

## SKRIPSI

PORTING PHP MENJADI PLAY FRAMEWORK (STUDI  
KASUS : KIRI *FRONT-END*)



STEVEN SUTANA

NPM: 2012730046

PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS  
UNIVERSITAS KATOLIK PARAHYANGAN  
«tahun»



**UNDERGRADUATE THESIS**

**PORTING PHP TO PLAY FRAMEWORK(CASE STUDY : KIRI  
*FRONT-END*)**



**STEVEN SUTANA**

**NPM: 2012730046**

**DEPARTMENT OF INFORMATICS  
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES  
PARAHYANGAN CATHOLIC UNIVERSITY  
«tahun»**



## **LEMBAR PENGESAHAN**

### **PORTING PHP MENJADI PLAY FRAMEWORK (STUDI KASUS : KIRI *FRONT-END*)**

**STEVEN SUTANA**

**NPM: 2012730046**

Bandung, «tanggal» «bulan» «tahun»

Menyetujui,

**Pembimbing Utama**

**Pembimbing Pendamping**

**Pascal Alfadian, M.Com.**  
**Ketua Tim Penguji**

**«pembimbing pendamping/2»**  
**Anggota Tim Penguji**

«penguji 1»

«penguji 2»

Mengetahui,

**Ketua Program Studi**

**Mariskha Tri Aditia, PDEng**



## **PERNYATAAN**

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

### **PORTING PHP MENJADI PLAY FRAMEWORK (STUDI KASUS : KIRI *FRONT-END*)**

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

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

Dinyatakan di Bandung,  
Tanggal «tanggal» «bulan» «tahun»

Meterai

Steven Sutana  
NPM: 2012730046



## **ABSTRAK**

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia» Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetur.

**Kata-kata kunci:** «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Indonesia»



## **ABSTRACT**

«Tuliskan abstrak anda di sini, dalam bahasa Inggris» Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetur.

**Keywords:** «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Inggris»



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



## KATA PENGANTAR

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

Bandung, «bulan» «tahun»

Penulis



## DAFTAR ISI

<b>KATA PENGANTAR</b>	<b>xv</b>
<b>DAFTAR ISI</b>	<b>xvii</b>
<b>DAFTAR GAMBAR</b>	<b>xix</b>
<b>DAFTAR TABEL</b>	<b>xx</b>
<b>1 PENDAHULUAN</b>	<b>1</b>
1.1 Latar Belakang . . . . .	1
1.2 Rumusan Masalah . . . . .	1
1.3 Tujuan . . . . .	2
1.4 Batasan Masalah . . . . .	2
1.5 Metode Penelitian . . . . .	2
1.6 Sistematika Penulisan . . . . .	2
<b>2 LANDASAN TEORI</b>	<b>3</b>
2.1 Play Framework . . . . .	3
2.1.1 Struktur Play Framework . . . . .	3
2.1.2 Body Parsers . . . . .	6
2.1.3 Internationalization . . . . .	6
2.2 OpenLayers . . . . .	7
2.2.1 Bing Maps . . . . .	8
2.2.2 Draw Point . . . . .	8
2.3 Zurb Foundation . . . . .	9
2.3.1 Sistem Grid . . . . .	9
2.4 Chrome DevTools . . . . .	9
2.4.1 Elements . . . . .	10
2.4.2 Network . . . . .	11
2.4.3 Sources . . . . .	13
2.4.4 Timeline . . . . .	14
2.4.5 Profile . . . . .	14
<b>3 ANALISIS</b>	<b>17</b>
3.1 Analisis Sistem Kini . . . . .	17
3.1.1 Peta . . . . .	17
3.1.2 Form Samping . . . . .	19
3.1.3 Internationalization . . . . .	29
3.2 Analisis Sistem Usulan . . . . .	29
3.2.1 Form Samping . . . . .	29
3.2.2 Internationalization . . . . .	33
3.2.3 Tampilan . . . . .	34
3.2.4 JavaScript dan stylesheet . . . . .	35

3.3	Analisis Use Case . . . . .	35
3.3.1	Skenario Use Case . . . . .	36
3.4	Analisis Activity Diagram . . . . .	38
<b>DAFTAR REFERENSI</b>		<b>41</b>
<b>A</b>	<b>THE PROGRAM</b>	<b>43</b>
<b>B</b>	<b>THE SOURCE CODE</b>	<b>45</b>

## DAFTAR GAMBAR

2.1 Struktur Play Framework . . . . .	4
2.2 Contoh <i>Routes</i> . . . . .	4
2.3 Interaksi <i>Body Parsers</i> dengan <i>Request</i> . . . . .	6
2.4 Sistem <i>grid</i> kosong sebelum memulai desain. . . . .	9
2.5 Contoh pembagian sistem <i>grid</i> . . . . .	9
2.6 Panel Elements . . . . .	10
2.7 Panel Network . . . . .	11
2.8 Contoh Header . . . . .	12
2.9 Contoh peninjauan sumber daya tersedia . . . . .	12
2.10 Contoh peninjauan sumber daya tidak tersedia . . . . .	12
2.11 Contoh Response . . . . .	12
2.12 Contoh Cookies . . . . .	13
2.13 Panel Sources dengan menyalakan Conditional <i>breakpoints</i> . . . . .	13
2.14 Panel Timeline saat melakukan pencarian rute . . . . .	14
2.15 Contoh CPU <i>profiler</i> . . . . .	15
2.16 Contoh Heap <i>profiler</i> . . . . .	15
3.1 Halaman Utama KIRI . . . . .	17
3.2 Peta pada KIRI . . . . .	18
3.3 Form pada KIRI . . . . .	19
3.4 Dropdown Menu Kota pada KIRI . . . . .	20
3.5 Dropdown Menu Bahasa pada KIRI . . . . .	22
3.6 Input User(Nama Tempat) . . . . .	23
3.7 Input User(Klik pada peta) . . . . .	23
3.8 Contoh Pencarian Rute pada KIRI . . . . .	26
3.9 Contoh Rute Alternatif pada KIRI . . . . .	27
3.10 Use Case Diagram KIRI . . . . .	36
3.11 Activity Diagram KIRI . . . . .	39
A.1 Interface of the program . . . . .	43

## **DAFTAR TABEL**

## BAB 1

### PENDAHULUAN

#### 1.1 Latar Belakang

KIRI [1] adalah sebuah aplikasi yang membantu pengguna dalam menggunakan kendaraan umum. Peran KIRI sangat sederhana, yaitu memberitahu dimana lokasi sekarang dan kemana lokasi tujuan, lalu KIRI akan memberitahu bagaimana cara sampai ke lokasi tujuan dengan menggunakan kendaraan umum.

Kode KIRI [2] menggunakan bahasa PHP. Bahasa PHP [3] merupakan bahasa *scripting* yang cocok untuk pengembangan halaman *web*. Tetapi menurut peneliti, bahasa PHP tidak cocok untuk proyek besar. Masalah yang sering dijumpai pada bahasa PHP adalah tidak ada *type safety*.

*Type safety*<sup>1</sup> adalah fitur keamanan untuk mencegah kesalahan tipe data. Kesalahan tipe data dapat disebabkan oleh perbedaan tipe untuk konstanta program, variabel, dan fungsi. Sebagai contoh tipe data yang dibutuhkan berupa Float tetapi dalam program tipe data yang dimasukkan berupa Integer. Beberapa bahasa pemrograman terdapat fitur *type safety*. Java mendukung Type Safety.

Play Framework adalah *framework* untuk aplikasi web dengan menggunakan bahasa Java dan Scala. Play Framework mempunyai antarmuka yang sederhana, nyaman, fleksibel, dan kuat. Play Framework menerapkan konsep MVC, yaitu Model, View, dan Controller[4].

*Porting* adalah proses adaptasi perangkat lunak yang awalnya tidak ditujukan untuk dieksekusi pada lingkungan tertentu. Istilah *porting* digunakan ketika mengacu pada perubahan yang dibuat ketika tidak kompatibel dengan lingkungan.

Pengembangan yang akan dilakukan adalah melakukan *porting* kode KIRI (PHP) menjadi Play Framework agar struktur kode KIRI menjadi rapih dan bahasa yang digunakan adalah bahasa Java. Dengan demikian, penulis bermaksud membuat proyek tugas akhir dengan judul “Porting PHP menjadi Play Framework (Studi Kasus: KIRI *Front-End*)”

#### 1.2 Rumusan Masalah

- Bagaimana memahami dan menganalisis kode KIRI yang sudah ada?
- Bagaimana melakukan porting kode KIRI *Front-End Server Side*(PHP) menjadi Play Framework (Java) ?

---

<sup>1</sup> [https://en.wikipedia.org/wiki/Type\\_safety](https://en.wikipedia.org/wiki/Type_safety), diakses 27 Oktober 2015

### 1.3 Tujuan

- Memahami dan menganalisis kode KIRI.
- Menjadikan kode KIRI *Front-End Server Side*(PHP) menjadi Play Framework (Java).

### 1.4 Batasan Masalah

1. Play Framework yang digunakan adalah versi 2.4.3.
2. Kode KIRI yang sudah ada diambil dari Github pascalalfadian[2].

### 1.5 Metode Penelitian

Berikut adalah metode penelitian yang digunakan dalam pembuatan skripsi ini:

1. Melakukan studi literatur tentang metode yang berkaitan dengan kode PHP dan Java (Play Framework).
2. Memahami dan melakukan analisis kode KIRI yang sudah ada.
3. Merancang dan mengimplementasikan kode KIRI yang sudah ada menjadi Play Framework.
4. Melakukan pengujian dan eksperimen.
5. Membuat dokumen skripsi.

