="
22     54.0913900" maxlon="12.2524800"/>
23   <node id="298884269" lat="54.0901746" lon="12.2482632" user="
24     SvenHRO" uid="46882" visible="true" version="1" changeset="
25     676636" timestamp="2008-09-21T21:37:45Z"/>
26   <node id="261728686" lat="54.0906309" lon="12.2441924" user="
27     PikoWinter" uid="36744" visible="true" version="1" changeset="
28     323878" timestamp="2008-05-03T13:39:23Z"/>
29   <node id="1831881213" version="1" changeset="12370172" lat="
30     54.0900666" lon="12.2539381" user="lafkor" uid="75625" visible
31     ="true" timestamp="2012-07-20T09:43:19Z">
32     <tag k="name" v="Neu_Broderstorf"/>
33     <tag k="traffic_sign" v="city_limit"/>
34   </node>
35   ...
36   <node id="298884272" lat="54.0901447" lon="12.2516513" user="
37     SvenHRO" uid="46882" visible="true" version="1" changeset="
38     676636" timestamp="2008-09-21T21:37:45Z"/>
39   <way id="26659127" user="Masch" uid="55988" visible="true"
40     version="5" changeset="4142606" timestamp="2010-03-16
41     T11:47:08Z">

```



```

1  <nd ref="292403538"/>
2  <nd ref="298884289"/>
3  ...
4  <nd ref="261728686"/>
5  <tag k="highway" v="unclassified"/>
6  <tag k="name" v="Pastower_Stra se"/>
7  </way>
8  <relation id="56688" user="kmvar" uid="56190" visible="true"
9    version="28" changeset="6947637" timestamp="2011-01-12
10    T14:23:49Z">
11    <member type="node" ref="294942404" role=""/>
12    ...
13    <member type="node" ref="364933006" role=""/>
14    <member type="way" ref="4579143" role=""/>
15    ...
16    <member type="node" ref="249673494" role=""/>
17    <tag k="name" v="K ijstenbus_Linie_123"/>
18    <tag k="network" v="VWV"/>
19    <tag k="operator" v="Regionalverkehr_K ijste"/>
20    <tag k="ref" v="123"/>
21    <tag k="route" v="bus"/>
22    <tag k="type" v="route"/>
23  </relation>
24  ...
25 </osm>

```

26 Struktur OSMXML:

- 27 • Dokumen OSMXML diawali dengan tag xml yang menjelaskan versi xml dan encoding
28 yang digunakan, pada contoh di atas digunakan xml versi 1.0 dan encoding UTF-8.
- 29 • Elemen osm memberikan informasi mengenai versi API dan generator yang diguna-
30 kan. Generator adalah alat untuk membuat dokumen XML pada saat fitur export
31 digunakan.
- 32 • Elemen bound memberikan informasi mengenai cakupan area pada dokumen XML
33 tersebut. Dilengkapi dengan atribut koordinat yaitu latitude dan longitude. Data
34 primitif pada OSM dibagi menjadi 3 bagian, yaitu node, way, dan relation.

35 1. Elemen Node merupakan informasi titik pada sebuah peta. Node memiliki bebe-
36 rapa atribut yaitu:

- 37 – id
38 Merupakan id dari node tersebut.
- 39 – user
40 Merupakan user yang melakukan editing pada node.
- 41 – uid
42 Id dari user.

- lat
berisi informasi koordinat pada garis lintang.
- lon
berisi informasi koordinat pada garis bujur.
- timestamp
Berisi informasi waktu saat node tersebut diperbaharui.

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 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,

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

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
    document.getElementById("demo").innerHTML = "Paragraph_changed
    .";
}
</script>
</head>
<body>
<h1>JavaScript in Head</h1>
<p id="demo">A Paragraph.</p>
<button type="button" onclick="myFunction()">Try it</button>
```

```
1 </body>
2 </html>
```

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

6 Seluruh browser yang terdapat pada komputer, konsol game, tablet, dan smartphone su-
7 dah disertai dengan javascript interpreter. Interpreter adalah suatu program yang berfungsi
8 untuk menerjemahkan kode program ke dalam bahasa mesin. Javascript adalah bagian
9 yang cukup penting pada sebuah halaman web, jika HTML berfungsi untuk menentukan
10 isi dari halaman dan CSS untuk menentukan tampilan pada halaman, javascript berfungsi
11 untuk menentukan “behavior” dari halaman web tersebut [4]. Berikut ini adalah uraian dari
12 struktur javascript dan beberapa contoh sintaks:

13 1. Struktur

14 • *Character Set*

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

17 • *Comments*

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

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

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

```
22 /*
```

```
23 * This is yet another comment.
```

```
24 * It has multiple lines.
```

```
25 */
```

26 • *Literal*

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

```
30 12 // The number twelve
```

```
31 1.2 // The number one point two
```

```
32 "hello world" // A string of text
```

```
33 'Hi' // Another string
```

```
34 true // A Boolean value
```

```
35 false // The other Boolean value
```

```
36 /javascript/gi // A "regular expression" literal (for pattern matching)
```

```
37 null // Absence of an object
```

38 • *Identifier*

39 *Identifier* pada javascript hanyalah nama yang digunakan untuk memberi nama
40 pada variabel atau fungsi. Digit tidak diperbolehkan sebagai karakter pertama
41 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 didefinisikan sekali, tapi 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);
```

```
}
```

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:

```

7 var objXMLHTTP = new XMLHttpRequest();
8
9 objXMLHTTP.open('GET', 'books.xml', false);
10 objXMLHTTP.send(null);
11
12 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:
6         0;}
7     </style>
8     <script type="text/javascript">
9       src="https://maps.googleapis.com/maps/api/js?key=API_KEY">
10    </script>
11    <script type="text/javascript">
12      function initialize() {
13        var mapOptions = {
14          center: { lat: -34.397, lng: 150.644},
15          zoom: 8
16        };
17        var map = new google.maps.Map(document.getElementById( 'map
18          -canvas '),
19          mapOptions);
20      }
21      google.maps.event.addDomListener(window, 'load', initialize)
22      ;
23    </script>
24  </head>
25  <body>
26    <div id="map-canvas"></div>
27  </body>
28 </html>

```

29 • Declaring

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

