

## SKRIPSI

VISUALISASI DATA HISTORI KIRI PADA GOOGLE MAPS



Jonathan Laksamana Purnomo

NPM: 2016730081

PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS  
UNIVERSITAS KATOLIK PARAHYANGAN  
2021



**UNDERGRADUATE THESIS**

**VISUALIZATION OF KIRI HISTORICAL DATA ON GOOGLE  
MAPS**



**Jonathan Laksamana Purnomo**

**NPM: 2016730081**

**DEPARTMENT OF INFORMATICS  
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES  
PARAHYANGAN CATHOLIC UNIVERSITY  
2021**



## ABSTRAK

KIRI adalah perangkat lunak yang berguna sebagai navigasi antar kota menggunakan transportasi publik dengan menggunakan perangkat peta digital.

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*.

Aplikasi ini berhasil membangun perangkat lunak berbasis website yang dapat menampilkan data histori kiri dalam bentuk *heat map* dan *marker clustering*. Pengujian dilakukan dengan dua cara pengujian fungsional dan pengujian eksperimental. Pengujian fungsional dilakukan secara otomatis menggunakan aplikasi pihak ketiga *cypress*. Pengujian eksperimental dilakukan dengan metode observasi dan telah berhasil menemukan pola-pola tertentu dari data histori KIRI.

**Kata-kata kunci:** Perangkat Lunak KIRI, Visualisasi Data, Data Histori



## ABSTRACT

KIRI is software that is useful for navigating between cities using public transportation using a digital map device.

Data Visualization is a technique for communicating data or information by using visual objects such as graphs, charts, diagrams, etc. One of the visual objects that can be used to represent data is Google Maps.

This application has succeeded in building a website-based software that can display left historical data in the form of *heat map* and *marker clustering*. Testing is carried out in two ways, functional testing and experimental testing. Functional testing is performed automatically using a third-party application *cypress*. Experimental testing was carried out using the observation method and has succeeded in finding certain patterns from the historical data of KIRI.

**Keywords:** KIRI software, Data Visualization, Historical Data



# DAFTAR ISI

<b>DAFTAR ISI</b>	<b>ix</b>
<b>DAFTAR GAMBAR</b>	<b>xi</b>
<b>1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	2
1.3 Tujuan . . . . .	2
1.4 Batasan Masalah . . . . .	3
1.5 Metodologi . . . . .	3
1.6 Sistematika Pembahasan . . . . .	3
<b>2 LANDASAN TEORI</b>	<b>5</b>
2.1 KIRI Website . . . . .	5
2.2 CSV . . . . .	6
2.2.1 CSV Format . . . . .	6
2.3 Node.js . . . . .	7
2.3.1 Struktur File Node.js (project) . . . . .	7
2.3.2 <i>Node Package Manager</i> . . . . .	7
2.3.3 NPM CLI . . . . .	8
2.4 <i>Express.js</i> . . . . .	8
2.4.1 Instalasi . . . . .	8
2.4.2 Struktur File Express.js . . . . .	9
2.4.3 Routing . . . . .	9
2.4.4 Menampilkan File Statis . . . . .	10
2.5 Google Maps Javascript API . . . . .	10
2.5.1 Map . . . . .	10
2.5.2 Sistem Kordinat Google Maps . . . . .	12
2.6 Marker . . . . .	14
2.7 Marker Clusterer . . . . .	16
2.8 HeatMap . . . . .	16
<b>3 ANALISIS</b>	<b>19</b>
3.1 Analisis Data Histori KIRI . . . . .	19
3.2 Deskripsi Perangkat Lunak . . . . .	20
3.3 Analisis Perangkat Lunak . . . . .	20
3.3.1 Analisis Kebutuhan Perangkat Lunak . . . . .	20
3.3.2 Use Case Diagram . . . . .	21
3.3.3 Use Case Skenario . . . . .	21
3.3.4 Data Flow Diagram . . . . .	23
3.3.5 Flow Chart Diagram . . . . .	25
3.3.6 Input dan Output . . . . .	26

<b>4 PERANCANGAN</b>	<b>27</b>
4.1 Perancangan Antarmuka . . . . .	27
4.2 Perancangan Diagram Fungsi . . . . .	28
4.3 Perancangan Diagram Kapabilitas . . . . .	28
4.3.1 Diagram Kapabilitas Web Client . . . . .	28
4.3.2 Diagram Kapabilitas Backend . . . . .	29
4.3.3 Diagram Kapabilitas Google API . . . . .	30
4.4 Perancangan Diagram Sequence . . . . .	31
4.5 Perancangan Pseudocode . . . . .	31
4.5.1 Perancangan Pseudocode Pada Web Client . . . . .	31
<b>5 IMPLEMENTASI DAN PENGUJIAN</b>	<b>35</b>
5.1 Implementasi . . . . .	35
5.1.1 Lingkungan Implementasi . . . . .	35
5.1.2 Implementasi Antarmuka Perangkat Lunak . . . . .	35
5.1.3 Implementasi Perangkat Lunak . . . . .	37
5.2 Pengujian . . . . .	50
5.2.1 Pengujian Fungsional . . . . .	50
5.2.2 Pengujian Eksperimental . . . . .	54
5.3 Masalah yang Dihadapi saat Implementasi . . . . .	62
<b>6 KESIMPULAN DAN SARAN</b>	<b>63</b>
6.1 Kesimpulan . . . . .	63
6.2 Saran . . . . .	63
<b>DAFTAR REFERENSI</b>	<b>65</b>
<b>A KODE PROGRAM</b>	<b>67</b>

## DAFTAR GAMBAR

1.1	Tampilan Utama Website KIRI . . . . .	1
1.2	Tampilan Heat Map . . . . .	2
1.3	Tampilan Marker Clustering . . . . .	2
2.1	Fitur-Fitur Pada Aplikasi KIRI . . . . .	5
2.2	Add Marker . . . . .	14
2.3	Contoh Marker Clustering . . . . .	16
2.4	Contoh HeatMap . . . . .	16
3.1	Alur Komunikasi . . . . .	20
3.2	Use Case Diagram . . . . .	21
3.3	Data Flow Diagram Level 0 . . . . .	24
3.4	Data Flow Diagram Level 1 . . . . .	24
3.5	KIRI Flow Chart . . . . .	26
4.1	Rancangan Antarmuka . . . . .	27
4.2	Rancangan Diagram Kelas . . . . .	28
4.3	Rancangan Diagram Kapabilitas . . . . .	29
4.4	Rancangan Diagram Kapabilitas . . . . .	30
4.5	Rancangan Diagram Kapabilitas . . . . .	30
4.6	Rancangan Diagram Interaksi . . . . .	31
5.1	Tampilan Awal Antarmuka . . . . .	36
5.2	Tampilan setelah memilih marker clustering . . . . .	36
5.3	Tampilan setelah memilih Heat Map . . . . .	37
5.4	Tampilan Filter Data Histori KIRI . . . . .	51
5.5	Tampilan Marker Data Histori KIRI . . . . .	51
5.6	Tampilan Heat Map Histori KIRI . . . . .	52
5.7	Visual regression sample result . . . . .	54
5.8	parameter tambahan . . . . .	55
5.9	Senin-Sabtu Operasional . . . . .	56
5.10	Senin-Sabtu Non Operasional . . . . .	56
5.11	Daerah titik awal terbanyak . . . . .	57
5.12	Daerah Taman Sari . . . . .	57
5.13	Daerah Taman Sari Zoomed . . . . .	58
5.14	Heat Map Taman Sari . . . . .	58
5.15	Daerah titik tujuan terbanyak . . . . .	59
5.16	Gambar Daerah Braga . . . . .	59
5.17	Daerah dengan titik start pada saat weekend . . . . .	60
5.18	Daerah Taman Sari Weekend . . . . .	61
5.19	Daerah dengan titik tujuan pada saat weekend . . . . .	61
5.20	Daerah Alun Alun Kota Bandung . . . . .	62



1

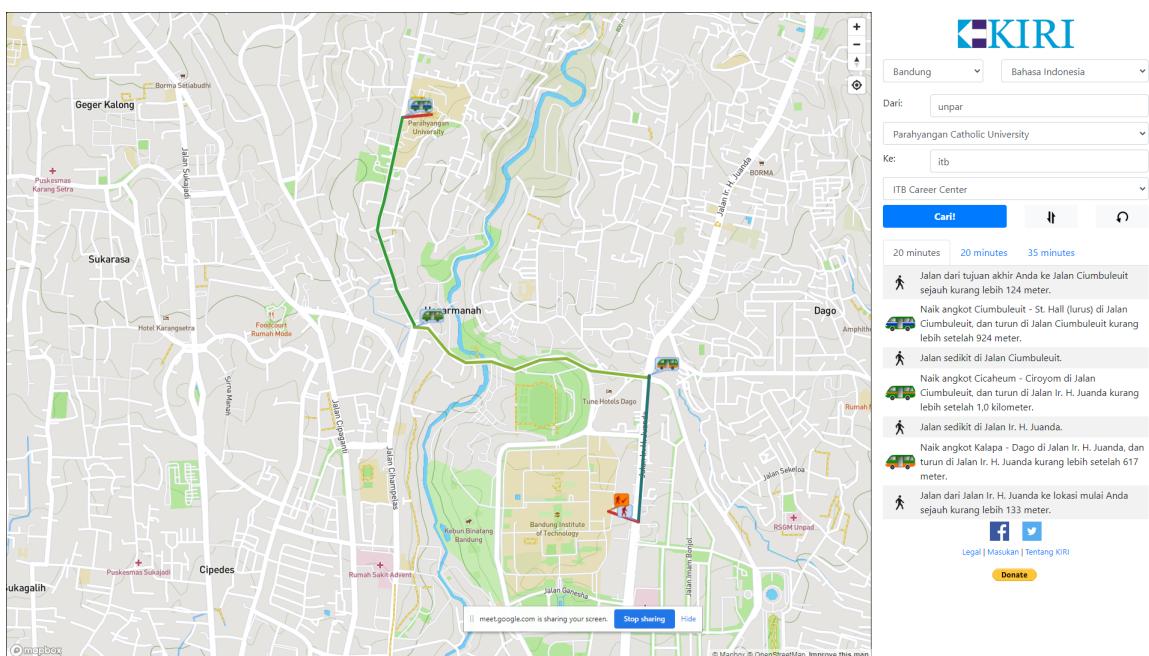
# BAB 1

2

## PENDAHULUAN

### 3 1.1 Latar Belakang

4 Kemajuan teknologi memudahkan manusia untuk mencari berbagai macam informasi. Salah satu  
5 informasi yang dapat diperoleh adalah informasi tentang navigasi transportasi publik. KIRI adalah  
6 perangkat lunak yang berguna sebagai navigasi antar kota menggunakan transportasi publik dengan  
7 menggunakan perangkat peta digital[1]. Pada awal pembuatannya KIRI dibuat untuk tujuan  
8 komersial. Namun karena dinilai kurang sukses, projek KIRI sekarang menjadi open source projek  
9 yang dapat di akses. Bentuk tampilan aplikasi KIRI dapat dilihat pada gambar 1.1.

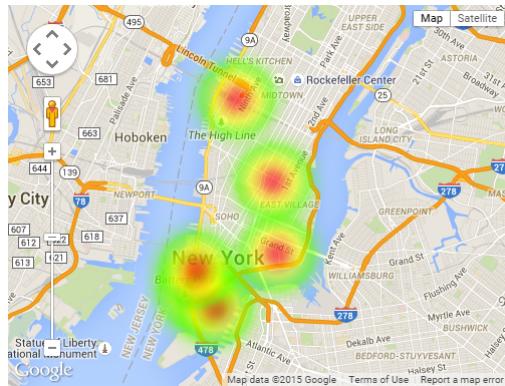


Gambar 1.1: Tampilan Utama Website KIRI

10 Pada perangkat lunak KIRI seluruh aktivitas yang dilakukan oleh user sudah tercatat. Data  
11 yang tercatat disebut juga dengan data histori. Data histori kiri memiliki jumlah record yang cukup  
12 banyak sehingga memungkinkan untuk mendapatkan informasi dari data tersebut. Tetapi data  
13 histori tersebut belum diolah secara maksimal.

14 Visualisasi Data adalah teknik untuk mengkomunikasikan data atau informasi dengan meng-  
15 gunakan objek visual seperti *graphic, chart, diagram, dll*. Salah satu objek visual yang dapat  
16 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. Contoh bentuk *heat map* dan *marker clustering* dapat dilihat pada gambar 1.2 dan 1.3



Gambar 1.2: Tampilan Heat Map



Gambar 1.3: Tampilan Marker Clustering

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:

- Menggunakan *Google Maps Javascript API*.

- 1 • Melakukan pengujian menggunakan metode observasi.
- 2 • Mengimplementasikan metode *Heat map* pada visualisasi data histori KIRI.
- 3 • Mengimplementasikan metode *Marker clustering* pada visualisasi data histori KIRI.

## 4 **1.4 Batasan Masalah**

- 5 Penelitian ini dibuat dengan batasan - batasan berikut:
- 6 1. Perangkat lunak ini hanya akan menampilkan data pada daerah Bandung saja.

## 7 **1.5 Metodologi**

- 8 Metodologi yang digunakan dalam penelitian ini adalah:
- 9 1. Mempelajari *Google Maps Javascript API* khususnya *Heat Map* dan *Marker Clustering*.
  - 10 2. Analisis perangkat lunak yang akan dibangun.
  - 11 3. Merancang perangkat lunak yang akan dibangun.
  - 12 4. Membangun perangkat lunak yang mengimplementasikan *Heat Map* atau *Marker Clustering* dengan memanfaatkan *Google Maps Javascript API*.
  - 13 5. Menentukan pola dari hasil visualisasi data.
  - 14 6. Analisis hasil pengujian dan mengambil kesimpulan.

## 16 **1.6 Sistematika Pembahasan**

- 17 Laporan penelitian tersusun ke dalam enam bab secara sistematis sebagai berikut.
- 18 • Bab 1 Pendahuluan  
Berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penelitian, dan sistematika pembahasan.
  - 19 • Bab 2 Dasar Teori  
Berisi metode penentuan pola, *library Google Maps* dan bahasa pemrograman *Javascript*
  - 20 • 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.
  - 21 • Bab 4 Perancangan Perangkat Lunak  
Berisi perancangan perangkat lunak yang akan dibangun, meliputi perancangan antarmuka, diagram kelas lengkap dan masukan perangkat lunak.
  - 22 • Bab 5 Implementasi dan Pengujian  
Berisi implementasi antarmuka perangkat lunak, pengujian fungsional, pengujian eksperimental serta kesimpulan dari pengujian.
  - 23 • Bab 6 Kesimpulan dan Saran  
Berisi kesimpulan dari awal hingga akhir penelitian dan saran untuk penelitian berikutnya.



1

## BAB 2

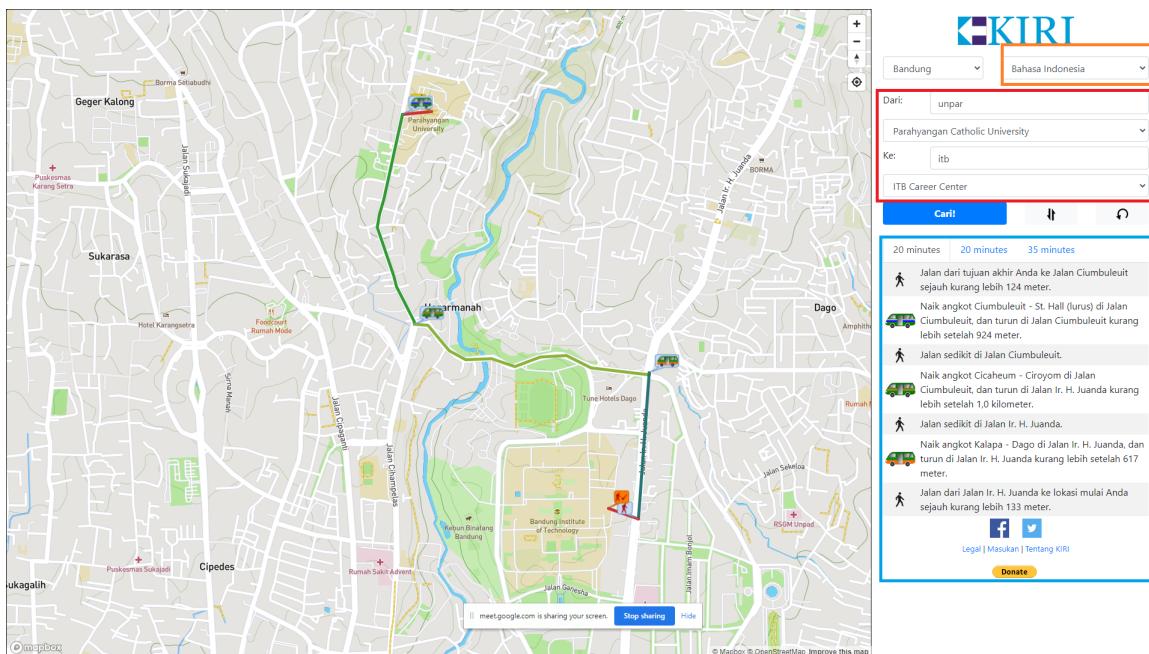
2

### LANDASAN TEORI

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

#### 5 2.1 KIRI Website

- 6 KIRI adalah aplikasi navigasi angkutan umum berbasis web yang melayani Bandung dan kota-kota  
7 lain di Indonesia.[1]. Pada awal pembuatannya KIRI ditujukan untuk penggunaan komersial, namun  
8 karena dinilai kurang sukses (project) KIRI sekarang menjadi (project) *open source*. Aplikasi KIRI  
9 memiliki beberapa fitur seperti:



Gambar 2.1: Fitur-Fitur Pada Aplikasi KIRI

- 10 • Pemilihan rute tercepat menggunakan angkutan kota. Dapat dilihat pada gambar 2.1 pada  
11 kotak berwarna merah.  
12 • Memiliki fitur multi bahasa. Dapat dilihat pada gambar 2.1 pada kotak berwarna oranye.  
13 • Dapat menampilkan instruksi lengkap mencapai tujuan. Dapat dilihat pada gambar 2.1 pada  
14 kotak berwarna biru.

## 2.2 CSV

CSV (*Comma Separated Values*) adalah format penyajian data dengan cara memisahkan setiap *value* dengan simbol titik koma(;) dan menggunakan baris baru sebagai penanda pemisah antar elemen data[2].

### 2.2.1 CSV Format

Tidak ada spesifikasi formal dalam penulisan csv. Format csv yang paling banyak diimplementasikan adalah sebagai berikut:

- Setiap field diletakan pada baris terpisah dan dipisahkan oleh *line break (CRLF)*, contohnya adalah sebagai berikut:

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

---

- *Field* terakhir pada format csv tidak harus menggunakan *line break (CRLF)*, contohnya adalah sebagai berikut:

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

---

- Memungkinkan adanya eksternal *header* yang memiliki aturan penulisan yang sama dengan elemen pada csv. Nilai pada eksternal *header* akan mewakili nama yang tercantum pada *field* jumlah eksternal *header* harus sama dengan jumlah kolom pada normal *record*, contohnya adalah sebagai berikut:

```
1      field_name , field_name , field_name  CRLF
2      aaa , bbb , ccc  CRLF
3      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 (,), contohnya adalah sebagai berikut:

```
1      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*. Contohnya adalah sebagai berikut:

```
1      "aaa" , "bbb" , "ccc"  CRLF
2      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 adalah sebagai berikut:

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

---

## 1 2.3 Node.js

2 *Node.js* merupakan *asynchronous event-driven JavaScript runtime*. Dengan menggunakan *Node.js*  
 3 memungkinkan untuk menjalankan perintah *javascript* tanpa menggunakan *web browser*, *Node.js*  
 4 memungkinkan untuk menjalankan dan melakukan *server-side scripting*[3]. Berikut ini adalah  
 5 contoh sintaks *Node.js*:

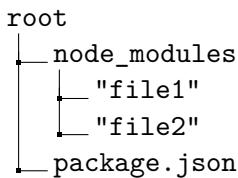
```
6 1 const http = require('http');
7 2
8 3 const hostname = '127.0.0.1';
9 4 const port = 3000;
10 5
11 6 const server = http.createServer((req, res) => {
12 7   res.statusCode = 200;
13 8   res.setHeader('Content-Type', 'text/plain');
14 9   res.end('Hello World');
15 10 });
16 11
17 12 server.listen(port, hostname, () => {
18 13   console.log(`Server running at http://${
19 14 }`);
```

---

### 21 2.3.1 Struktur File Node.js (project)

22 *Node.js* secara otomatis dapat membuat *file-file* dasar. *File-file* tersebut memiliki fungsi tersendiri.  
 23 Folder *node\_modules* berfungsi sebagai penyimpanan semua *library* yang diinstall melalui npm.  
 24 Semua *library* yang diinstall melalui npm akan dicatat pada file *package.json*, Hal ini bertujuan  
 25 untuk mempermudah proses *maintenance library* pada (project) *node.js*.

26 Struktur dari *file-file* akan berbentuk seperti:



### 27 2.3.2 Node Package Manager

28 *Node Package Manager* (NPM) adalah *software registry* yang dimiliki oleh *Node.js*. NPM memung-  
 29 kinkan pengguna untuk mempublikasi atau menggunakan *software library*[4]. NPM terdiri dari tiga  
 30 komponen penting yaitu:

- 31 • NPM website.
- 32 • NPM CLI (*Command Line Interface*).
- 33 • NPM *Registry*.

34 NPM memiliki beberapa kegunaan antara lain:

- 35 • Mendownload *Software Library*.
- 36 • Menjalankan *packages* tanpa harus mendownload *npx*.
- 37 • Mempublikasikan *Tools* atau *Software Library*.

### 2.3.3 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:

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

NPM CLI memiliki perintah-perintah yang dapat digunakan untuk membantu melakukan *maintenance* terhadap *library* pada (*project*) *node.js*. Berikut perintah-perintah pada NPM CLI:

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

```
10  1       npm init [--force|-f|--yes|-y|--scope]
11  2       npm init <@scope> (same as `npx <@scope>/create`)
12  3       npm init [<@scope>/]<name> (same as `npx [<@scope>/] create-<name>`)
```

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

```
15  1       npm install (with no args, in package dir)
16  2       npm install [<@scope>/]<name>
17  3       npm install [<@scope>/]<name>@<tag>
18  4       npm install [<@scope>/]<name>@<version>
19  5       npm install [<@scope>/]<name>@<version range>
20  6       npm install <alias>@npm:<name>
21  7       npm install <git-host>:<git-user>/<repo-name>
22  8       npm install <git repo url>
23  9       npm install <tarball file>
24 10      npm install <tarball url>
25 11      npm install <folder>
```

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

```
28  1       npm uninstall [<@scope>/]<pkg>[@<version>]
```

## 2.4 Express.js

*Express.js* merupakan *web application framework* untuk *node.js*. *Express.js* menyediakan *robust feature* dalam pembuatan perangkat lunak berbentuk situs web maupun perangkat begerak[5].

### 2.4.1 Instalasi

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

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

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

### 2.4.2 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[5]. Untuk dapat menggunakan *express generator* pengembang harus menambahkan *library express generator* melalui npm dengan menggunakan perintah sebagai berikut:

```
7 npx express-generator
```

8 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
```

### 2.4.3 Routing

10 *Routing* adalah proses untuk menentukan cara perangkat lunak merespon *input* melalui beberapa  
 11 *endpoint*. Dalam pembuatan *routing* pada *express.js* terdapat empat komponen penting yaitu:  
 12 • Instansi dari *Express.js* (*app*).  
 13 • *Http method*.  
 14 • *Path*  
 15 • Perintah yang akan dijalankan jika *route* dijalankan *Handler*.