### 1.6 Sistematika Penulisan

Setiap bab dalam penulisan ini memiliki sistematika yang dijelaskan ke dalam poin-poin sebagai berikut:

1. Bab 1: Pendahuluan, yaitu membahas tentang latar belakang, rumusan masalah, tujuan, batasan masalah, metode penelitian dan sistematika penulisan.
2. Bab 2: Dasar Teori, yaitu membahas mengenai teori-teori yang mendukung berjalannya skripsi ini yang berisi tentang penggunaan Play Framework.
3. Bab 3: Analisis, yaitu membahas mengenai analisis masalah yang berisi tentang kode KIRI *Front-End Server Side* serta melakukan *porting* kode KIRI *Front-End Server Side* menjadi Play Framework.

## BAB 2

### LANDASAN TEORI

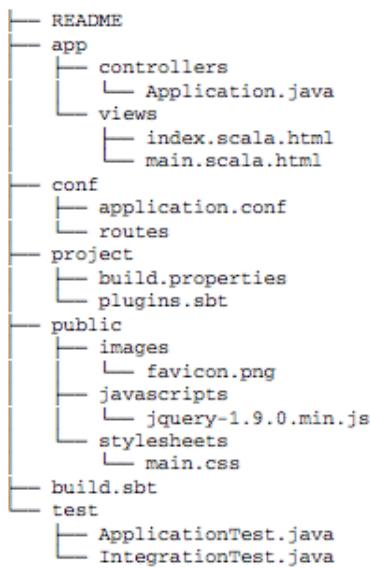
#### 2.1 Play Framework

##### 2.1.1 Struktur Play Framework

Play Framework [4] merupakan *framework* untuk aplikasi web dengan menggunakan bahasa Java dan Scala. Play Framework tidak sepenuhnya menggunakan bahasa Java, tetapi ada juga bahasa Scala. Terdapat bahasa Scala bukan berarti harus mempelajari bahasa Scala karena dalam Play 2 dilengkapi dengan Java API yang komplit, memberikan opsi untuk memilih bahasa pemrograman yang cocok. Play Framework mempunyai antarmuka yang sederhana, nyaman, fleksibel, dan kuat. Referensi yang digunakan membahas Play Framework 2.2, sedangkan versi Play Framework yang dipakai dalam penelitian ini adalah versi 2.4. Ada sedikit perbedaan sintaks yang akan dijelaskan pada bagian yang berbeda. Beberapa fitur utama yang membuat Play Framework produktif dan penggunaan yang nyaman:

1. Penggunaan Play Framework sederhana.
2. Konfigurasi skema URL aplikasi deklaratif.
3. Pemetaan type-safety.
4. Play Framework menyediakan contoh sintaks type-safety.
5. Arsitektur yang mencakup teknologi HTML5.
6. Kode langsung aktif berubah ketika memuat kembali halaman web.
7. Fitur *full-stack web-framework*, termasuk *persistence*, keamanan, dan *internationalization*. *Persistence* adalah ide yang menggunakan koneksi TCP yang sama untuk mengirim dan menerima beberapa HTTP *requests/responses* tanpa membuka TCP baru untuk setiap *requests/responses* dengan tujuan untuk meningkatkan kinerja HTTP.
8. Mendukung aplikasi *event-driven* dan dinamis.

Play Framework memiliki struktur yang dapat dilihat pada gambar 2.1.



Gambar 2.1: Struktur Play Framework

### Direktori *conf*.

Konfigurasi Play Framework terdapat pada direktori *conf*. Dalam direktori *conf*, terdapat file *application.conf* dan *routes*. File *application.conf* mengandung informasi data konfigurasi aplikasi, seperti *logging*, koneksi basis data, dan port berapa server berjalan. File *routes* menentukan *routes* aplikasi, yaitu pemetaan dari URL HTTP ke kode aplikasi. Setiap *routes* memiliki tiga bagian, yaitu HTTP *method*, URL *path*, dan *action method*. HTTP *method* merupakan metode yang dipakai dalam pengiriman HTTP. URL *path* adalah URL yang dipakai untuk mengakses halaman. *Action method* merupakan metode yang dipanggil ketika mengakses halaman pada URL *path*. Sebagai contoh dapat dilihat pada 2.2, HTTP *method* yang dipakai pada URL /list adalah HTTP *method* GET dan akan memanggil *method* list pada kelas Products di controllers.



Gambar 2.2: Contoh *Routes*

### Direktori *public*

Direktori *public* mengandung semua sumber daya yang disediakan langsung tanpa melalui proses terlebih dahulu. Direktori *public* biasanya mengandung file gambar, *stylesheets*, JavaScript, dan halaman statis HTML. Contoh direktori *public* dapat dilihat pada gambar 2.1.

### Direktori *app*.

Direktori *app* merupakan direktori utama pada aplikasi. Direktori *app* berisi kode aplikasi dan berbagai kebutuhan untuk menyusun aplikasi, seperti sumber file Java dan file *template*. Contoh

direktori *app* pada saat pertama kali membuat aplikasi Play dapat dilihat pada gambar 2.1.

Dalam *controller*, terdapat file Application.java yang berisi kode Java untuk memuat halaman *view*. *Controller* adalah kelas untuk menerima HTTP *request* dan mengembalikan nilai dari HTTP *request* berupa *view*. Ada dua file template, yaitu index.scala.html dan main.scala.html yang berfungsi untuk menentukan halaman HTML yang akan dimuat. Semua konten yang dihasilkan di server dan dikirimkan ke klien seperti halaman HTML disebut *view*. *View* dapat menerima parameter yang didefinisikan pada template *view*. Contoh *view* dengan menerima parameter berupa String dapat dilihat pada kode listing 2.1. *Method* pada *Controller* menghasilkan hasil berupa *Result* yang berupa *view* dengan memberi parameter "Hello World". *Method* pada *controller* dan *view* dihubungkan melalui pendefinisian pada *routes*. Sebagai contoh pada kode listing 2.2, *method ok* membangun HTTP *response* yang mengandung *response body* sebagai hasil dari *template list*. *Method ok* menerima *parameter* berupa "Hello World". Pada [4], hasil *Result* dalam **static**, tetapi pada Play Framework 2.4 **static** dihilangkan.

Listing 2.1: Contoh View

```

1 @(title: String)
2
3     <!DOCTYPE html>
4
5     <html>
6         <head>
7             <title>@title</title>
8             <link rel="stylesheet" media="screen" href="@routes.Assets.at("stylesheets/main.css")">
9             <link rel="shortcut-icon" type="image/png" href="@routes.Assets.at("images/favicon.png")">
10            <script src="@routes.Assets.at("javascripts/jquery-1.9.0.min.js")" type="text/javascript"></
11               script>
12
13             </head>
14             <body>
15                 @title
16             </body>
17         </html>
```

Listing 2.2: Contoh Controller

```

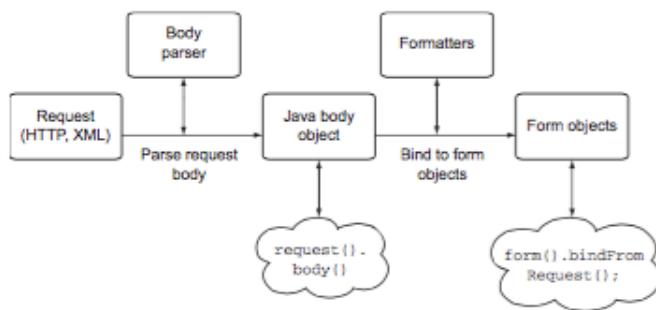
1 public Result index() {
2     return ok(index.render("Hello_World"));
3 }
```

Beberapa alternatif hasil *Results*, yaitu:

1. **Method ok** Method yang berfungsi untuk mengembalikan *Result* dengan kode *HTTP Response* 200 atau sukses dalam mengirim **HTTP Request**.
2. **Method notFound** Method yang berfungsi untuk mengembalikan *Result* dengan kode *HTTP Response* 404 atau tidak ada halaman yang dituju.
3. **Method badRequest** Method yang berfungsi untuk mengembalikan *Result* dengan kode *HTTP Response* 400 atau adanya kesalahan masukan pengguna.
4. **Method internalServerError** Method yang berfungsi untuk mengembalikan *Result* dengan kode *HTTP Response* 500 atau adanya kesalahan pada server.
5. **Method status** Method yang berfungsi untuk mengembalikan *Result* dengan kode *HTTP Response* yang dapat ditentukan sendiri beserta pesannya.
6. **Method redirect** Method yang berfungsi untuk mengembalikan *Result* untuk mengalihkan halaman.

### 2.1.2 Body Parsers

*Body parsers* bertugas untuk melakukan pemetaan *request body* menjadi objek. Setiap *action method* POST dan PUT mengandung *body*. Jumlah *body* dapat satu atau banyak, dan dapat berupa XML, JSON, data biner, atau dapat berupa apapun sesuai Content-Type pada *header request*. *Body parsers* akan menguraikan *body* menjadi objek Java. *Body parsers* mengubah *request* menjadi objek yang dapat digunakan oleh komponen Play. Karena *body* JSON dan *body* XML berbeda penguraiannya, Play menggunakan *body parsers* yang berbeda pula implementasinya. Berbeda Content-Type pada *header request*, *body parsers* spesifik dapat mengubah data yang masuk menjadi sesuatu yang dapat dimengerti oleh Play. Ilustrasi dapat dilihat pada gambar 2.3.