```

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

```

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

41 • Loading Google Maps API

42 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.

Parameter:

- mapDiv:Node

node yang digunakan untuk membuat peta.

- 1 • `opts?:MapOptions`
2 Ops dari *map* yang akan dibuat.
- 3 2. `google.maps.LatLng` class
4 Membuat objek *LatLng* yang merepresentasikan titik geografis.
5 Parameter:
6 • `lat:number`
7 *Latitude* dalam derajat.
8 • `lng:number`
9 *Longitude* dalam derajat.
10 • `noWrap?:boolean`
11 *Latitude* ditentukan dalam rentang derajat -90 hingga 90 dan *longitude* ditent-
12 ukan dalam rentang derajat -180 hingga 180. Nilai *true* pada boolean `noWrap`
13 untuk mengaktifkan nilai di luar batas tersebut.
- 14 3. `google.maps.event.addListener()`
15 Menambahkan fungsi *listener*
16 Parameter:
17 • `instance:Object`
18 Objek yang ditambahkan *listener*.
19 • `eventName:string`
20 Nama *Event*.
21 • `handler:Function`
22 Fungsi yang dipanggil ketika *event* terjadi.
- 23 Return:
24 `MapsEventListener`

25 2.3.3.2 Menggambar pada Peta

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

- 30 1. `google.maps.Marker` class
31 Membuat *marker* pada peta dengan opsi tertentu.
32 Parameter:
33 • `opts?:MarkerOptions`
34 Ops dari *marker* yang dibuat.
- 35 2. `google.maps.Polyline` class
36 Membuat *polyline* pada peta dengan opsi tertentu.
37 Parameter:

- `opts?:PolylineOptions`

Opsi dari *polyline* yang dibuat.

3. `setMap`

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

Parameter:

- `map:Map|StreetViewPanorama`

Peta yang disisipkan *marker* atau *polyline*.

4. `setIcon`

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, perubahan *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);
```

```

1      var beachMarker = new google.maps.Marker({
2          position: myLatLng,
3          map: map,
4          icon: image
5      });

```

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

7 • *Draggable*

8 Draggable memungkinkan pengguna untuk menyeret marker ke lokasi yang berbe-
9 da, hanya memerlukan pengaturan atribut pada konstruktor `google.maps.Marker`.

10 Contoh:

```

11      var marker = new google.maps.Marker({
12          position: myLatLng,
13          map: map,
14          draggable: true,
15          title:"Hello World!"
16      });