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

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

18 Berikut ini beberapa contoh perintah pembuatan *route* pada *Express.js*

```
19 1     app.get('/', function (req, res) {
20 2         res.send('Hello World!')
21 3     })
22 4
23 5     app.post('/', function (req, res) {
24 6         res.send('Got a POST request')
25 7     })
26 8
27 9     app.put('/user', function (req, res) {
28 10        res.send('Got a PUT request at /user')
```

---

```

1 11      })
2 12
3 13     app.delete('/user', function (req, res) {
4 14       res.send('Got a DELETE request at /user')
5 15   })

```

---

#### 7 2.4.4 Menampilkan File Statis

8 Untuk dapat menampilkan *file* yang bersifat statis seperti foto, *javascript*, dan *css*, *Express.js* telah  
9 menyediakan *object* sebagai berikut:

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

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

```
13 1 app.use('/static', express.static('public'))
```

---

15 Parameter '*/static*' berutujuan untuk membuat *virtual path* yang memiliki *prefix* '*/static*', sedangkan  
16 parameter '*public*' menandakan bahwa *file* statis berada didalam *folder public*. Perintah diatas  
17 ketika dijalankan maka *Express.js* akan secara otomatis membuat *route* seperti berikut yang dapat  
18 digunakan untuk mengakses *file* statis.

```

19 1 http://localhost:3000/static/images/kitten.jpg
20 2 http://localhost:3000/static/css/style.css
21 3 http://localhost:3000/static/js/app.js
22 4 http://localhost:3000/static/images/bg.png
23 5 http://localhost:3000/static/hello.html

```

---

### 25 2.5 Google Maps Javascript API

26 Pada subbab ini akan menjelaskan tentang *Google Maps Javascript API* beserta kelas-kelas yang  
27 dimilikinya. *Google Maps* adalah layanan pemetaan web yang dikembangkan oleh *Google*. *Google*  
28 *Maps* menawarkan citra satelit, foto udara, dan peta jalan yang interaktif, kondisi lalu lintas  
29 secara *real time*. Dalam pengembangannya, *Google Maps* memiliki akses pendukung untuk bahasa  
30 pemrograman *Javascript*. Berikut ini beberapa layanan yang telah disediakan oleh *Google Maps*  
31 *javascript API*[6]:

- 32 • *Maps*.
- 33 • *Drawing Object*.
- 34 • *Street Views*.
- 35 • *Routes*

#### 36 2.5.1 Map

37 *Map* adalah sebuah *object* pada *Google Maps Javascript API* yang digunakan untuk membuat  
38 *object map* dalam elemen *html*. Untuk dapat menginisialisasi *object map*, pengembang harus dapat  
39 menggunakan *constructor* yang memiliki perintah sebagai berikut:

---

```

1   1     Map(mapDiv[, opts])
2   2     Parameters:
3   3       mapDiv: Element
4   4       opts: MapOptions optional

```

---

6 Pada *constructor* diatas terdapat dua parameter:

- 7 • mapDiv.
- 8 • MapOptions

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

---

```

12  1     function initMap() {
13  2         let mapOptions = {
14  3             center: {lat: -6.914744, lng: 107.609810},
15  4             zoom: 12,
16  5         }
17  6         let map = new google.maps.Map(document.getElementById("map"), mapOptions);
18  7         return map;
19  8     }

```

---

21 Ketika fungsi *initMap()* dijalankan maka *Google Maps Javascript API* akan membuat *object map*  
22 didalam *html dom* yang memiliki id='map' dan akan mengatur *property* yang dimiliki sesuai dengan  
23 *object mapOptions*.

24 *Object map* telah menyediakan fungsi-fungsi yang bisa langsung digunakan oleh pengembang,  
25 beberapa fungsi yang disediakan oleh *object map* adalah:

- 26 • Fungsi *fitBounds()* berguna untuk menentukan *view port* dari *object map* sesuai dengan *bounds*  
27 yang diberikan. Fungsi ini memiliki struktur perintah seperti berikut:

---

```

28  1     fitBounds(bounds[, padding])
29  2     Parameters:
30  3       bounds: LatLngBounds|LatLngBoundsLiteral
31  4       padding: number|Padding optional
32  5     Return Value: None

```

---

- 34 • Fungsi *getBounds()* berguna untuk mendapatkan *view port* dari *object map*. Fungsi ini memiliki  
35 struktur perintah seperti berikut:

---

```

36  1     getBounds()
37  2     Parameters: None
38  3     Return Value: LatLngBounds

```

---

- 40 • Fungsi *getCenter()* berguna untuk mendapatkan posisi yang ditunjukan pada *object map*  
41 posisi yang didapatkan akan berbentuk *longitude* dan *latitude*. Fungsi ini memiliki struktur  
42 perintah seperti berikut:

---

```

43  1     getCenter()
44  2     Parameters: None
45  3     Return Value: LatLng

```

---

- 47 • Fungsi *getClickableIcons()* berguna untuk mendapatkan *clickable icon*. Setiap *clickable icon*  
48 merupakan *point of interest* pada *map*. Fungsi ini memiliki struktur perintah seperti berikut:

---

```

1      1      getClickableIcons()
2      2      Parameters: None
3      3      Return Value: boolean

```

---

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

---

```

7      1      getMapTypeId()
8      2      Parameters: None
9      3      Return Value: MapTypeId|string

```

---

*Google Maps Javascript API* menyediakan variabel konstanta / *constant* yang berfungsi untuk menentukan jenis peta yang akan digunakan pada *object map*. Konstanta 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()* beguna untuk membuat atau mengubah *mapTypeId*. Fungsi ini memiliki struktur perintah seperti berikut:

---

```

22     1      setMapTypeId(mapTypeId)
23     2      Parameters:
24     3      mapTypeId: MapTypeId|string

```

---

- Fungsi *setZoom()* berguna untuk mengubah *zoom value* yang dimiliki oleh *object map*.

---

```

27     1      setZoom(zoom)
28     2      Parameters:
29     3      zoom: number

```

---

- Fungsi *setMapOption()* berguna untuk mengubah *property mapOption* yang dimiliki oleh *object map*.

---

```

33     1      setOptions(options)
34     2      Parameters:
35     3      options: MapOptions

```

---

### 37 2.5.2 Sistem Kordinat Google Maps

38 *Google Maps Javascript API* menggunakan kelas *LatLng* untuk merepresentasikan koordinat secara 39 geografis pada *object map*. Kelas *LatLng* merupakan sebuah kelas yang merepresentasikan *latitude* 40 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.

1 Untuk dapat menginisialisasi kelas *LatLng* pada *Google Maps Javascript API* pengembang perlu  
2 membuat *constructor* yang memiliki struktur seperti berikut:

```
3 1     LatLng(lat, lng[, noWrap])
4 2     Parameters:
5 3     lat:  number
6 4     lng:  number
7 5     noWrap: boolean optional
```

---

9 *Object LatLng* dibuat untuk yang mewakili titik geografis. *latitude* ditentukan dalam derajat dalam  
10 rentang [-90, 90]. *longitude* ditentukan dalam derajat dalam rentang [-180, 180]. Parameter *noWrap*  
11 dibuat *true* untuk mengaktifkan nilai di luar rentang ini. Contoh penggunaan kelas *LatLng* adalah  
12 sebagai berikut:

```
13 1         map.setCenter(new google.maps.LatLng(-34, 151));
14 2         map.setCenter({lat: -34, lng: 151});
```

---

16 *Google Maps Javascript API* telah menyediakan fungsi bawaan yang dapat diakses ketika menggu-  
17 nakan kelas *LatLng*. Fungsi tersebut adalah sebagai berikut:

- 18 • Fungsi *equals()* bertujuan untuk membandingkan posisi antara *object map*.

```
19 1         equals(other)
20 2         Parameters:
21 3         other:  LatLng
22 4         Return Value: boolean
```

---

- 24 • Fungsi *lat()* bertujuan untuk mendapatkan posisi *latitude* dari *object map*.

```
25 1         lat()
26 2         Parameters: None
27 3         Return Value: number
28 4         Returns the latitude in degrees.
```

---

- 30 • Fungsi *lng()* bertujuan untuk mendapatkan posisi *longitude* dari *object map*.

```
31 1         lng()
32 2         Parameters: None
33 3         Return Value: number
34 4         Returns the longitude in degrees.
```

---

- 36 • Fungsi *toJSON()* akan mengembalikan format *json* terhadap posisi latlng pada *object map*
- 37 • Fungsi *toUrlValue()* akan mengembalikan string dalam bentuk "lat, lng" untuk *LatLang* ini.

## 1 2.6 Marker



Gambar 2.2: Add Marker

- 2 *Marker* adalah sebuah kelas yang dapat memunculkan *mark* / tanda pada *object map*. Untuk dapat  
3 menginisialisasi kelas *marker*, pengembang dapat menggunakan *constructor* yang memiliki struktur  
4 seperti berikut:

```
5 1     Marker([opts])
6 2     Parameters:
7 3     opts: MarkerOptions optional
```

---

- 9 Pada *constructor marker* terdapat parameter *MarkerOptions* yang bersifat optional. *MarkerOptions*  
10 merupakan sebuah *object* yang dapat digunakan pada *object mark*. *MarkerOptions* memiliki properti  
11 yang dapat digunakan seperti:

Tabel 2.1: Tabel Properti Pada Objek Marker

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

<sup>1</sup> Contoh inisialisasi *marker* pada *object maps* adalah sebagai berikut:

```

1   return new google.maps.Marker({
2       position: location,
3       label: labels[i % labels.length],
4   });

```

<sup>7</sup> Kelas *marker* memiliki fungsi bawaan yang telah disediakan oleh antara lain seperti:

Tabel 2.2: Tabel Fungsi Pada Kelas Marker

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

## 1 2.7 Marker Clusterer

- 2 *MarkerClustererPlus* merupakan sebuah *library* tambahan untuk dapat mengelompokan *object*  
 3 *marker* pada 2.6. Visualisasi dari *MarkerClustererPlus* dapat dilihat pada gambar 2.3.



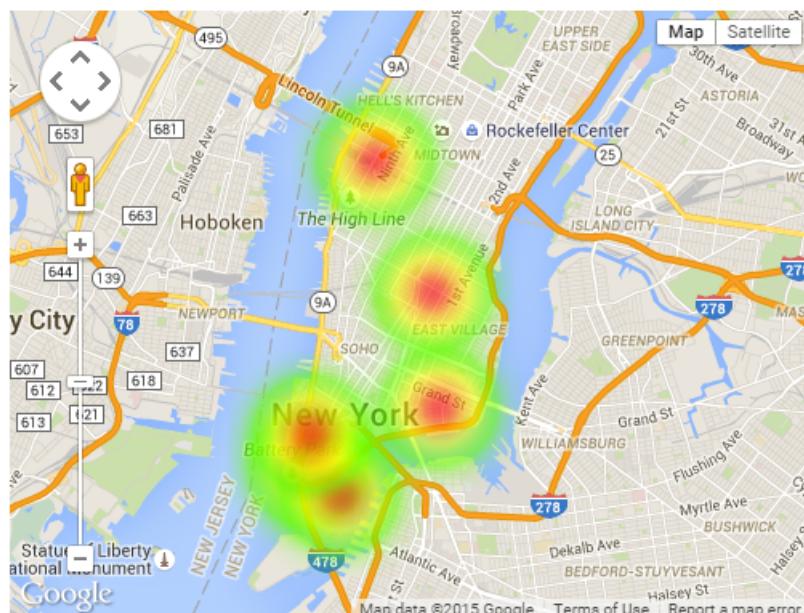
Gambar 2.3: Contoh Marker Clustering

- 4 Untuk dapat menginisialisasi kelas tersebut, pihak pengembang perlu menuliskan perintah  
 5 sebagai berikut:

```
6 1 var markerCluster = new MarkerClusterer(map, markers,
7 2   {imagePath: `${path}/m`});
```

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

## 11 2.8 HeatMap



Gambar 2.4: Contoh HeatMap

1 *Google Maps Javascript API* telah menyediakan kelas *heatmap* untuk menampilkan *heatmap* pada  
2 *object maps* 2.4. Untuk dapat menginisialisasi kelas ini, pengembang perlu memanggil perintah  
3 *constructor* yang memiliki struktur seperti berikut:

```
4   1     HeatmapLayer([opts])
5   2       Parameters:
6   3         opts: HeatmapLayerOptions optional
```

---

8 .  
9 Kelas *HeatMap* dilengkapi dengan parameter *HeatmapLayerOptions* yang bersifat opsional, *object*  
10 ini bertujuan untuk dapat mengatur *property* dari kelas *HeatMap*. Parameter *HeatMapLayerOptions*  
11 merupakan sebuah *object* yang memiliki atribut sebagai berikut:

- 12 • Titik data yang diperlukan.
- 13 • *Dissipate variable* yang menentukan apakah *heatmap* akan menghilang jika *maps* diperbesar  
atau diperkecil.
- 14 • Gradien dari warna *heatmap*
- 15 • *Map attribute* untuk menunjukkan peta dimana *heatmap* akan ditampilkan.
- 16 • *MaxIntensity* yang merupakan nilai maximal dari intensitas warna pada *heatmap*.
- 17 • *Opacity* yang merupakan nilai *opacity* dari *heatmap*.
- 18 • Radius yang merupakan nilai radius dari *heatmap*.

20 Contoh inisialisasi kelas *heatmap* adalah sebagai berikut:

```
21 1   let heatmap = new google.maps.visualization.HeatmapLayer({
22 2     data: heatmapData
23 3   }) ;
```

---

25 Kelas *HeatMap* memiliki fungsi bawaan yang telah disediakan oleh *Google Maps* beberapa fungsi  
26 tersebut adalah:

- 27 • *getData()* fungsi ini akan mengembalikan data point pada *object heatmap*.
- 28 • *getMap()* fungsi ini akan mengembalikan *object map*.
- 29 • *setData(data)* fungsi ini akan memasukan data pada *object heatmap*.
- 30 • *setMap(map)* fungsi ini akan memasukan *object map* pada *object heatmap*.
- 31 • *setOption(option)* fungsi ini akan memasukan *object HeatMapLayerOptions* pada *object*  
*heatmap*.



1

## BAB 3

2

### ANALISIS

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

6 **3.1 Analisis Data Histori KIRI**

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

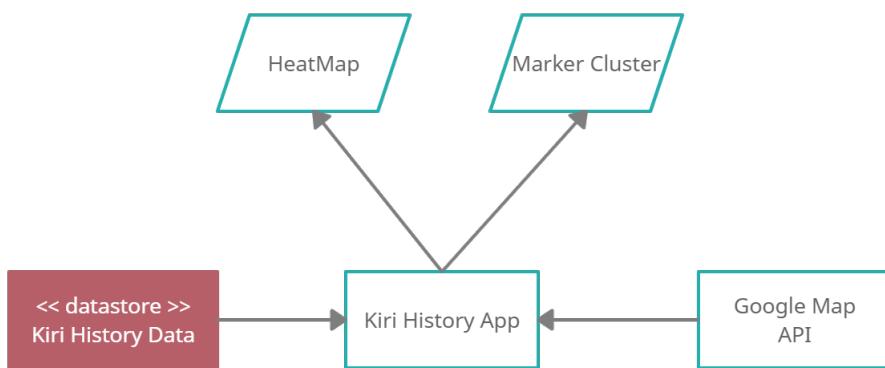
- 10 • LogId sebagai penanda satu record didalam data histori KIRI.
- 11 • APIKey sebagai atribut *API Key* yang digunakan ketika melakukan perintah pada perangkat  
12 lunak KIRI.
- 13 • Timestamp (UTC) sebagai atribut untuk mencatat waktu perintah dilakukan format berbentuk  
14 *timestamp*.
- 15 • Action yang dilakukan user pada saat menggunakan perangkat lunak KIRI memiliki empat  
16 nilai *action* pada data histori KIRI yaitu:
  - 17 – *PAGELOAD*.
  - 18 – *SEARCHPLACE*.
  - 19 – *WIDGETLOAD*.
  - 20 – *FINDROUTE*.
- 21 • AdditionalData yang digunakan untuk mencatat informasi tambahan berdasarkan *action* yang  
22 dipilih. Nilai pada additionalData akan bergantung pada *action* yang dipilih:
  - 23 – Jika *action* bernilai *PAGELOAD* maka additionalData akan bernilai ip dari pengakses.
  - 24 – Jika *action* bernilai *SEARCHPLACE* maka additionalData akan bernilai *keyword* yang  
25 dituliskan oleh pengakses.
  - 26 – Jika *action* bernilai *FINDROUTE* maka additionalData akan bernilai posisi tempat dan  
27 tujuan dalam bentuk *latitude* dan *longitude* yang dicari oleh pengakses.
  - 28 – Jika *action* bernilai *WIDGETLOAD* maka additionalData akan bernilai alamat url dari  
29 penyedia *widget*.

## 3.2 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 node.js* 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. Gambaran tentang alur komunikasi perangkat lunak dapat dilihat pada gambar 3.1.



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 ke dalam 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*.

## 3.3 Analisis Perangkat Lunak

Pada subbab ini akan dilakukan analisis untuk aplikasi visualisasi data histori KIRI.

### 3.3.1 Analisis Kebutuhan Perangkat Lunak

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

1     • Data histori KIRI

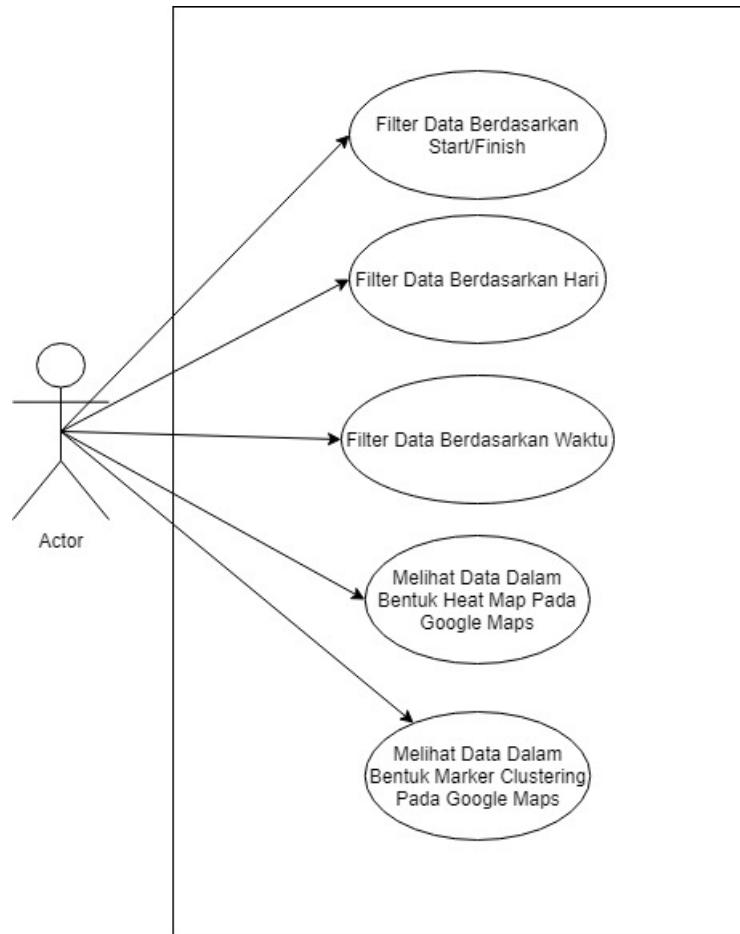
2       Aplikasi dapat melakukan proses filter pada data histori dan menggunakan data histori KIRI  
3       untuk dapat divisualisasikan oleh karena itu diperlukan raw data histori KIRI.

4     • Google Maps Javascript API

5       Aplikasi dapat melakukan visualisasi data menggunakan *heat map* dan *marker clustering* yang  
6       merupakan fitur dari *Google Maps Javascript API*.

7     **3.3.2 Use Case Diagram**

8     Interaksi antara pengguna dengan perangkat lunak yang dibangun pada skripsi ini digambarkan  
9     pada diagram *use case* pada Gambar 3.2.



