

SKRIPSI

VISUALISASI DATA HISTORI KIRI PADA GOOGLE MAPS



Jonathan Laksamana Purnomo

NPM:

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2020

UNDERGRADUATE THESIS

VISUALIZATION OF KIRI HISTORICAL DATA ON GOOGLE MAPS



Jonathan Laksamana Purnomo

NPM:

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2020

LEMBAR PENGESAHAN

VISUALISASI DATA HISTORI KIRI PADA GOOGLE MAPS

Jonathan Laksamana Purnomo

NPM:

Bandung, 3 Desember 2020

Menyetujui,

Pembimbing Utama

Pembimbing Pendamping

Pascal Alfadian, M.Comp.

Ketua Tim Penguji

Anggota Tim Penguji

«penguji 1»

«penguji 2»

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

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

VISUALISASI DATA HISTORI KIRI PADA GOOGLE MAPS

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 dijatuhkan 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 3 Desember 2020

Meterai Rp. 6000

Jonathan Laksamana Purnomo
NPM:

ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia»

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Kata-kata kunci: KIRI, *Navigation System, Data History, Marker Clustering, Heat Map, Google Javascript API*

ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Keywords: KIRI, *Navigation System, Data History, Marker Clustering, Heat Map, Google Javascript API*

Dipersembahkan untuk Tuhan YME, keluarga, para dosen, teman-teman yang telah memberi dukungan dalam pembuatan skripsi ini, serta diri sendiri.

KATA PENGANTAR

«Tuliskan kata pengantar dari anda di sini ...»

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.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Bandung, Desember 2020

Penulis

DAFTAR ISI

KATA PENGANTAR	xv
DAFTAR ISI	xvii
DAFTAR GAMBAR	xix
DAFTAR TABEL	xxi
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Tujuan	3
1.4 Batasan Masalah	3
1.5 Metodologi	4
1.6 Sistematika Pembahasan	4
2 LANDASAN TEORI	5
2.1 KIRI Website	5
2.2 JSON	6
2.2.1 JSON Grammar	6
2.2.2 Values	7
2.2.3 Objects	7
2.2.4 Arrays	7
2.2.5 Numbers	7
2.2.6 Strings	8
2.2.7 JSON Example	9
2.3 CSV	10
2.3.1 CSV Format	10
2.4 Node.js	11
2.4.1 Struktur File Node.js Project	11
2.4.2 <i>Node Package Manager</i>	11
2.4.3 NPM CLI	12
2.5 <i>Expressjs</i>	12
2.5.1 Instalasi	12
2.5.2 Struktur File Express.js	13
2.5.3 Routing	13
2.5.4 Menampilkan File Statis	14
2.6 Google Maps Javascript API	14
2.6.1 Map	14
2.6.2 Sistem Koordinat Google Maps	16
2.6.3 Drawing System	17
2.6.4 Marker	18
2.6.5 Marker Clusterer	19

2.6.6	HeatMap	20
3	ANALISIS	23
3.1	Deskripsi Perangkat Lunak	23
3.2	Analisis Data Histori KIRI	24
3.3	Analisis Perangkat Lunak	25
3.3.1	Analisis Kebutuhan Perangkat Lunak	25
3.4	Use Case Diagram	25
3.4.1	<i>Use Case</i> Skenario	25
	DAFTAR REFERENSI	29
	A KODE PROGRAM	31
	B HASIL EKSPERIMEN	33

DAFTAR GAMBAR

1.1	Tampilan utama website KIRI	1
1.2	Multi Bahasa	2
1.3	Instruksi KIRI	2
2.1	Tampilan utama website KIRI	5
2.2	Multi Bahasa	6
2.3	Instruksi KIRI	6
2.4	Add Marker	18
2.5	Contoh Marker Clustering	19
2.6	Contoh HeatMap	20
3.1	Alur Komunikasi	23
3.2	Use Case	25
B.1	Hasil 1	33
B.2	Hasil 2	33
B.3	Hasil 3	33
B.4	Hasil 4	33

DAFTAR TABEL

2.1	Tabel Properti Pada Objek Marker	18
2.2	Tabel Fungsi Pada Kelas Marker	19
3.1	Tabel Skenario Memfilter Data Berdasarkan Atribut Waktu	26
3.2	Tabel Skenario Memfilter Data Berdasarkan Atribut Hari	26
3.3	Tabel Skenario Memfilter Data Berdasarkan Atribut Start/Finish	27
3.4	Tabel Skenario Penggunaan Google Maps Javascript API	27

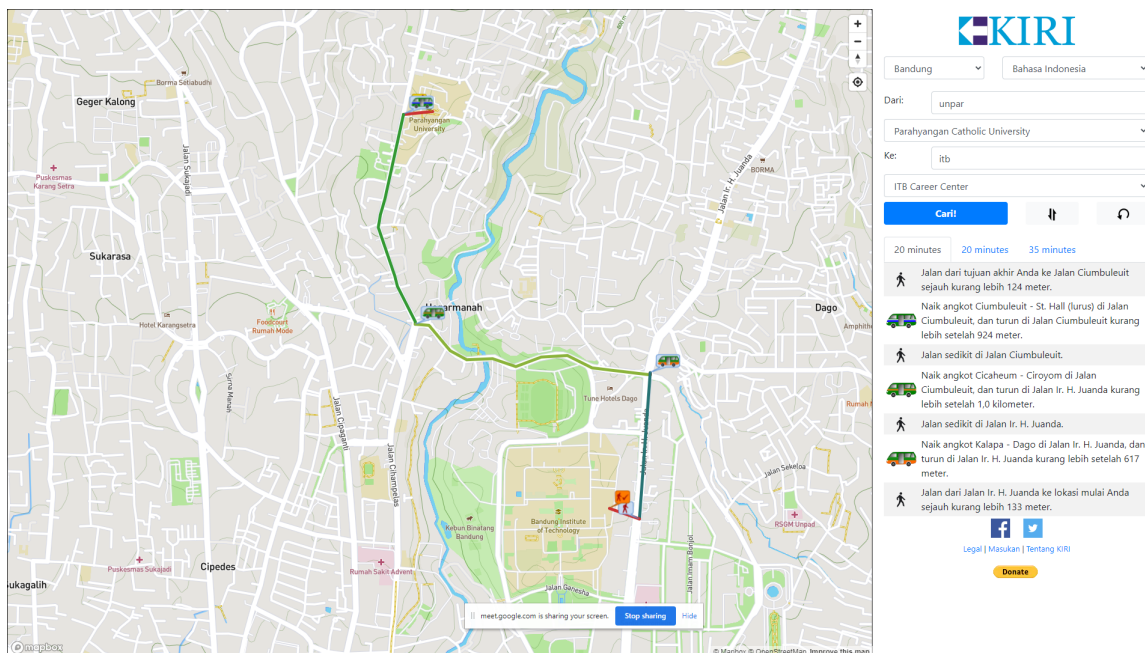
BAB 1

PENDAHULUAN

1.1 Latar Belakang

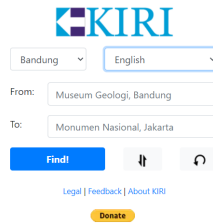
Kemajuan teknologi memudahkan manusia untuk mencari berbagai macam informasi. Salah satu informasi yang dapat diperoleh adalah informasi tentang navigasi transportasi publik. KIRI adalah perangkat lunak yang berguna sebagai navigasi antar kota menggunakan transportasi publik dengan menggunakan perangkat peta digital[1]. Pada awal pembuatannya KIRI dibuat untuk tujuan komersial. Namun karena dinilai kurang sukses, proyek KIRI sekarang menjadi open source proyek yang dapat di akses. Aplikasi KIRI memiliki beberapa fitur sebagai berikut:

- Pemilihan rute tercepat menggunakan angkutan kota. perangkat lunak KIRI dapat menentukan rute terbaik untuk berpegian dengan angkutan umum.



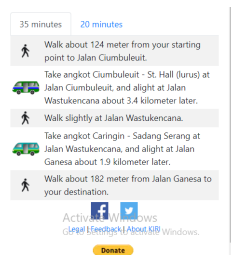
Gambar 1.1: Tampilan utama website KIRI

- Memiliki fitur multi bahasa. perangkat lunak kiri dilengkapi dengan fitur multi bahasa.



Gambar 1.2: Multi Bahasa

- Dapat menampilkan instruksi lengkap mencapai tujuan. perangkat lunak KIRI juga menampilkan seluruh instruksi lengkap penggunaan angkutan umum.