```

17 2. *Polyline*

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

22 • *strokeColor*

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

25 • *strokeOpacity*

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

27 • *strokeWeight*

28 Atribut *strokeWeight* menentukan lebar garis dalam piksel.

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

```

31      var flightPlanCoordinates = [
32          new google.maps.LatLng(37.772323, -122.214897),
33          new google.maps.LatLng(21.291982, -157.821856),
34          new google.maps.LatLng(-18.142599, 178.431),
35          new google.maps.LatLng(-27.46758, 153.027892)
36      ];
37      var flightPath = new google.maps.Polyline({
38          path: flightPlanCoordinates,
39          strokeColor: '#FF0000',
40          strokeOpacity: 1.0,
41          strokeWeight: 2

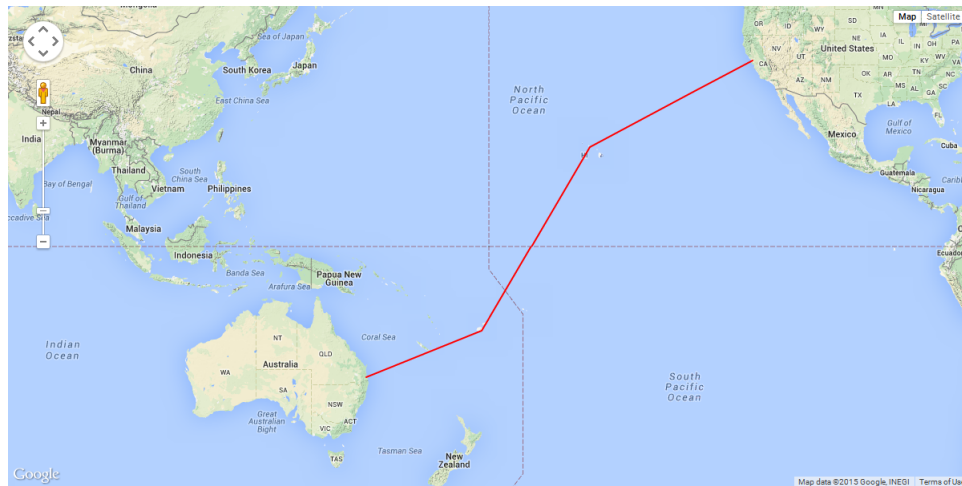
```

```

1      });
2
3      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

6

7 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()`.

- 17 1. `google.maps.geometry.spherical` namespace
 18 Fungsi utilitas untuk menghitung sudut, jarak, dan area. Secara *default*, radius bumi
 19 yang digunakan adalah 6378137 meter.
- 20 2. `computeDistanceBetween()`
 21 Menghitung jarak antara dua titik.
 22 Parameter:
 - 23 • `from:LatLng`
 24 Koordinat titik pertama.
 - 25 • `to:LatLng`
 26 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* dan 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:

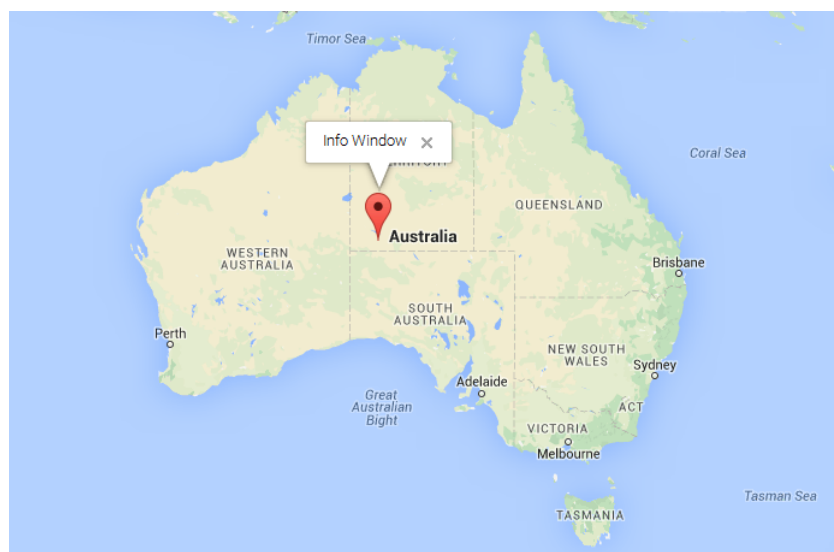
1. `google.maps.InfoWindow` class

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

Parameter:

- 1 • `opts?:InfoWindowOptions`
- 2 Opsi dari *info window* yang dibuat.
- 3 2. `open()`
- 4 Membuka *info window* pada peta.
- 5 Parameter
- 6 • `map?:Map|StreetViewPanorama`
- 7 Membuka *info window* pada peta yang diberikan.
- 8 • `anchor?:MVCObject`
- 9 Objek yang berhubungan dengan *info window*, contoh: *marker*.

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



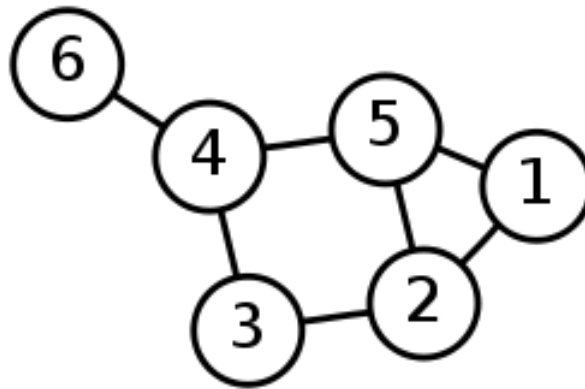
Gambar 2.3: Info Window pada Peta

10

11 2.4 Graf

12 Graf adalah himpunan objek yang terdiri dari node (simpul) dan edge (sisi), graf digambarkan
 13 an sebagai node yang dihubungkan oleh edge. Konsep graf telah digunakan pada banyak
 14 aplikasi komputer dan menggunakan beberapa jenis graf seperti graf sederhana, graf tidak
 15 berarah, graf berarah, graf berbobot, dan lain-lain. Contoh graf dapat dilihat pada Gambar
 16 2.4. Graf mengikuti aturan berikut ²:

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



Gambar 2.4: Contoh Graf

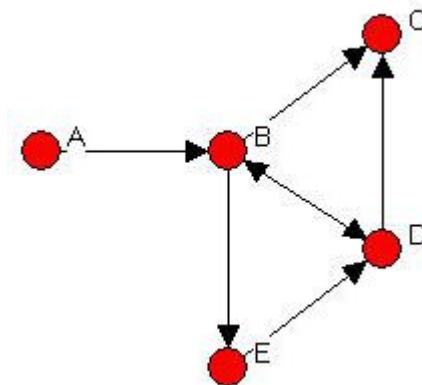
- 1 Berdasarkan contoh pada Gambar 2.4 didapatkan informasi tipe dari node adalah bi-
 2 langan bulat
 3 Himpunan node = 1,2,3,4,5,6
 4 Himpunan edge = (6,4),(4,5),(4,3),(3,2),(5,2),(2,1),(5,1)