Gambar 2.3: Interaksi *Body Parsers* dengan *Request*

### 2.1.3 Internationalization

Pengguna aplikasi mungkin berasal dari beda negara dan bahasa, juga punya format yang berbeda untuk tanggal, angka, dan waktu. Kombinasi dari bahasa aturan format disebut *locale*. Adaptasi program untuk berbeda *locale* disebut *internationalization (i18n)* atau *localization (l10n)*. Perbedaan *internationalization* dan *localization* adalah *internationalization* melakukan *refactor* untuk menghapus kode lokal dari aplikasi, sedangkan *localization* membuat versi lokal dari aplikasi. Program yang sudah diproses Internationalization mempunyai karakteristik:

- Dengan penambahan data lokalisasi, eksekusi yang sama dapat dijalankan di seluruh dunia.
- Unsur tekstual, seperti pesan status dan komponen GUI, tidak ada *hard-code* dalam program. Sebaliknya, pesan status dan komponen GUI disimpan di luar *source code* dan diambil secara dinamis.
- Dengan adanya bahasa baru, program tidak perlu dikompilasi ulang.
- Data culturally-dependent, seperti tanggal dan mata uang, tampil dalam format yang sesuai dengan wilayah pengguna.
- Internationalization dapat dilakukan proses lokalisasi dengan cepat.

Untuk mendukung beberapa bahasa dalam Play, membuat kunci pesan yang dipetakan pesan sesungguhnya pada file pesan. Pesan ini dimuat dalam file messages.LANG dimana LANG merupakan bahasa yang dipakai. Messages.LANG disimpan dalam direktori conf. Sebagai contoh,

terdapat dua bahasa yang dipakai pada aplikasi, yaitu Bahasa Inggris dan Bahasa Indonesia seperti pada kode listing 2.3 dan 2.4

Listing 2.3: Contoh messages.en untuk i18n

```

1 from = From:
2 ph_from = e.g. Stasiun
3 ph_to = e.g. Monas, Jakarta
4 find = Find!
5 to = To:
```

Listing 2.4: Contoh messages.id untuk i18n

```

1 from = Dari:
2 ph_from = misal: Stasiun
3 ph_to = misal: Monas, Jakarta
4 find = Cari!
5 to = Ke:
```

Play harus mengetahui bahasa apa saja yang ada pada aplikasi sesuai dengan file messages.LANG. Untuk itu, daftarkan bahasa pada application.conf seperti pada kode listing 2.5. Pada [4], konfigurasi bahasa dalam satu String. Tetapi pada Play Framework 2.4 konfigurasi bahasa dalam bentuk array of String.

Listing 2.5: Konfigurasi Bahasa i18n

```

1 ...
2 # The application languages
3 # ~~~~~
4 play.i18n.langs = [ "en", "id" ]
5 ...
```

## 2.2 OpenLayers

OpenLayers [5] merupakan *library* yang memiliki performa tinggi dan fitur yang dikemas untuk kebutuhan menampilkan peta menggunakan JavaScript. Dalam pengembangan aplikasi yang menggunakan fitur peta, tugas yang paling penting dan utama adalah membuat peta tersebut. Peta menjadi inti untuk menambahkan dan menampilkan data. Fitur yang terdapat pada OpenLayers adalah:

- *Tiled Layers*

OpenLayers dapat menggunakan banyak *map provider*, seperti OSM, Bing, MapBox, Stamen, MapQuest, dan berbagai sumber lain yang dapat ditemukan. Dengan menggunakan OpenLayers, tidak perlu menulis ulang kode yang sudah ada dan dapat mengganti kapanpun sumber *map provider* yang ingin digunakan.

- *Vector Layers*

OpenLayers dapat mengubah data vektor dari berbagai tipe sumber, seperti GeoJSON, TopoJSON, KML, dan GML.

- Cepat dan Siap untuk Perangkat *Mobile*

OpenLayers mendukung perangkat *mobile*. OpenLayers dapat membangun profil kustom yang berisi komponen yang dibutuhkan saja.

- Mudah menyesuaikan peta dan *cutting edge*

OpenLayers menyesuaikan peta WebGL, Canvas 2D, dan semua kelebihan dari HTML 5. Atur tampilan peta dengan mengubah langsung CSS.

Modul OpenLayers yang dipakai dalam penelitian ini adalah:

- Bing Maps untuk menampilkan peta menggunakan Bing. KIRI menggunakan *map provider* Bing Maps sebagai peta pada halaman utama KIRI.
- Draw untuk menggambar poin pada peta. Saat peta KIRI menangkap *event mouseclick*, muncul poin yang berupa kustom gambar pada peta KIRI sebagai asal tempat dan tujuan.

### 2.2.1 Bing Maps

Peta yang digunakan merupakan Bing Maps dari Microsoft. Jika ingin menggunakan Bing Maps pada Openlayers, harus mendapatkan kunci yang didapat dari Bing Maps serta memilih tipe peta yang akan ditampilkan. Sebagai contoh pada kode listing 2.6.

Listing 2.6: Penggunaan Bing Maps pada Openlayers

```

1 var mapLayer = new ol.layer.Tile(
2 {
3     source : new ol.source.BingMaps(
4         {
5             key : 'AuV7xD6_UMiQ5BLoZr0xkpjLpzWqMT55772Q8XtLIQeuDebHPKiNXSiZXxEr1GA',
6             imagerySet : 'Road'
7         })
8 });

```

### 2.2.2 Draw Poin

Pada peta KIRI, pengguna dapat memilih tempat awal dan tujuan dengan memasukkan nama tempat atau memilih tempat pada peta. Pada Openlayers, disediakan fitur untuk menangani kursor klik pengguna. Pertama menentukan gambar yang akan ditampilkan pada peta seperti pada kode listing 2.7. Setelah itu, menggambar pada peta seperti pada figur 2.8.

Listing 2.7: Menentukan gambar yang akan dipakai pada peta

```

1
2
3 var markers = {start: null, finish: null};
4
5 var inputVectorSource = new ol.source.Vector();
6
7 markers[ 'start' ] = new ol.Feature(
8 {
9     geometry: new ol.geom.Point(event.coordinate)
10})
11
12 markers[ 'start' ].setStyle(new ol.style.Style(
13 {
14     image: new ol.style.Icon({
15         src: '/assets/images/start.png',
16         anchor: [1.0, 1.0]
17     })
18 }));
19
20 inputVectorSource.addFeature(markers[ 'start' ]);

```

Listing 2.8: Menambahkan vektor pada peta

```

1
2 var map = new ol.Map({
3     target: 'map',
4     layers : [ mapLayer, new ol.layer.Vector({source: inputVectorSource}), new ol.layer.Vector({source:
5         resultVectorSource}) ],
6     view: new ol.View({
7         center: ol.proj.fromLonLat([107.60981, -6.91474]),
8         zoom: 12
9     });

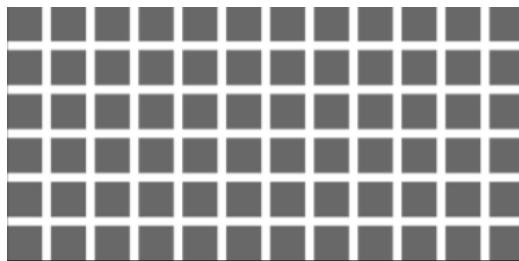
```

## 2.3 Zurb Foundation

Zurb Foundation [6] merupakan sebuah *toolkit* yang membantu untuk desain dan mengembangkan sebuah halaman *web*. Zurb Foundation menggunakan sistem *grid*, banyak komponen CSS serta JavaScript.

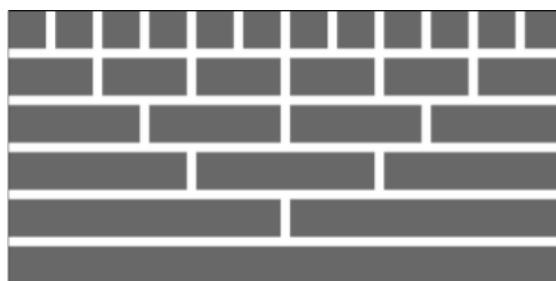
### 2.3.1 Sistem Grid

Sistem *grid* pada Zurb Foundation adalah semacam *spreadsheet*, graf atau tabel yang digunakan untuk mengatur tampilan HTML. Gambaran sistem *grid* dapat dilihat pada gambar 2.4.



Gambar 2.4: Sistem *grid* kosong sebelum memulai desain.

Setiap sel merupakan area konten yang dapat digabung dengan sel lain yang bersebelahan untuk memperbesar area konten. Sel yang digunakan pada Zurb Foundation adalah 12 sel pada setiap baris. Gambaran pembagian sel pada Zurb Foundation dapat dilihat pada gambar 2.5.



Gambar 2.5: Contoh pembagian sistem *grid*.

Komponen Foundation adalah kode CSS yang membantu untuk desain dan merepresentasikan konten halaman *web*. Dengan menggunakan komponen Foundation, tidak perlu desain sendiri dari awal. CSS pada Foundation juga dapat dikustomisasi sesuai dengan kebutuhan. Jika ingin menggunakan komponen Foundation, kita hanya perlu menggunakan tag class pada elemen yang diinginkan. Contoh penggunaan komponen Foundation dapat dilihat pada figur 2.9.