Gambar 1.3: Instruksi KIRI

Pada perangkat lunak KIRI seluruh aktivitas yang dilakukan oleh user sudah tercatat. Data yang tercatat disebut juga dengan data histori. Data histori KIRI memiliki jumlah record yang cukup banyak sehingga memungkinkan untuk mendapatkan informasi dari data tersebut. Tetapi data histori tersebut belum diolah secara maksimal. Data histori KIRI memiliki format *comma separated values* (csv) yang memiliki lima atribut yaitu:

- LogId unik sebagai penanda satu record didalam data histori KIRI.
- APIKey atribut api key yang digunakan ketika melakukan perintah pada perangkat lunak KIRI.
- Timestamp (UTC) atribut untuk mencatat waktu perintah dilakukan format berbentuk *timestamp*.
- Action jenis action yang dilakukan user pada saat menggunakan perangkat lunak KIRI. Terdapat empat nilai action pada data histori KIRI yakni:
 - *PAGELOAD*.
 - *SEARCHPLACE*.
 - *WIDGETLOAD*.
 - *FINDROUTE*.
- AdditionalData atribut yang digunakan untuk mencatat informasi tambahan berdasarkan action yang dipilih. Nilai pada additionaldata akan bergantung pada action yang dipilih:
 - jika action bernilai *PAGELOAD* maka additionaldata akan bernilai ip dari pengakses.
 - jika action bernilai *SEARCHPLACE* maka additionaldata akan bernilai keyword yang dituliskan oleh pengakses.
 - jika action bernilai *FINDROUTE* maka additionaldata akan bernilai posisi tempat dan tujuan dalam bentuk longitude dan longitude yang dicari oleh pengakses.

- jika action bernilai *WIDGETLOAD* maka additionaldata akan bernilai alamat url dari penyedia widget.

Visualisasi Data adalah teknik untuk mengkomunikasikan data atau informasi dengan menggunakan objek visual seperti *graphic*, *chart*, *diagram*, dll. Salah satu objek visual yang dapat digunakan untuk merepresentasikan data adalah *Google Maps*.

Metode yang akan digunakan dalam memvisualisasikan data adalah *Heat Map* dan *Marker Clustering*. *Heat Map* adalah teknik visualisasi data yang menunjukkan besarnya suatu fenomena sebagai warna dalam dua dimensi. Sedangkan *Marker Clustering* adalah teknik visualisasi data yang mengelompokkan *marker* atau *pointer* yang jarak *latitude* dan *longitude* nya saling berdekatan antara suatu *marker* dengan marker yang lainnya.

Pada skripsi ini akan dibangun perangkat lunak yang dapat memvisualisasikan data histori KIRI. Perangkat lunak ini akan menggunakan metode visualisasi *heat map* dan *marker clustering* dari hasil visualisasi tersebut akan diambil suatu pola kesimpulan dari data histori KIRI.

1.2 Rumusan Masalah

Rumusan masalah dari topik ini adalah sebagai berikut:

- Bagaimana memvisualisasikan data histori KIRI?
- Bagaimana menemukan pola dari data histori KIRI?
- Bagaimana penerapan metode *heat map* pada visualisasi data histori KIRI?
- Bagaimana penerapan metode *marker clustering* pada visualisasi data histori KIRI?

1.3 Tujuan

Tujuan dari topik ini adalah sebagai berikut:

- Mempelajari *Google Maps Javascript API*.
- Melakukan observasi data.
- Mengimplementasikan metode *Heat map* pada visualisasi data histori KIRI.
- Mengimplementasikan metode *Marker clustering* pada visualisasi data histori KIRI.

1.4 Batasan Masalah

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetur a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetur. Nullam elementum, urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.

1.5 Metodologi

Metodologi yang digunakan dalam penelitian ini adalah:

1. Mempelajari *Google Maps Javascript API* khususnya *Heat Map* dan *Marker Clustering*.
2. Analisis masalah perangkat lunak yang akan dibangun.
3. Merancang perangkat lunak yang akan dibangun.
4. Membangun perangkat lunak yang mengimplementasikan *Heat Map* atau *Marker Clustering* dengan memanfaatkan *Google Maps Javascript API*.
5. Menentukan pola dari hasil visualisasi data.
6. Analisis hasil pengujian dan mengambil kesimpulan.

1.6 Sistematika Pembahasan

Laporan penelitian tersusun ke dalam enam bab secara sistematis sebagai berikut.

- Bab 1 Pendahuluan
Berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, dan sistematika pembahasan.
- Bab 2 Dasar Teori
Berisi metode penentuan pola, *library Google Maps* dan bahasa pemrograman *Javascript*
- Bab 3 Analisis
Berisi analisis masalah terkait implementasi *Goole Map*, studi kasus teknik penentuan pola yang diimplementasikan, dan gambaran umum perangkat lunak yang meliputi diagram aktivitas dan diagram kelas.
- Bab 4 Perancangan Perangkat Lunak
Berisi perancangan perangkat lunak yang akan dibangun, meliputi perancangan antarmuka, diagram kelas lengkap dan masukan perangkat lunak.
- Bab 5 Implementasi dan Pengujian
Berisi implementasi antarmuka perangkat lunak, pengujian fungsional, pengujian eksperimental serta kesimpulan dari pengujian.
- Bab 6 Kesimpulan dan Saran
Berisi kesimpulan dari awal hingga akhir penelitian dan saran untuk penelitian berikutnya.

BAB 2

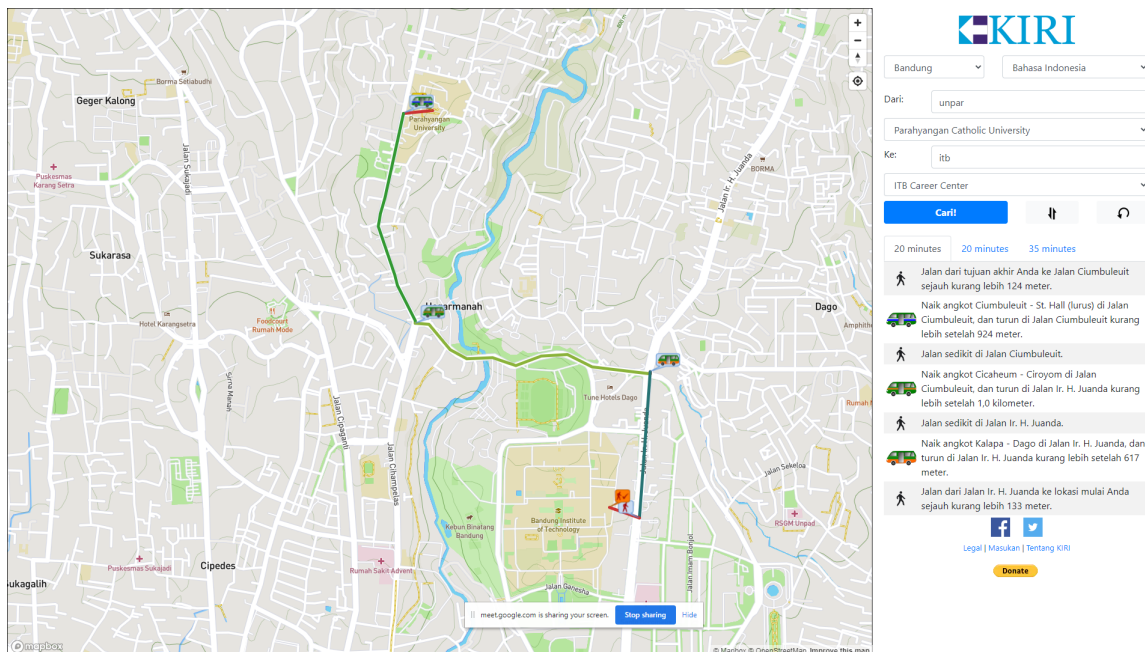
LANDASAN TEORI

Pada bab ini dijelaskan dasar-dasar teori mengenai KIRI website, JSON, CSV, dan *Google Maps Javascript API*

2.1 KIRI Website

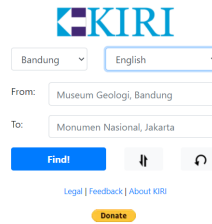
KIRI adalah aplikasi navigasi angkutan umum berbasis web yang melayani Bandung dan kota-kota lain di Indonesia.[1]. Pada awal pembuatannya KIRI dibuat untuk tujuan komersial. Namun karena dinilai kurang sukses proyek KIRI sekarang menjadi open source proyek yang dapat di akses. Aplikasi KIRI memiliki beberapa fitur sebagai berikut:

- Pemilihan rute tercepat menggunakan angkutan kota. perangkat lunak KIRI dapat menentukan rute terbaik untuk berpegangan dengan angkutan umum.