Gambar 3.2: Use Case Diagram

10    **3.3.3 Use Case Skenario**

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

Tabel 3.1: Tabel Skenario Memfilter Data Berdasarkan Start/Finish

Nama	Memfilter Data Berdasarkan Start/Finish.
Deskripsi	Melakukan filter data berdasarkan kategori dari data histori kiri
Aktor	Pengguna
Pre-kondisi	Aplikasi sudah dijalankan dan sudah dapat mengolah raw data histori KIRI.
Alur Skenario Utama	<ol style="list-style-type: none"> <li>1. Sistem memuat aplikasi.</li> <li>2. Sistem menampilkan data.</li> <li>3. Pengguna dapat memfilter data berdasarkan <i>start / finish</i>.</li> <li>4. Sistem melakukan proses filtering berdasarkan atribut yang telah dipilih pengguna.</li> </ol>

Tabel 3.2: Tabel Skenario Memfilter Data Berdasarkan Hari

Nama	Memfilter Data Berdasarkan Hari.
Deskripsi	Melakukan filter data berdasarkan kategori dari data histori kiri
Aktor	Pengguna
Pre-kondisi	Aplikasi sudah dijalankan dan sudah dapat mengolah raw data histori KIRI.
Alur Skenario Utama	<ol style="list-style-type: none"> <li>1. Sistem memuat aplikasi.</li> <li>2. Sistem menampilkan data.</li> <li>3. Pengguna dapat memfilter data berdasarkan hari.</li> <li>4. Sistem melakukan proses filtering berdasarkan atribut yang telah dipilih pengguna.</li> </ol>

Tabel 3.3: Tabel Skenario Memfilter Data Berdasarkan Jam

Nama	Memfilter Data Berdasarkan Jam.
Deskripsi	Melakukan filter data berdasarkan kategori dari data histori kiri
Aktor	Pengguna
Pre-kondisi	Aplikasi sudah dijalankan dan sudah dapat mengolah raw data histori KIRI.
Alur Skenario Utama	<ol style="list-style-type: none"> <li>1. Sistem memuat aplikasi.</li> <li>2. Sistem menampilkan data.</li> <li>3. Pengguna dapat memfilter data berdasarkan jam.</li> <li>4. Sistem melakukan proses filtering berdasarkan atribut yang telah dipilih pengguna.</li> </ol>

Tabel 3.4: Tabel Skenario Melihat Visualisasi Data Dalam Bentuk Marker Cluster

Nama	Melihat Visualisasi Data Dalam Bentuk Marker Cluster.
Deskripsi	Melihat hasil visualisasi data pada google map
Aktor	Pengguna
Pre-kondisi	Aplikasi sudah dijalankan dan sudah dapat menampilkan data histori KIRI.
Alur Skenario Utama	<ol style="list-style-type: none"> <li>1. Sistem menampilkan data yang telah difilter oleh pengguna.</li> <li>2. Pengguna dapat memilih akan menggunakan metode visualisasi <i>Marker Cluster</i></li> <li>3. Sistem menampilkan hasil visualisasi data berdasarkan metode yang telah dipilih pengguna.</li> </ol>

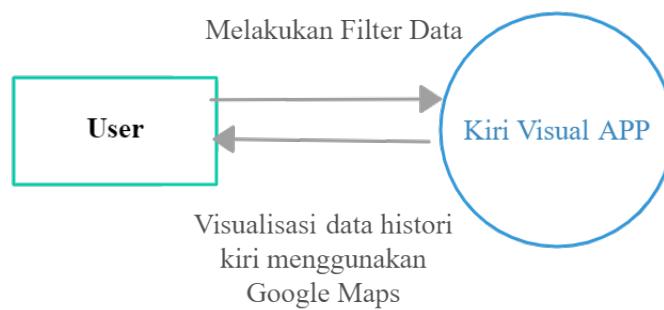
Tabel 3.5: Tabel Skenario Melihat Visualisasi Data Dalam Bentuk Marker Clustering

Nama	Melihat Visualisasi Data Dalam Bentuk Heat Map.
Deskripsi	Melihat hasil visualisasi data pada google map
Aktor	Pengguna
Pre-kondisi	Aplikasi sudah dijalankan dan sudah dapat menampilkan data histori KIRI.
Alur Skenario Utama	<ol style="list-style-type: none"> <li>1. Sistem menampilkan data yang telah difilter oleh pengguna.</li> <li>2. Pengguna dapat memilih akan menggunakan metode visualisasi <i>Marker Clustering</i></li> <li>3. Sistem menampilkan hasil visualisasi data berdasarkan metode yang telah dipilih pengguna.</li> </ol>

#### <sup>1</sup> 3.3.4 Data Flow Diagram

- <sup>2</sup> Interaksi data dalam perangkat lunak yang dibangun pada skripsi ini digambarkan pada Gambar <sup>3</sup> *data flow diagram* 3.3.

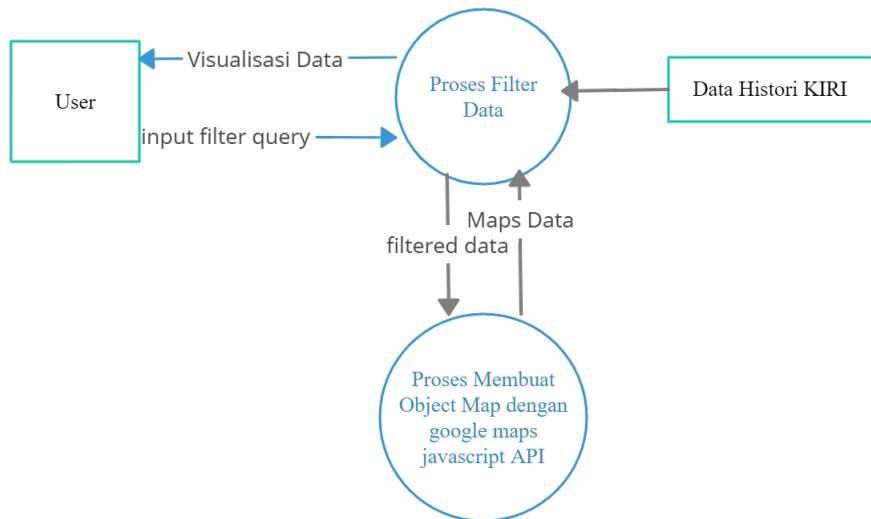
LEVEL 0:



Gambar 3.3: Data Flow Diagram Level 0

- 1 Pada level 0 ini dapat dilihat pada gambar 3.3, perangkat lunak yang akan dibuat memiliki dua buah flow utama yaitu user dapat melakukan penyaringan data, dan user dapat menerima hasil penyaringan data dalam bentuk visualisasi pada *google maps*. Setelah dilakukan pengujian lebih lanjut maka didapatkan Gambar *data flow diagram level 1* 3.4.

LEVEL 1:



Gambar 3.4: Data Flow Diagram Level 1

- 5 Pada level 1 ini dapat dilihat pada Gambar 3.4, perangkat lunak akan memiliki dua buah proses utama yaitu:
  - 7 • Filter data
  - 8 Proses untuk melakukan penyaringan data histori kiri berdasarkan *query input* dari user.
  - 9 • Proses pembuatan object *maps*

1      Proses yang akan menerima data histori KIRI dan membuat objek map menggunakan *Google*  
2      *Maps Javascript API*.

3      Berdasarkan *data flow diagram* pada gambar 3.4, pada perangkat lunak yang akan dibuat user  
4      dapat melakukan tiga buah aksi, yaitu:

5      1. Filter data histori KIRI

6      User dapat memasukan filter parameter untuk melakukan proses penyaringan data histori  
7      KIRI, dan perangkat lunak akan melakukan proses penyaringan berdasarkan filter parameter  
8      yang dipilih user.

9      2. Select mode visualisasi

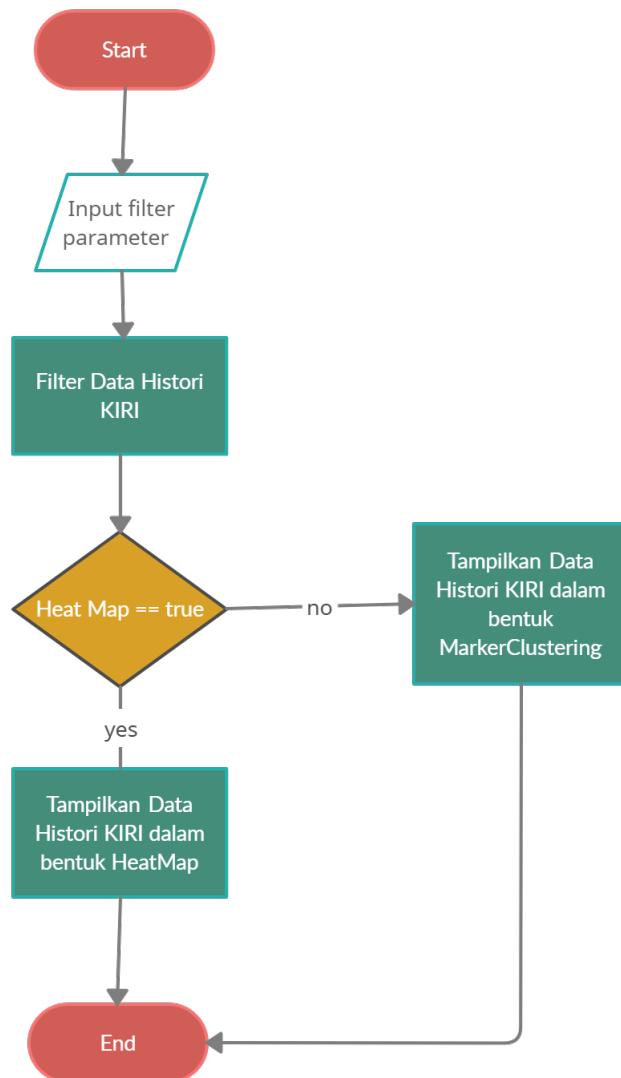
10     User dapat memilih mode visualisasi pada perangkat lunak ini, akan disediakan dua mode  
11     visualisasi *heatmap* dan *marker clustering*.

12     3. Visualisasi data dengan google maps pada perangkat ini

13     User akan menerima output berupa data yang telah tervisualisasi dengan google map.

14     **3.3.5 Flow Chart Diagram**

15     Alur perangkat lunak yang akan dibuat dapat digambarkan kedalam bentuk *flow chart* 3.5



Gambar 3.5: KIRI Flow Chart

### **3.3.6 Input dan Output**

- 2 Input dan output yang akan diterima oleh perangkat lunak yang akan dibuat adalah sebagai berikut.
  - Input Filter Parameter
 

Filter Parameter adalah semua parameter yang dapat dipilih user untuk dapat melakukan penyaringan data histori KIRI, pada perangkat lunak yang akan dibangun user dapat memilih tiga filter parameter.
  - Input mode visualisasi
 

Pada perangkat lunak yang akan dibangun user dapat memilih dua mode visualisasi yaitu *mode heatmap* dan *mode markerclustering*.
  - Output visualisasi data
 

Output pada perangkat lunak yang akan dibangun adalah hasil visualisasi data histori KIRI dalam bentuk *marker clustering* atau bentuk *heat map*.

1

## BAB 4

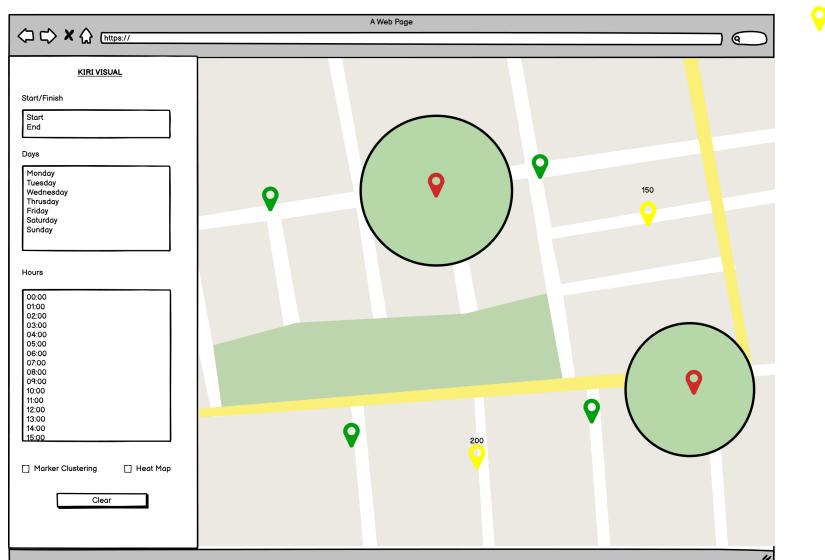
2

### PERANCANGAN

3 Bab ini akan menjelaskan perancangan aplikasi visualisasi data histori KIRI pada Google Maps.

#### 4 4.1 Perancangan Antarmuka

5 Pada aplikasi yang dibuat, disediakan antarmuka untuk memudahkan pengguna dalam berinteraksi  
6 dengan perangkat lunak. Berikut adalah Gambar 4.1 yang merupakan antarmuka dari perangkat  
7 lunak yang dibuat.

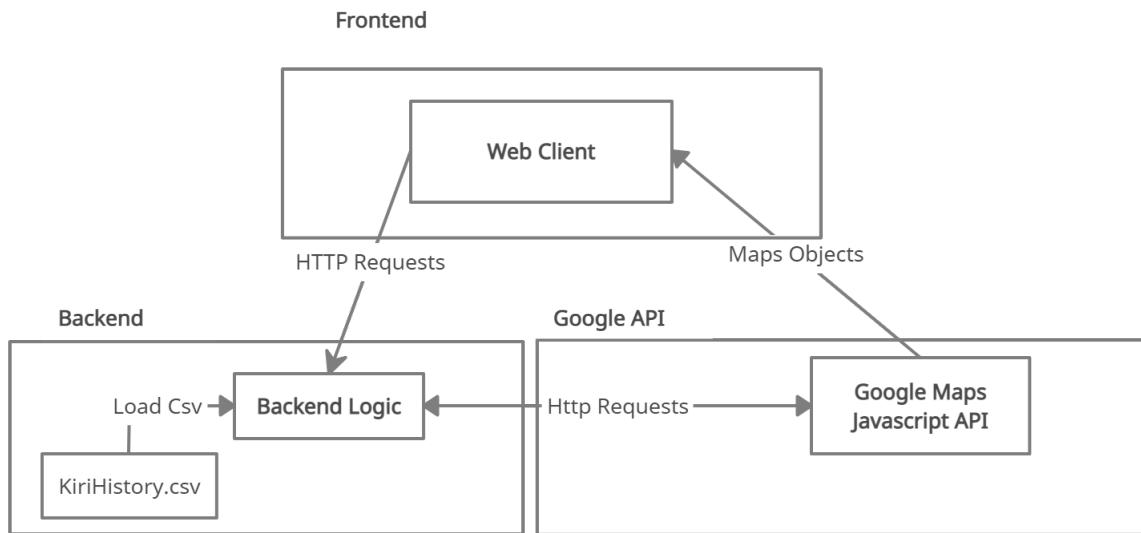


Gambar 4.1: Rancangan Antarmuka

8 Berdasarkan rancangan diatas, berikut adalah fungsi dari setiap komponen dalam antarmuka.  
9 • *Map*: digunakan untuk menampilkan peta *google map*.  
10 • *Checkbox Start*: untuk memfilter data berdasarkan tempat keberangkatan.  
11 • *Checkbox End*: untuk memilfter data berdasarkan tempat tujuan.  
12 • *Selection Box Hours*: untuk memfilter data berdasarkan jam.  
13 • *Selection Box Days*: untuk memfilter data berdasarkan hari.  
14 • *Checkbox Heat Map*: untuk menampilkan data dalam bentuk *heat map*.  
15 • *Checkbox Marker Clustering*: untuk menampilkan data dalam bentuk *marker clustering*.  
16 • Button *Clear*: menghapus seluruh *overlay* pada objek *map*.

## 1 4.2 Perancangan Diagram Fungsi

- 2 Berdasarkan hasil analisis pada 3.3.1, maka untuk aplikasi ini dapat dibuat diagram fungsi seperti  
 3 Gambar 4.2.



Gambar 4.2: Rancangan Diagram Fungsi

- 4 Aplikasi ini merupakan aplikasi berbasis website dimana akan memanfaatkan *Node.js* pada 2.3  
 5 yang akan berperan sebagai *asynchronous event-driven*, sehingga backend logic dapat menggunakan  
 6 *javascript* tanpa menggunakan *web browser*. Pada aplikasi ini terdiri dari tiga bagian yaitu:

- *Frontend*

8 Pada bagian *Frontend* terdiri dari *web client* yang berfungsi untuk menerima input dan  
 9 mengeluarkan output dari *user*. Webclient akan berkomunikasi dengan *backend* dengan  
 10 menggunakan *http request*.

- *Backend*

12 Pada bagian *Backend* akan menggunakan *node.js* akan menerima input dari web client pada  
 13 bagian ini juga akan dilakukan *load* data histori kiri yang berbentuk csv untuk dapat diolah.

- *Google API*

15 Aplikasi ini akan memanfaat *Google Maps Javascript API* untuk dapat memvisualisasikan  
 16 data history KIRI.

## 17 4.3 Perancangan Diagram Kapabilitas

- 18 Berdasarkan kapabilitas dari masing masing komponen pada 4.2, maka dapat dibuat diagram  
 19 kapabilitas.

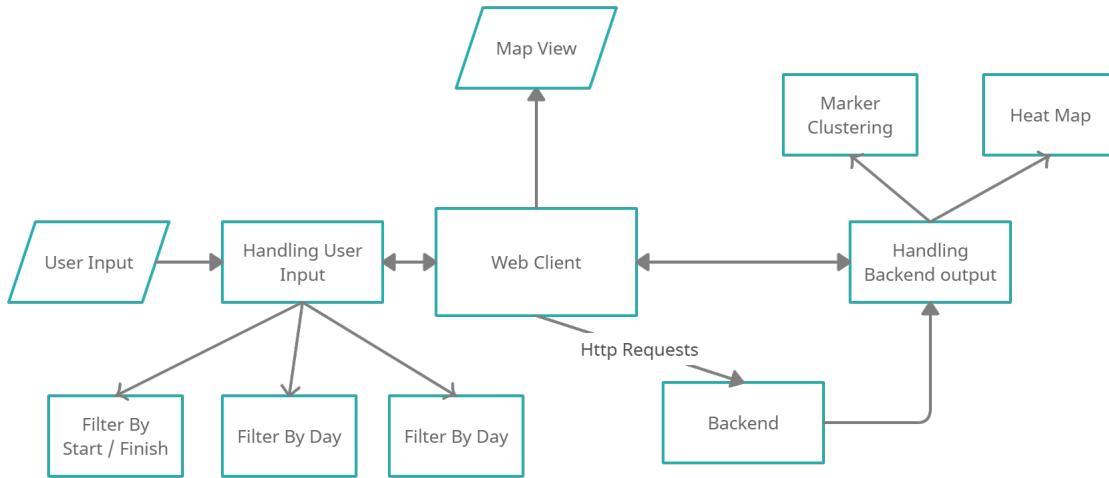
### 20 4.3.1 Diagram Kapabilitas Web Client

- 21 Pada bagian *web client* akan memiliki kapabilitas sebagai berikut:

- Menampilkan *Google Maps*.
- Menampilkan *heatmap* atau *marker clustering object*.
- Menerima input dari user  
Input dari user berupa parameter yang berguna untuk memfilter data histori kiri dan menampilkannya dalam bentuk *heatmap* atau *marker clustering*.

Berdasarkan kapabilitas pada 4.3.1 maka dapat dibuat diagram yang berbentuk seperti Gambar

4.3



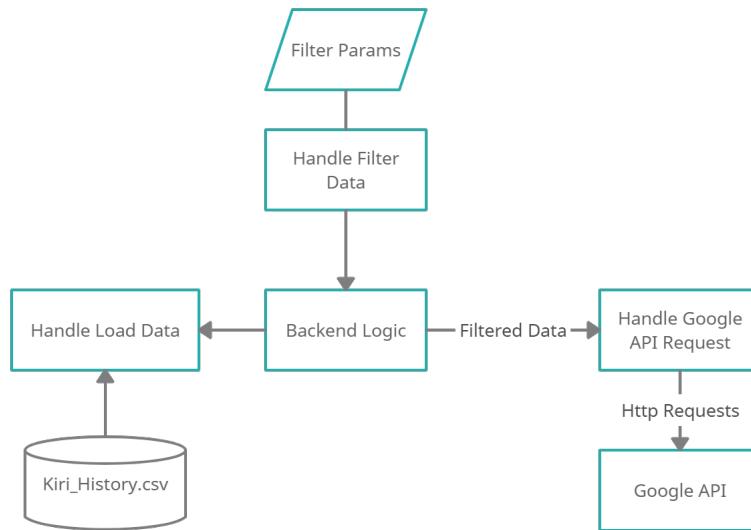
Gambar 4.3: Diagram Kapabilitas Web Client

### 4.3.2 Diagram Kapabilitas Backend

Pada aplikasi yang akan dibangun, *backend* memiliki kapabilitas sebagai berikut:

- Menerima *input* dari *web client*. Input yang diterima berupa filter parameter yang telah diolah oleh *web client*.
- Melakukan filterisasi data berdasarkan *input parameter* dari *web client*.
- Melakukan komunikasi *via http request* terhadap *Google API*.

Berdasarkan kapabilitas 4.3.2 maka dapat dibuat diagram yang berbentuk seperti Gambar 4.4.



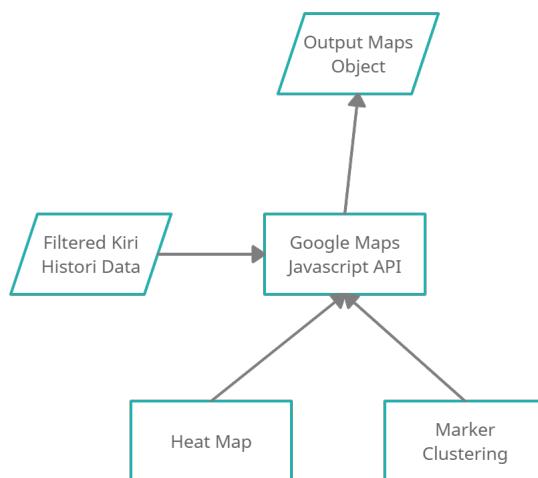
Gambar 4.4: Diagram Kapabilitas Backend

#### **4.3.3 Diagram Kapabilitas Google API**

Pada aplikasi ini akan menggunakan *Google Maps Javascript API* sebagai *third party library* dalam menampilkan hasil visualisasi data yang dilakukan oleh *backend*. *Google Maps Javascript API* pada perangkat lunak ini memiliki kapabilitas sebagai berikut:

- Menerima *input* dari *Backend*. Input yang diterima berupa data histori KIRI yang telah disaring sesuai dengan *input* dari user.
- Mengeluarkan *output* berupa *object map* yang akan dirender oleh *web client*. *Object map* dapat berupa *marker cluster object* atau *heatmap object* tergantung pada input yang diterima dari *backend*.

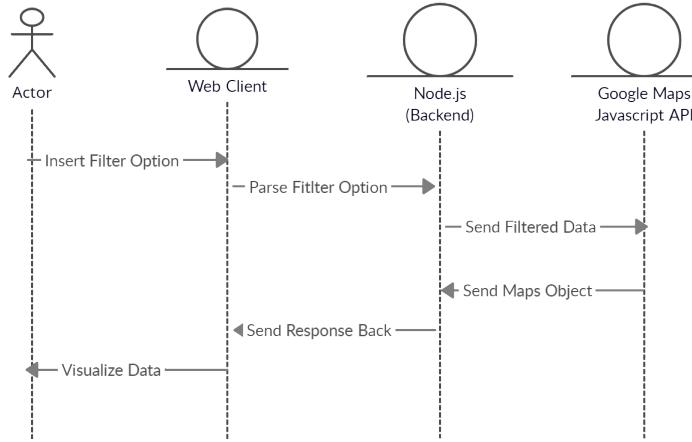
Berdasarkan kapabilitas pada 4.3.3, maka dapat dibuat diagram yang berbentuk seperti Gambar 4.5.