5 2.4.1 Graf Tidak Berarah

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

9 2.4.2 Graf Berarah

- 10 Graf berarah memiliki arah pada setiap edgenya. Pada graf berarah, edge biasanya digam-
 11 barkan 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



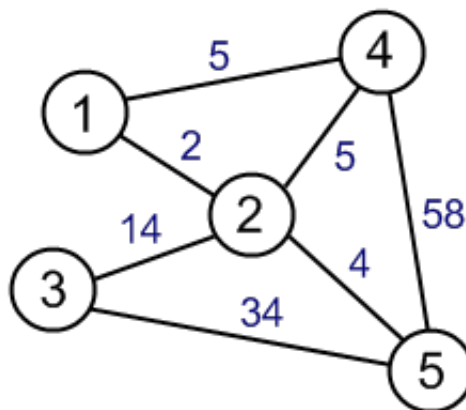
Gambar 2.5: Contoh Graf Berarah

- 12
 13 kapital.
 14 Himpunan node = A, B, C, D, E
 15 Himpunan edge = (A, B), (B, C), (D, C), (B, D), (D, B), (E, D), (B, E)

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

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. Berdasarkan contoh pada Gambar 2.6 didapatkan informasi tipe dari



Gambar 2.6: Contoh Graf Berbobot

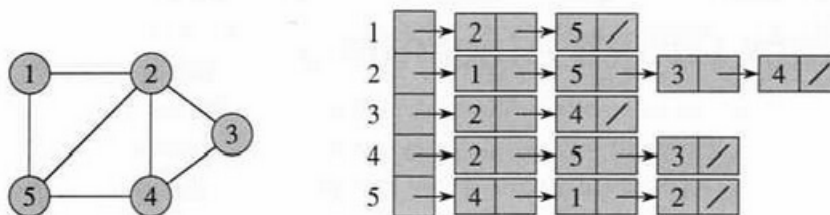
node adalah bilangan bulat dan tipe dari bobot adalah bilangan bulat.
 Himpunan node = 1,2,3,4,5
 Himpunan edge = (1,4,5), (4,5,58), (3,5,34), (2,4,5), (2,5,4), (3,2,14), (1,2,2)

2.4.4 Representasi Graf

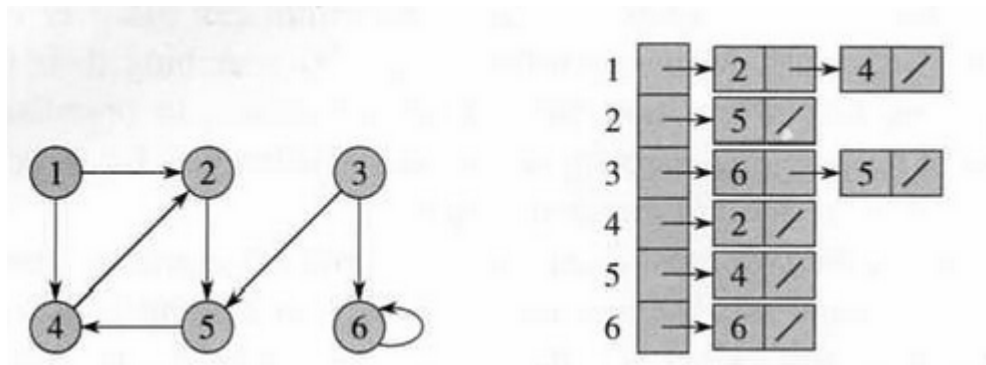
Terdapat dua cara untuk merepresentasikan graf yaitu dengan *adjacency list* dan *adjacency matrix* [1]. Keduanya dapat merepresentasikan graf berarah ataupun graf tidak berarah. *Adjacency list* merepresentasikan graf ke dalam bentuk array, sedangkan *adjacency matrix* merepresentasikan graf ke dalam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

- *Adjacency List*

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



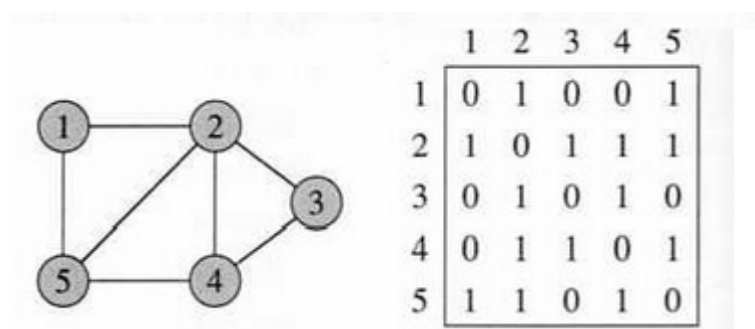
Gambar 2.7: Contoh Adjacency List (Graf Tidak Berarah)



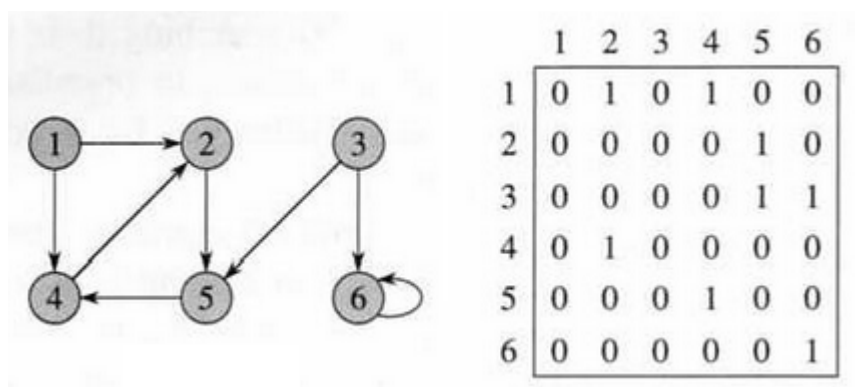
Gambar 2.8: Contoh Adjacency List (Graf Berarah)

- *Adjacency Matrix*

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



Gambar 2.9: Contoh Adjacency Matrix (Graf Tidak Berarah)



Gambar 2.10: Contoh Adjacency Matrix (Graf Berarah)

1 2.5 Algoritma Dijkstra

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

Algorithm 1 *Dijkstra*

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

BAB 3

ANALISIS

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

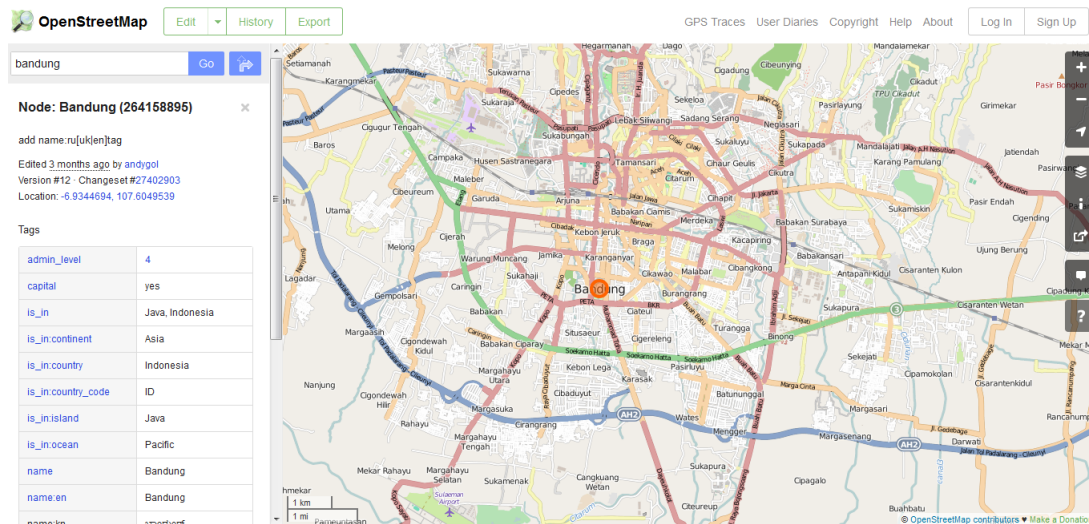
3.1 Analisis OpenStreetMap

OpenStreetMap adalah portal peta terbuka yang menyediakan data dalam bentuk peta ataupun dokumen XML. Aplikasi yang dibuat akan berbasis OpenStreetMap, hal ini berarti aplikasi yang dibuat akan menggunakan data yang diperoleh dari situs 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 akan 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.

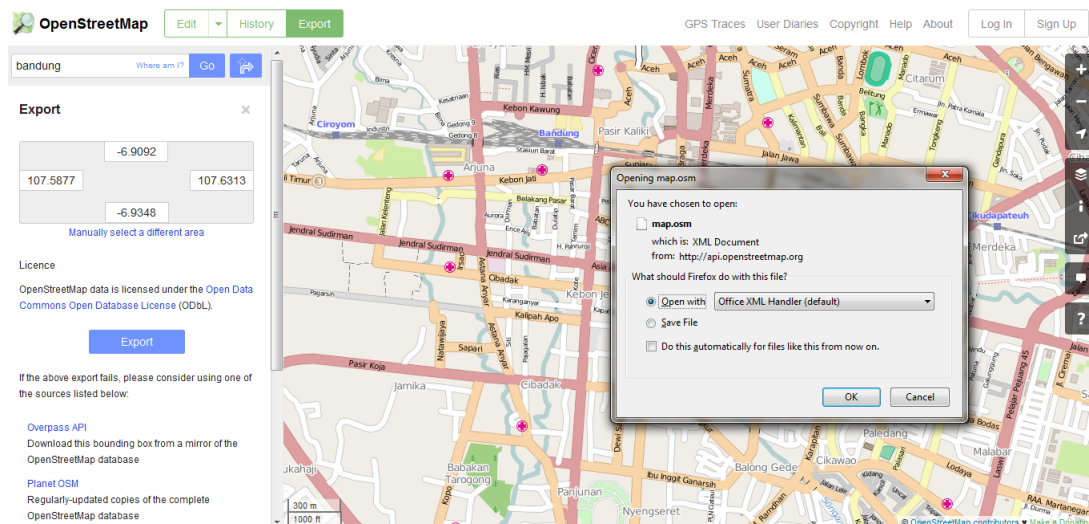
3.1.1 Langkah-Langkah Pengambilan Data OSMXML

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

1. Mengunjungi situs www.openstreetmap.org.
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 XML. 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. Contoh data yang sudah diunduh dapat dilihat di bawah ini.