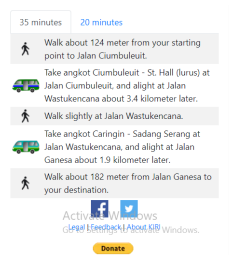
Gambar 2.1: Tampilan utama website KIRI

- Memiliki fitur multi bahasa. perangkat lunak kiri dilengkapi dengan fitur multi bahasa.



Gambar 2.2: Multi Bahasa

- Dapat menampilkan instruksi lengkap mencapai tujuan. perangkat lunak kiri juga menampilkan seluruh instruksi lengkap penggunaan angkutan umum.



Gambar 2.3: Instruksi KIRI

2.2 JSON

JSON (*Javascript Object Notation*) adalah format teks untuk melakukan serialisasi data terstruktur[2]. JSON berasal objek literal *javascript*, Seperti yang didefinisikan oleh bahasa pemrograman *ECMAScript*, JSON dapat mewakili empat tipe data primitif (*strings*, *numbers*, *booleans*, *null*) dan dua tipe data terstruktur (*objects*, *arrays*).

2.2.1 JSON Grammar

JSON teks terdiri dari sekumpulan Token. Set Token mencakup enam *structural characters*, *strings*, *numbers* dan tiga nama literal. JSON teks merupakan *serialized value*. JSON membatasi JSON teks menjadi *object* atau *array*. Implementasi JSON hanya boleh menghasilkan *object* atau *array*. JSON memiliki enam *structural characters* yaitu :

- `begin-array` = `ws %x5B ws` ; [left square bracket.
- `begin-object` = `ws %x7B ws` ; { left curly bracket.
- `end-array` = `ws %x5D ws` ;] right square bracket.
- `end-object` = `ws %x7D ws` ; } right curly bracket.
- `name-separator` = `ws %x3A ws` ; : colon.
- `value-separator` = `ws %x2C ws` ; , comma.

Penggunaan *whitespace* sebelum atau setelah enam *structural characters* diperbolehkan.

```
ws = *(
    %x20 /           ; Space
    %x09 /           ; Horizontal tab
    %x0A /           ; Line feed or New line
    %x0D )           ; Carriage return
```

2.2.2 Values

JSON value harus berupa *object*, *array*, *number*, atau *string*, atau salah satu dari tiga *literal names*:

- `true`.
- `false`.
- `null`.

Setiap *literal names* harus tersusun dari huruf kecil. Selain *literal names* yang disebutkan diatas tidak ada *literals names* lain yang izinkan.

```
value = false / null / true / object / array / number / string
```

```
false = %x66.61.6c.73.65      ; false
```

```
null  = %x6e.75.6c.6c        ; null
```

```
true   = %x74.72.75.65       ; true
```

2.2.3 Objects

Object direpresentasikan dengan sepasang kurung kurawal () kosong atau berisi *name/value pair* atau (*members*). Nama pada *objects* berupa *string*. Setelah nama akan diikuti oleh simbol *colon* (:) yang berfungsi untuk memisahkan nama dengan *value*. Setelah *value* akan diikuti dengan simbol koma (,). Nama pada *Object* harus bersifat unik.

```
object = begin-object [ member *( value-separator member ) ]
              end-object
```

```
member = string name-separator value
```

Object dengan susunan nama yang unik akan dapat beroperasi dan diimplementasikan oleh semua software. Jika susunan nama pada *object* tidak bersifat unik maka respon perangkat lunak yang menerima *object* tersebut tidak dapat diprediksi.

2.2.4 Arrays

Arrays direpresentasikan dengan sepasang kurung siku ([]) kosong atau berisi satu atau lebih element. Elements pada array akan dipisahkan dengan simbol koma (,).

```
array = begin-array [ value *( value-separator value ) ] end-array
```

Tidak ada persyaratan bahwa *value* pada array harus memiliki tipe data yang sama.

2.2.5 Numbers

Numbers direpresentasikan menggunakan basis 10 *decimal*, *Number* bisa memiliki *prefix* seperti simbol minus (-), *Number* juga biasanya dilengkapi dengan pecahan dan eksponen. Dalam penulisan *number leading zero* tidak diperbolehkan. *Pecahan* adalah *decimal point* yang diikuti oleh satu atau lebih digit. *Eksponen* biasanya diawali dengan *character* E yang bisa diikuti dengan simbol opsional *plus* atau *minus*. Huruf E dan simbol opsional akan diikuti oleh satu atau lebih digit. Nilai *Numeric* yang tidak diperbolehkan dalam aturan penulisan dibawah seperti (*NaN* dan *Infinity*).

```

number = [ minus ] int [ frac ] [ exp ]

decimal-point = %x2E          ; .

digit1-9 = %x31-39           ; 1-9

e = %x65 / %x45              ; e E

exp = e [ minus / plus ] 1*DIGIT

frac = decimal-point 1*DIGIT

int = zero / ( digit1-9 *DIGIT )

minus = %x2D                  ; -

plus = %x2B                   ; +

zero = %x30                   ; 0

```

Spesifikasi ini memungkinkan implementasi untuk menetapkan batasan pada rentan dan ketepatan angka yang diterima. Sejak banyak *software* mengimplementasikan *IEEE 754-2008 binary64 (double precision) numbers*, Untuk mendapatkan performa yang baik maka *implementasi* tidak boleh melebihi *precision* atau *range* yang disediakan.

2.2.6 Strings

String direpresentasikan dengan simbol *quotation marks* semua *unicode* akan diletakan diantara simbol *quotation marks*, Kecuali beberapa *character* yang khusus yang harus melalui proses *escape*:

- Quotation mark.
- Reverse solidus.
- Control characters (*U+0000,U+001F*). .

Untuk melakukan *escape* pada *extended character* atau pada *character* yang tidak dalam bentuk penulisan standart, *character* dapat direpresentasikan kedalam *12-character sequence*.

```

string = quotation-mark *char quotation-mark

char = unescaped /
  escape (
    %x22 /          ; "      quotation mark  U+0022
    %x5C /          ; \      reverse solidus  U+005C
    %x2F /          ; /      solidus          U+002F
    %x62 /          ; b      backspace        U+0008
    %x66 /          ; f      form feed        U+000C
    %x6E /          ; n      line feed        U+000A
    %x72 /          ; r      carriage return  U+000D
    %x74 /          ; t      tab              U+0009
    %x75 4HEXDIG ) ; uXXXX  U+XXXX

escape = %x5C          ; \

```

quotation-mark = %x22 ; "

unescaped = %x20-21 / %x23-5B / %x5D-10FFFF

2.2.7 JSON Example

Berikut ini adalah contoh penulisan *JSON*

- Contoh JSON Object

```
{
  "Image": {
    "Width": 800,
    "Height": 600,
    "Title": "View from 15th Floor",
    "Thumbnail": {
      "Url": "http://www.example.com/image/481989943",
      "Height": 125,
      "Width": 100
    },
    "Animated" : false ,
    "IDs": [116, 943, 234, 38793]
  }
}
```

Image member adalah sebuah *object* yang memiliki *Thumbnail member* yang merupakan sebuah *object* dan *IDs Member* yang merupakan sebuah *array*.

- Contoh JSON Array dengan dua buah *elements* yang berupa *object*

```
[
  {
    "precision": "zip",
    "Latitude": 37.7668,
    "Longitude": -122.3959,
    "Address": "",
    "City": "SAN FRANCISCO",
    "State": "CA",
    "Zip": "94107",
    "Country": "US"
  },
  {
    "precision": "zip",
    "Latitude": 37.371991,
    "Longitude": -122.026020,
    "Address": "",
    "City": "SUNNYVALE",
    "State": "CA",
    "Zip": "94085",
    "Country": "US"
  }
]
```

- Contoh JSON teks yang hanya memiliki *values*

```
"hello world"
32
false
```

2.3 CSV

CSV (*Comma Separated Values*) adalah format penyajian data yang sudah umum digunakan di banyak program *spreadsheet*, CSV memiliki format memisahkan setiap value dengan simbol titik koma (;) dan menggunakan baris baru sebagai penanda pemisah antar element data[3].