Gambar 4.5: Diagram Kapabilitas Goolge API

## 1 4.4 Perancangan Diagram Sequence

- 2 Pada subbab ini akan dijelaskan interaksi antar komponen yang sesuai dengan yang dijelaskan pada  
 3 subbab 3.2, dengan diagram interaksi untuk setiap fitur pada aplikasi visualisasi data histori KIRI.



Gambar 4.6: Diagram Sequence Visualisasi Data Histori KIRI

- 4 Fitur pada perangkat lunak ini akan diawali dengan user dapat memilih *input filter*. *Input*  
 5 *filter* yang dapat dipilih adalah filter berdasarkan atribut *start* atau *finish*, *days*, dan *time*. Setelah  
 6 filter *options* dipilih oleh user maka *web client* akan mengubah hasil *options* tersebut menjadi  
 7 *filter params*. *Filter params* adalah sebuah objek yang akan digunakan oleh *backend logic* untuk  
 8 melakukan proses *filter* terhadap data histori KIRI. Setelah *backend logic* menerima *filter params*,  
 9 maka proses *filter* akan dilakukan sesuai dengan isi dari *filter params*. Aksi ini akan menghasilkan  
 10 data histori KIRI yang sudah disaring sesuai dengan isi dari *filter params*. *Backend logic* akan  
 11 mengirimkan hasil data yang telah disaring tersebut ke *Google Maps Javascript API* dengan lalu  
 12 *Google Maps Javascript API* akan mengeluarkan maps object berdasarkan data yang dikirim oleh  
 13 *backend logic*. Data tersebut akan ditampilkan kembali oleh *web client* dalam bentuk data visualisasi  
 14 menggunakan *Google Maps*.

## 15 4.5 Perancangan Pseudocode

- 16 Pada subbab ini dirancang *pseudocode* untuk membuat aplikasi visualisasi data histori KIRI.  
 17 *Pseudocode* yang dibuat akan mengacu pada kapabilitas setiap komponen pada 4.3.

### 18 4.5.1 Perancangan Pseudocode Pada Web Client

- 19 Berdasarkan kapabilitasnya pada 4.3.1, pseudocode dapat dibuat berdasarkan dua fungsi utama  
 20 yaitu:
- 21 • Handling User Input. Pada tahap ini *web client* akan menerima input dari user dan akan  
 22 mengubah nya menjadi *filter params*.
- 23 • Menampilkan data histori KIRI dengan menggunakan *Google Maps*.

Tabel 4.1: Handling User Input

Nama	Handling User Input.
Deskripsi	Fungsi yang bertugas untuk menerima input user dan mengubah inputan user menjadi <i>filter params</i>
Pre-kondisi	Aplikasi sudah menerima hasil input dari user.
Input	<p>Input yang akan diterima oleh aksi ini antara lain:</p> <ul style="list-style-type: none"> <li>• <i>Days</i> aksi akan menerima <i>input days</i> yang bertipe <i>array of number</i> dengan <i>range</i> antara 0 sampai 6 yang merepresentasikan hari.</li> <li>• <i>Hours</i> aksi akan menerima <i>input hours</i> yang bertipe <i>array of number</i> dengan <i>range</i> antara 0 sampai 23 yang merepresntasikan jam.</li> <li>• <i>isHeatMap</i> aksi akan menerima parameter bertipe <i>boolean</i> yang menandakan apakah user memilih menggunakan <i>map</i> bertipe <i>heat map</i>.</li> <li>• <i>isMarkerClustering</i> aksi akan menerima parameter bertipe <i>boolean</i> yang menandakan apakah user memilih menggunakan <i>map</i> bertipe <i>marker clustering</i>.</li> </ul>
Output	<p>Output yang akan dihasilkan oleh aksi ini adalah object <i>filter params</i>. <i>Filter params</i> merupakan sebuah objek yang memiliki dua buah atribut antara lain:</p> <ul style="list-style-type: none"> <li>• Atribut <i>day</i> atribut ini adalah atribut untuk menampung <i>array of number</i> yang merepresentasikan hari.</li> <li>• Atribut <i>hour</i> atribut ini adalah atribut untuk menampung <i>array of number</i> yang merepresentasikan jam.</li> </ul>

**Pseudocode** untuk aksi 4.5.1 dapat digambarkan seperti berikut:

---

**Algorithm 1** Handle User Input

---

```

1: function CREATEFILTEROBJ
2:   days  $\leftarrow$  arrayofdays
3:   hours  $\leftarrow$  arrayofhours
4:   filterParams  $\leftarrow$  objectofselectedinput
5:   filterParams.days  $\rightarrow$  days
6:   filterParams.hours  $\rightarrow$  hours
7:   return filterParams

```

---

Tabel 4.2: Menampilkan data histori kiri dengan menggunakan *google maps*

Nama	Handling Output.
Deskripsi	Fungsi yang bertugas untuk menampilkan <i>maps object</i> yang dikirimkan oleh <i>backend</i> .
Pre-kondisi	Aplikasi sudah menerima objek <i>map</i> dari <i>backend</i> .
Input	Aksi memerlukan input antara lain <ul style="list-style-type: none"> <li>• Selected <i>map type</i>. tipe <i>map</i> yang dapat dipilih user adalah <i>heat map</i> atau <i>marker clustering</i>.</li> <li>• Objek <i>map</i> yang telah dikirimkan oleh <i>Google Maps Javascript API</i>.</li> </ul>
Output	Output yang akan dihasilkan oleh aksi ini adalah hasil visualisasi dari objek <i>map</i> yang diterima dari <i>backend</i> . Hasil visualisasi dapat berbentuk <i>Heat Map</i> atau <i>Marker Clustering</i>

**Algorithm 2** Handle Output

```

1: function SETMAP(isMarkerCluster,data)
2:   map ← googlemapsobject
3:   data ← response.data
4:   if isMarkerCluster == true then
5:     markers ← setMarkers(data)
6:     markerCluster ← MarkerClusterer(map,data)
7:   else
8:     heatMap ← setHeatmap(data)
9:   end if
10:

```

**1 Perancangan Pseudocode Pada Backend Server**

- 2 Berdasarkan kapabilitas pada 4.3.2, pseudocode dapat dibuat berdasarkan dua fungsi utama yaitu:
- 3     • Handling load data histori KIRI.
- 4     • Handling filter data histori KIRI.

Tabel 4.3: Load data histori KIRI

Nama	Load Data.
Deskripsi	Fungsi ini adalah fungsi yang berperan untuk meload data histori KIRI.
Pre-kondisi	Data histori kiri sudah tersedia didalam folder asset dan harus bertipe <i>csv</i> .
Input	aksi memerlukan input antara lain <ul style="list-style-type: none"> <li>• Data histori KIRI berbentuk <i>csv</i>.</li> </ul>
Output	<i>Array of Object</i> data histori kiri.

---

**Algorithm 3** Load Data

---

```

1: function LOADDATA
2:   data  $\leftarrow$  arrayofobjects
3:   data  $\leftarrow$  fs.readFile("KiriHistory.csv")
4:   return data
5:
6: end function=0

```

---

Tabel 4.4: Handling Filter Data

Nama	Filter Data.
Deskripsi	Fungsi ini bertugas untuk memfilter data berdasarkan filter parameter yang telah dikirimkan oleh <i>web client</i> .
Pre-kondisi	<i>Web client</i> telah mengirimkan <i>filter param</i> .
Input	Aksi memerlukan input antara lain <ul style="list-style-type: none"> <li>• Filter param.</li> </ul>
Output	<i>Array of Object</i> data histori kiri yang telah disaring.

1

## BAB 5

2

### IMPLEMENTASI DAN PENGUJIAN

3 Bab ini terdiri atas implementasi, pengujian, dan masalah yang dihadapi. Pada bagian implementasi  
4 akan dijelaskan mengenai lingkungan implementasi dan hasil dari implementasi. Pada bagian  
5 pengujian akan berisi hasil dari pengujian. Pada bagian masalah yang dihadapi akan dijelaskan  
6 masalah-masalah yang dihadapi pada saat implementasi.

7

#### 5.1 Implementasi

8

##### 5.1.1 Lingkungan Implementasi

9 Berikut adalah spesifikasi *laptop* yang digunakan untuk implementasi.

- 10 1. *Processor* : AMD Ryzen 7  
11 2. *Memory* : 16 GB DDR4 2400MHz SDRAM  
12 3. *Storage* : 512 GB SSD  
13 4. *VGA* : NVDIA GTX 1660TI  
14 5. *OS* : Windows 10 64-bit

15 Berikut adalah spesifikasi *browser* yang digunakan.

- 16 1. *Name* : Google Chrome  
17 2. *Version* : 90.0.4430.212

18 Berikut adalah spesifikasi perangkat lunak yang digunakan untuk implementasi.

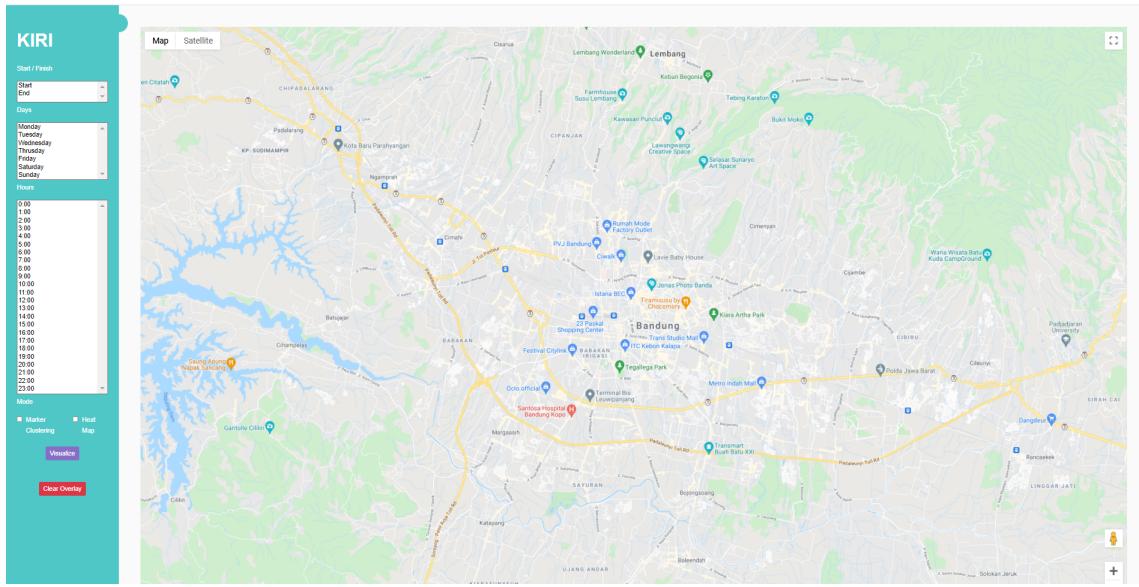
- 19 1. *IDE* : Webstrom 2021.1.1  
20 2. Bahasa Pemrograman : Javascript  
21 3. *Runtime Enviroment* : node.js 14.17.0 LTS

22

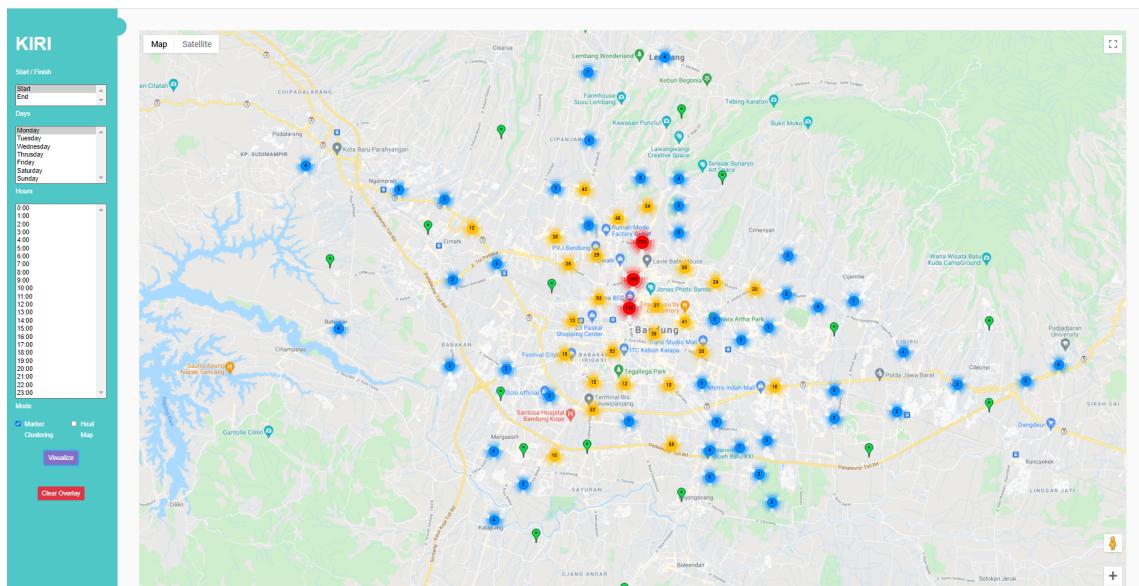
##### 5.1.2 Implementasi Antarmuka Perangkat Lunak

23 Pada implementasinya, antarmuka perangkat lunak telah berhasil dirancang sesuai dengan subbab

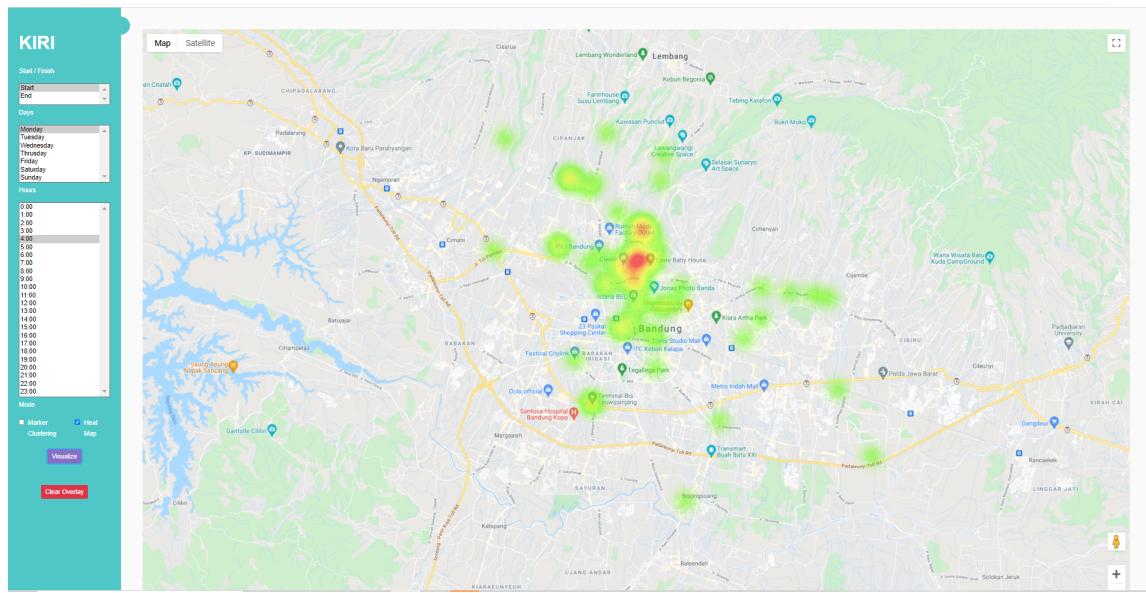
24 4.1. Berikut adalah gambar hasil implementasi.



Gambar 5.1: Tampilan Awal Antarmuka



Gambar 5.2: Tampilan setelah memilih marker clustering



Gambar 5.3: Tampilan setelah memilih Heat Map

### **1 5.1.3 Implementasi Perangkat Lunak**

2 Perangkat lunak yang dibuat sesuai dengan perancangan pada Bab 4. Implementasi aplikasi ini  
3 menggunakan bahasa pemrograman *JavaScript*. Terdapat tiga bagian utama dalam perangkat lunak  
4 yang dibuat.

- 5 • *Model*

6 Bagian ini adalah representasi dari data, bagian ini juga dapat disebut sebagai *backend*. Pada  
7 bagian ini terdapat kelas *model* untuk data histori KIRI. Pada bagian ini juga terdapat kelas  
8 *helper* untuk melakukan operasi pengolahan data.

- 9 • *Router*

10 Bagian ini adalah bagian yang menjadi *controller* dalam perangkat lunak. *Router* akan  
11 berguna sebagai *media* komunikasi antara *web client* dengan *backend server*

- 12 • *View*

13 Bagian ini adalah bagian tampilan atau yang bisa disebut *web client* karena pada perangkat  
14 lunak ini menggunakan *platform website*. Bagian ini bertugas untuk menampilkan hasil  
15 visualisasi dan mengolah input dari user.

### **16 Model**

17 Bagian ini adalah representasi dari data, bagian ini juga dapat disebut sebagai *backend*. Pada  
18 bagian ini terdapat kelas *model* untuk data histori KIRI. Pada bagian ini juga terdapat kelas *helper*  
19 untuk melakukan operasi pengolahan data. Berikut ini adalah gambaran kelas *model* data histori  
20 KIRI:

Kode 5.1: Metode Load Data

```
21 1 const fs = require("fs")
22 2 const {csvToObject, buildFilter, filterData} = require("./Utils");
23 3
24 4 class KiriHistory {
```

```

1   5     constructor() {
2   6       fs.readFile(_dirname + "/data/KIRIStatistics.csv", "utf8", ((err, data
3   7         ) => {
4   8           if (err === null) {
5   9             let arrCSV = data.split("\n")
6   9             arrCSV.shift();
7 10             this.data = arrCSV.map(csvToObject).filter(item => item !==
8 11               undefined);
9 11         }
10 12       }))
11 13     }
12 14
13 15     //return promise
14 16     getData = (filterParams) => {
15 17       let query = buildFilter(filterParams);
16 18       this.data = filterData(this.data, query);
17 19       return this.data;
18 20     }
19 21
20 22   }
21 23
22 24 module.exports = KiriHistory;

```

Pada kelas Model ini memiliki tiga *action* utama yaitu:

### 1. Load Data

Action ini akan dijalankan begitu *object model* dibuat. Action ini akan melakukan load dan normalisasi data histori KIRI. Action ini akan menggunakan method *readFile* yang berasal dari *package filesystem*. Berikut ini adalah contoh implementasi dari action ini.

Kode 5.2: Metode Load Data

```

29   1  constructor() {
30   2    fs.readFile(_dirname + "/data/KIRIStatistics.csv", "utf8", ((err, data) =
31   3      ) => {
32   4        if (err === null) {
33   5          let arrCSV = data.split("\n")
34   5          arrCSV.shift();
35   6          this.data = arrCSV.map(csvToObject).filter(item => item !==
36   6            undefined);
37   7        }
38   8      })
39   9    }

```

Method *readFile* akan melakukan action load pada data histori KIRI pada 3.1. Data yang akan diload bertipe *csv* dan memiliki format data sebagai berikut.

Kode 5.3: Histori Data KIRI

```

43   1 logId , APIKey , Timestamp (UTC) , Action , AdditionalData
44   2 113909 , E5D9904F0A8B4F99 , 2/1/2014 0:07 , PAGELOAD , /5.10.83.30/
45   3 113914 , A44EB361A179A49E , 2/1/2014 0:11 , SEARCHPLACE , taman+fot/10
46   4 113915 , A44EB361A179A49E , 2/1/2014 0:11 , FINDROUTE , "-6.8972513 , 107.6385574 /
47   -6.91358 , 107.62718/1"

```

---

```
1   5 114256 ,308201BB30820124 ,2/1/2014 5:24 ,SEARCHPLACE ,soekarno+hatta+643/10
```

---

3 Ketika *readFile* dijalankan fungsi ini akan memiliki *callback* yang akan mengembalikan data  
4 histori KIRI, data yang telah diload oleh *readFile* akan berbentuk seperti berikut.

Kode 5.4: Raw Data

```
5  [
6    'logId ,APIKey ,Timestamp (UTC) ,Action ,AdditionalData\r',
7    '113909 ,E5D9904F0A8B4F99 ,2/1/2014 0:07 ,PAGELOAD ,/5.10.83.30\r',
8    '113910 ,E5D9904F0A8B4F99 ,2/1/2014 0:07 ,PAGELOAD ,/5.10.83.49\r',
9    '113911 ,E5D9904F0A8B4F99 ,2/1/2014 0:09 ,PAGELOAD ,/5.10.83.30\r',
10   ]
```

---

12 Data tersebut akan ditampung ke dalam variabel yang diberi nama *arrCSV*. Baris pertama  
13 dari data ini adalah *header* dari csv. Pada proses normalisasi data baris pertama ini tidaklah  
14 dibutuhkan oleh karena itu akan dihilangkan baris pertama menggunakan method *shift()*.  
15 Sehingga data akan berbentuk sebagai berikut.

Kode 5.5: Raw Data 2

```
16 [
17   '113909 ,E5D9904F0A8B4F99 ,2/1/2014 0:07 ,PAGELOAD ,/5.10.83.30\r',
18   '113910 ,E5D9904F0A8B4F99 ,2/1/2014 0:07 ,PAGELOAD ,/5.10.83.49\r',
19   '113911 ,E5D9904F0A8B4F99 ,2/1/2014 0:09 ,PAGELOAD ,/5.10.83.30\r',
20   ]
```

---

22 Data tersebut kemudian akan dipetakan ke dalam *map* dengan menggunakan fungsi *mapper*  
23 yang diberi nama *csvToObject*. Fungsi ini akan memetakan dan melakukan *formatting* dari  
24 setiap *value* dari 5.5.

Kode 5.6: CSV Mapper

```
25 1 const csvToObject = item => {
26 2   let cols = item.split(",")
27 3   let action = cols[3];
28 4   if (action === "FINDROUTE") {
29 5     let startLng = cols[5].split("/") [0]
30 6     let endLat = cols[5].split("/") [1]
31 7     let fullDate = new Date(cols[2])
32 8     return {
33 9       apiKey: cols[0],
34 10      timestamp: fullDate,
35 11      action,
36 12      startCor: {lat: parseFloat(cols[4].substr(1)), lng: parseFloat
37 13        (startLng)},
38 14      endCor: {lat: parseFloat(endLat), lng: parseFloat(cols[6].
39 15        split("/") [0])},
40 16      day: fullDate.getDay(),
41 17      hour: fullDate.getHours()
42 18    }
43 19  }
44 20 }
```

---

1 Fungsi ini akan akan memanfaatkan *method split* untuk membagi raw data 5.5 menjadi array  
 2 yang berbentuk seperti berikut.