```
<?xml version="1.0" encoding="UTF-8"?>
<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"/>
```

```

1 <node id="25433683" visible="true" version="3" changeset="839915"
2   timestamp="2009-03-21T14:18:48Z" user="adhitya" uid="7748"
3   lat="-6.9067659" lon="107.5989458"/>
4 ...
5 <node id="25433687" visible="true" version="2" changeset="839915"
6   timestamp="2009-03-21T14:18:36Z" user="adhitya" uid="7748"
7   lat="-6.9040267" lon="107.5969508"/>
8 <node id="25433688" visible="true" version="2" changeset="839915"
9   timestamp="2009-03-21T14:18:58Z" user="adhitya" uid="7748"
10  lat="-6.9039393" lon="107.5963723"/>
11 <node id="25500626" visible="true" version="3" changeset="839915"
12  timestamp="2009-03-21T14:22:17Z" user="adhitya" uid="7748"
13  lat="-6.9070329" lon="107.6019401"/>
14 ...
15 <node id="3030289971" visible="true" version="1" changeset="
16   24892866" timestamp="2014-08-20T18:40:31Z" user="albahrimaraxsa
17   " uid="2162153" lat="-6.9066710" lon="107.5982569"/>
18 <node id="2325451442" visible="true" version="4" changeset="
19   27916144" timestamp="2015-01-04T18:06:33Z" user="isonpurba"
20   uid="2552445" lat="-6.9045011" lon="107.6024922"/>
21 <way id="4567626" visible="true" version="4" changeset="15861148"
22   timestamp="2013-04-25T13:56:12Z" user="mrdoggie94" uid="
23   1331966">
24   <nd ref="25433682"/>
25   <nd ref="25433681"/>
26   <nd ref="25433680"/>
27   <tag k="avgspeed" v="15"/>
28   <tag k="highway" v="residential"/>
29   <tag k="name" v="Dr. J. Rubini"/>
30 </way>
31 <way id="4567634" visible="true" version="2" changeset="7821743"
32   timestamp="2011-04-10T11:15:30Z" user="evo2mind" uid="234610">
33   <nd ref="25433681"/>
34   <nd ref="28802396"/>
35   <tag k="avgspeed" v="15"/>
36   <tag k="highway" v="residential"/>
37   <tag k="name" v="Dr. J. Susilo"/>
38 </way>
39 ...
40 </osm>

```

Node dan way memiliki informasi penting yang akan 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*) pada node tersebut. Informasi yang didapatkan akan disimpan ke dalam bentuk node pada graf. Tag

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

8 3.2 Analisis Javascript

9 Javascript diperlukan untuk membaca dokumen XML, sehingga seluruh informasi yang di-
 10 perlukan dapat diubah ke dalam bentuk graf yang akan diproses lebih lanjut. XMLHttpRequest-
 11 quest adalah salah satu objek pada javascript yang dapat digunakan untuk mendapatkan *file*
 12 XML. Berikut ini adalah contoh kode program yang akan digunakan untuk mendapatkan
 13 *file* XML:

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

18 Setelah mendapatkan *file* XML, diperlukan XML DOM untuk membaca informasi yang
 19 diperlukan pada dokumen XML. Berikut ini adalah contoh kode program yang akan digu-
 20 nakan:

```
21 var node = xmlDoc.getElementsByTagName("node");
22 var way = xmlDoc.getElementsByTagName("way");
23
24 var id_node = node[0].getAttribute('id');
```

25 Di bawah ini adalah contoh kode program yang digunakan untuk menampilkan informasi
 26 dari dokumen XML dalam bentuk tabel:

```
27 <html>
28 <head>
29 <style>
30 table, th, td {
31     border: 1px solid black;
32     border-collapse: collapse;
33 }
34 th, td {
35     padding: 5px;
36 }
37 </style>
38 </head>
39 <body>
40 <script>
```