2.3.1 CSV Format

Tidak ada spesifikasi *formal* dalam penulisan csv. Format csv yang akan dituliskan pada dokumen ini adalah format yang paling banyak diimplementasikan:

- Setiap field diletakkan pada baris terpisah dan dipisahkan oleh *line break (CRLF)* contoh:

```
aaa , bbb , ccc CRLF
zzz , yyy , xxx CRLF
```

- *Field* terakhir pada format csv tidak harus menggunakan *line break (CRLF)* contoh:

```
aaa , bbb , ccc CRLF
zzz , yyy , xxx
```

- Memungkinkan adanya eksternal *header* yang memiliki aturan penulisan yang sama dengan element pada csv. Nilai pada eksternal *header* akan mewakili nama yang tercantum pada *field* jumlah eksternal *header* harus sama dengan jumlah kolom pada normal record contoh:

```
field_name , field_name , field_name CRLF
aaa , bbb , ccc CRLF
zzz , yyy , xxx CRLF
```

- Memungkinkan ada satu atau lebih, *field* yang dipisahkan oleh koma (,). Setiap baris harus memiliki jumlah *field* yang sama pada satu *file*. Spasi dianggap sebuah element pada *field* yang tidak dapat diabaikan. Pada field terakhir sebuah baris tidak boleh diberikan simbol koma (,) contoh:

```
aaa , bbb , ccc
```

- Setiap *Field* bisa menggunakan simbol *double quotes*. Jika *field* tidak menggunakan simbol *double quotes* maka *double quotes* tidak akan ditampilkan pada *fields*

```
"aaa" , "bbb" , "ccc" CRLF
zzz , yyy , xxx
```

- Jika simbol *double quotes* merupakan salah satu elemen pada *field*, Maka simbol *double quotes* tersebut harus diescape dengan memberikan simbol *double quotes* lain didalam *field* tersebut contoh:

```
"aaa" , "b" "bb" , "ccc"
```


2.4 Node.js

Node.js merupakan *asynchronous event-driven JavaScript runtime*. Dengan menggunakan *Node.js* memungkinkan untuk menjalankan perintah *javascript* tanpa menggunakan *web browser*, *Node.js* memungkinkan untuk menjalankan dan melakukan *server-side scripting*[4]. *Node.js* didesain untuk membuat *network applications* yang *scaleable*. Berikut ini contoh syntax *Node.js*:

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

2.4.1 Struktur File Node.js Project

Pada saat membuat proyek baru, *Node.js* membuat *file-file* dasar secara otomatis. *File-file* tersebut memiliki fungsi tersendiri. Folder *node_modules* berfungsi sebagai penyimpanan semua *library* yang diinstal melalui npm. Semua *library* yang diinstall melalui npm akan dicatat pada file *package.json*, Hal ini bertujuan untuk mempermudah proses *maintenance library* pada proyek *node.js*.

Struktur dari *file-file* akan berbentuk seperti:

```
root
├── node_modules
│   ├── "file1"
│   └── "file2"
└── package.json
```

2.4.2 Node Package Manager

Node Package Manager(NPM) adalah *software registry* yang dimiliki oleh *Node.js*, NPM memungkinkan pengguna untuk mempublikasi atau menggunakan *software library*[5]. NPM terdiri dari tiga komponen penting yaitu:

- NPM website.
- NPM CLI (*Command Line Interface*).
- NPM *Registry*.

NPM memiliki beberapa kegunaan antara lain:

- Mendownload *Software Library*.
- Menjalankan *packages* tanpa harus mendownload *npm*.
- Mempublikasikan *Tools* atau *Software Library*.

2.4.3 NPM CLI

NPM akan terinstall secara otomatis pada perangkat keras ketika menambahkan *node.js*. Untuk mempermudah proses pembuatan perangkat lunak, Penggunaan *Command Line Interface* (CLI) akan menjadi salah satu *point* penting. *NPM* memiliki tiga komponen utama dalam penulisan perintah CLI, Komponen ini akan berbentuk seperti:

```
npm <command> [args]
```

NPM CLI memiliki perintah-perintah yang dapat digunakan untuk membantu melakukan *maintenance* terhadap *library* pada projek *node.js*. Berikut perintah-perintah pada *npm cli*:

- Command untuk menginisialisasi *npm package* file dapat menggunakan perintah:

```
npm init [--force|-f|--yes|-y|--scope]
npm init <@scope> (same as 'npx <@scope>/create ')
npm init [<@scope>/]<name> (same as 'npx [<@scope>/]create --<name>')
```

- Command untuk menambahkan *package / library* dapat menggunakan perintah:

```
npm install (with no args, in package dir)
npm install [<@scope>/]<name>
npm install [<@scope>/]<name>@<tag>
npm install [<@scope>/]<name>@<version>
npm install [<@scope>/]<name>@<version range>
npm install <alias>@npm:<name>
npm install <git-host>:<git-user>/<repo-name>
npm install <git repo url>
npm install <tarball file>
npm install <tarball url>
npm install <folder>
```

- Command untuk menghapus *package / library* dapat menggunakan perintah:

```
npm uninstall [<@scope>/]<pkg>[@<version>]
```

2.5 Expressjs

Express.js merupakan *web application framework* untuk *node.js*. *Express.js* menyediakan *robust feature* dalam pembuatan perangkat lunak berbentuk situs web maupun perangkat bergerak[6].

2.5.1 Instalasi

Express.js dapat diinstal dengan menggunakan *npm*. Untuk menambahkan *package express.js* dapat menggunakan perintah:

```
npm install express --save
```

Ketika menjalankan perintah diatas maka secara otomatis akan menambahkan *library express.js* yang akan disimpan pada *folder package.json*.

2.5.2 Struktur File Express.js

Express.js tidak memberikan aturan baku dalam penyusunan struktur file dalam pengembangan perangkat lunak. namun *express.js* menyediakan *application generator* yang disebut *express generator* untuk mempermudah pihak pengembang dalam pembuatan perangkat lunak[6]. Untuk dapat menggunakan *express generator* pengembang harus menambahkan *library express generator* melalui npm dengan menggunakan perintah:

```
npx express-generator
```

Secara otomatis *express* akan menghasilkan sebuah folder yang memiliki struktur file seperti:

```
app.js
├── bin
│   └── www
├── package.json
├── public
│   ├── images
│   ├── javascripts
│   └── stylesheets
├── routes
│   ├── index.js
│   └── users.js
└── views
    ├── index.pug
    ├── error.pug
    └── layout.pug
```

2.5.3 Routing

Routing adalah proses untuk menentukan cara perangkat lunak merespon *input* melalui beberapa *endpoint*. Dalam pembuatan *routing* pada *express.js* terdapat empat komponen penting yaitu:

- Instansi dari *express.js* (*app*).
- *Method http method*.
- *Path*
- Perintah yang akan dijalankan jika *route* dijalankan *Handler*.

Pembuatan *routing* pada *express.js* memiliki struktur perintah sebagai berikut:

```
app.METHOD(PATH, HANDLER)
```

Berikut ini beberapa contoh perintah pembuatan *route* pada *express.js*

```
app.get('/', function (req, res) {
  res.send('Hello World!')
})

app.post('/', function (req, res) {
  res.send('Got a POST request')
})

app.put('/user', function (req, res) {
  res.send('Got a PUT request at /user')
```

```

    })

    app.delete('/user', function (req, res) {
        res.send('Got a DELETE request at /user')
    })

```

2.5.4 Menampilkan File Statis

Untuk dapat menampilkan *file* yang bersifat statis seperti foto, *javascript*, dan *css*, *express.js* telah menyediakan objek

```
express.static(root, [options])
```

Parameter *root* menandakan *root directory* untuk menampilkan *file* statis. Contoh perintah untuk menampilkan *file* statis:

```
app.use('/static', express.static('public'))
```

Parameter *'/static'* bertujuan untuk membuat *virtual path* yang memiliki *prefix* *'/static'*, sedangkan parameter *'public'* menandakan bahwa *file* statis berada didalam *folder public*. Perintah diatas ketika dijalankan maka *express* akan secara otomatis membuat *route* seperti:

```

http://localhost:3000/static/images/kitten.jpg
http://localhost:3000/static/css/style.css
http://localhost:3000/static/js/app.js
http://localhost:3000/static/images/bg.png
http://localhost:3000/static/hello.html

```

yang dapat digunakan untuk mengakses *file* statis.