Listing 2.9: Menggunakan kelas yang sudah disediakan dari Foundation

```
1| 
```

## 2.4 Chrome DevTools

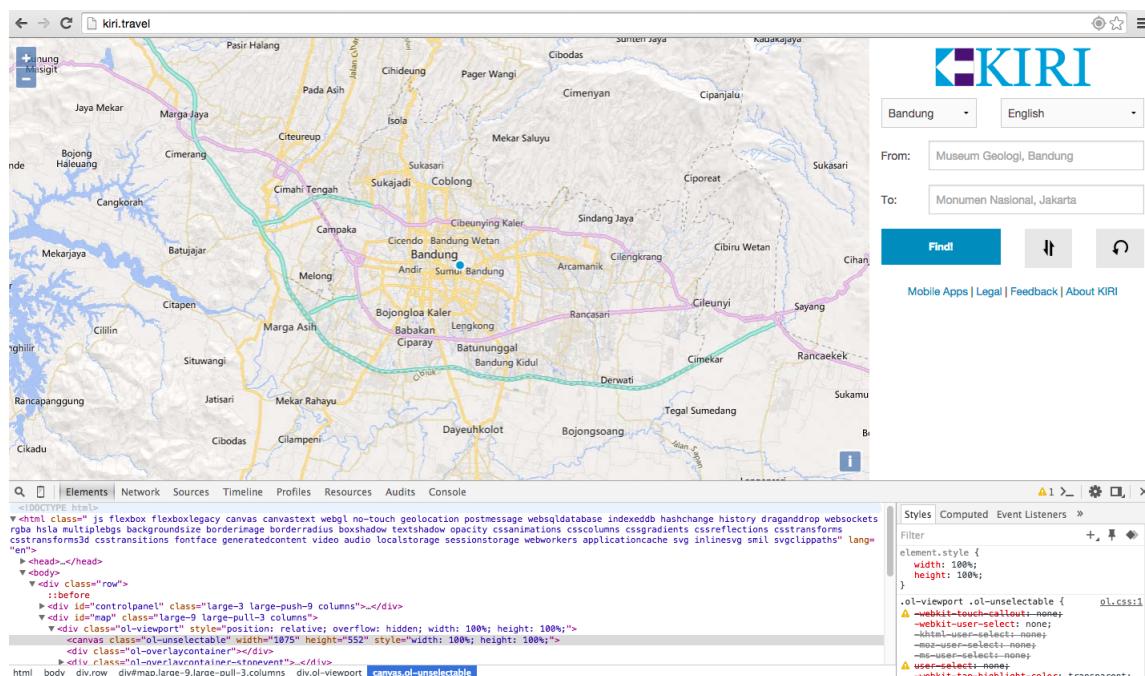
Chrome DevTools[7] merupakan perangkat untuk memperhatikan dan melakukan *debugging* halaman web yang terdapat pada *browser* Google Chrome. DevTools dapat digunakan secara efisien

untuk memeriksa tampilan, mengatur *breakpoints* JavaScript, dan optimasi kode. DevTools dapat diakses dengan melakukan klik kanan pada halaman web lalu klik periksa elemen. DevTools disusun dalam beberapa panel *task-orientated*. Beberapa panel tersebut adalah:

1. **Elements**, untuk memeriksa, melihat, dan mengubah tampilan halaman web.
2. **Network**, untuk memantau aktivitas jaringan pada halaman web secara *real-time*.
3. **Sources**, untuk melakukan *debugging* pada JavaScript dengan menentukan *breakpoints*.
4. **Timeline**, untuk merekam dan analisis aktivitas halaman web.
5. **Profiles**, untuk menggambarkan waktu eksekusi dan penggunaan memori dari halaman web.
6. **Resources**, untuk memeriksa sumber daya halaman web, seperti basis data, *cookies*, *cache*, gambar, dan tampilan halaman web.
7. **Console**, untuk mencatat informasi diagnostik pada proses pengembangan serta menyediakan *prompt shell* yang dapat digunakan untuk interaksi dengan dokumen dan DevTools.

#### 2.4.1 Elements

Panel Elements dapat memperlihatkan struktur halaman web dalam bentuk *Document Object Model* (DOM), dan dapat mengubah elemen DOM dengan cepat. DOM adalah struktur logis dokumen serta cara dokumen diakses dan diubah<sup>1</sup>. Sebagai contoh pada gambar 2.6, pemeriksaan elemen akan memperlihatkan dua bagian, yaitu DOM dan CSS yang digunakan pada DOM.



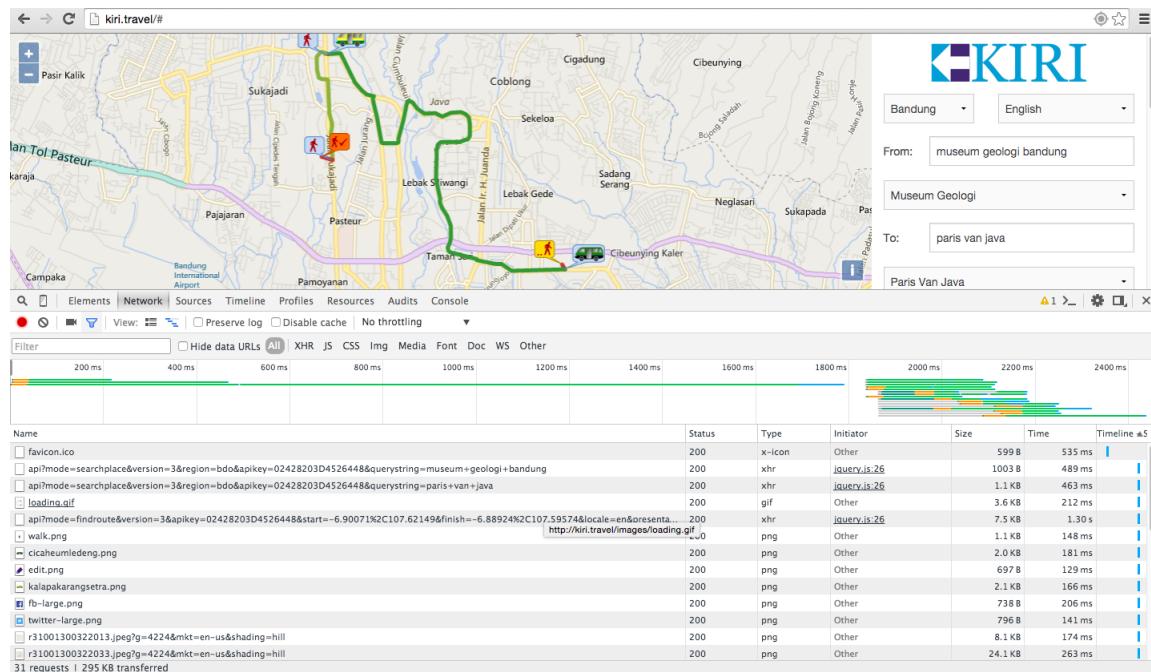
Gambar 2.6: Panel Elements

<sup>1</sup><http://www.w3.org/TR/DOM-Level-2-Core/introduction.html>, diakses 2 Oktober 2015

### 2.4.2 Network

Panel Network memberikan informasi tentang sumber daya yang diminta dan sumber daya yang diunduh melalui jaringan secara *real-time*. Panel Network juga memperlihatkan waktu yang dibutuhkan untuk permintaan sumber daya. Sebagai contoh pada gambar 2.7, saat melakukan pencarian rute, panel Network memperlihatkan apa saja sumber daya yang diperlukan serta waktu yang dibutuhkan pada proses tersebut. Tiap sumber daya pada panel Network terdapat kolom:

- **Name**, nama sumber daya.
- **Status**, kode status HTTP *request*.
- **Type**, tipe sumber daya.
- **Initiator**, asal dari sumber daya yang diminta.
- **Size**, ukuran sumber daya.
- **Time**, waktu yang dibutuhkan dalam permintaan sumber daya.



Gambar 2.7: Panel Network

Ketika sumber daya diklik, maka akan muncul bagian baru disamping sumber daya tersebut yang berisi kolom:

- **Header**

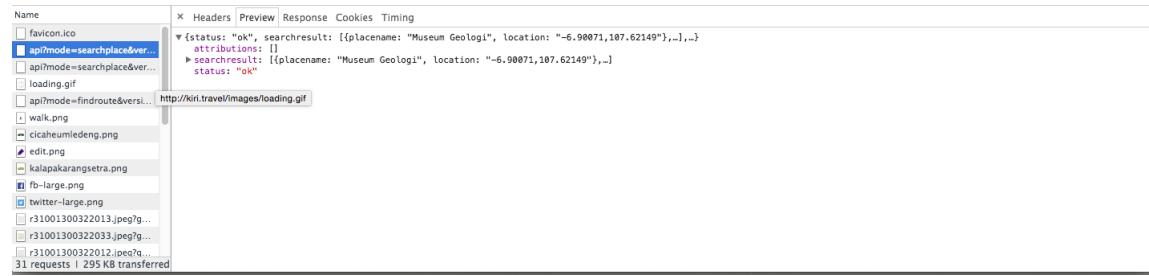
Header menampilkan *request URL*, *request method*, *status code*, *response headers*, *request headers*, dan *query string parameters* beserta nilainya.



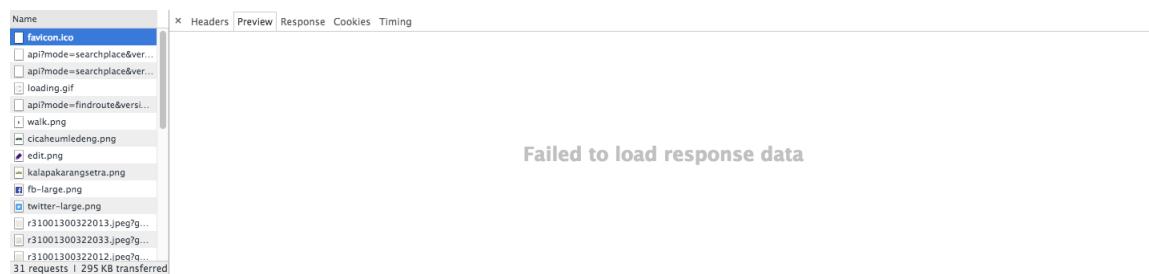
Gambar 2.8: Contoh Header

### • Preview

Preview menampilkan peninjauan sumber daya jika sumber daya tersebut tersedia. Gambar 2.9 menunjukkan adanya peninjauan sumber daya, sedangkan gambar 2.10 menunjukkan tidak ada peninjauan sumber daya.