```

1  xmlhttp=new XMLHttpRequest();
2  xmlhttp.open("GET","map.xml",false);
3  xmlhttp.send();
4  xmlDoc=xmlhttp.responseXML;
5
6  document.write("<div_style='float:_left '>");
7  document.write("<table><tr><th>Node</th><th>Id</th><th>Latitude</th><th>Longitude</th></tr>");
8      th>Longitude</th></tr>");
9  document.write("<caption>Node</caption>");
10 var node=xmlDoc.getElementsByTagName("node");
11 for (ct=0;ct<node.length;ct++)
12 {
13     document.write("<tr><td>");
14     document.write(ct);
15     document.write("</td><td>");
16     document.write(node[ct].getAttribute('id'));
17     document.write("</td><td>");
18     document.write(node[ct].getAttribute('lat'));
19     document.write("</td><td>");
20     document.write(node[ct].getAttribute('lon'));
21     document.write("</td></tr>");
22 }
23 document.write("</table>");
24 document.write("</div>");
25
26 function isHighway(way,index){
27     var tag = way[index].getElementsByTagName("tag");
28     for (hg=0;hg<tag.length;hg++)
29     {
30         if(tag[hg].getAttribute('k') == "highway"){
31             return true;
32         }
33     }
34     return false;
35 }
36
37 document.write("<div_style='margin-left:_20px;float:_left '>");
38 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>");
39     >th>Id_Node_1</th><th>Id_Node_2</th></tr>");
40 document.write("<caption>Edge</caption>");
41 var way = xmlDoc.getElementsByTagName("way");
42 var nd;
43 for (i=0;i<way.length;i++)
44 {

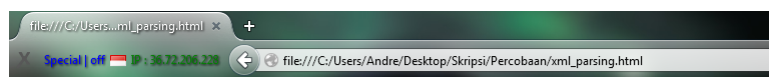
```

```

1      nd = way[i].getElementsByTagName("nd");
2      if (isHighway(way, i)) {
3          for (j=0; j<nd.length-1; j++)
4              {
5                  document.write("<tr><td>");
6                  document.write(i);
7                  document.write("</td><td>");
8                  document.write(way[i].getAttribute('id'));
9                  document.write("</td><td>");
10                 document.write(j);
11                 document.write("</td><td>");
12                 document.write(nd[j].getAttribute('ref'));
13                 document.write("</td><td>");
14                 document.write(nd[j+1].getAttribute('ref'));
15                 document.write("</td></tr>");
16             }
17     }
18 }
19 document.write("</div>");
20 </script>
21 </body>
22 </html>

```

Hasil dari kode program di atas dapat dilihat pada Gambar 3.3.



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

23

24 Pada Gambar 3.3 dapat dilihat terdiri dari dua tabel yang menunjukkan informasi node
 25 dan edge yang sudah dibaca. Tabel node menunjukkan atribut penting yang akan digunakan

1 yaitu id node, *latitude*, dan *longitude*. Sedangkan tabel edge menunjukkan informasi yang
2 didapatkan dari tag way yang sudah dilakukan *filter*, yaitu hanya tag highway saja yang
3 akan digunakan. Pada tabel edge terdapat informasi penting yang akan digunakan, yaitu id
4 way, id node pertama, dan id node kedua. Informasi yang sudah didapatkan, selanjutnya
5 akan dimodelkan ke dalam bentuk graf yang akan dibahas pada subbab 3.3.

6 3.3 Pemodelan OSMXML Menjadi Graf

7 Pada subbab 3.2, dokumen OSMXML sudah dibaca dan tahap selanjutnya adalah memo-
8 delkan data OSMXML tersebut ke dalam bentuk graf. Seperti yang sudah diketahui, graf
9 terdiri dari node dan edge. Informasi node dan edge yang sudah didapatkan akan dimodelkan
10 ke dalam bentuk graf berarah, hal ini karena jalan yang menghubungkan node-node tersebut
11 memiliki arah, baik searah maupun dua arah. Untuk merepresentasikan graf tersebut akan
12 digunakan *adjacency list*. Di bawah ini adalah contoh kode program yang digunakan untuk
13 memodelkan dokumen OSMXML menjadi graf:

```
14 <!DOCTYPE html>
15 <html>
16 <head>
17 </head>
18 <body>
19 <script>
20 xmlhttp=new XMLHttpRequest();
21 xmlhttp.open("GET","map.xml",false);
22 xmlhttp.send();
23 xmlDoc=xmlhttp.responseXML;
24
25 function Neighbor(vnum, nbr){
26     this.vertexNum = vnum;
27     this.next = nbr;
28 }
29
30 function Node(id, neighbors){
31     this.id = id;
32     this.adjList = neighbors;
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     }
```

```

1      return false;
2  }
3
4  function indexForName(adjLists, id) {
5      for (var i=0; i < adjLists.length; i++){
6          if (adjLists[i].id == id){
7              return i;
8          }
9      }
10     return -1;
11 }
12
13 function Graph(node, way){
14     var adjLists = [];
15     var nd;
16
17     for(v=0; v < node.length; v++){
18         adjLists.push(new Node(node[v].getAttribute('id'), null));
19     }
20
21     for (i=0; i < way.length; i++)
22     {
23         nd = way[i].getElementsByTagName("nd");
24         if (isHighway(way, i)){
25             for (j=0; j < nd.length-1; j++)
26             {
27                 v1 = indexForName(adjLists, nd[j].getAttribute('ref
28                     '));
29                 v2 = indexForName(adjLists, nd[j+1].getAttribute('
30                     ref'));
31
32                 adjLists[v1].adjList = new Neighbor(v2, adjLists[
33                     v1].adjList);
34                 adjLists[v2].adjList = new Neighbor(v1, adjLists[
35                     v2].adjList);
36             }
37         }
38     }
39     return adjLists;
40 }
41
42 var node = xmlDoc.getElementsByTagName("node");
43 var way = xmlDoc.getElementsByTagName("way");
44