2.6 Google Maps Javascript API

Pada subbab ini akan menjelaskan tentang *google maps javascript api* beserta kelas-kelas yang dimilikinya. Google Maps adalah layanan pemetaan web yang dikembangkan oleh Google. Menawarkan citra satelit, foto udara, dan peta jalan yang interaktif, kondisi lalu lintas secara *real time*. Dalam pengembangannya *google maps* memiliki akses pendukung untuk bahasa pemrograman *javascript*. Berikut ini beberapa layanan yang telah disediakan oleh *google maps javascript api*^[7]:

- *Maps*.
- *Drawing Object*.
- *Street Views*.
- *Routes*

2.6.1 Map

Map adalah sebuah objek pada *google maps javascript api* yang digunakan untuk membuat objek *map* didalam *html* elemen. Untuk dapat menginisialisasi objek *map* pengembang harus dapat menggunakan constructor yang memiliki perintah:

```

Map(mapDiv[, opts])
Parameters:
mapDiv: Element
opts: MapOptions optional

```

Pada *constructor* diatas terdapat dua paramete:

- mapDiv.
- MapOptions

MapDiv adalah elemen html dimana objek map akan diinisialisasi, parameter ini bersifat wajib. *MapOptions* adalah sebuah objek yang dapat digunakan untuk mengatur *property* yang dimiliki oleh objek *map*. Berikut ini adalah contoh menginisialisasi objek *map*:

```
function initMap() {
    let mapOptions = {
        center: {lat: -6.914744, lng: 107.609810},
        zoom: 12,
    }
    let map = new google.maps.Map(document.getElementById("map"), mapOptions);
    return map;
}
```

Ketika fungsi *initMap()* dijalankan maka *google maps javascript api* akan membuat objek map didalam *html dom* yang memiliki id='map' dan akan mengatur *property* yang dimiliki sesuai dengan objek *mapOptions*.

Objek *map* telah menyediakan fungsi-fungsi yang bisa langsung digunakan oleh pengembang, beberapa fungsi yang disediakan oleh objek *map* adalah:

- Fungsi *fitBounds* berguna untuk menentukan *view port* dari objek *map* sesuai dengan *bounds* yang diberikan. Fungsi ini memiliki struktur perintah seperti:

```
fitBounds(bounds[, padding])
Parameters:
bounds: LatLngBounds|LatLngBoundsLiteral
padding: number|Padding optional
Return Value: None
```

- Fungsi *getBounds* berguna untuk mendapatkan *view port* dari objek *map*. Fungsi ini memiliki struktur perintah seperti:

```
getBounds()
Parameters: None
Return Value: LatLngBounds
```

- Fungsi *getCenter* berguna untuk mendapatkan posisi yang ditunjukan pada objek *map* posisi yang didapatkan akan berbentuk *longitude* dan *latitude*. Fungsi ini memiliki struktur perintah seperti:

```
getCenter()
Parameters: None
Return Value: LatLng
```

- Fungsi *getClickableIcons* berguna untuk mendapatkan *clickable icon* setiap *clickable icon* merupakan *point of interest* pada *map*. Fungsi ini memiliki struktur perintah seperti:

```
getClickableIcons()
Parameters: None
Return Value: boolean
```

- Fungsi *getMapTypeId* berguna untuk mendapatkan *id* dari jenis *map* yang digunakan . Fungsi ini memiliki struktur perintah seperti:

```
getMapTypeId()
Parameters:  None
Return Value: MapTypeId | string
```

Google Maps Javascript API menyediakan variabel konsanta / *constant* yang berfungsi untuk menentukan jenis peta yang akan digunakan pada objek *map*. Konsanta ini dapat memiliki empat nilai yaitu:

- HYBRID jenis peta ini akan menampilkan layar *transparan* pada jalan-jalan utama pada citra satelit.
 - ROADMAP jenis peta ini akan menampilkan *street map*.
 - SATELLITE jenis peta ini akan menampilkan citra satellite.
 - TERRAIN jenis peta ini akan menampilkan bentuk nyata dari kondisi geologi suatu tempat.
- Fungsi *setMapTypeId* berguna untuk membuat atau mengubah *mapTypeId*. Fungsi ini memiliki struktur perintah seperti:

```
setMapTypeId(mapTypeId)
Parameters:
mapTypeId:  MapTypeId | string
```

- Fungsi *setZoom* berguna untuk mengubah *zoom value* yang dimiliki oleh objek *map*.

```
setZoom(zoom)
Parameters:
zoom:  number
```

- Fungsi *setMapOption* berguna untuk mengubah *property mapOption* yang dimiliki oleh objek *map*.

```
setOptions(options)
Parameters:
options:  MapOptions
```

2.6.2 Sistem Koordinat Google Maps

Google Maps Javascript API menggunakan kelas *LatLng* untuk merepresentasikan koordinat secara geografis pada objek *map*. Kelas *LatLng* merupakan sebuah kelas yang merepresentasikan latitude dan longitude

- Latitude memiliki batas antara -90 sampai 90 derajat, jika ada nilai yang diluar batasan tersebut maka nilai tersebut akan dibulatkan kebatas terdekat.
- Longitude memiliki batas antara -180 sampai 180 derajat, jika ada nilai yang diluar batasan tersebut maka nilai tersebut akan dibulatkan kebatas terdekat.

Untuk dapat menginisialisasi kelas *LatLng* pada *google maps javascript api* pengembang perlu membuat *constructor* yang memiliki struktur:

```

LatLng(lat, lng[, noWrap])
Parameters:
lat:    number
lng:    number
noWrap: boolean optional

```

Membuat objek `LatLng` yang mewakili titik geografis. *latitude* ditentukan dalam derajat dalam rentang `[-90, 90]`. *longitude* ditentukan dalam derajat dalam rentang `[-180, 180]`. Set *noWrap* `true` untuk mengaktifkan nilai di luar rentang ini. Contoh penggunaan kelas *LatLng*:

```

map.setCenter(new google.maps.LatLng(-34, 151));
map.setCenter({lat: -34, lng: 151});

```

Google Maps Javascript API telah menyediakan fungsi bawaan yang dapat diakses ketika menggunakan kelas *LatLng*. Fungsi tersebut adalah:

- Fungsi *equals* fungsi ini bertujuan untuk membandingkan posisi antara objek *map*.

```

equals(other)
Parameters:
other:    LatLng
Return Value: boolean

```

- Fungsi *lat* fungsi ini bertujuan untuk mendapatkan posisi *latitude* dari objek *map*.

```

lat()
Parameters:    None
Return Value:  number
Returns the latitude in degrees.

```

- Fungsi *lng* fungsi ini bertujuan untuk mendapatkan posisi *longitude* dari objek *map*.

```

lng()
Parameters:    None
Return Value:  number
Returns the longitude in degrees.

```

- Fungsi *toJSON* fungsi ini akan mengembalikan format *json* terhadap posisi *latlng* pada objek *map*
- Fungsi *toUrlValue* Mengembalikan string dalam bentuk "lat, lng" untuk *LatLng* ini.

2.6.3 Drawing System

Pada subbab ini akan dijelaskan cara penggunaan kelas *overlay* yang disediakan oleh *google maps javascript api*. *Google Maps Javascript API* memiliki kelas-kelas yang digunakan untuk menggambar suatu objek didalam objek *map*. Beberapa kelas *overlay* yang disediakan adalah sebagai berikut:

- *Marker*.
- *Heat Map*.

2.6.4 Marker



Gambar 2.4: Add Marker

Marker adalah sebuah kelas yang dapat memunculkan *mark* / tanda pada objek *map*. Untuk dapat menginisialisasi kelas *marker* pengembang dapat menggunakan *constructor* yang memiliki struktur seperti:

```
Marker ([ opts ])
Parameters:
opts: MarkerOptions optional
```

Pada *constructor marker* terdapat parameter *markerOptions* yang bersifat optional. *MarkerOptions* merupakan sebuah objek yang dapat digunakan pada objek *mark*. *MarkerOptions* memiliki properti yang dapat digunakan seperti:

Tabel 2.1: Tabel Properti Pada Objek Marker

Properti	Deksripsi
Properti <i>anchorPoint</i> .	Nilai offset dari objek marker terhadap info window.
Properti <i>animation</i> .	Animasi yang akan digunakan ketika objek marker diinisialisasikan.
Properti <i>clickable</i> .	Properti penanda jika objek marker diklik.
Properti <i>crossOnDrag</i> .	Properti penanda jika objek marker didrag oleh pengguna.
Properti <i>cursor</i> .	Properti yang akan menunjukkan <i>cursor</i> ketika objek marker di <i>hover</i> .
Properti <i>dragable</i> .	Properti bertipe <i>boolean</i> yang akan menandakan apakah objek marker dapat dilakukan operasi <i>drag</i> .
Properti <i>icon</i> .	Properti untuk memberikan icon pada objek marker.
Properti <i>label</i> .	Properti untuk memberikan label pada objek marker.
Properti <i>map</i> .	Properti untuk menentukan objek map yang akan dipakai oleh marker.
Properti <i>opacity</i> .	Properti untuk mengakses nilai opacity dari objek marker
Properti <i>position</i> .	Properti untuk mengakses nilai posisi dari objek marker.