Kode 5.7: Data Split Result

```
3 [  
4   '115566',  
5   'A44EB361A179A49E',  
6   '2/2/2014 9:07',  
7   'FINDROUTE',  
8   '"-0.7819376',  
9   '100.2871169/-6.90359',  
10  '107.60040/1"\r'  
11 ]
```

13 Berdasarkan data 5.7 kita dapat mengekstrak *action type* 3.1 pada posisi ketiga dari array  
 14 tersebut. Dalam aplikasi ini hanya akan digunakan data yang memiliki *action FINDROUTE*  
 15 hal ini dikarenakan hanya pada *action* ini data mengandung *value* dari posisi *start* dan *finish*.  
 16 Setelah function *csvToObject* ini dijalankan maka akan mengembalikan kumpulan *object* data  
 17 yang berbentuk sebagai berikut.

Kode 5.8: Result Data

```
18 [  
19   {  
20     apiKey: '113915',  
21     timestamp: 2014-01-31T17:11:00.000Z,  
22     action: 'FINDROUTE',  
23     startCor: { lat: -6.8972513, lng: 107.6385574 },  
24     endCor: { lat: -6.91358, lng: 107.62718 },  
25     day: 6,  
26     hour: 0  
27   }  
28 ]
```

## 30 2. Get Data

31 Fungsi ini akan dijalankan setiap ada request dari *web client*. *Action* ini akan mengembalikan  
 32 data histori KIRI sesuai dengan filter parameter yang diberikan. Berikut ini adalah contoh  
 33 implementasi dari *action* ini.

Kode 5.9: Function Get Data

```
34   getData = (filterParams) => {  
35     let query = buildFilter(filterParams);  
36     this.data = filterData(this.data, query);  
37     return this.data;  
38   }
```

40 Pada fungsi ini akan menerima parameter *filterParams* yang berasal dari *web client*. Fungsi ini  
 41 akan menjalankan dua perintah *buildFilter* dan *filterData* dan akan mengembalikan data  
 42 histori yang sesuai dengan *filterParams* yang diberikan. Fungsi *buildFilter* akan mengubah  
 43 *filterParams* yang berbentuk seperti berikut.

Kode 5.10: Filter Parameter

```

1  1  {
2      2      day: [ 0 ],
3          3      hour: [ 8, 9, 10, 11 ]
4  }

```

---

Menjadi *query params*. Hal ini dilakukan karena fungsi *filter* yang dirancang pada perangkat lunak ini akan memanfaatkan *function filter* dari javascript.

### 3. Filter Data

Berikut ini adalah hasil implementasi dari *function filter*:

Kode 5.11: Fungsi Filter

```

10 1  const filterData = (data, query) => {
11 2      const keysWithMinMax = [];
12 3      const filteredData = data.filter((item) => {
13 4          for (let key in query) {
14 5              if (item[key] === undefined) {
15 6                  return false;
16 7              } else if (keysWithMinMax.includes(key)) {
17 8                  if (query[key]['min'] !== null && item[key] < query[key]['min']) {
18 9                      return false;
19 10                 }
20 11             if (query[key]['max'] !== null && item[key] > query[key]['max']) {
21 12                 return false;
22 13             }
23 14         } else if (!query[key].includes(item[key])) {
24 15             return false;
25 16         }
26 17     }
27 18     return true;
28 19 });
29 20     return filteredData;
30 21 };

```

---

Fungsi ini akan memanfaatkan *method filter* dan *includes javascript*. Fungsi ini akan menghasilkan data yang telah disaring sesuai dengan *query params* yang diberikan.

## 36 Router

- 37 Bagian ini merupakan bagian yang bertugas sebagai *controller* antara *backend logic* dan *web client*.
- 38 Implementasi router dapat dilihat sebagai berikut.

Kode 5.12: Router Implementation

```

39 1  const express = require('express')
40 2  const app = express()
41 3  const Kiri = require("../model/Kirihistory");
42 4  const path = require('path');
43 5  let modelKiri = new Kiri();
44 6  app.use(express.json())

```

```

1   7 app.get("/", (req, res) => {
2     8   res.sendFile(path.join(`_${__dirname}../../views/index.html`));
3   9 })
4 10
5 11
6 12 app.post('/searchRoute', (req, res) => {
7   13   let filterParams = req.body;
8   14   let data = modelKiri.filterData(filterParams);
9   15   res.status(200).json({
10   16     'status': 'OK',
11   17     'messages': 'Data',
12   18     'data': data,
13   19   })
14 20
15 21 })
16 22
17 23
18 24 module.exports = {app, express};

```

---

Bagian ini memiliki dua fungsi utama yaitu melakukan render untuk *web client* dan mengatur *response* dan *request* antara *web client* dan *backend logic*.

### 1. Render HTML

Perangkat lunak ini menggunakan *node.js* sebagai *runtime manager*. Perangkat lunak ini memanfaatkan *method* *sendFile* milik *node.js* untuk dapat melakukan *rendering* pada *route* yang diinginkan. Implementasi pada bagian ini dapat dilihat sebagai berikut.

Kode 5.13: Render HTML

```

26 1 app.get("/", (req, res) => {
27   2   res.sendFile(path.join(`${viewPath}/index.html`));
28   3 })

```

---

Pada potongan kode diatas menunjukan bahwa *index.html* akan dirender pada *route* '/'.

### 2. Controller antara *Web client* dan *Backend Logic*

Implementasi pada bagian ini dapat dilihat sebagai berikut.

Kode 5.14: Get Filtered Data

```

33 1 app.post('/searchRoute', (req, res) => {
34   2   let filterParams = req.body;
35   3   let data = modelKiri.getData(filterParams);
36   4   res.status(200).json({
37   5     'status': 'OK',
38   6     'messages': 'Data',
39   7     'data': data,
40   8   })
41   9
42 10 })

```

---

Pada bagian ini perangkat lunak akan membuat *endpoint* yang akan menerima parameter berupa *filter parameter* dan akan mengembalikan hasil data histori yang telah difilter.

## 1 View

- 2 Bagian ini adalah bagian disebut *web client* pada 4.3.1. Pada bagian ini terdapat fungsi untuk  
 3 menerima *input user* dan menampilkan hasil visualisasi menggunakan *Google Maps Javascript API*.  
 4 Pada bagian ini format *user interface* akan ditulis dalam format *Hypertext Markup Language (html)*  
 5 hal ini dapat dilihat pada file *index.html*. Selain untuk tampilan bagian ini juga berfungsi untuk  
 6 menerima input user dan mengubahnya menjadi *filter params*, hal ini dapat dilihat pada file *maps.js*.  
 7 Bentuk implementasi dari bagian ini dapat dilihat sebagai berikut.

Kode 5.15: Map Implementation

```

8 1 const IS_LOCAL_TEST = true;
9 2 const host = IS_LOCAL_TEST ? "http://localhost:3000" : "https://
10      kiri-app.herokuapp.com";
11 3
12 4 function initMap() {
13 5     let map = new google.maps.Map(document.getElementById("map"), {
14 6         center: {lat: -6.914744, lng: 107.609810},
15 7         zoom: 13,
16 8
17 9     });
18 10    return map;
19 11 }
20 12
21 13 function setMarkers(data, isStart, isEnd) {
22 14     let locations = [];
23 15     if (isStart) {
24 16         data.forEach(item => {
25 17             let startMark = {
26 18                 name: "start",
27 19                 pos: item.startCor,
28 20                 icon: "http://maps.google.com/mapfiles/ms/icons/green-dot.png"
29 21             }
30 22             locations.push(startMark)
31 23         })
32 24     }
33 25
34 26     if (isEnd) {
35 27         data.forEach(item => {
36 28             let endMark = {name: "end", pos: item.endCor, icon: "http://
37 29                 maps.google.com/mapfiles/ms/icons/red-dot.png"}
38 30             locations.push(endMark)
39 31         })
40 32     }
41 33     // console.log("filtered markers size", data.length)
42 34     let markers = locations.map((item) => {
43 35         return new google.maps.Marker({
44 36             position: {lat: parseFloat(item.pos.lat), lng: parseFloat(
45 37                 item.pos.lng)},
46 38             icon: item.icon
47 39         });
48 40     });
49 41

```

```
1  40      return markers;
2  41  }
3  42
4  43  function setHeatMap(arrData, map, isStart, isEnd) {
5  44      let heatmapData = [];
6  45      if (isStart) {
7  46          arrData.forEach(item => {
8  47
9  48              let startCoor = {location: new google.maps.LatLng(item.startCor.lat
10         , item.startCor.lng)}
11 49              heatmapData.push(startCoor)
12 50
13 51      })
14 52  }
15 53  if (isEnd) {
16 54      arrData.forEach(item => {
17 55          // {location: new google.maps.LatLng(37.782, -122.447), weight:
18         0.5},
19 56          let endLocationObj = {location: new google.maps.LatLng(
20         item.endCor.lat, item.endCor.lng)}
21 57          heatmapData.push(endLocationObj)
22 58      })
23 59  }
24 60
25 61      let heatmap = new google.maps.visualization.HeatmapLayer({
26 62          data: heatmapData,
27 63          radius: 50
28 64      });
29 65      heatmap.setMap(map);
30 66
31 67      return heatmap;
32 68  }
33 69
34 70  function sendRequest(url, data = {}) {
35 71      return axios.post(url, data);
36 72  }
37 73
38 74
39 75  function docReady(fn) {
40 76      // see if DOM is already available
41 77      if (document.readyState === "complete" || document.readyState === "
42         interactive") {
43 78          // call on next available tick
44 79          setTimeout(fn, 1);
45 80      } else {
46 81          document.addEventListener("DOMContentLoaded", fn);
47 82      }
48 83  }
49 84
50 85
51 86  createFilterObj = () => {
```

```
1  87      let days = Array.from(document.getElementById("day").selectedOptions).map(
2          option => parseInt(option.value))
3  88      let hours = Array.from(document.getElementById("hour").selectedOptions).map(
4          option => parseInt(option.value));
5  89      let filter = {day: days, hour: hours};
6  90      return filter
7  91  }
8  92
9  93
10 94  deleteMarkes = (markers) => {
11 95      markers.forEach(item => {
12 96          item.setMap(null)
13 97      })
14 98      return null
15 99  }
16 100
17 101
18 102 docReady(function () {
19 103     let map = initMap()
20 104     let markerCluster = null;
21 105     let markers;
22 106     let heatMap;
23 107
24 108     document.getElementById("send-btn").onclick = function (e) {
25 109         e.preventDefault();
26 110         let filterParams = createFilterObj();
27 111         if (filterParams.day.length <= 0) {
28 112             console.log("fill day");
29 113         } else if (filterParams.hour.length <= 0) {
30 114             console.log("hour need to be filled");
31 115         } else {
32 116             let isMarkerCluster = document.getElementById("marker-cluster").
33                 checked
34 117             let isHeatMap = document.getElementById("heat-map").checked
35 118             let statusSelected = Array.from(document.getElementById("start-finish").selectedOptions).map(option => option.value)
36 119             let statusStartChecked = statusSelected.includes("start");
37 120             let statusEndChecked = statusSelected.includes("end")
38 121             if (markerCluster) {
39 122                 markerCluster.removeMarkers(markers);
40 123                 markers = deleteMarkes(markers)
41 124             }
42 125             if (heatMap) {
43 126                 heatMap.setMap(null)
44 127             }
45 128             sendRequest(` ${host}/searchRoute`, filterParams).then(res => {
46 129                 document.getElementById("result-length").innerHTML =
47 130                     res.data.data.length;
48 131                 if (isMarkerCluster) {
49 132                     map.setMapTypeId(google.maps.MapTypeId.ROADMAP);
50 133                     let data = res.data.data;
51 134                     if (!Array.isArray(markers)) {
```

```

1 134                         markers = setMarkers(data, statusStartChecked,
2                               statusEndChecked)
3 135                         markerCluster = new MarkerClusterer(map, markers, {
4 136                             imagePath:
5 137                                 "https://developers.google.com/maps/
6                                     documentation/javascript/examples/
7                                     markerclusterer/m",
8 138                         });
9 139                     }
10 140                 }
11 141             if (isHeatMap) {
12 142                 map.setMapTypeId(google.maps.MapTypeId.ROADMAP);
13 143                 if (!heatMap || heatMap.getMap() === null) {
14 144                     heatMap = setHeatMap(res.data.data, map,
15                           statusStartChecked, statusEndChecked)
16 145                 }
17 146             }
18 147         });
19 148     }
20 149
21 150 }
22 151 });

```

---

Bagian ini memiliki tiga tugas utama yaitu:

1. Menerima *input user* dan mengubah *input user* menjadi *filter params*

Implementasi bagian ini dapat dilihat menjadi:

Kode 5.16: Input User

```

27 1  createFilterObj = () => {
28 2     let days = Array.from(document.getElementById("day").selectedOptions).
29     map(option => parseInt(option.value))
30 3     let hours = Array.from(document.getElementById("hour").selectedOptions
31     ).map(option => parseInt(option.value));
32 4     let filter = {day: days, hour: hours}
33 5     return filter
34 6 }

```

---

Ketika fungsi ini dijalankan maka akan mengambil *value* dari elemen dengan id *day* dan *hour*. Setelah itu data tersebut akan dibuat ke dalam bentuk *map* yang disebut sebagai *filter parameter*. Pada bagian ini juga terdapat fungsi *docReady* yang bertugas untuk berkomunikasi dengan *node.js* atau *Google Maps Javascript API*. Hasil implementasi fungsi tersebut adalah sebagai berikut:

Kode 5.17: docReady Method

```

41 1  function docReady(fn) {
42 2     if (document.readyState === "complete" || document.readyState === "
43     interactive") {
44 3         setTimeout(fn, 1);
45 4     } else {
46 5         document.addEventListener("DOMContentLoaded", fn);
47 6     }

```

---

```
1    7  }
```

3 Pada fungsi ini memiliki parameter sebuah fungsi dimana fungsi ini akan dijalankan setiap  
4 *tick*. Hal ini dilakukan agar koneksi antar *backend* bisa tetap terus dilakukan. Berikut ini  
5 implementasi fungsi *docReady* pada perangkat lunak ini.

Kode 5.18: docReady Method

```
6  1  docReady(function () {
7      2      let map = initMap()
8      3      let markerCluster = null;
9      4      let markers;
10     5      let heatMap;
11     6      document.getElementById("send-btn").onclick = function (e) {
12         7          e.preventDefault();
13         8          let filterParams = createFilterObj()
14         9          let isMarkerCluster = document.getElementById("marker-cluster").
15             checked
16         10         let isHeatMap = document.getElementById("heat-map").checked
17         11         let statusSelected = Array.from(document.getElementById(""
18             start-finish").selectedOptions).map(option => option.value)
19         12         let statusStartChecked = statusSelected.includes("start");
20         13         let statusEndChecked = statusSelected.includes("end")
21         14         sendRequest("http://localhost:3000/searchRoute", filterParams).
22             then(res => {
23                 15                 if (isMarkerCluster) {
24                     16                     map.setMapTypeId(google.maps.MapTypeId.ROADMAP);
25                     17                     let data = res.data.data;
26                     18                     if (!Array.isArray(markers)) {
27                         19                         markers = setMarkers(data, statusStartChecked,
28                             statusEndChecked)
29                         20                         markerCluster = new MarkerClusterer(map, markers, {
30                             21                             imagePath:
31                                 "https://developers.google.com/maps/
32                                     documentation/javascript/examples/
33                                     markerclusterer/m",
34                         22                         });
35                         23                     });
36                         24                 }
37                         25             });
38
39             27             if (isHeatMap) {
40                 28                 map.setMapTypeId(google.maps.MapTypeId.ROADMAP);
41                 29                 if (!heatMap || heatMap.getMap() === null) {
42                     30                     heatMap = setHeatMap(res.data.data, map,
43                         statusStartChecked, statusEndChecked)
44                     31                 }
45                     32             }
46             33         });
47             34     }
48
49             36     document.getElementById('clear-btn').onclick = function (e) {
50                 37                 e.preventDefault();
51                 38                 if (markerCluster) {
```

```

1      39             markerCluster.removeMarkers(markers);
2      40             markers = deleteMarkes(markers)
3      41         }
4      42     if (heatMap) {
5      43         heatMap.setMap(null)
6      44     }
7      45   }
8 46 }) ;

```

---

## 2. Mengirim dan menerima data dari router

Implementasi untuk bagian ini dapat dilihat pada *method docReady*.

Kode 5.19: communication method

```

12    1   document.getElementById("send-btn").onclick = function (e) {
13    2     e.preventDefault();
14    3     let filterParams = createFilterObj()
15    4     let isMarkerCluster = document.getElementById("marker-cluster").
16      checked
17    5     let isHeatMap = document.getElementById("heat-map").checked
18    6     let statusSelected = Array.from(document.getElementById(""
19      start-finish").selectedOptions).map(option => option.value)
20    7     let statusStartChecked = statusSelected.includes("start")
21    8     let statusEndChecked = statusSelected.includes("end")
22    9     sendRequest("http://localhost:3000/searchRoute", filterParams).
23      then(res => {
24 10       if (isMarkerCluster) {
25 11         map.setMapTypeId(google.maps.MapTypeId.ROADMAP);
26 12         let data = res.data.data;
27 13         if (!Array.isArray(markers)) {
28 14           markers = setMarkers(data, statusStartChecked,
29             statusEndChecked)
30 15           markerCluster = new MarkerClusterer(map, markers, {
31 16             imagePath:
32 17               "https://developers.google.com/maps/
33               documentation/javascript/examples/
34               markerclusterer/m",
35 18             });
36 19           }
37 20         }
38 21       }
39 22     if (isHeatMap) {
40 23       map.setMapTypeId(google.maps.MapTypeId.ROADMAP);
41 24       if (!heatMap || heatMap.getMap() === null) {
42 25         heatMap = setHeatMap(res.data.data, map,
43           statusStartChecked, statusEndChecked)
44 26       }
45 27     }
46 28   });
47 29 }

```

---

Pada potongan kode di atas dapat dilihat jika tombol send-btn ditekan, maka akan memanggil *method sendRequest*. Method ini bertujuan untuk berkomunikasi dengan *backend* untuk

1 mendapatkan data yang histori yang telah disaring berdasarkan input dari user. Fungsi ini  
 2 juga bertujuan untuk mendapatkan hasil visualisasi dari data yang telah disaring dengan  
 3 menggunakan *Google Maps Javascript API*.

### 4 3. Berkommunikasi dengan *Google Maps Javascript API*

5 Perangkat lunak ini menggunakan *Google Maps Javascript API* sebagai salah satu *third*  
 6 *party library* untuk melakukan visualisasi data. Terdapat dua buah opsi yang disediakan oleh  
 7 perangkat lunak ini dalam melakukan visualisasi data yaitu *Heat Map* dan *Marker Clustering*.  
 8 Untuk dapat menggunakan opsi tersebut *web client* perlu membuat input parameter yang  
 9 sesuai dengan kebutuhan *Google Maps Javascript API 2.5*. Implementasi dari bagian ini  
 10 dapat dilihat sebagai berikut.

Kode 5.20: Map Input

```

11 1 function setMarkers(data, isStart, isEnd) {
12 2     let locations = [];
13 3     if (isStart) {
14 4         data.forEach(item => {
15 5             let startMark = {
16 6                 name: "start",
17 7                 pos: item.startCor,
18 8                 icon: "http://maps.google.com/mapfiles/ms/icons/
19 9                     green-dot.png"
20 10            }
21 11            locations.push(startMark)
22 12        })
23 13    }
24 14    if (isEnd) {
25 15        data.forEach(item => {
26 16            let endMark = {name: "end", pos: item.endCor, icon: "http://
27 17                maps.google.com/mapfiles/ms/icons/red-dot.png"}
28 18            locations.push(endMark)
29 19        })
30 20    }
31 21    // console.log("filtered markers size" , data.length)
32 22    let markers = locations.map((item) => {
33 23        return new google.maps.Marker({
34 24            position: {lat: parseFloat(item.pos.lat), lng: parseFloat(
35 25                item.pos.lng)},
36 26            icon: item.icon
37 27        });
38 28    return markers;
39 29  }
40 30
41 31
42 32    function setHeatMap(arrData, map, isStart, isEnd) {
43 33        let heatmapData = [];
44 34        if (isStart) {
45 35            arrData.forEach(item => {

```

```

1      36
2      37          let startCoor = {location: new google.maps.LatLng(
3          item.startCor.lat, item.startCor.lng)}
4      38          heatmapData.push(startCoor)
5      39
6      40      })
7      41  }
8      42  if (isEnd) {
9      43      arrData.forEach(item => {
10     44          // {location: new google.maps.LatLng(37.782, -122.447),
11     45          // weight: 0.5},
12     46          let endLocationObj = {location: new google.maps.LatLng(
13          item.endCor.lat, item.endCor.lng)}
14     47          heatmapData.push(endLocationObj)
15     48      })
16     49  }
17
18     50  let heatmap = new google.maps.visualization.HeatmapLayer({
19          data: heatmapData,
20          radius: 50
21      });
22     54  heatmap.setMap(map);
23
24     56  return heatmap;
25 }
26

```

---

## 27 5.2 Pengujian

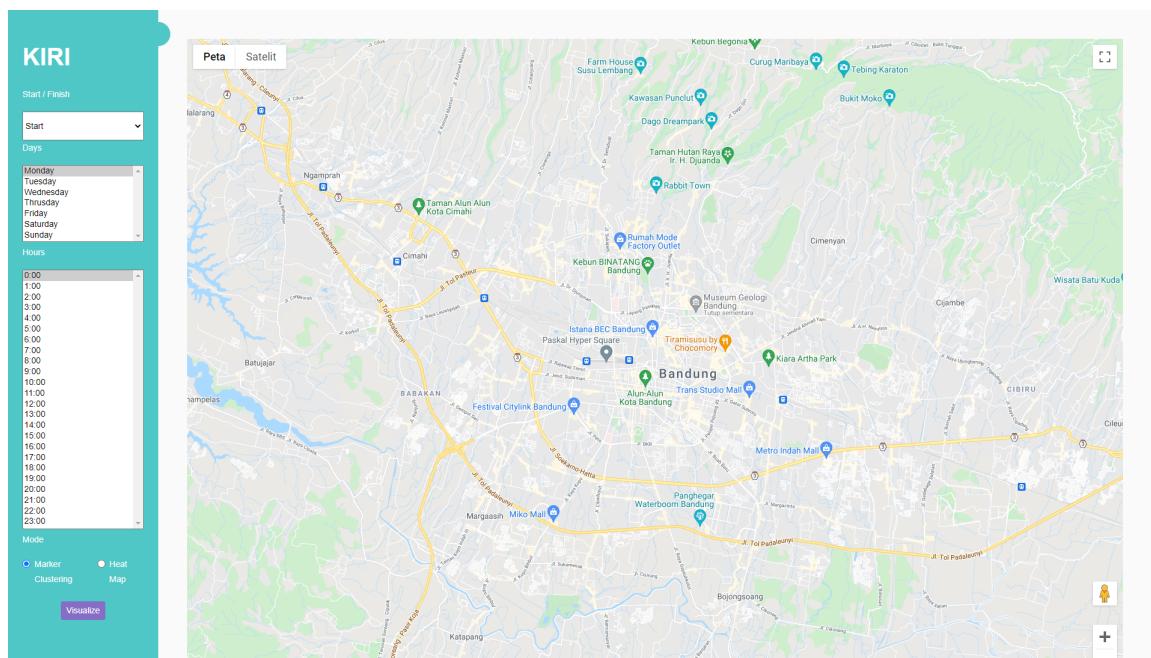
28 Pengujian dilakukan dengan menggunakan dua metode yaitu pengujian fungsional dan pengujian  
29 eksperimental. Pengujian fungsional bertujuan untuk menguji fitur-fitur yang telah disediakan.  
30 Pengujian Eksperimental bertujuan untuk mencari pola-pola tertentu pada data histori KIRI.