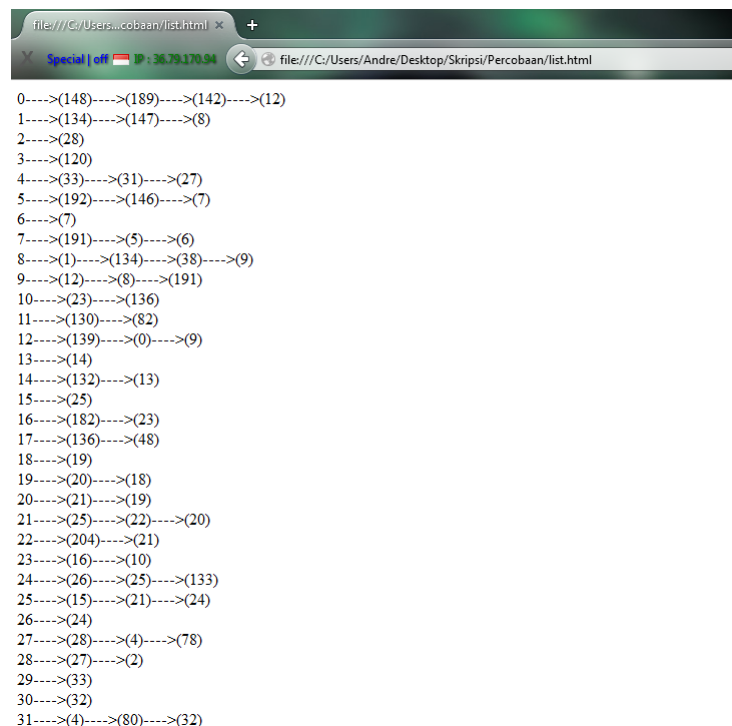
```

```

1  var list = new Graph(node, way);
2
3  for (j=0; j < list.length; j++){
4      document.write(j);
5      for (nbr=list[j].adjList; nbr != null; nbr=nbr.next) {
6          document.write("----->");
7          document.write('(' + nbr.vertexNum + ') ');
8      }
9      document.write("<br>");
10 }
11
12 </script>
13 </body>
14 </html>

```

Hasil dari kode program di atas dapat dilihat pada Gambar 3.4. Pada Gambar 3.4, data pada



Gambar 3.4: Pemodelan OSMXML menjadi Graf

OSMXML sudah dimodelkan ke dalam bentuk graf menggunakan representasi *adjacency list*. Graf tersebut selanjutnya akan divisualisasikan menggunakan Google Maps Javascript API yang akan dibahas pada subbab 3.4.

3.4 Visualisasi Graf

Data OSMXML yang sudah dimodelkan ke dalam bentuk graf, selanjutnya akan dibuat visualisasi menggunakan Google Maps Javascript API. Peta yang akan ditampilkan dalam bentuk *roadmap*, karena peta jenis ini memberikan informasi mengenai nama jalan, sehingga peta jenis ini lebih cocok untuk aplikasi yang akan dikembangkan. Berikut ini adalah

- 1 contoh potongan kode program untuk membuat visualisasi graf ke dalam bentuk peta digital
- 2 menggunakan Google Maps Javascript API.

```

3 function getLatByAtt(str)
4 {
5     for (n=0;n<node.length;n++)
6     {
7         if (node[n].getAttribute('id') == str)
8         {
9             return node[n].getAttribute('lat');
10        }
11    }
12 }
13 function getLonByAtt(str)
14 {
15     for (m=0;m<node.length;m++)
16     {
17         if (node[m].getAttribute('id') == str)
18         {
19             return node[m].getAttribute('lon');
20        }
21    }
22 }
23
24 function isHighway(way,index){
25     var tag = way[index].getElementsByTagName("tag");
26     for (hg=0;hg<tag.length;hg++)
27     {
28         if (tag[hg].getAttribute('k') == "highway"){
29             return true;
30        }
31    }
32     return false;
33 }
34
35 var currentId = 0;
36 var uniqueId = function() {
37     return ++currentId;
38 }
39
40 var mapProp = {
41     center: new google.maps.LatLng
42         (-6.906845432118958,107.59851515293121),
43     zoom: 17,
44     mapTypeId: google.maps.MapTypeId.ROADMAP

```

```

1  };
2
3  var map=new google.maps.Map(document.getElementById("googleMap"),
4      mapProp);
5
6  var path = [];
7  var markers = {};
8  var nd,marker,line,content,id;
9  var lat1,lon1,lat2,lon2;
10 var image = 'icon/dot_blue.png';
11
12 for (i=0;i<way.length;i++)
13 {
14     nd = way[i].getElementsByTagName("nd");
15     if (isHighway(way,i))
16     {
17         id = uniqueId();
18         marker = new google.maps.Marker({
19             id: id,
20             position: new google.maps.LatLng(getLatByAtt(nd[0].
21                 getAttribute('ref')), getLonByAtt(nd[0].
22                 getAttribute('ref'))),
23             map: map,
24             icon: image,
25         });
26         markers[id] = marker;
27
28         content = 'Id Node : '+nd[0].getAttribute('ref')+'<br>'+
29             'Index Node : '+indexOfName(list,nd[0].getAttribute('
30             ref'))+'<br>'+
31             '<a href="#" onclick="setAsal(\''+_id_+'\'")>Pilih
32             sebagai asal</a>'+<br>'+
33             '<a href="#" onclick="setTujuan(\''+_id_+'\'")>Pilih
34             sebagai tujuan</a>';
35
36         addInfoWindow(marker, content);
37
38         for (j=0;j<nd.length-1;j++)
39         {
40             id = uniqueId();
41             marker = new google.maps.Marker({
42                 id: id,
43                 position: new google.maps.LatLng(getLatByAtt(nd[j
44                     +1].getAttribute('ref')), getLonByAtt(nd[j+1].

```