Contoh penginisialisasian *marker* pada objek *maps* adalah:

```
return new google.maps.Marker({
    position: location,
    label: labels[i % labels.length],
});
```

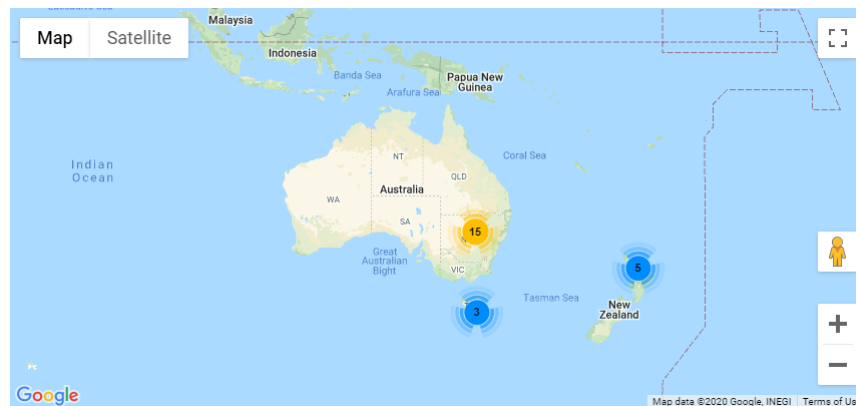
Kelas *marker* memiliki fungsi bawaan yang telah disediakan oleh antara lain seperti:

Tabel 2.2: Tabel Fungsi Pada Kelas Marker

Package	Deksripsi
<code>getAnimation</code>	Fungsi untuk mendapatkan animasi yang digunakan oleh objek marker
<code>getClickable</code>	Fungsi untuk mendapatkan status <i>clickable</i>
<code>getCursor</code>	Fungsi untuk mendapatkan nilai <i>cursor</i>
<code>getDraggable</code>	Fungsi untuk mendapatkan nilai <i>draggable</i>
<code>getIcon</code>	Fungsi untuk mendapatkan nilai <i>icon</i>
<code>getMap</code>	Fungsi untuk mendapatkan objek <i>map</i>
<code>getOpacity</code>	Fungsi untuk mendapatkan objek <i>opacity</i>
<code>getPosition</code>	Fungsi untuk mendapatkan objek <i>position</i>
<code>getShape</code>	Fungsi untuk mendapatkan <i>shape</i> dari objek <i>marker</i>
<code>getTitle</code>	Fungsi untuk mendapatkan <i>title</i> dari objek <i>marker</i>

2.6.5 Marker Clusterer

Google Maps Javascript API telah menyediakan kelas *MarkerClustererPlus* untuk dapat mengabungkan objek *marker* 2.6.4.



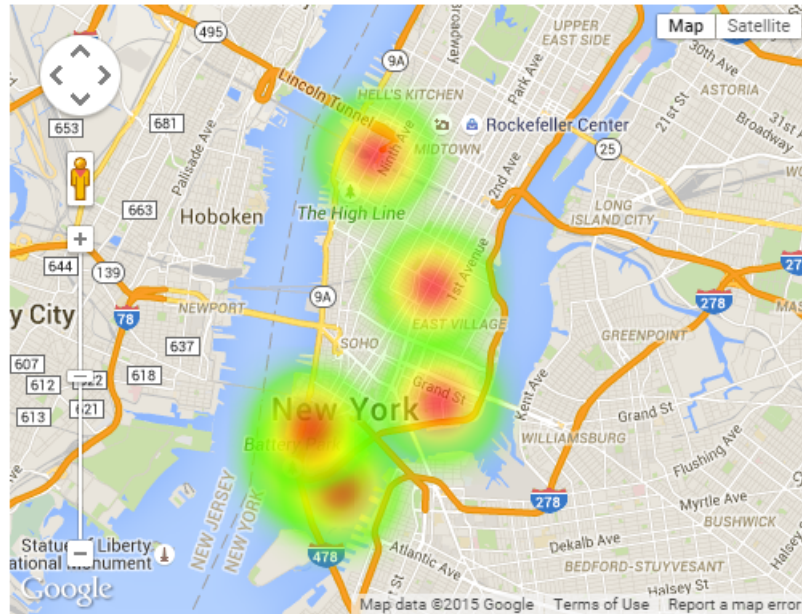
Gambar 2.5: Contoh Marker Clustering

Untuk dapat menginisialisasi kelas tersebut pihak pengembang perlu menuliskan perintah:

```
var markerCluster = new MarkerClusterer(map, markers,
    {imagePath: '${path}/m'});
```

Marker Clusterer sendiri merupakan suatu library untuk kelas *mark* sehingga kelas ini dapat menggunakan fungsi turunan dari kelas *mark*.

2.6.6 HeatMap



Gambar 2.6: Contoh HeatMap

Google Maps Javascript API telah menyediakan kelas *heatmap* untuk menampilkan *heatmap* pada objek *maps*. Untuk dapat menginisialisasi kelas ini pengembang perlu memanggil perintah *constructor* yang memiliki struktur seperti:

```
HeatmapLayer ([ opts ] )
Parameters:
opts: HeatmapLayerOptions optional
```

Kelas *HeatMap* dilengkapi dengan parameter *HeatmapLayerOptions* yang bersifat opsional, objek ini bertujuan untuk dapat mengatur *property* dari kelas *HeatMap*. Parameter *HeatMapLayerOptions* merupakan sebuah objek yang memiliki atribut sebagai berikut:

- data *titik-titik* data yang diperlukan.
- *dissipating* variabel yang menentukan apakah *heatmap* akan menghilang jika *maps* diperbesar atau diperkecil.
- *gradient* dari warna *heatmap*
- *map* attribute untuk menunjukan peta dimana *heatmap* akan ditampilkan.
- *maxIntencity* nilai maximal dari intensitas warna pada *heatmap*.
- *opacity* nilai *opacity* dari *heatmap*.
- *radius* nilai *radius* dari *heatmap*.

Contoh penginisialisasian kelas *heatmap*:

```
let heatmap = new google.maps.visualization.HeatmapLayer({
  data: heatmapData
});
```

Kelas *HeatMap* memiliki fungsi-fungsi bawaan yang telah disediakan oleh *google maps* beberapa fungsi tersebut adalah:

- `getData()` fungsi ini akan mengembalikan data point pada objek *heatmap*.
- `getMap()` fungsi ini akan mengembalikan objek *map*.
- `setData(data)` fungsi ini akan memasukan data pada objek *heatmap*.
- `setMap(map)` fungsi ini akan memasukan objek *map* pada objek *heatmap*.
- `setOption(option)` fungsi ini akan memasukan objek *HeatMapLayerOptions* pada objek *heatmap*.

BAB 3

ANALISIS

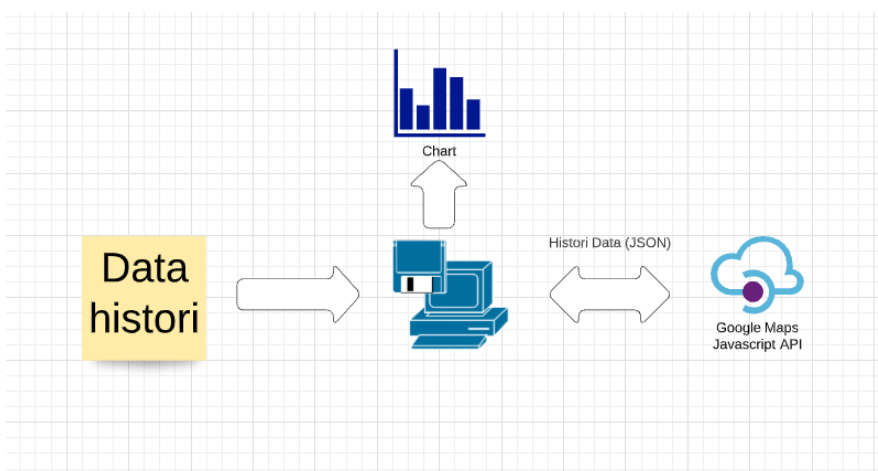
Pada bab ini dijelaskan mengenai deskripsi perangkat lunak, analisis data histori KIRI , analisis perangkat lunak, dan analisis *heat map* dan *marker clustering* menggunakan *Google Maps Javascript API*.