### 31 5.2.1 Pengujian Fungsional

32 Pengujian fungsional dilakukan dengan menjalankan fitur-fitur yang ada. Berikut adalah fitur-fitur  
33 yang ada pada perangkat lunak.

#### 34 1. Filter Data Histori KIRI

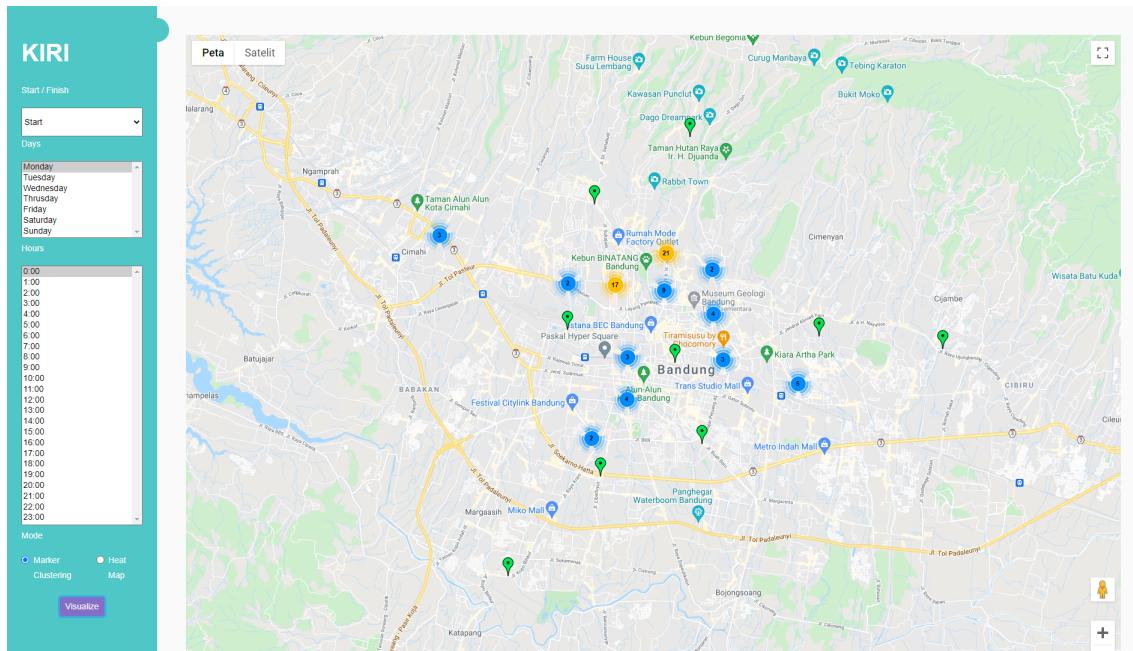
35 Fitur ini bertujuan untuk melakukan proses filterisasi pada data histori KIRI sesuai dengan  
36 input yang diberikan pengguna. Pengguna dapat memasukan tiga jenis empat jenis input  
37 sesuai dengan yang telah disediakan pada tampilan 5.4, pengguna dapat melakukan filter  
38 berdasarkan atribut *day* dapat dilihat pada kotak bewarna merah, *hour* dapat dilihat pada  
39 kotak bewarna *orange*, dan *start/finsih* dapat dilihat pada kotak bewarna hitam.



Gambar 5.4: Tampilan Filter Data Histori KIRI

1      2. Menampilkan visualisasi data dalam bentuk *marker*

Fitur ini bertujuan untuk memvisualisasikan data dalam bentuk *marker*. Jika pengguna memilih *option marker clustering* dan menekan tombol *visualize*, maka program akan menampilkan data dalam bentuk *marker* pada *object map* seperti Gambar 5.5.

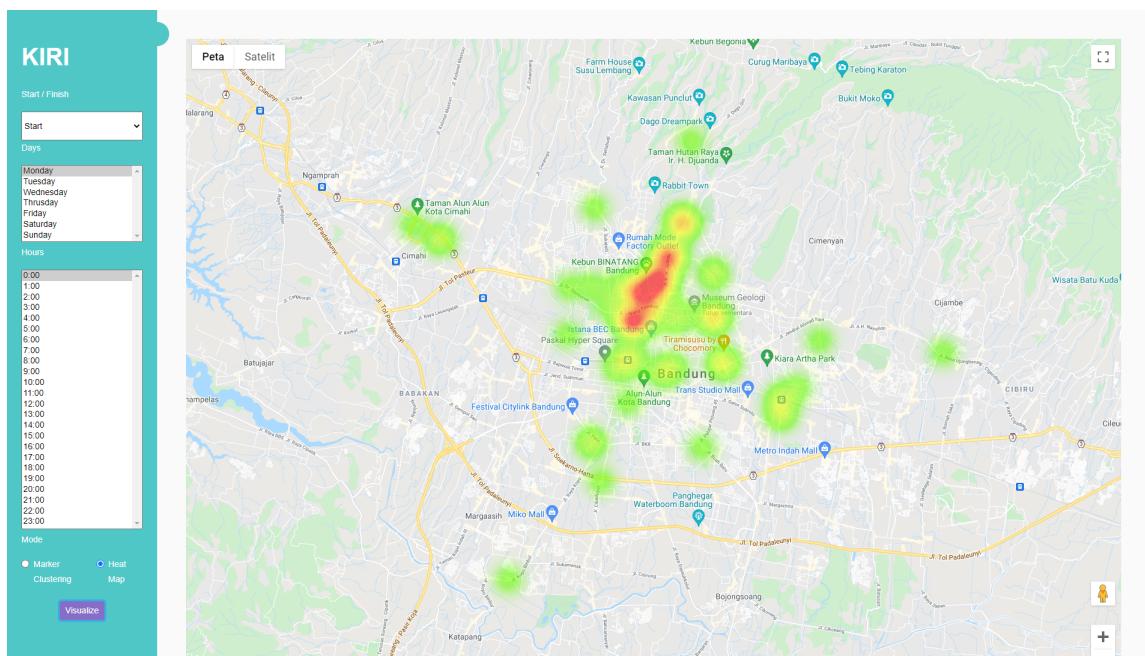


Gambar 5.5: Tampilan Marker Data Histori KIRI

5      6      7      3. Menampilkan visualisasi data dalam bentuk *heat map*

Fitur ini bertujuan untuk memvisualisasikan data dalam bentuk *heat map*. Jika pengguna memilih *option heat map* dan menekan tombol *visualize*, maka program akan menampilkan

1 data dalam bentuk *heat map* pada *object map* seperti Gambar 5.6.



Gambar 5.6: Tampilan Heat Map Data Histori KIRI

2 Pengujian fungsional pada perangkat lunak ini dilakukan dengan menggunakan *automated test*  
3 dengan bantuan *third party app cypress.io*. Pengujian dibagi dalam tiga tahap, yaitu:

4 1. Pengujian tampilan

5 Pengujian ini bertujuan untuk memastikan tampilan telah tersedia / berhasil ditampilkan  
6 pada *web client*. Pengujian dilakukan dengan membuat *test case*:

Kode 5.21: UI Test Case

```

8   1 describe('KIRI history html dom end to end testing', () => {
9     2   it('Check website succesfully loaded ', () => {
10    3     cy.visit(WEB_DOMAIN_NAME)
11    4     cy.get('h1').should('have.text', "KIRI");
12    5   })
13    6   it('Check start/finish html already exist or already loaded', () => {
14    7     cy.get('#start-finish').select(COMPLETION_STATUS[0]).should(
15       have.value, 'start');
16    8     cy.get('#start-finish').select(COMPLETION_STATUS[1]).should(
17       have.value, 'end');
18    9   })
19    10  it('Check days html already exist or already loaded', () => {
20    11    let expectedValue = ['0', "1", "2", "3", "4", "5", "6"];
21    12    cy.get('#day')
22    13      .select(DAYS)
23    14      .invoke('val')
24    15      .should('deep.equal', expectedValue)
25    16  })
26    17  it('Check time html dom already exist or already loaded', () => {
27    18    let expectedValue = ['0', "1", "2", "3", "4", "5", "6", "7", "8",

```

```

1           "9", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19"
2           ", "20", "21", "22", "23"];
3   19     cy.get('#hour')
4   20       .select(HOURS)
5   21       .invoke('val')
6   22       .should('deep.equal', expectedValue)
7   23   })
8   24   it('Check visualization mode html dom already exist or already
9       loaded', () => {
10  25     cy.get('#marker-cluster').check();
11  26     cy.get('#heat-map').click().check();
12  27   })
13  28 }

```

---

15 2. Pengujian fungsional dengan menggunakan metode *visual regression*

16 Pengujian ini bertujuan untuk menguji fungsi *heat map* dan *marker clustering* metode yang  
17 digunakan adalah membandingkan gambar *map* sebelum dan sesudah pemanggilan fungsi  
18 tersebut. *Syntax test case* yang dibuat akan berbentuk seperti berikut.

19

Kode 5.22: Funcional Test Case Visual Regression

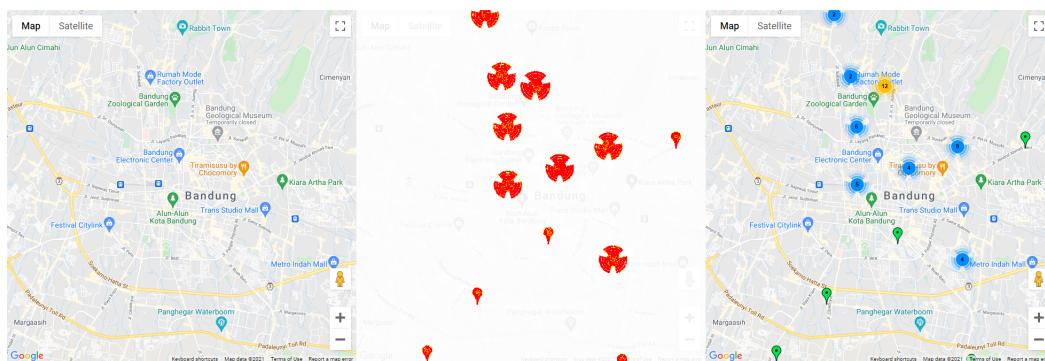
```

20 1 COMPLETION_STATUS.forEach(status => {
21 2     DAYS.forEach(day => {
22 3         HOURS_TEST.forEach(hour => {
23 4             describe('Functional Testing check with visual regression',
24 5                 () => {
25 6                 it(`Test for marker cluster functional in status ${status}
26 7                     , day ${day}, and time ${hour}`, () => {
27 8                     cy.get('#marker-cluster').click();
28 9                     cy.get('#start-finish').select(status).get("#day").
29 10                     select(day);
30 11                     cy.get("#hour").select(hour).get("#send-btn").click().
31 12                     get("#map").notMatchImageSnapshot();
32 13
33 14                     cy.get('#start-finish').select(status).get("#day").
34 15                     select(day);
35 16                     cy.get("#hour").select(hour).get("#send-btn").click().
36 17                     get("#map").wait(2000).notMatchImageSnapshot();
37 18
38 19             })
39 20         })
40 21     })
41 22   })
42 23 }

```

---

44 Pengujian ini akan memiliki hasil seperti pada Gambar 5.7.



Gambar 5.7: Visual regression sample result

3. Pengujian fungsional dengan menggunakan metode *http response status*

Pengujian ini bertujuan untuk menguji fungsi *heat map* dan *marker clustering* metode yang digunakan adalah melihat *response status* yang dikeluarkan oleh *node server*. *Syntax test case* yang dibuat akan berbentuk seperti berikut.

Kode 5.23: Funcional Test Case

```

6   1 COMPLETION_STATUS.forEach(status => {
7     2   DAYS.forEach(day => {
8       3     HOURS_TEST.forEach(hour => {
9         4       describe("Functional Testing check with http request status",
10           5         () => {
11             6           it(`Test for marker cluster in status ${status}, day ${day}
12               , and time ${hour}`), () => {
13               7                 cy.get('#marker-cluster').click();
14               8                 cy.get('#start-finish').select(status).get("#day").
15                   select(day);
16               9                 cy.get("#hour").select(hour).get("#send-btn").click().
17                   request('POST', '/searchRoute').then(res =>
18                     9                   expect(res.status).to.eq(200)
19                   );
20               10             });
21             11           });
22           12         });
23         13       });
24       14     });
25     15   });

```

Total pengujian yang berjumlah 47 *test case* yang sudah dilengkapi dengan bukti *screenshot* dan *video* yang dapat diakses pada tautan [dashboard](#).

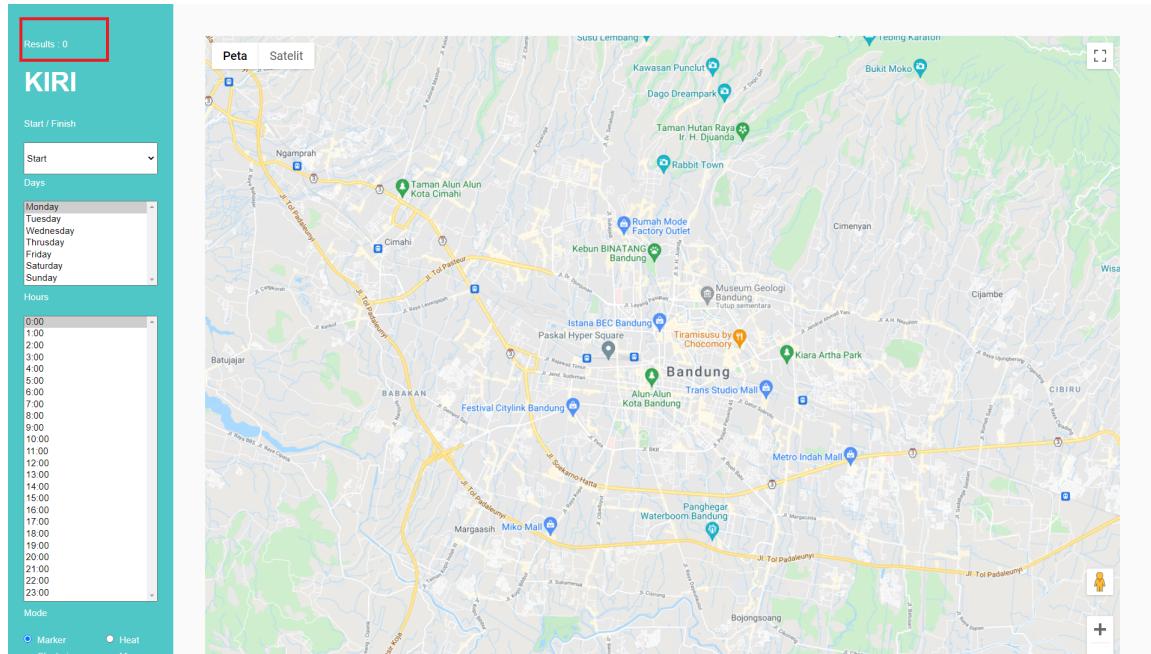
### 5.2.2 Pengujian Eksperimental

Tujuan dari pengujian ini adalah mencari pola-pola tertentu pada data histori KIRI dengan menggunakan metode observasi. Pengujian akan dilakukan untuk menjawab beberapa pertanyaan yaitu:

1. Bagaimana penggunaan perangkat lunak KIRI pada saat jam kerja?
2. Bagaimana penggunaan KIRI pada saat non jam kerja ?

1. Daerah manakah paling sering dijadikan titik awal pencarian rute ?
2. Daerah manakah paling sering dijadikan tujuan akhir pencarian rute ?
3. Daerah manakah paling sering dijadikan titik awal pencarian rute pada saat *weekend* ?
4. Daerah manakah paling sering dijadikan titik tujuan pencarian rute pada saat *weekend* ?

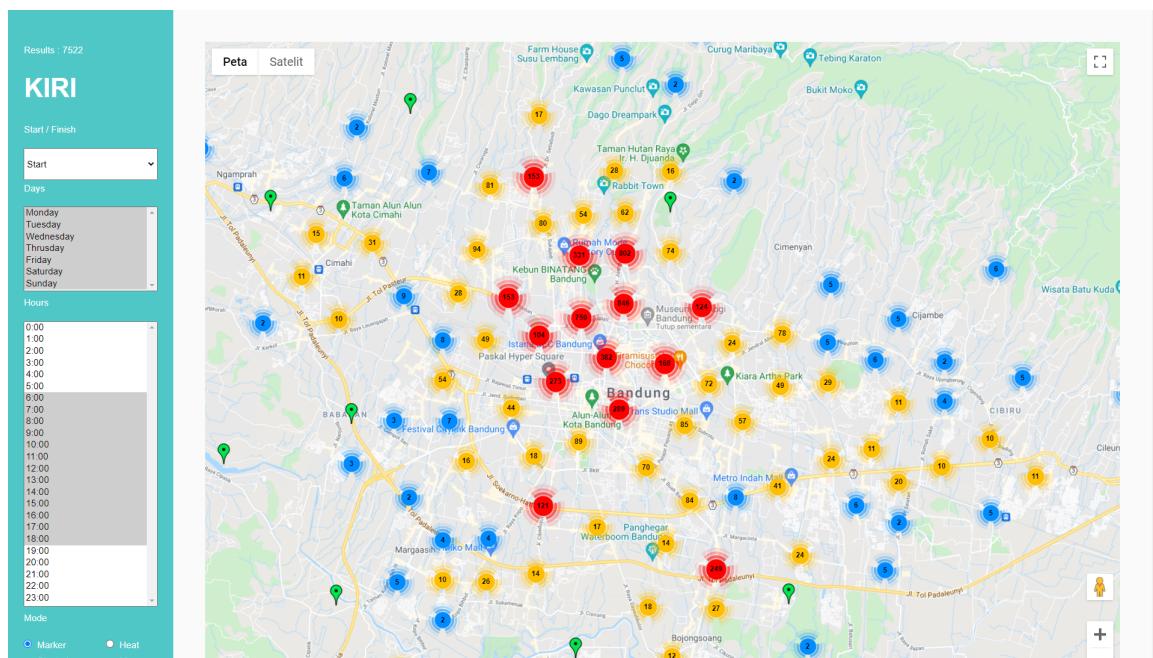
Untuk membantu menjawab beberapa pertanyaan diatas akan ditambahkan sebuah parameter untuk mengetahui berapa jumlah data yang didapat saat pengguna melakukan visualisasi pada Gambar 5.8.



Gambar 5.8: Parameter Tambahan

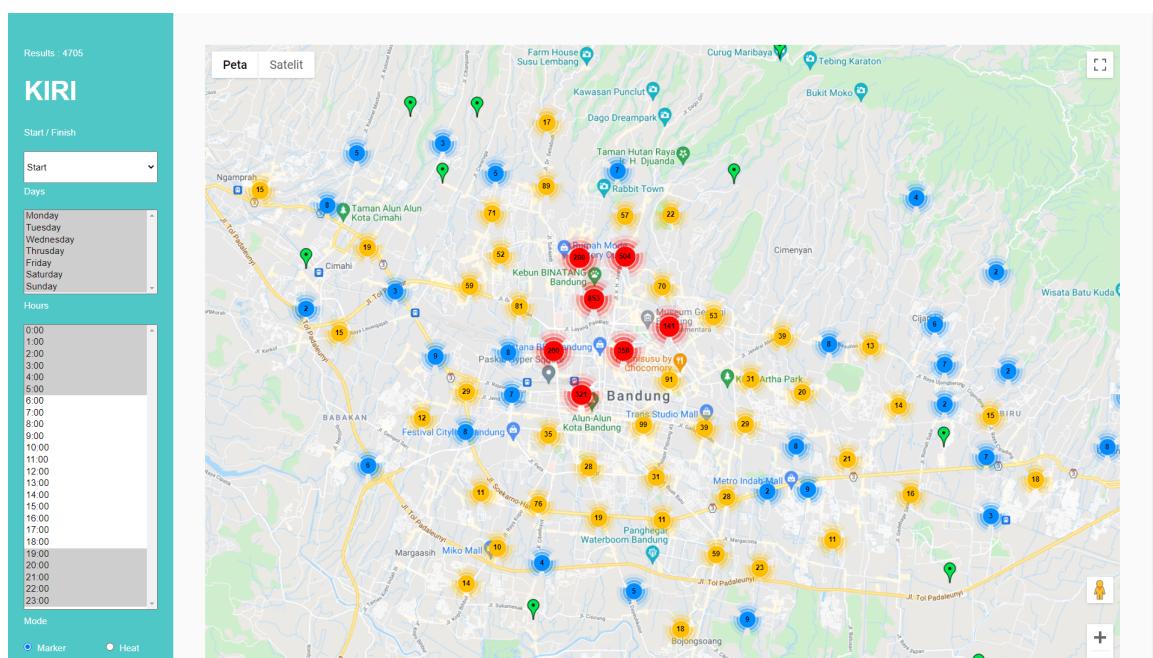
#### 8 Pengujian 1: Bagaimana penggunaan perangkat lunak KIRI pada saat jam kerja?