```

1         getAttribute('ref'))),
2         map: map,
3         icon: image,
4     });
5     markers[id] = marker;
6
7
8     content = 'Id Node : '+nd[j+1].getAttribute('ref')+'<
9         br>'+
10    'Index Node : '+indexOfName(list, nd[j+1].getAttribute
11    ('ref'))+'<br>'+
12    '<a href="#" onclick="setAsal(\''+id+'\')">Pilih
13    sebagai asal</a>'+<br>'+
14    '<a href="#" onclick="setTujuan(\''+id+'\')">Pilih
15    sebagai tujuan</a>';
16
17    addInfoWindow(marker, content);
18
19    line = new google.maps.Polyline({
20        path: [new google.maps.LatLng(getLatByAtt(nd[j].
21            getAttribute('ref')), getLonByAtt(nd[j].
22            getAttribute('ref'))),
23            new google.maps.LatLng(getLatByAtt(nd[j+1].
24            getAttribute('ref')), getLonByAtt(nd[j+1].
25            getAttribute('ref')))],
26        strokeColor: "#000000",
27        strokeOpacity: 1.0,
28        strokeWeight: 3,
29        map: map
30    });
31    }
32    }
33    }
34
35    function addInfoWindow(marker, message) {
36        var infoWindow = new google.maps.InfoWindow({
37            content: message
38        });
39
40        google.maps.event.addDomListener(marker, 'click', function ()
41        {
42            infoWindow.open(map, marker);
43        });
44    }

```

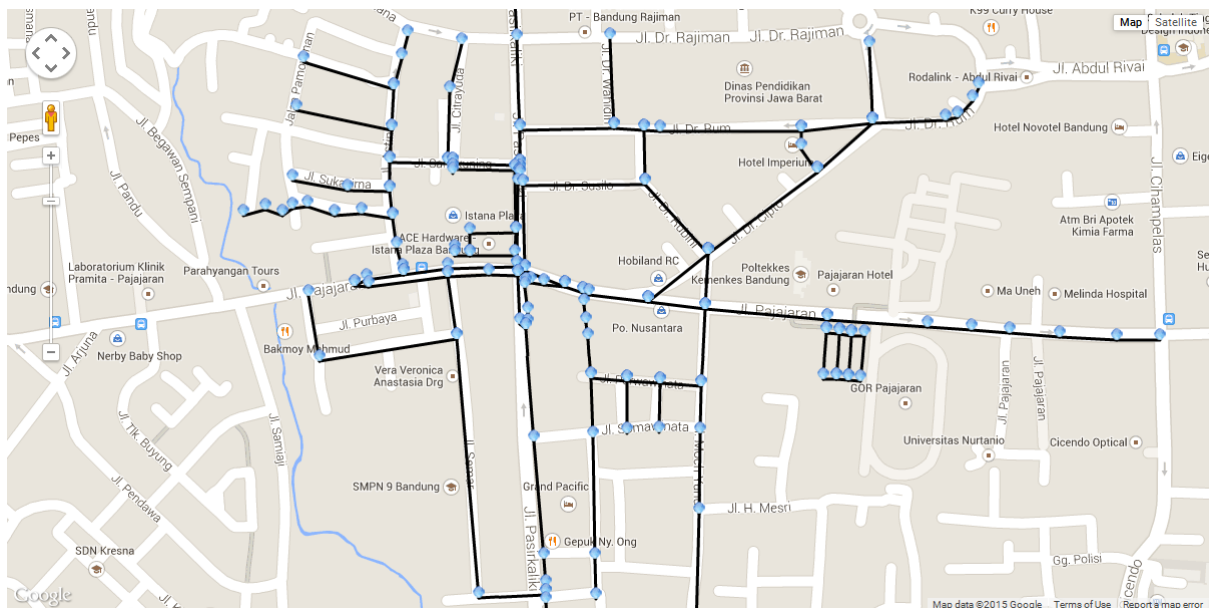


```

1  var isAsalClicked = false;
2  var isTujuanClicked = false;
3  var asal , tujuan;
4
5  function setAsal(id){
6      if (!isAsalClicked){
7          markers[id].setIcon('icon/dot_green.png');
8          isAsalClicked = true;
9          asal = id;
10     }else{
11         markers[asal].setIcon('icon/dot_blue.png');
12         markers[id].setIcon('icon/dot_green.png');
13         asal = id;
14     }
15 }
16
17 function setTujuan(id){
18     if (!isTujuanClicked){
19         markers[id].setIcon('icon/dot_red.png');
20         isTujuanClicked = true;
21         tujuan = id;
22     }else{
23         markers[tujuan].setIcon('icon/dot_blue.png');
24         markers[id].setIcon('icon/dot_red.png');
25         tujuan = id;
26     }
27 }

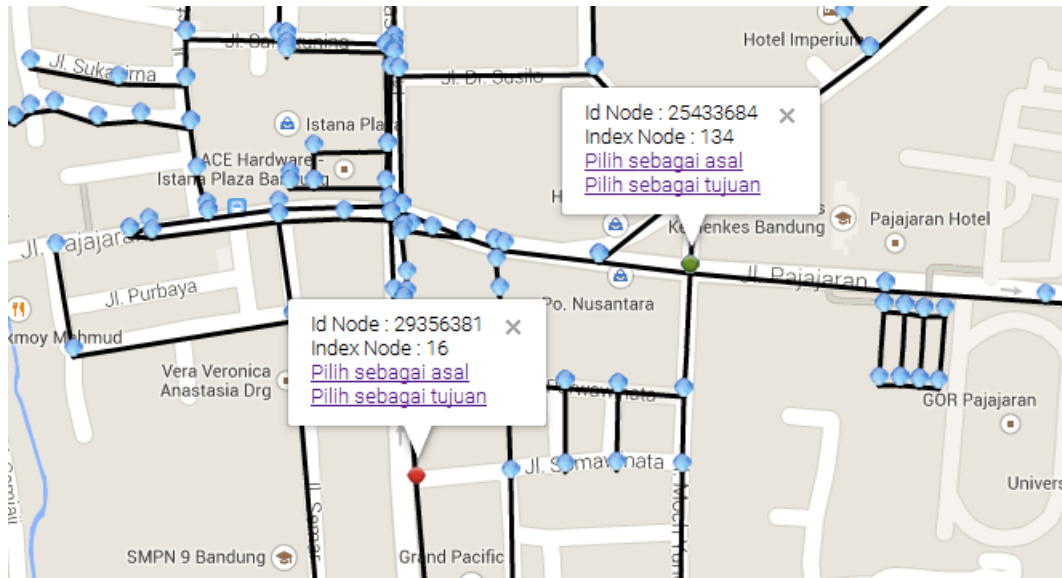
```

Hasil dari kode program di atas dapat dilihat pada Gambar 3.5.



Gambar 3.5: Visualisasi Graf

- 1 Setiap node pada graf yang sudah dilakukan *filter* akan diwakili oleh marker. Setiap
2 marker tersebut akan memiliki *info window* yang akan memberikan informasi seperti id
3 node, index node pada graf, dan dua buah *hyperlink* yang berfungsi untuk menjadikan
4 marker yang dipilih menjadi asal atau tujuan. Contoh *info window* yang ditampilkan pada
5 peta dapat dilihat pada Gambar 3.6. Setiap edge pada graf akan menjadi garis pada peta
yang dibuat menggunakan *polyline*. 3.6.



Gambar 3.6: Info Window

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].