3.1 Deskripsi Perangkat Lunak

Pada skripsi ini akan dibangun aplikasi yang bertujuan untuk menemukan pola dengan tool visualisasi. Aplikasi ini akan menggunakan data histori perangkat lunak KIRI sebagai sumber data. Aplikasi ini akan dibangun menggunakan *tools* nodejs dan akan mengimplementasikan *google maps javascript api* sebagai tool untuk melakukan visualisasi data. Beberapa fitur yang dirancang dalam perangkat lunak ini:

- Perangkat lunak dapat memfilter data berdasarkan atribut *start/finish*.
- Perangkat lunak dapat memfilter data berdasarkan atribut waktu.
- Perangkat lunak dapat memfilter data berdasarkan atribut hari.
- Perangkat lunak dapat menampilkan data dalam bentuk *heat map*
- Perangkat lunak dapat menampilkan data dalam bentuk *marker clustering*.

Perangkat lunak ini akan melakukan filter data dan memproses data histori kiri dan akan menampilkan data tersebut kedalam bentuk *heat map* dan *marker clustering* untuk mendapatkan pola-pola tertentu berdasarkan data histori tersebut. Berikut ini adalah gambaran alur komunikasi perangkat lunak yang akan dibuat:



Gambar 3.1: Alur Komunikasi

1. Perangkat lunak akan mengolah data histori kiri sesuai dengan input yang diberikan.
2. Data yang sudah diolah akan diubah kedalam format *JSON*.
3. Data yang sudah diolah akan digunakan oleh *Google Maps Javascript API* untuk dapat dilakukan visualisasi data.
4. Perangkat lunak akan memvisualisasikan data kedalam bentuk *heat map* dan *marker clustering*.

3.2 Analisis Data Histori KIRI

Perangkat lunak yang akan menggunakan data histori KIRI sebagai sumber data untuk melakukan visualisasi data. Data histori KIRI memiliki format *csv*. Data histori KIRI memiliki lima atribut yaitu:

- LogId Id unik sebagai penanda satu record didalam data histori KIRI.
- APIKey atribut api key yang digunakan ketika melakukan perintah pada perangkat lunak KIRI.
- Timestamp (UTC) atribut untuk mencatat waktu perintah dilakukan format berbentuk *timestamp*.
- Action jenis action yang dilakukan user pada saat menggunakan perangkat lunak KIRI. Terdapat empat nilai action pada data histori KIRI yakni:
 - *PAGELOAD*.
 - *SEARCHPLACE*.
 - *WIDGETLOAD*.
 - *FINDROUTE*.
- AdditionalData atribut yang digunakan untuk mencatat informasi tambahan berdasarkan action yang dipilih. Nilai pada additionaldata akan bergantung pada action yang dipilih:
 - jika action bernilai *PAGELOAD* maka additionaldata akan bernilai ip dari pengakses.
 - jika action bernilai *SEARCHPLACE* maka additionaldata akan bernilai keyword yang dituliskan oleh pengakses.
 - jika action bernilai *FINDROUTE* maka additionaldata akan bernilai posisi tempat dan tujuan dalam bentuk longitude dan longitude yang dicari oleh pengakses.
 - jika action bernilai *WIDGETLOAD* maka additionaldata akan bernilai alamat url dari penyedia widget.

Perangkat lunak akan mengubah format data histori KIRI dari bentuk *csv* menjadi *json* dimana format atribut json akan berupa:

```
[{"timestamp": "2014-1-2:0:11",
  "start": "-6.8972513,107.6385574",
  "finish": "-6.91358,107.62718"},
 {"timestamp": "2014-1-2:0:13",
  "start": "-6.8972513,107.6385574",
  "finish": "-6.91358,107.62718"}]
```

agar dapat diparsing menggunakan *google maps javascript api*.

3.3 Analisis Perangkat Lunak

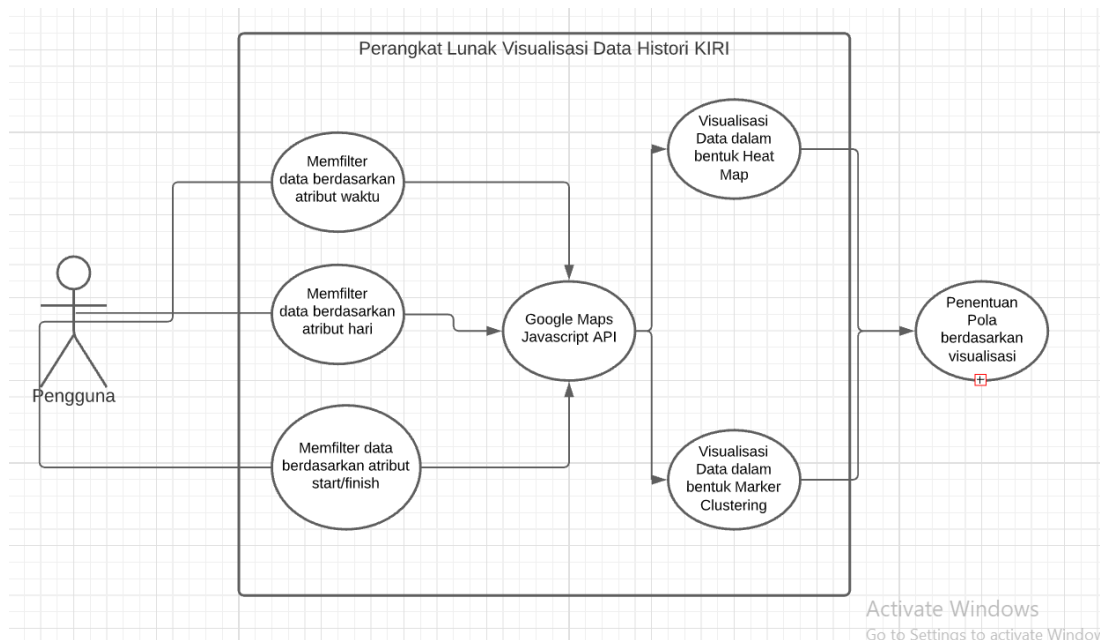
3.3.1 Analisis Kebutuhan Perangkat Lunak

Berdasarkan latar belakang, rumusan masalah, dan tujuan maka dapat didefinisikan kebutuhan - kebutuhan perangkat lunak visualisasi data histori KIRI sebagai berikut:

- Data histori KIRI Aplikasi dapat melakukan proses filter pada data histori dan menggunakan data histori KIRI untuk dapat divisualisasikan oleh karena itu diperlukan raw data histori KIRI.
- Google Maps Javascript API Aplikasi dapat melakukan visualisasi data menggunakan *heat map* dan *marker clustering* yang merupakan fitur dari *google maps javascript api*.

3.4 Use Case Diagram

Interaksi antara pengguna dengan perangkat lunak yang dibangun pada skripsi ini digambarkan pada diagram *use case* dan skenario berikut ini



Gambar 3.2:
fig:usecase

3.4.1 Use Case Skenario

Setiap fungsi yang ada pada diagram *use case* dijelaskan dengan skenario untuk memberi gambaran interaksi pengguna dengan sistem.

Tabel 3.1: Tabel Skenario Memfilter Data Berdasarkan Atribut Waktu

Nama	Memfilter Data Berdasarkan Atribut Waktu.
Deskripsi	Melakukan filter data berdasarkan input waktu yang diberikan oleh pengguna
Aktor	Pengguna
Pre-kondisi	Aplikasi sudah dijalankan dan sudah dapat mengolah raw data histori KIRI.
Alur Skenario Utama	<ol style="list-style-type: none"> 1. Sistem memuat aplikasi. 2. Sistem menampilkan data. 3. Pengguna memfilter data berdasarkan waktu". 4. Sistem melakukan proses filtering menggunakan atribut waktu. 5. Sistem memvisualisasikan data yang telah difilter oleh pengguna.

Tabel 3.2: Tabel Skenario Memfilter Data Berdasarkan Atribut Hari

Nama	Memfilter Data Berdasarkan Atribut Hari.
Deskripsi	Melakukan filter data berdasarkan input Hari yang diberikan oleh pengguna.
Aktor	Pengguna
Pre-kondisi	Aplikasi sudah dijalankan dan sudah dapat mengolah raw data histori KIRI.
Alur Skenario Utama	<ol style="list-style-type: none"> 1. Sistem memuat aplikasi. 2. Sistem menampilkan data. 3. Pengguna memfilter data berdasarkan hari". 4. Sistem melakukan proses filtering menggunakan atribut hari. 5. Sistem memvisualisasikan data yang telah difilter oleh pengguna.