9 Pada datalog yang tersedia dapat diketahui bahwa hasil log tersebut diambil pada tahun 2014. Kita dapat mengasumsikan bahwa jadwal kerja yang berlaku pada saat itu adalah dari pukul 06:00 - 18:00. Berdasarkan informasi tersebut kita akan melakukan observasi pada hari senin sampai jumat 12 dari jam 06:00 - 18:00. Hasil dari pengujian dapat dilihat pada gambar 5.9.



Gambar 5.9: Senin-Sabtu Jam Kerja

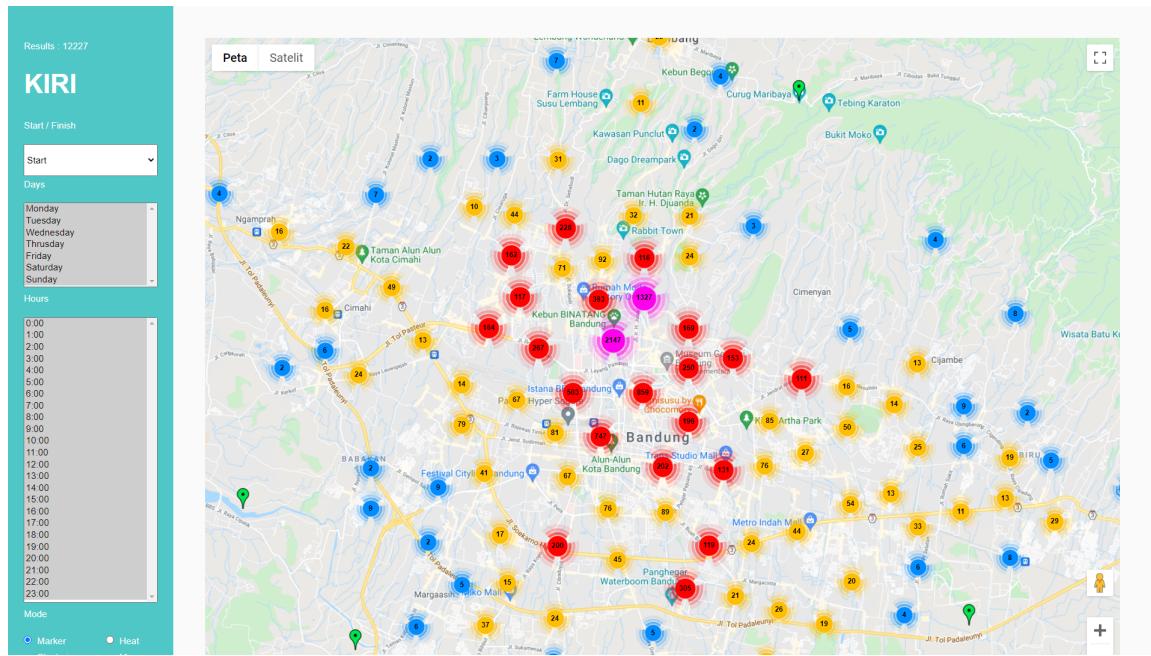
- 1 Berdasarkan hasil observasi pada gambar 5.9 terdapat pemanggilan data sebanyak 7522 kali.
- 2 **Pengujian 2: Bagaimana penggunaan perangkat lunak KIRI pada saat non-jam kerja?**
- 3 Pada datalog yang tersedia akan dilakukan observasi untuk melihat jumlah penggunaan perangkat
- 4 lunak KIRI pada jam non-operasional angkutan umum. Jam non-kerja dapat kita asumsikan dari
- 5 pukul 19:00-07:00.



Gambar 5.10: Senin-Sabtu Non Jam Kerja

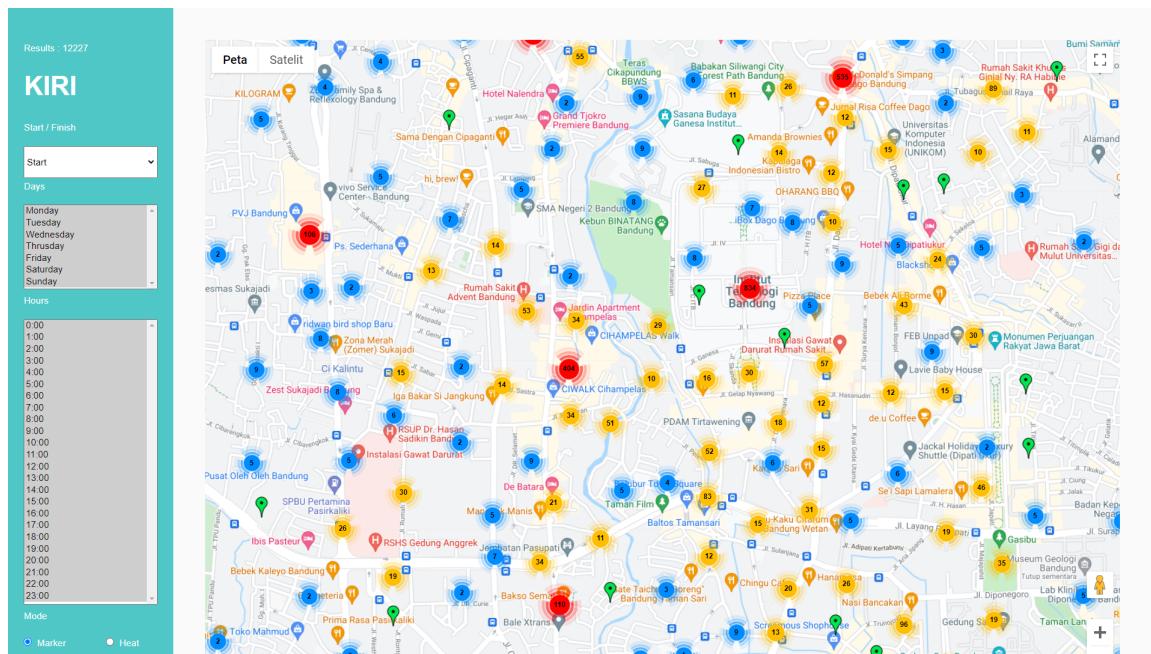
- 6 Berdasarkan hasil observasi pada gambar 5.10 terdapat pemanggilan data sebanyak 4705 kali.

### 1 Pengujian 3: Daerah manakah paling sering dijadikan titik awal pencarian rute?



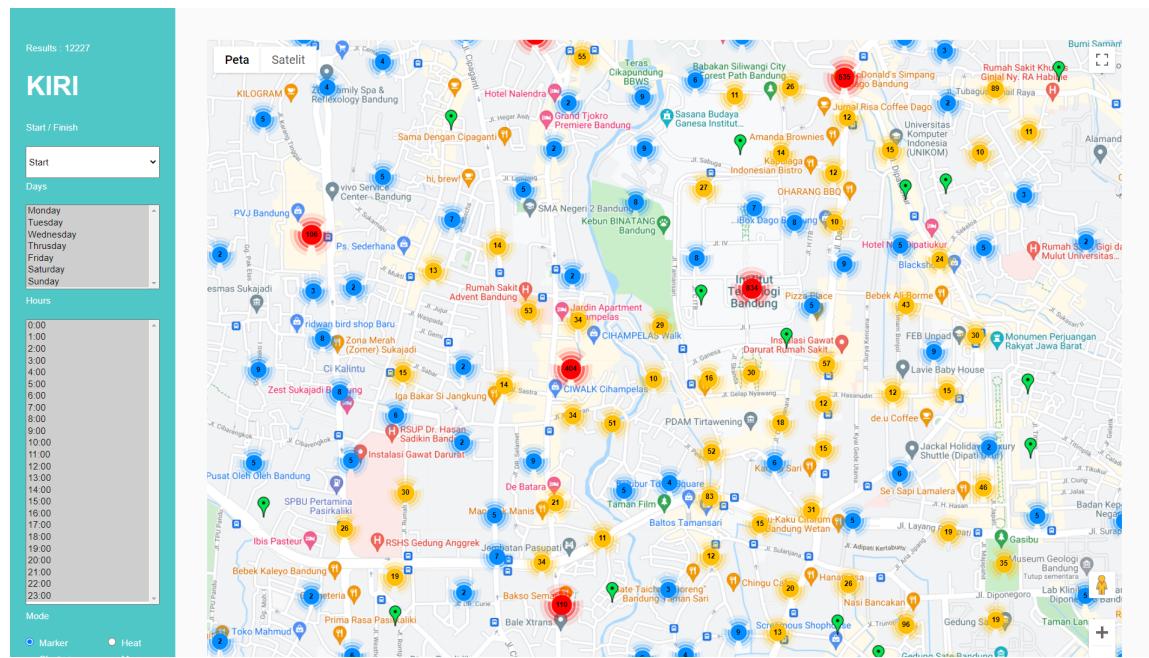
Gambar 5.11: Daerah titik awal terbanyak

- 2 Berdasarkan gambar 5.11 dapat dilihat terdapat satu daerah yang memiliki jumlah pencarian terbanyak sebanyak 2147 kali. Ketika dilakukan observasi lebih lanjut dapat diketahui bahwa daerah tersebut adalah daerah Taman Sari seperti pada Gambar 5.12.



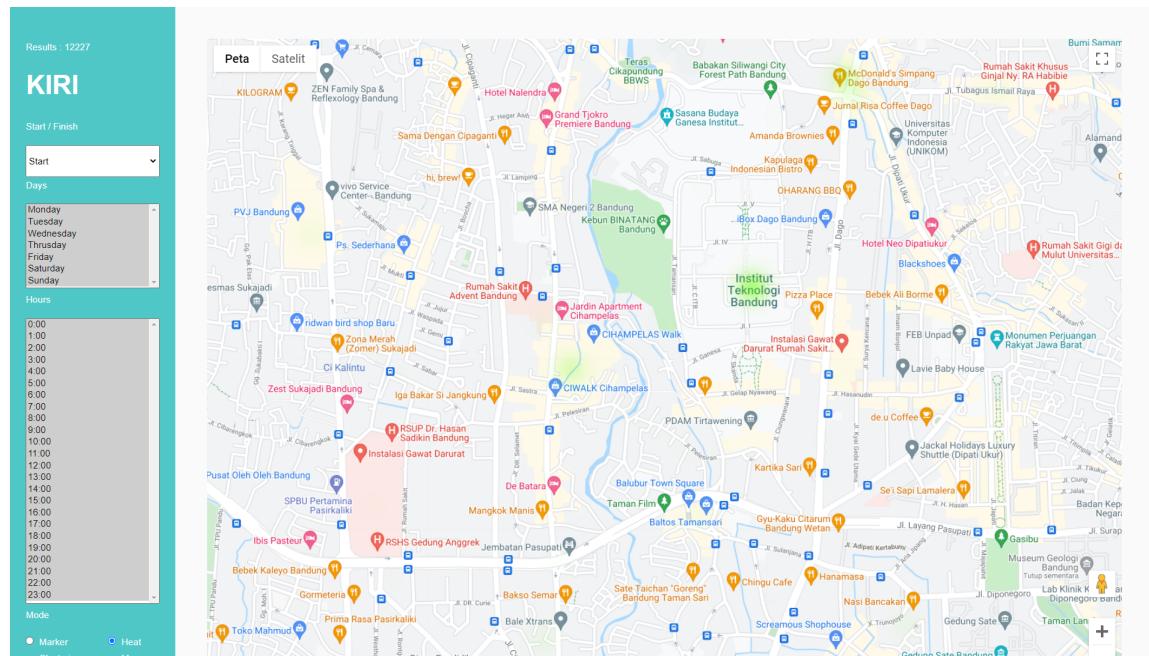
Gambar 5.12: Daerah Taman Sari

- 5 Berdasarkan hasil visualisasi pada gambar 5.12 akan dilakukan proses observasi lebih dalam untuk daerah Taman Sari yang dapat dilihat pada gambar 5.13.



Gambar 5.13: Daerah Taman Sari Zoomed

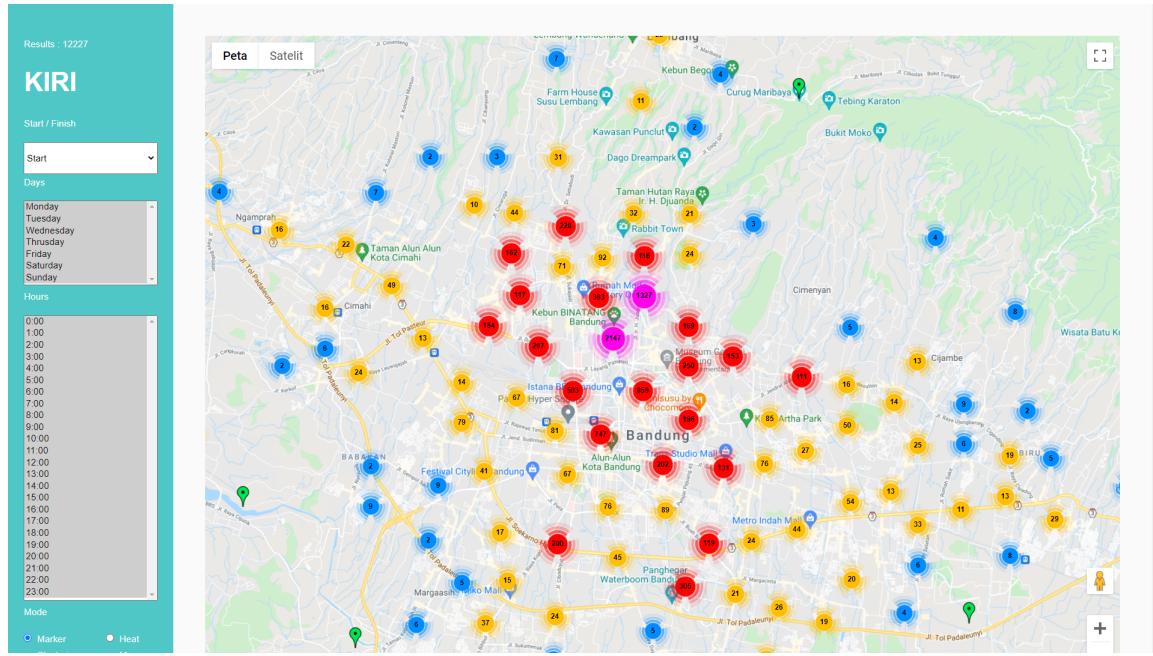
- 1 Berdasarkan gambar 5.13 terdapat lima daerah yang paling sering dijadikan titik awal pencarian  
 2 rute. Daerah tersebut adalah Paris Van Java , Cihampelas Walk, Institut Teknologi Bandung (ITB),  
 3 dan McDonald's Simpang Dago. Untuk memperkecil hasil pencarian mode visualisasi akan diganti  
 4 menjadi *heat map*. Hal ini dilakukan agar dapat melihat titik-titik yang intensitas terbesar.



Gambar 5.14: Heat Map Taman Sari

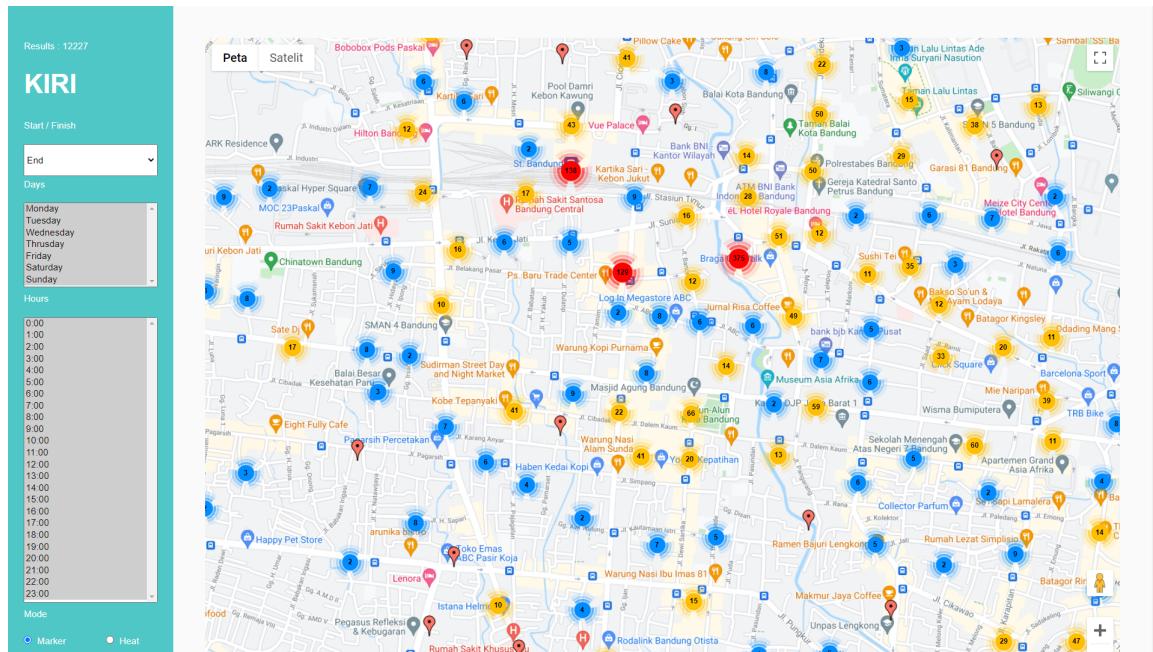
- 5 Berdasarkan gambar 5.14 dapat disimpulkan bahwa titik yang menjadi tempat tujuan terbanyak  
 6 adalah Institut Teknologi Bandung (ITB).

### 1 Pengujian 4: Daerah manakah paling sering dijadikan tujuan akhir pencarian rute?



Gambar 5.15: Daerah titik tujuan terbanyak

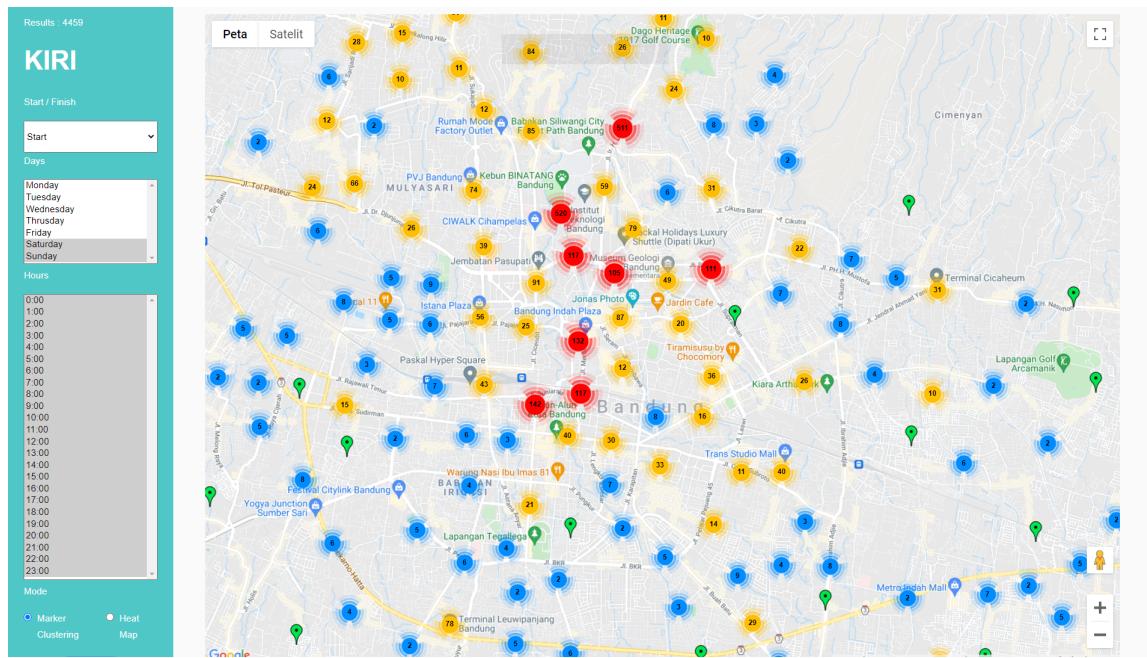
- 2 Berdasarkan gambar 5.15 dapat dilihat terdapat satu daerah yang memiliki jumlah pencarian terbanyak sebanyak 1350 kali. Ketika dilakukan observasi lebih lanjut dapat diketahui bahwa daerah tersebut adalah daerah Braga City Walk. Untuk mendapatkan hasil yang lebih mendetail akan dilakukan proses observasi lebih dalam untuk daerah Braga City Walk yang dapat dilihat pada gambar 5.16.



Gambar 5.16: Gambar Daerah Braga

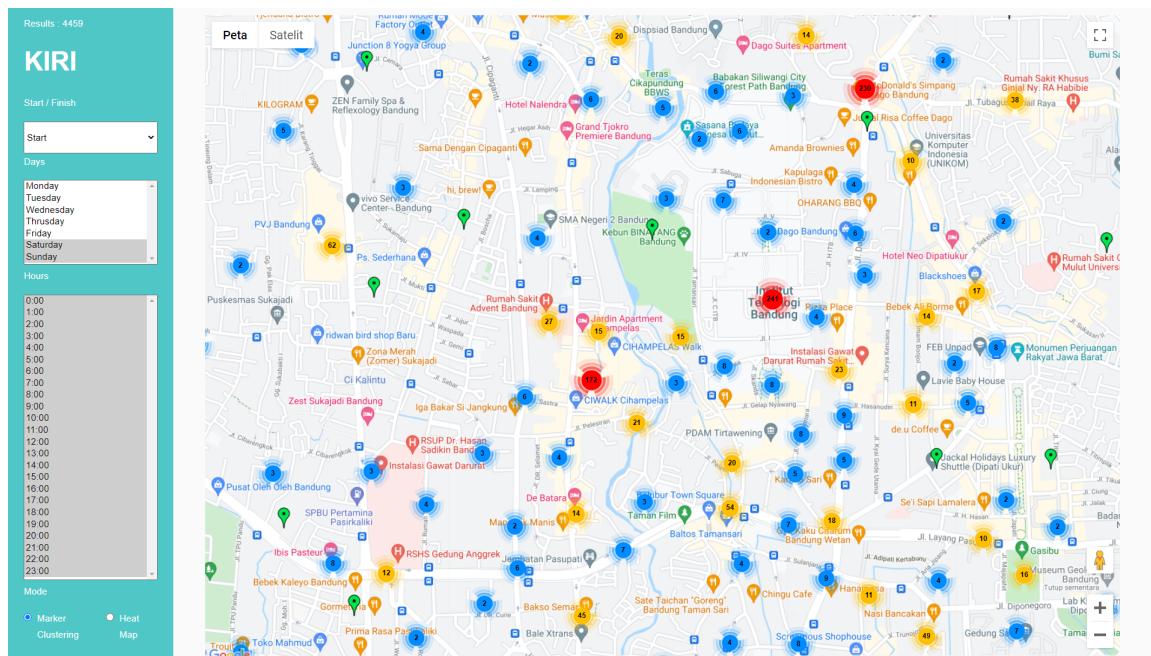
- 1 Berdasarkan hasil gambar 5.16 terdapat tiga titik yang paling sering menjadi destinasi tujuan  
 2 pengguna perangkat lunak KIRI. titik tersebut ialah Stasiun Bandung, Braga City Walk, dan Pasar  
 3 Baru Trade Center.