Gambar 2.9: Contoh peninjauan sumber daya tersedia



Gambar 2.10: Contoh peninjauan sumber daya tidak tersedia

### • Response

Response menampilkan respon dari sumber daya yang dipilih. Gambar 2.11 menunjukkan respon dari sumber daya.

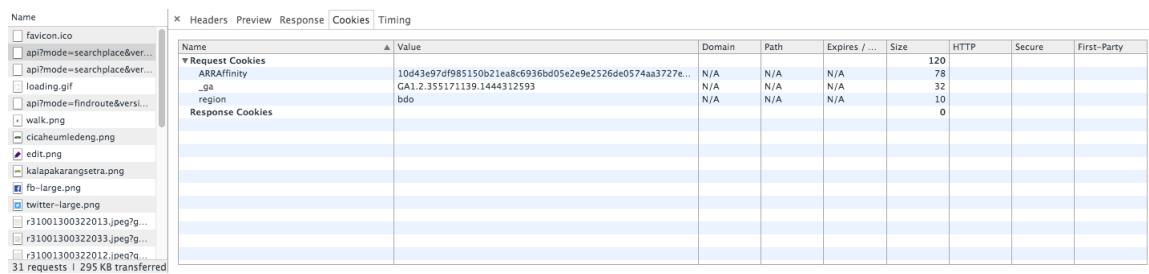


Gambar 2.11: Contoh Response

- **Cookies**

Cookies digunakan server web untuk menyimpan data pada *browser* klien. Kolom Cookies menampilkan seluruh *cookie* yang terdapat pada halaman web. Pada gambar 2.12 terdapat kolom:

- **Name**, nama *cookie*.
- **Value**, nilai *cookie*.
- **Domain**, asal *cookie*.
- **Path**, URL *cookie*.
- **Expires / Max-Age**, batas habis *cookie*.
- **Size**, ukuran *cookie*.
- **HTTP**
- **Secure**
- **First-Party**

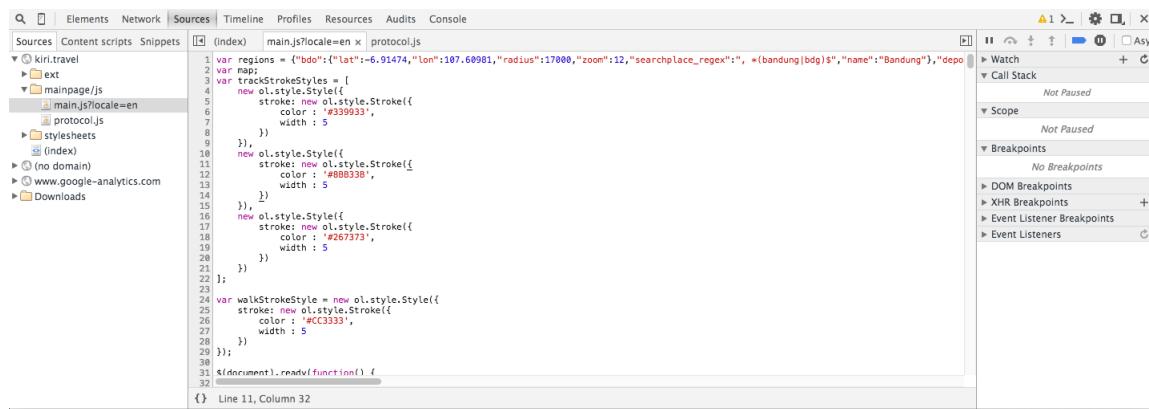


Name	Value	Domain	Path	Expires / ...	Size	HTTP	Secure	First-Party
ARRAffinity	10443e97df985150b21ea8cf6936bd05e2e9e2526de0574aa3727e...	N/A	N/A	N/A	120			
_ga	GA1.2.355171139.1444312593	N/A	N/A	N/A	78			
region	bdo	N/A	N/A	N/A	32			
Request Cookies					10			
Response Cookies					0			

Gambar 2.12: Contoh Cookies

### 2.4.3 Sources

Panel Sources memungkinkan untuk melakukan *debugging* JavaScript dengan menggunakan *breakpoints*<sup>2</sup>. Pengembang membutuhkan alat *debugging* untuk menemukan penyebab masalah dan memperbaikinya dengan cepat.

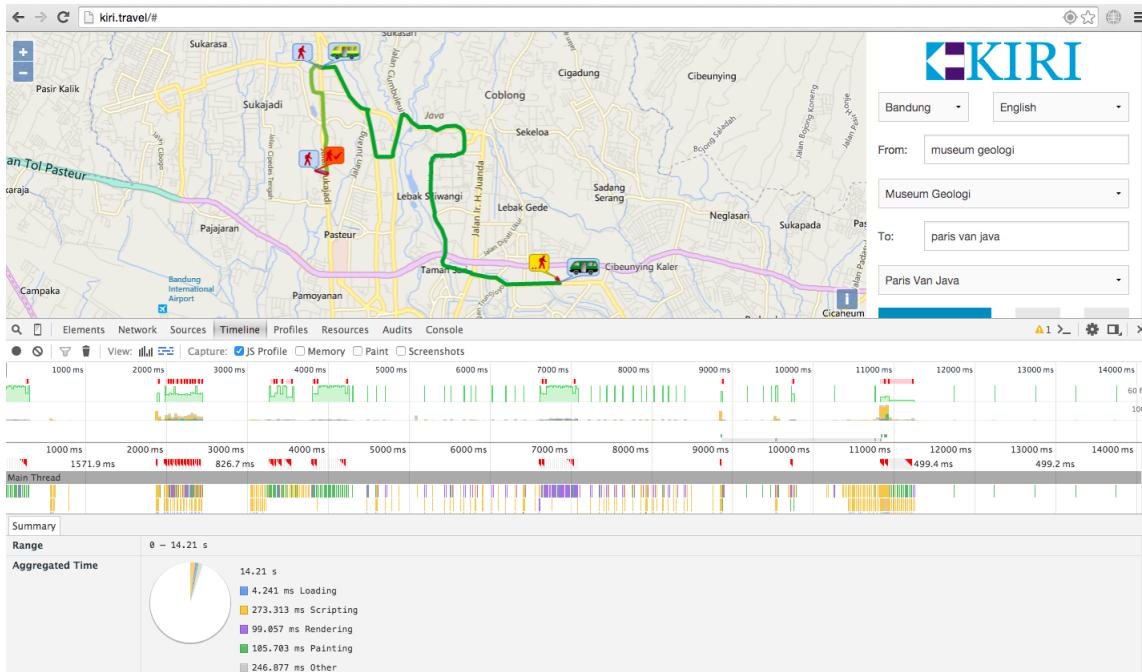


Gambar 2.13: Panel Sources dengan menyalakan Conditional *breakpoints*

<sup>2</sup>Terdapat dua cara untuk menambahkan *breakpoints*. Cara pertama adalah Manual *breakpoints*, yaitu mengatur *breakpoints* pada baris kode. Cara kedua adalah Conditional *breakpoints*, yaitu *breakpoints* secara otomatis muncul ketika suatu kondisi terpenuhi, misal ketika *on click*

#### 2.4.4 Timeline

Panel Timeline memberikan gambaran lengkap waktu yang dibutuhkan semua sumber daya yang dibutuhkan ketika memuat dan menggunakan halaman web. Sebagai contoh pada gambar 2.14, panel Timeline memberikan gambaran lengkap ketika melakukan pencarian rute.