Tabel 3.3: Tabel Skenario Memfilter Data Berdasarkan Atribut Start/Finish

Nama	Memfilter Data Berdasarkan Atribut Start/Finish.
Deskripsi	Melakukan filter data berdasarkan input Start/Finish yang diberikan oleh pengguna.
Aktor	Pengguna
Pre-kondisi	Aplikasi sudah dijalankan dan sudah dapat mengolah raw data histori KIRI.
Alur Skenario Utama	<ol style="list-style-type: none"> 1. Sistem memuat aplikasi. 2. Sistem menampilkan data. 3. Pengguna memfilter data berdasarkan hari". 4. Sistem melakukan proses filtering menggunakan atribut Start/Finish. 5. Sistem memvisualisasikan data yang telah difilter oleh pengguna.

Tabel 3.4: Tabel Skenario Penggunaan Google Maps Javascript API

Nama	M Penggunaan Google Maps Javascript API.
Deskripsi	Menggunakan Google Maps Javascript API untuk memvisualisasikan data histori kiri
Aktor	Pengguna
Pre-kondisi	Aplikasi sudah berhasil memproses data berdasarkan filter dari pengguna
Alur Skenario Utama	<ol style="list-style-type: none"> 1. Sistem menerima input filter dari pengguna. 2. Sistem mengubah data kedalam format JSON. 3. Sistem menggunakan Google Maps Javascript API". 4. Sistem memvisualisasikan data yang telah difilter oleh pengguna.

DAFTAR REFERENSI

- [1] Nugroho, P. dan Natali, V. (2017) Open sourcing proprietary application case study: Kiri website. *International Journal of New Media Technology*, **4**, 82–86.
- [2] Bray, T. dan Google (2014) the javascript object notation (json) data interchange format. Technical Report 8259.
- [3] Shafranovich, Y. (2005) Common format and mime type for comma-separated values (csv) files. Technical Report 7111.
- [4] Dahl, R. nodejs. <https://nodejs.org/en/about/>.
- [5] Dahl, R. npmjs. <https://docs.npmjs.com/about-npm>.
- [6] Holowaychuk, T. expressjs. <https://expressjs.com/en/starter/installing.html>.
- [7] Google Google maps reference. <https://developers.google.com/maps/documentation/javascript/reference>.

LAMPIRAN A

KODE PROGRAM

Listing A.1: MyCode.c

```

1 // This does not make algorithmic sense,
2 // but it shows off significant programming characters.
3
4 #include<stdio.h>
5
6 void myFunction( int input, float* output ) {
7     switch ( array[i] ) {
8         case 1: // This is silly code
9             if ( a >= 0 || b <= 3 && c != x )
10                 *output += 0.005 + 20050;
11             char = 'g';
12             b = 2^n + ~right_size - leftSize * MAX_SIZE;
13             c = (--aaa + &daa) / (bbb++ - ccc % 2 );
14             strcpy(a,"hello_$@?");
15         }
16         count = ~mask | 0x00FF00AA;
17     }
18 }
19
20 // Fonts for Displaying Program Code in LATEX
21 // Adrian P. Robson, nepsweb.co.uk
22 // 8 October 2012
23 // http://nepsweb.co.uk/docs/progfonts.pdf

```

Listing A.2: MyCode.java

```

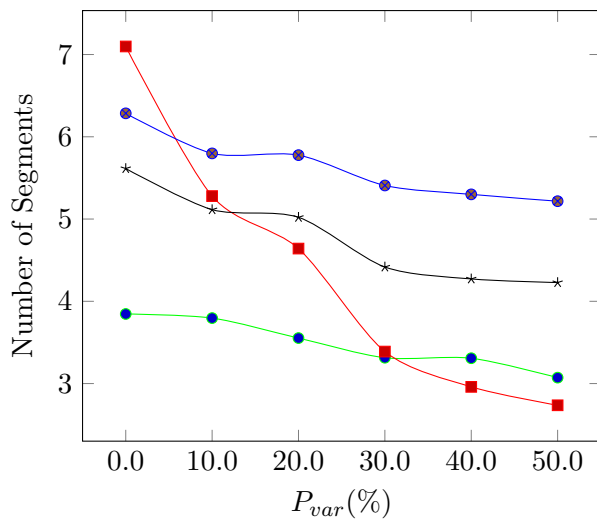
1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.HashSet;
4
5 //class for set of vertices close to furthest edge
6 public class MyFurSet {
7     protected int id; //id of the set
8     protected MyEdge FurthestEdge; //the furthest edge
9     protected HashSet<MyVertex> set; //set of vertices close to furthest edge
10    protected ArrayList<ArrayList<Integer>> ordered; //list of all vertices in the set for each trajectory
11    protected ArrayList<Integer> closeID; //store the ID of all vertices
12    protected ArrayList<Double> closeDist; //store the distance of all vertices
13    protected int totaltrj; //total trajectories in the set
14
15    /*
16     * Constructor
17     * @param id : id of the set
18     * @param totaltrj : total number of trajectories in the set
19     * @param FurthestEdge : the furthest edge
20     */
21    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
22        this.id = id;
23        this.totaltrj = totaltrj;
24        this.FurthestEdge = FurthestEdge;
25        set = new HashSet<MyVertex>();
26        ordered = new ArrayList<ArrayList<Integer>>();
27        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
28        closeID = new ArrayList<Integer>(totaltrj);
29        closeDist = new ArrayList<Double>(totaltrj);
30        for (int i = 0;i <totaltrj;i++) {
31            closeID.add(-1);
32            closeDist.add(Double.MAX_VALUE);
33        }
34    }
35
36 }

```

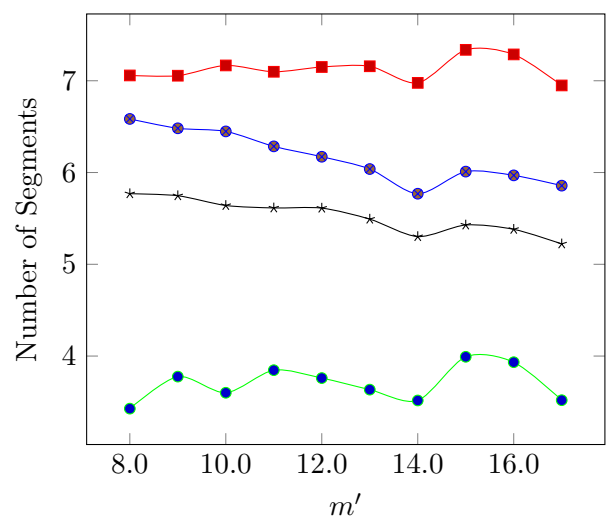

LAMPIRAN B

HASIL EKSPERIMEN

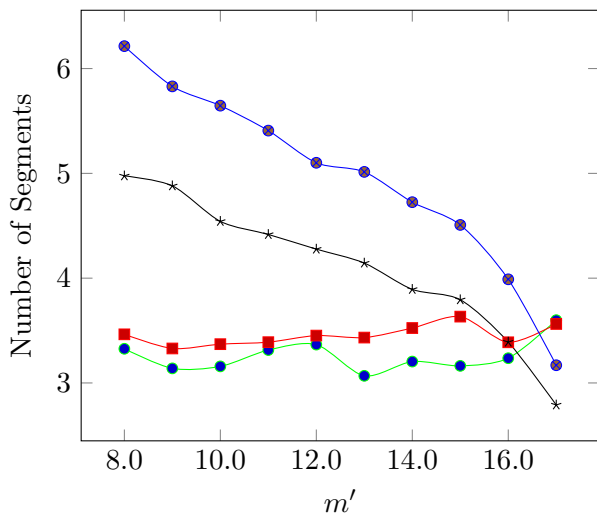
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



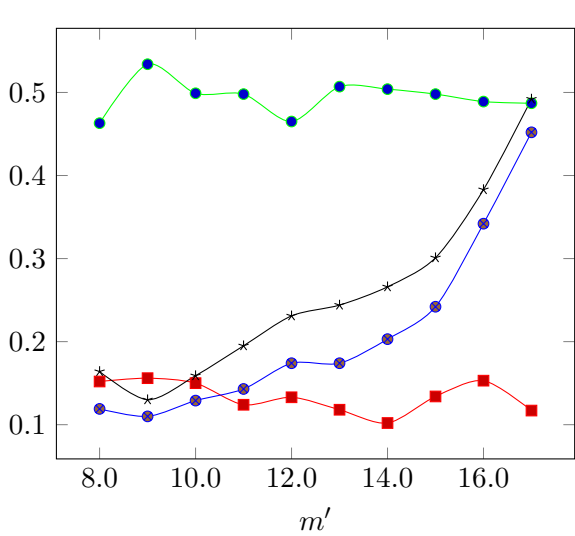
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4