4 **Pengujian 5: Daerah manakah paling sering dijadikan titik awal pencarian rute pada  
 5 saat weekend?**



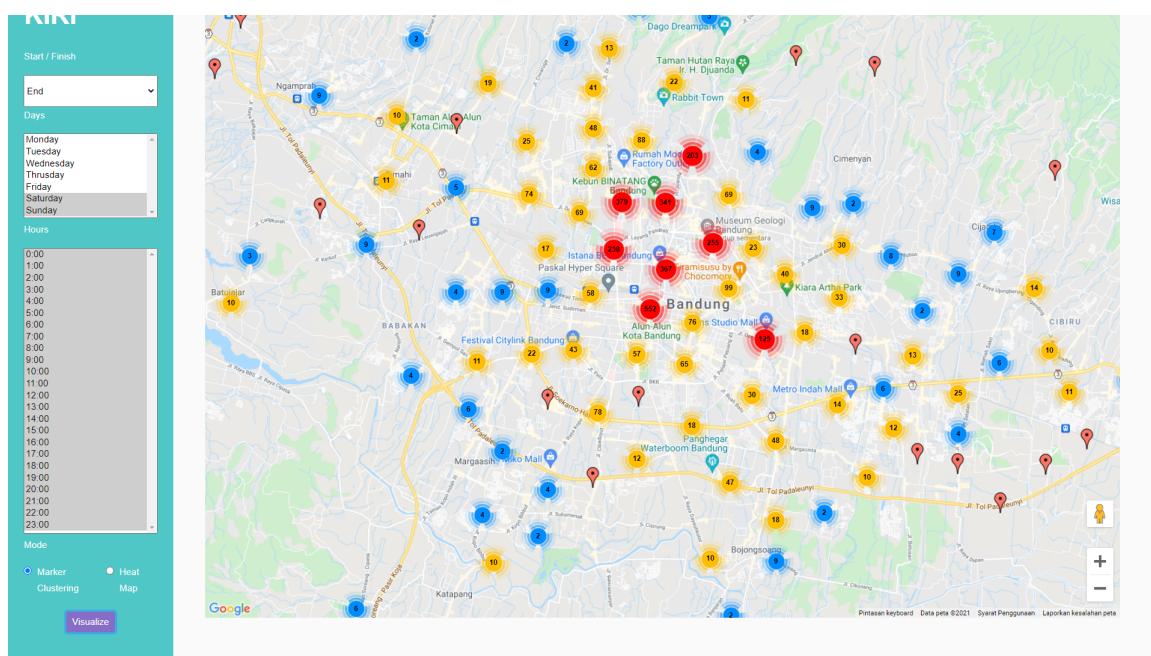
Gambar 5.17: Daerah dengan titik start pada saat weekend

- 6 Berdasarkan gambar 5.17 dapat dilihat terdapat satu daerah yang memiliki jumlah pencarian  
 7 terbanyak sebanyak 775 kali. Ketika dilakukan observasi lebih lanjut dapat diketahui bahwa daerah  
 8 tersebut merupakan daerah Taman Sari.



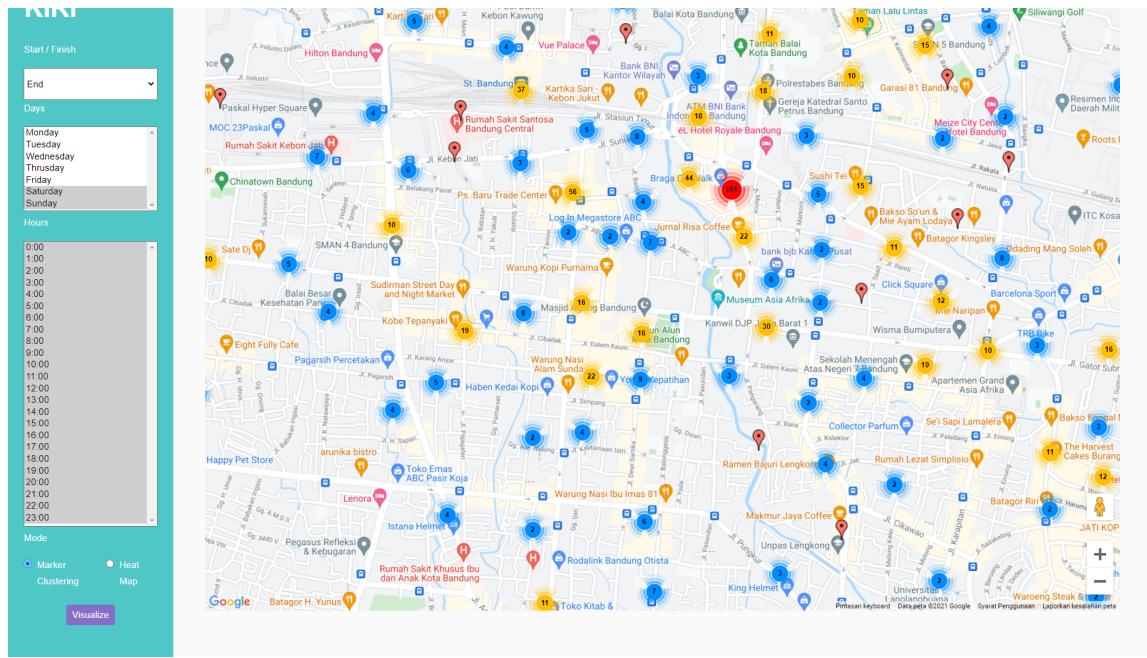
Gambar 5.18: Daerah Taman Sari Weekend

- 1 Ketika dilakukan observasi lanjutan seperti pada gambar 5.18, dapat terdapat tiga hotspot yang
- 2 paling sering dijadikan tempat awal rute pencarian tempat ini adalah Cihampelas Walk, Institut
- 3 Teknologi Bandung (ITB), dan McDonald's Simpang Dago.
  
- 4 **Pengujian 6: Daerah manakah paling sering dijadikan titik tujuan pencarian rute pada saat weekend?**



Gambar 5.19: Daerah dengan titik tujuan pada saat weekend

- 1 Berdasarkan gambar 5.19 dapat dilihat terdapat satu daerah yang memiliki jumlah pencarian  
 2 terbanyak sebanyak 551 kali. Ketika dilakukan observasi lebih lanjut dapat diketahui bahwa daerah  
 3 tersebut merupakan daerah Alun-Alun Kota Bandung.



Gambar 5.20: Daerah Alun Alun Kota Bandung

- 4 Ketika dilakukan observasi lanjutan seperti pada gambar 5.20, dapat terdapat sebuah hotspot  
 5 yang paling sering dijadikan tempat tujuan akhir rute pencarian tempat ini adalah Cafe Jurnal  
 6 Risa.

### 7 5.3 Masalah yang Dihadapi saat Implementasi

- 8 Penulis memiliki beberapa masalah yang dihadapi dalam pengembangan aplikasi visualisasi data  
 9 histori KIRI menggunakan *Google Maps Javascript API*:
- 10 1. Perlu dilakukan proses pembatasan data, hal ini dikarenakan data histori yang diberikan  
 11 sangatlah luas bahkan mencakup pencarian rute untuk daerah Jakarta. Hal ini mengakibatkan  
 12 mode visualisasi *heat map* kurang dapat memberikan informasi dikarenakan persebaran data  
 13 yang terlalu banyak.
- 14 2. Pengujian pada perangkat lunak ini hanya berdasarkan tiga atribut yaitu tempat awal /  
 15 tempat tujuan, hari, dan jam, dan belum mencakup atribut seperti tanggal, bulan, tahun.
- 16 3. Dataset yang diberikan meruupakan *data log* yang tercatat pada tahun 2014 sehingga ada  
 17 beberapa hotspot yang mungkin saja sudah tidak relevan dengan keadaan saat ini.
- 18 4. Proses *functional testing* tidak dapat dilakukan secara maksimal dikarenakan adanya limitasi  
 19 penggunaan API dari pihak *Google*.

1

## BAB 6

2

### KESIMPULAN DAN SARAN

3 Bab ini membahas kesimpulan berdasarkan implementasi dan pengujian, serta saran-saran untuk  
4 pengembangan berikutnya.

#### 5 6.1 Kesimpulan

6 Berdasarkan hasil penelitian yang dilakukan, diperoleh kesimpulan-kesimpulan sebagai berikut:

- 7 1. Berdasarkan hasil perbandingan pengujian satu [5.2.2](#) dengan pengujian dua [5.2.2](#). Dapat  
8 disimpulkan bahwa mayoritas pengguna aplikasi KIRI akan menggunakan aplikasinya pada  
9 saat jam kerja.
- 10 2. Berdasarkan hasil pengujian tiga [5.2.2](#) dan pengujian lima [5.2.2](#). Dapat disimpulkan bahwa  
11 titik yang paling sering dijadikan tempat awal pencarian rute terdapat di daerah Taman  
12 Sari, lebih detail nya terdapat pada tiga titik utama yaitu Paris Van Java, Institut Teknologi  
13 Bandung, McDonald's Simpang Dago.
- 14 3. Berdasarkan hasil pengujian empat [5.2.2](#). Dapat disimpulkan terdapat tiga titik yang paling  
15 sering menjadi destinasi tujuan pencarian rute yaitu Pasar Baru Trade Center, Stasiun Kereta  
16 Bandung, Braga City Walk.
- 17 4. Berdasarkan hasil pengujian enam [5.2.2](#). Dapat disimpulkan terdapat sebuah titik yang paling  
18 sering menjadi destinasi tujuan pencarian rute pada saat weekend yaitu Cafe Jurnal Risa.
- 19 5. Berdasarkan hasil seluruh pengujian perangkat lunak ini telah berhasil memvisualisasikan  
20 data histori KIRI kedalam bentuk *heat map* dan *marker clustering* dengan menggunakan  
21 *Google Maps*.

#### 22 6.2 Saran

23 Penulis memiliki beberapa saran untuk pengembangan aplikasi visualisasi data histori KIRI dengan  
24 *Google Maps*:

- 25 1. Perlu diperhatikan bahwa data log histori KIRI memiliki banyak potensi untuk dapat diolah,  
26 pada perangkat lunak ini hanya akan mengolah data berdasarkan tiga atribut yaitu start/finish,  
27 hari, dan waktu. Jika atribut dapat ditambah seperti atribut tanggal pastinya hasil kesimpulan  
28 akan dapat lebih baik.
- 29 2. Pengujian pada perangkat lunak ini hanya dengan menggunakan metode observasi. Pengujian  
30 dapat dilakukan dengan metode *Exploratory data analysis* (EDA) memberikan hasil kesimpulan  
31 yang lebih baik.



## **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] Shafranovich, Y. (2005) Common format and mime type for comma-separated values (csv) files. Technical Report 7111.
- [3] Dahl, R. nodejs. <https://nodejs.org/en/about/>.
- [4] Dahl, R. npmjs. <https://docs.npmjs.com/about-npm>.
- [5] Holowaychuk, T. expressjs. <https://expressjs.com/en/starter/installing.html>.
- [6] Google Google maps refrence. <https://developers.google.com/maps/documentation/javascript/reference>.



## LAMPIRAN A

### KODE PROGRAM

Kode A.1: Router Implementation

```
1 const express = require('express')
2 const app = express()
3 const Kiri = require("../model/Kirihistory");
4 const path = require('path');
5 let modelKiri = new Kiri();
6 app.use(express.json())
7 app.get("/", (req, res) => {
8     res.sendFile(path.join(`${__dirname}../../views/index.html`));
9 })
10
11
12 app.post('/searchRoute', (req, res) => {
13     let filterParams = req.body;
14     let data = modelKiri.filterData(filterParams);
15     res.status(200).json({
16         'status': 'OK',
17         'messages': 'Data',
18         'data': data,
19     })
20
21 })
22
23
24 module.exports = {app, express};
```

---

Kode A.2: Map Implementation

```
1 const IS_LOCAL_TEST = true;
2 const host = IS_LOCAL_TEST ? "http://localhost:3000" : "https://
    kiri-app.herokuapp.com";
3
4 function initMap() {
5     let map = new google.maps.Map(document.getElementById("map"), {
6         center: {lat: -6.914744, lng: 107.609810},
7         zoom: 13,
8
9     });
10    return map;
11 }
12
13 function setMarkers(data, isStart, isEnd) {
14     let locations = [];
15     if (isStart) {
16         data.forEach(item => {
17             let startMark = {
18                 name: "start",
19                 pos: item.startCor,
20                 icon: "http://maps.google.com/mapfiles/ms/icons/green-dot.png"
```

```
21         }
22         locations.push(startMark)
23     })
24 }
25
26 if (isEnd) {
27     data.forEach(item => {
28         let endMark = {name: "end", pos: item.endCor, icon: "http://
29             maps.google.com/mapfiles/ms/icons/red-dot.png"}
30         locations.push(endMark)
31     })
32
33 // console.log("filtered markers size" , data.length)
34 let markers = locations.map((item) => {
35     return new google.maps.Marker({
36         position: {lat: parseFloat(item.pos.lat), lng: parseFloat(
37             item.pos.lng)},
38         icon: item.icon
39     });
40 }
41 return markers;
42
43 function setHeatMap(arrData, map, isStart, isEnd) {
44     let heatmapData = [];
45     if (isStart) {
46         arrData.forEach(item => {
47
48             let startCoor = {location: new google.maps.LatLng(item.startCor.lat
49                 , item.startCor.lng)}
50             heatmapData.push(startCoor)
51
52         })
53     if (isEnd) {
54         arrData.forEach(item => {
55             // {location: new google.maps.LatLng(37.782, -122.447), weight:
56             0.5},
57             let endLocationObj = {location: new google.maps.LatLng(
58                 item.endCor.lat, item.endCor.lng)}
59             heatmapData.push(endLocationObj)
60         })
61     let heatmap = new google.maps.visualization.HeatmapLayer({
62         data: heatmapData,
63         radius: 50
64     });
65     heatmap.setMap(map);
66
67     return heatmap;
68 }
69
70 function sendRequest(url, data = {}) {
71     return axios.post(url, data);
72 }
73
74
75 function docReady(fn) {
76     // see if DOM is already available
77     if (document.readyState === "complete" || document.readyState === "
78         interactive") {
```

```
78     // call on next available tick
79     setTimeout(fn, 1);
80 } else {
81     document.addEventListener("DOMContentLoaded", fn);
82 }
83 }
84
85
86 createFilterObj = () => {
87     let days = Array.from(document.getElementById("day").selectedOptions).map(
88         option => parseInt(option.value))
89     let hours = Array.from(document.getElementById("hour").selectedOptions).map(
90         option => parseInt(option.value));
91     let filter = {day: days, hour: hours};
92     return filter
93 }
94
95 deleteMarkes = (markers) => {
96     markers.forEach(item => {
97         item.setMap(null)
98     })
99     return null
100 }
101
102 docReady(function () {
103     let map = initMap()
104     let markerCluster = null;
105     let markers;
106     let heatMap;
107
108     document.getElementById("send-btn").onclick = function (e) {
109         e.preventDefault();
110         let filterParams = createFilterObj();
111         if (filterParams.day.length <= 0) {
112             console.log("fill day");
113         } else if (filterParams.hour.length <= 0) {
114             console.log("hour need to be filled");
115         } else {
116             let isMarkerCluster = document.getElementById("marker-cluster").
117                 checked
118             let isHeatMap = document.getElementById("heat-map").checked
119             let statusSelected = Array.from(document.getElementById(""
120                 "start-finish").selectedOptions).map(option => option.value)
121             let statusStartChecked = statusSelected.includes("start");
122             let statusEndChecked = statusSelected.includes("end")
123             if (markerCluster) {
124                 markerCluster.removeMarkers(markers);
125                 markers = deleteMarkes(markers)
126             }
127             if (heatMap) {
128                 heatMap.setMap(null)
129             }
130             sendRequest(` ${host}/searchRoute`, filterParams).then(res => {
131                 document.getElementById("result-length").innerHTML =
132                     res.data.data.length;
133                 if (isMarkerCluster) {
134                     map.setMapTypeId(google.maps.MapTypeId.ROADMAP);
135                     let data = res.data.data;
136                     if (!Array.isArray(markers)) {
137                         markers = setMarkers(data, statusStartChecked,
138                             statusEndChecked)
```

```

135             markerClusterer = new MarkerClusterer(map, markers, {
136                 imagePath:
137                     "https://developers.google.com/maps/
138                         documentation/javascript/examples/
139                         markerclusterer/m",
140             });
141         }
142         if (isHeatMap) {
143             map.setMapTypeId(google.maps.MapTypeId.ROADMAP);
144             if (!heatMap || heatMap.getMap() === null) {
145                 heatMap = setHeatMap(res.data.data, map,
146                     statusStartChecked, statusEndChecked)
147             }
148         }
149     });
150 }
151 });

```

---

Kode A.3: Kiri Histori Model

```

1 const fs = require("fs")
2 const {csvToObject, buildFilter, filterData} = require("./Utils");
3
4 class KiriHistory {
5     constructor() {
6         this.fetchData();
7     }
8
9     fetchData = () => {
10         fs.readFile(__dirname + "/data/KIRIStatistics.csv", "utf8", ((err, data
11             ) => {
12                 if (err === null) {
13                     let arrCSV = data.split("\n")
14                     //remove header
15                     arrCSV.shift();
16                     // console.log(arrCSV);
17                     this.data = arrCSV.map(csvToObject).filter(item => item !==
18                         undefined);
19                 }
20             }))
21     }
22     //trigger heroku
23     filterData = (filterParams) => {
24         this.fetchData();
25         let query = buildFilter(filterParams);
26         this.data = filterData(this.data, query);
27         // selasa 21:00 data cuman 1?
28         // console.log(this.data)
29         return this.data;
30     }
31 }
32 module.exports = KiriHistory;

```

---

Kode A.4: Index Page

```

1 <!doctype html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">

```

```
5      <meta name="viewport"
6          content="width=device-width, user-scalable=no, initial-scale=1.0,
7              maximum-scale=1.0, minimum-scale=1.0">
8      <meta http-equiv="X-UA-Compatible" content="ie=edge">
9
10     <link rel="stylesheet" href="css/main.css">
11 </head>
12 <body>
13
14 <div class="wrapper d-flex align-items-stretch">
15     <nav id="sidebar">
16
17         <div class="p-4 pt-5">
18             <p>Results : <span id="result-length">0</span></p>
19             <h1><a href="index.html" class="logo" id="logo">KIRI</a></h1>
20             <li>
21                 <p>Start / Finish</p>
22                 <div>
23                     <select style="width: 100% ; height: 3rem" id="start-finish">
24                         <option value="start">Start</option>
25                         <option value="end">End</option>
26                     </select>
27                 </div>
28             </li>
29
30             <li>
31                 <p>Days</p>
32                 <select style="width: 100% ; height: 8rem" id="day" multiple>
33                     <option selected value="0">Monday</option>
34                     <option value="1">Tuesday</option>
35                     <option value="2">Wednesday</option>
36                     <option value="3">Thursday</option>
37                     <option value="4">Friday</option>
38                     <option value="5">Saturday</option>
39                     <option value="6">Sunday</option>
40                 </select>
41
42             </li>
43             <!-- ini jam -->
44             <li>
45                 <p>Hours</p>
46                 <select style="width: 100% ; height: 27rem" id="hour" multiple>
47                     <option selected value="0">0:00</option>
48                     <option value="1">1:00</option>
49                     <option value="2">2:00</option>
50                     <option value="3">3:00</option>
51                     <option value="4">4:00</option>
52                     <option value="5">5:00</option>
53                     <option value="6">6:00</option>
54                     <option value="7">7:00</option>
55                     <option value="8">8:00</option>
56                     <option value="9">9:00</option>
57                     <option value="10">10:00</option>
58                     <option value="11">11:00</option>
59                     <option value="12">12:00</option>
60                     <option value="13">13:00</option>
61                     <option value="14">14:00</option>
62                     <option value="15">15:00</option>
63                     <option value="16">16:00</option>
64                     <option value="17">17:00</option>
```

```
66          <option value="18">18:00</option>
67          <option value="19">19:00</option>
68          <option value="20">20:00</option>
69          <option value="21">21:00</option>
70          <option value="22">22:00</option>
71          <option value="23">23:00</option>
72      </select>
73
74
75      </li>
76      <li>
77          <p>Mode </p>
78
79          <div class="d-flex justify-content-between">
80              <div class="form-check">
81                  <input class="form-check-input" type="radio"
82                      name="mode"
83                      checked
84                      id="marker-cluster">
85                  <label class="form-check-label">
86                      Marker Clustering
87                  </label>
88              </div>
89              <div class="form-check">
90                  <input class="form-check-input" type="radio" name="mode"
91                      " id="heat-map">
92                  <label class="form-check-label">
93                      Heat Map
94                  </label>
95              </div>
96          </div>
97
98
99      </div>
100
101
102      <div class="d-flex justify-content-center mb-5">
103          <input type="button" id="send-btn" value="Visualize" class="btn
104              btn-sm btn-primary">
105      </div>
106
107  </nav>
108
109  <!-- Page Content -->
110  <div id="content" class="p-4 p-md-5 pt-5">
111      <div style="height: 100vh">
112          <div id="map"></div>
113      </div>
114  </div>
115 </div>
116
117
118 <script
119     src="https://maps.googleapis.com/maps/api/js?
120         key=AIzaSyCeLYwPPPpDiL1RjRscCPShSCBJAWamzu-I&callback=initMap&
121         libraries=visualization&v=weekly"
122         defer></script>
123
124 <script src="https://cdnjs.cloudflare.com/ajax/libs/axios/0.21.0/axios.min.js"
125         integrity="sha512-DZqqY3Pi0vTP9HkjIWgj06ouCbq+dxqWoJZ/Q+
```

---

```
125      zPYNHmlnI2dQnbJ5bxAHpAMw+LXRm4D72EIRXzvcHQte8/VQ=="  
126  crossorigin="anonymous"></script>  
126  <script src="https://unpkg.com/@googlemaps/markerclustererplus/dist/  
127    index.min.js"></script>  
127  <script src="js/maps.js"></script>  
128  
129  </body>  
130  </html>
```

---