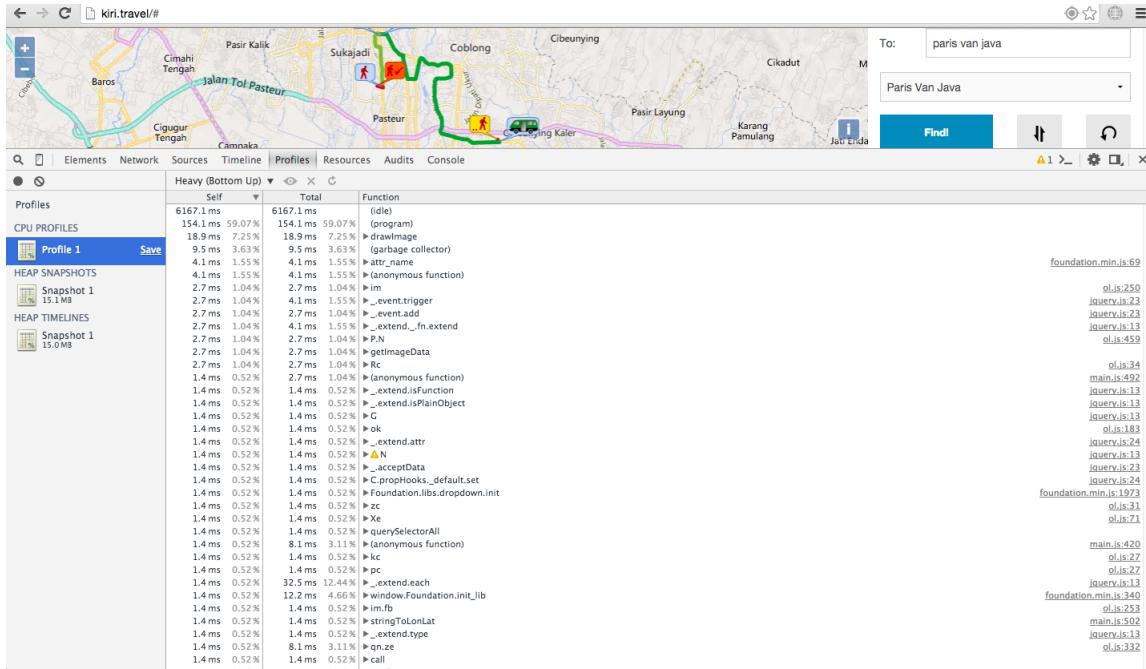


Gambar 2.14: Panel Timeline saat melakukan pencarian rute

#### 2.4.5 Profile

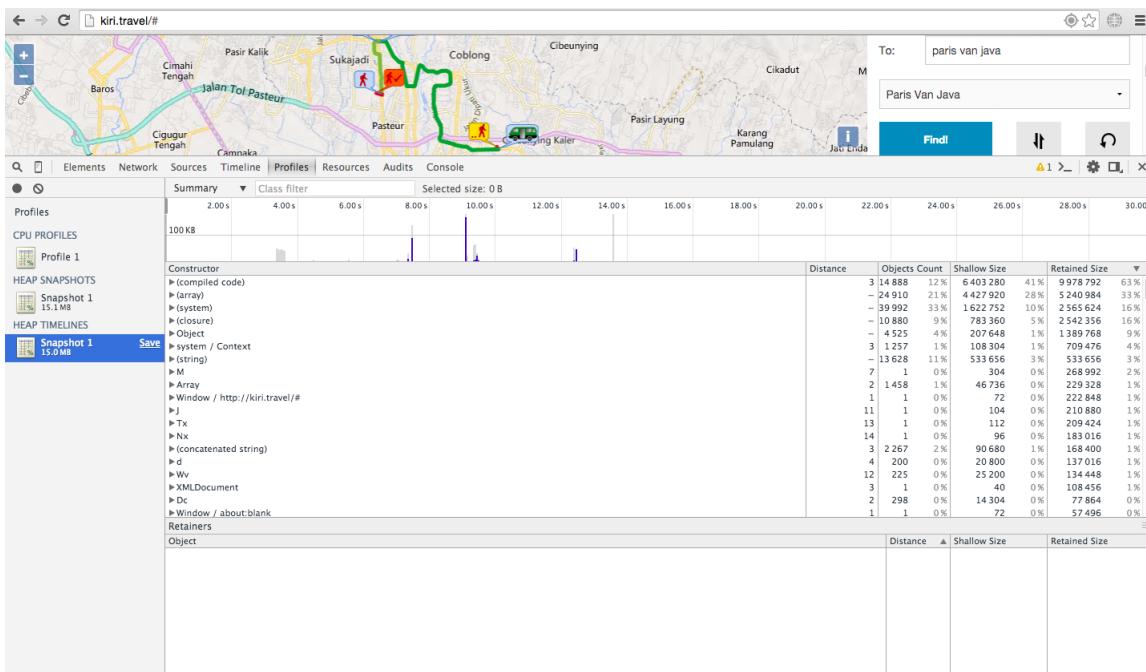
Panel Profile memberikan riwayat waktu pelaksanaan dan penggunaan memori dari halaman web. Profile yang tersedia adalah:

- CPU *profiler* menunjukkan waktu eksekusi yang dihabiskan oleh fungsi JavaScript. Gambar 2.15 menunjukkan waktu eksekusi yang dihabiskan oleh JavaScript.



Gambar 2.15: Contoh CPU profiler

- Heap profiler menunjukkan distribusi memori oleh JavaScript dan DOM yang berhubungan pada halaman web. Gambar 2.16 menunjukkan disribusi memori.



Gambar 2.16: Contoh Heap profiler

- JavaScript profiler menunjukkan dimana waktu eksekusi dihabiskan pada skrip.

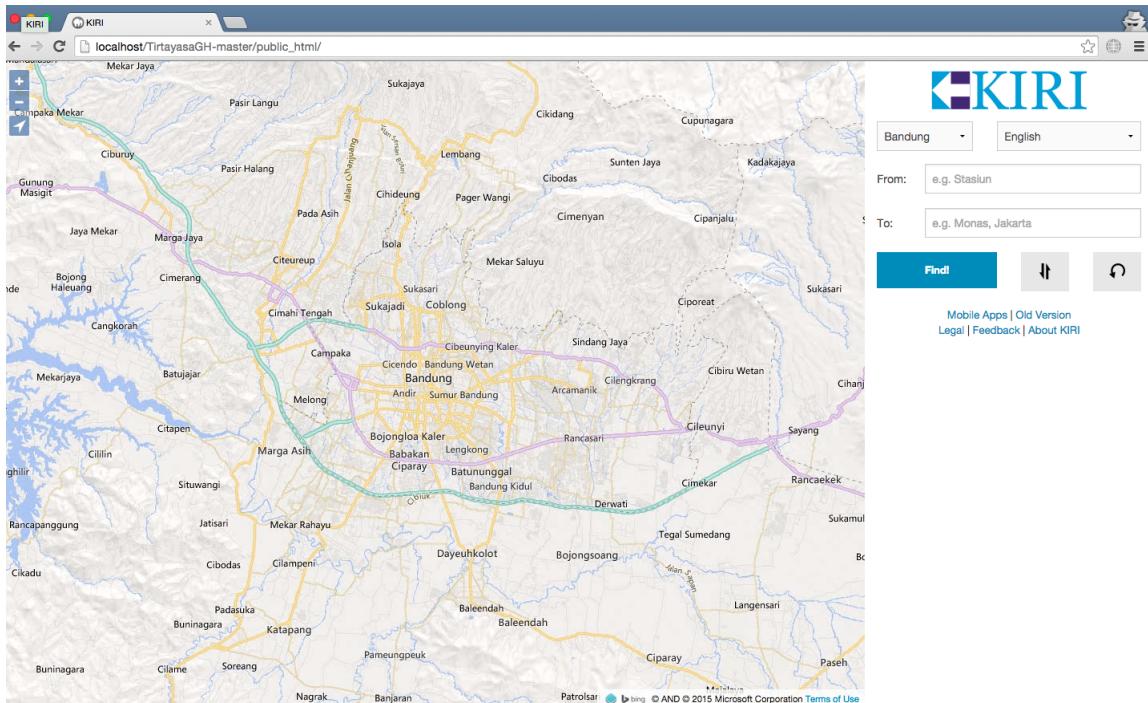


## BAB 3

### ANALISIS

#### 3.1 Analisis Sistem Kini

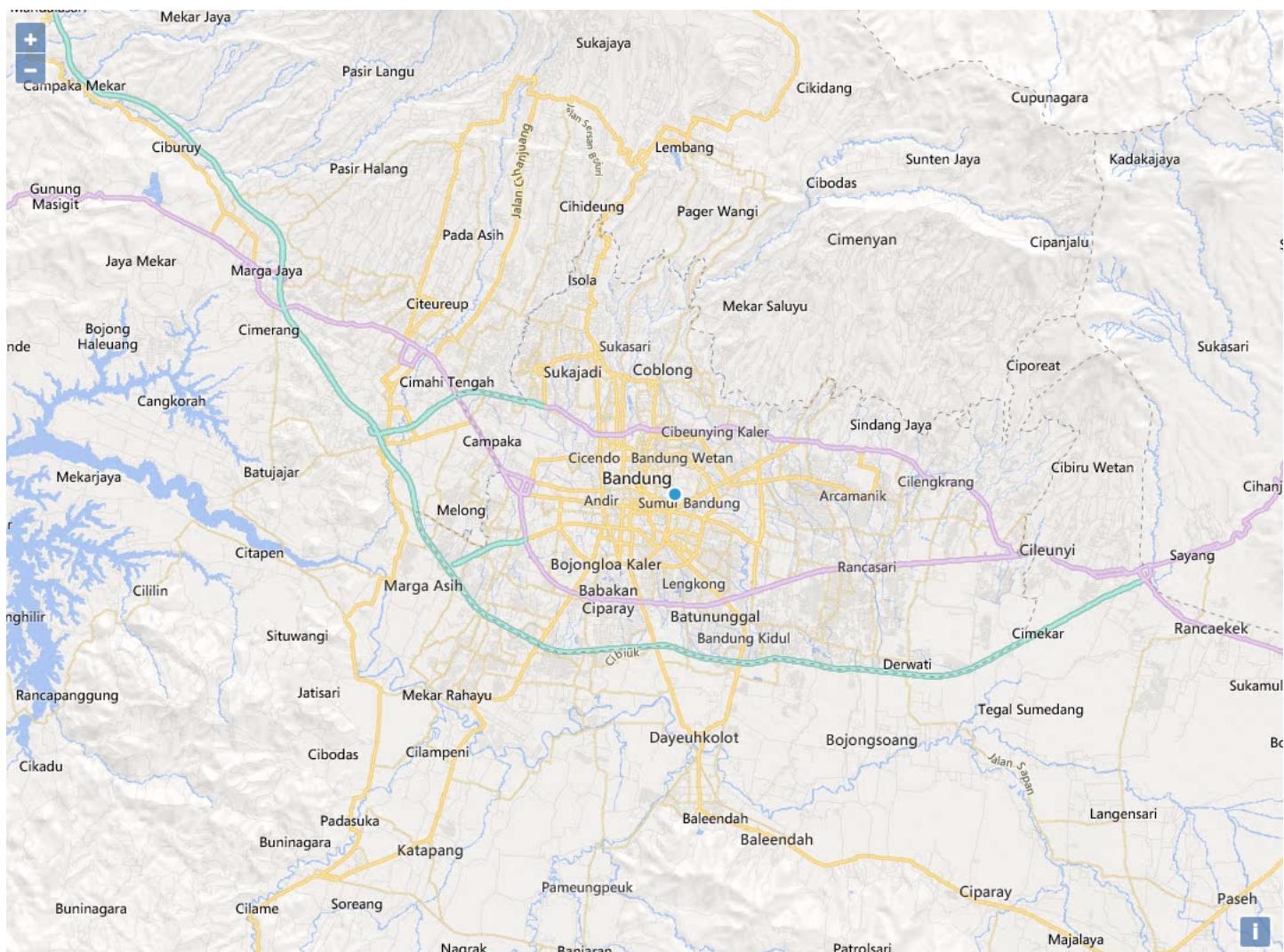
Pada halaman utama KIRI (dapat dilihat pada gambar 3.1), terdapat beberapa bagian yaitu:



Gambar 3.1: Halaman Utama KIRI

##### 3.1.1 Peta

Peta pada KIRI (Gambar 3.2) berfungsi untuk menampilkan peta pada pengguna berdasarkan kota pengguna dan juga menentukan tempat asal dan tujuan pengguna dengan melakukan klik pada peta.



Gambar 3.2: Peta pada KIRI

KIRI menggunakan OpenLayers yang berbasis JavaScript untuk memuat peta pada halaman web. Pertama melakukan deklarasi peta yang digunakan menggunakan BingMaps. Penggunaan BingMaps membutuhkan dua *parameter*, yaitu *key* yang merupakan kunci untuk menggunakan BingMaps dan *imagerySet* yang merupakan tipe peta pada BingMaps. dan tipe peta pada BingMaps tersebut seperti pada kode listing 3.1.

Listing 3.1: Deklarasi peta BingMaps

```
1 var mapLayer = new ol.layer.Tile(  
2 {  
3     source : new ol.source.BingMaps(  
4     {  
5         key : 'AuV7xXD6_UMiQ5BLoZr0xkpjLpzWqMT55772Q8XtLIQuDebHPKiNXSIZXxEr1GA',  
6         imagerySet : 'Road'  
7     })  
8});
```

Untuk menambahkan fitur pada peta OpenLayers, seperti membuat marker pada peta dan membuat rute pada peta dapat dicapai dengan membuat objek ol.source.Vector seperti pada kode listing 3.2.

Listing 3.2: Objek ol.source.Vector

```
1 | var resultVectorSource = new ol.source.Vector();
```

```
2| var inputVectorSource = new ol.source.Vector();
```

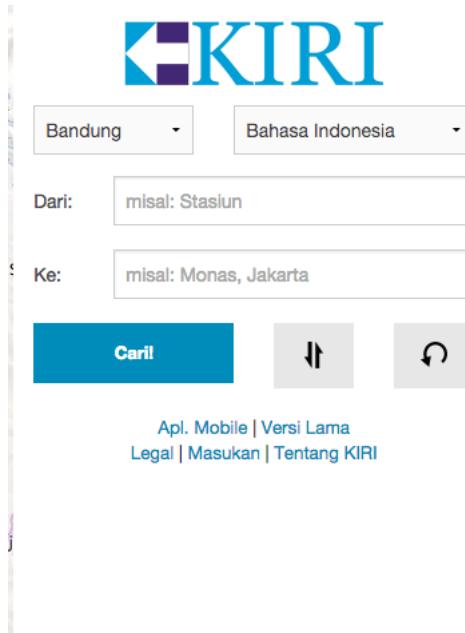
Setelah deklarasi peta beserta konfigurasi fitur yang terdapat pada peta, memasukkan semua fitur pada *layers* dan *target* untuk memasukkan *id tag* yang digunakan pada HTML seperti pada kode listing 3.3.

Listing 3.3: Instansiasi peta

```
1| var map = new ol.Map(  
2|   {  
3|     ...  
4|     layers : [ mapLayer, new ol.layer.Vector({source: inputVectorSource}), new ol.layer.Vector({source  
5|       : resultVectorSource}) ],  
6|     target : 'map'  
6|   });
```

### 3.1.2 Form Samping

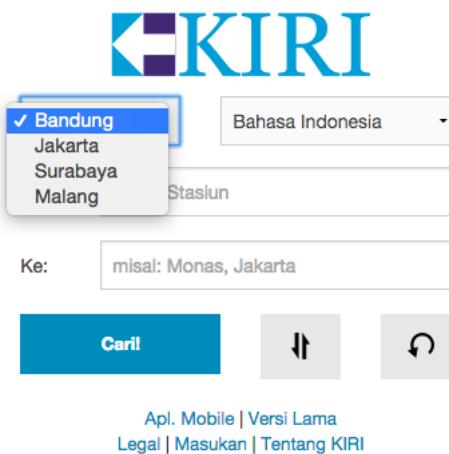
Form yang terdapat pada halaman utama KIRI (Gambar 3.3) terdiri dari:



Gambar 3.3: Form pada KIRI

#### Dropdown Menu Kota

*Dropdown* yang berfungsi untuk memilih kota yang akan ditampilkan pada peta (Gambar 3.4).



Gambar 3.4: Dropdown Menu Kota pada KIRI

Melakukan deklarasi variabel regioninfos sebagai *associated array* pada file constants.php. Setiap kota direpresentasikan sebagai proto\_region\_KOTA dimana KOTA adalah kota yang ada pada KIRI. Setiap proto\_region\_KOTA terdapat *lat* sebagai garis lintang, *lon* sebagai garis bujur dan *zoom* sebagai tingkat *zoom* untuk memperbarui peta. Proto\_region\_KOTA juga terdapat *name* untuk menampilkan pilihan kota dan *searchplace\_regex* untuk pencarian rute pada pilihan kota seperti pada kode listing 3.4.

Listing 3.4: Deklarasi variabel regioninfos

```

1 ...
2 /**
3  * Different parameters for different regions. */
4 $regioninfos = array(
5     $proto_region_bandung => array(
6         'lat' => -6.91474,
7         'lon' => 107.60981,
8         'radius' => 17000,
9         'zoom' => 12,
10        'searchplace_regex' => ', *(bandung|bdg)$',
11        'name' => 'Bandung'
12    ),
13    $proto_region_jakarta => array(
14        'lat' => -6.21154,
15        'lon' => 106.84517,
16        'radius' => 15000,
17        'zoom' => 11,
18        'searchplace_regex' => ', *(jakarta|jkt)$',
19        'name' => 'Jakarta'
20    ),
21    $proto_region_surabaya => array(
22        'lat' => -7.27421,
23        'lon' => 112.71908,
24        'radius' => 15000,
25        'zoom' => 12,
26        'searchplace_regex' => ', *(surabaya|sby)$',
27        'name' => 'Surabaya'
28    ),
29    $proto_region_malang => array(
30        'lat' => -7.9812985,
31        'lon' => 112.6319264,
32        'radius' => 15000,
33        'zoom' => 13,
34        'searchplace_regex' => ', *(malang|mlg)$',
35        'name' => 'Malang'
36    )
37);

```

Untuk menampilkan pilihan kota, pertama mengambil array regioninfos, lalu melakukan pengulangan sebanyak nilai yang terdapat pada regioninfos. Dalam pengulangan tersebut, menulis

*tag* HTML **option** sesuai dengan *name* yang terdapat pada *regioninfos*, jika *name* tersebut sama dengan *region* pengguna, maka opsi tersebut akan terpilih. Kode dapat dilihat pada kode listing 3.5

Listing 3.5: Menampilkan pilihan kota kepada pengguna

```

1 ...
2 <select class="fullwidth" id="regionselect">
3   <?php
4     foreach ($regioninfos as $key => $value) {
5       print "<option value=\"$key\"";
6       if ($key == $region) {
7         print " selected";
8       }
9       print ">" . $value['name'] . "</option>\n";
10    }
11  ?>
12 </select>
13 ...

```

Untuk memperbarui peta, KIRI menggunakan fungsi JavaScript dengan menerima dua parameter, yaitu *newRegion* dan *updateCookie*. Pertama membuat *cookie* dengan kunci *region*, lalu membuat variabel *point* dengan mengubah String menjadi LonLat dari titik tengah peta yang dituju. Untuk memperbarui peta dengan mengatur titik tengah pada peta yaitu memanggil *method setCenter* yang menerima parameter *ol.proj.transform* yang berisi garis lintang dan bujur serta kode dari Sistem dan Transformasi Koordinat. Setelah itu, mengatur tingkat *zoom* dengan memanggil *method setZoom* dengan parameter berupa tingkat *zoom* dari peta yang dituju. Kode dapat dilihat pada kode listing 3.6

Listing 3.6: Fungsi JavaScript untuk memperbarui peta

```

1 /**
2  * Updates the region information in this page.
3  */
4 function updateRegion(newRegion, updateCookie) {
5   region = newRegion;
6   setCookie('region', region);
7   var point = stringToLonLat(regions[region].center);
8   map.getView().setCenter(ol.proj.transform(point, 'EPSG:4326', 'EPSG:3857'));
9   map.getView().setZoom(regions[region].zoom);
10 }

```

Untuk melakukan pengubahan dari tipe data String menjadi *array* Float yang berguna menjadi garis lintang dan garis bujur dengan cara memanggil *method split* dengan *parameter* ‘,’ yang berfungsi membuang ‘,’ dan menjadikan *array*. Setelah menjadi *array*, String tersebut masing-masing dijadikan ke tipe data Float dengan cara memanggil *method parseFloat* dengan *parameter* String yang ingin dijadikan Float. Kode dapat dilihat pada kode listing 3.7.

Listing 3.7: Fungsi JavaScript untuk mengubah String menjadi *array* Float

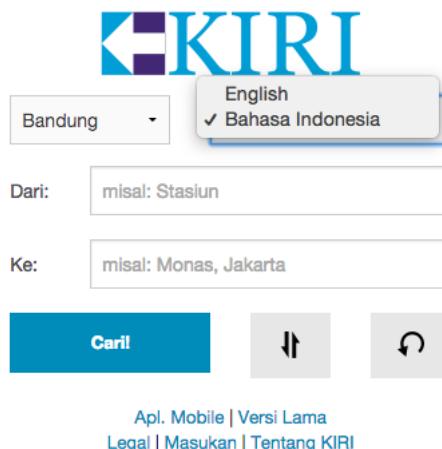
```

1 /**
2  * Converts "lat,lng" into lonlat array
3  * @return the converted lonlat array
4  */
5 function stringToLonLat(text) {
6   var latlon = text.split(/,\s*/);
7   return [parseFloat(latlon[1]), parseFloat(latlon[0])];
8 }

```

## Dropdown Menu Bahasa

*Dropdown* yang berfungsi untuk memilih bahasa yang akan digunakan pada KIRI (Gambar 3.5).



Gambar 3.5: Dropdown Menu Bahasa pada KIRI

Untuk menampilkan pilihan bahasa, menggunakan *tag* HTML `option`. Pada bagian ini, hanya cek jika sudah dilakukan lokalisasi ke Bahasa Indonesia, maka opsi yang terpilih adalah Bahasa Indonesia. Kode dapat dilihat pada 3.8

Listing 3.8: Menampilkan pilihan bahasa kepada pengguna

```

1 ..
2 <select class="fullwidth" id="localeselect">
3   <option value="en">English</option>
4   <option value="id"
5     <?php if ($locale == $proto_locale_indonesia) print " selected"; ?>>Bahasa
6     Indonesia</option>
7 </select>
8 ..

```

Ketika memilih *dropdown* bahasa, memanggil *method* JavaScript dengan *parameter* berupa fungsi yang berisi menambahkan URL dengan *query locale=id* atau *locale=en*. Kode dapat dilihat pada kode listing 3.9.

Listing 3.9: Fungsi JavaScript untuk Internationalization

```

1 ..
2 // Event handlers
3 var localeSelect = $('#localeselect');
4 localeSelect.change(function() {
5   // IE fix: when window.location.origin is not available
6   if (!window.location.origin) {
7     window.location.origin = window.location.protocol + "//" + window.location.hostname + (window.
8       location.port ? ':' + window.location.port : '');
9   }
10   window.location.replace(window.location.origin + "?locale=" + localeSelect.val());
11 });

```

## Textfield

*Textfield* pada KIRI menggunakan PHP agar dapat dilakukan proses Internationalization, seperti pada kode listing 3.10 untuk *textfield* tempat asal dan kode listing 3.11 untuk *textfield* tempat tujuan. Textfield pada KIRI dapat menerima dua masukan pengguna, yaitu:

Listing 3.10: Menampilkan *textfield* tempat awal kepada pengguna

```

1 ..
2 <div class="small-2 columns">

```

```

3   <label for="startInput" class="inline"><?php print $index_from; ?></label>
4 </div>
5 <div class="small-10 columns">
6   <input type="text" id="startInput" value=""
7     placeholder="<?php print $index_placeholder_start; ?>">
8 </div>
9 ...

```

Listing 3.11: Menampilkan *textfield* tempat tujuan kepada pengguna

```

1 ...
2 <div class="small-2 columns">
3   <label for="finishInput" class="inline"><?php print $index_to; ?></label>
4 </div>
5 <div class="small-10 columns">
6   <input type="text" id="finishInput" value=""
7     placeholder="<?php print $index_placeholder_finish; ?>">
8 </div>
9 ...

```

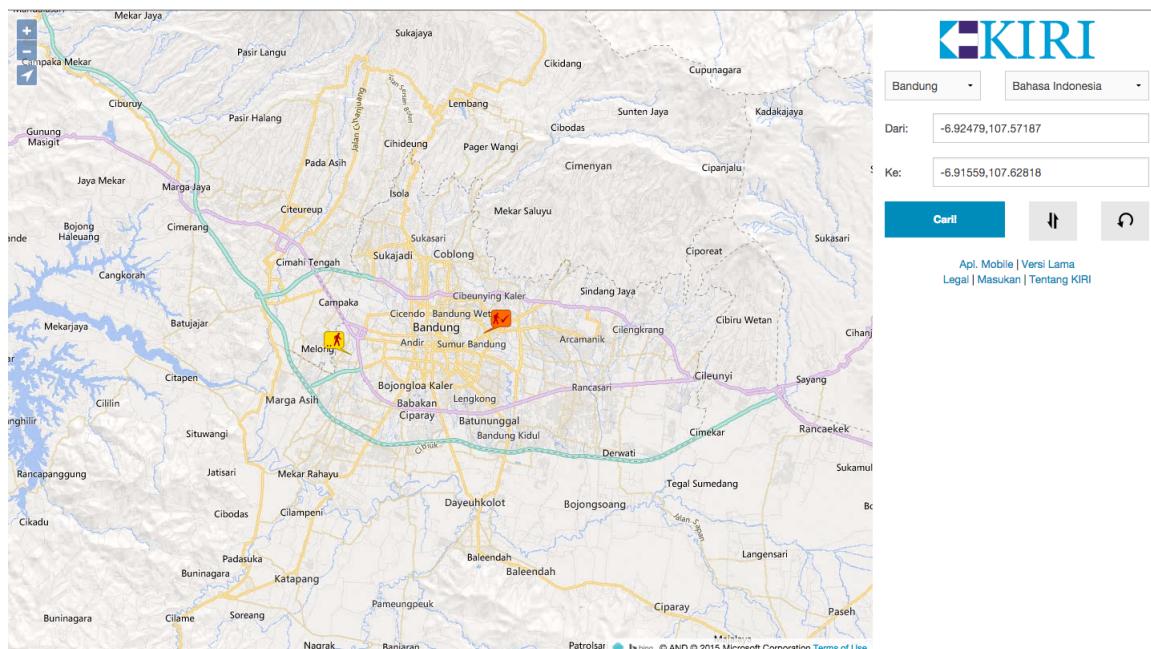
1. **Textfield dengan Masukan Nama Tempat**, pengguna dapat memasukkan nama tempat asal dan tujuan (Gambar 3.6)

Dari: Stasiun

Ke: Paris van Java

Gambar 3.6: Input User(Nama Tempat)

2. **Textfield dengan Masukan Klik Peta**, pengguna memasukkan koordinat tempat asal dan tujuan dengan klik pada peta. Dengan melakukan klik pada peta, *textfield* tempat asal dan tujuan akan secara otomatis terisi oleh koordinat masing-masing tempat (Gambar 3.7).



Gambar 3.7: Input User(Klik pada peta)

Agar peta dapat diklik, maka memanggil method `on` dengan parameter ‘click’ dan fungsi yang akan diimplementasikan ketika melakukan klik pada peta. Isi fungsi tersebut adalah

pertama melakukan pengecekan apabila textfield tempat asal atau tempat tujuan kosong, maka membuat geometry yang merupakan objek ol.geom.Point dengan parameter koordinat pada peta yang diklik oleh pengguna, lalu membuat marker dengan gambar start.png bila textfield tempat asal kosong atau finish.png bila textfield tempat tujuan kosong. Setelah itu, menambahkan fitur marker ke inputVectorSource yang akan ditampilkan pada peta dan menulis koordinat pada textfield tempat asal atau tempat tujuan. Kode dapat dilihat pada kode listing 3.12.

Listing 3.12: Membuat *event* klik pada peta

```

1 ..
2 // Map click event
3 map.on('click', function(event) {
4     if ($('#startInput').val() === '') {
5         markers['start'] = new ol.Feature({
6             geometry: new ol.geom.Point(event.coordinate)
7         })
8         markers['start'].setStyle(new ol.style.Style({
9             image: new ol.style.Icon({
10                 src: 'images/start.png',
11                 anchor: [1.0, 1.0]
12             })
13         }));
14         inputVectorSource.addFeature(markers['start']);
15         $('#startInput').val(latLngToString(ol.proj.transform(event.coordinate, 'EPSG:3857',
16                                         'EPSG:4326')));
17     } else if ($('#finishInput').val() === '') {
18         markers['finish'] = new ol.Feature({
19             geometry: new ol.geom.Point(event.coordinate)
20         })
21         markers['finish'].setStyle(new ol.style.Style({
22             image: new ol.style.Icon({
23                 src: 'images/finish.png',
24                 anchor: [0.0, 1.0]
25             })
26         }));
27         inputVectorSource.addFeature(markers['finish']);
28         $('#finishInput').val(latLngToString(ol.proj.transform(event.coordinate, 'EPSG:3857',
29                                         'EPSG:4326')));
30     }
31 });

```

### Tombol Swap

Pengguna dapat menukar isi dari *textfield* tempat asal dan tujuan. Pertama kali yang dilakukan adalah mencari pada dokumen dengan *id* swapbutton dan memanggil *method* *click* dengan *parameter* fungsi swapInput seperti pada kode listing 3.13. Fungsi swapInput berisi melakukan pencarian pada dokumen dengan id startInput dan finishInput. Melakukan penampungan sementara dengan mengambil isi dari textfield tempat asal, lalu mengubah isi dari textfield tempat asal dengan tujuan dan mengubah isi dari textfield tempat tujuan dengan isi dari penampungan sementara. Setelah itu, jika kedua textfield ada isinya, melakukan pencarian rute. Kode dapat dilihat pada kode listing 3.14.

Listing 3.13: *Method* untuk memanggil fungsi JavaScript ketika tombol *swap* ditekan

```

1 ..
2 $('#swapbutton').click(swapInput);
3 ..

```

Listing 3.14: Fungsi JavaScript untuk menukar isi *textfield* tempat asal dan tujuan

```

1 /**
2 * Swap the inputs
3 */

```

```

4| function swapInput() {
5|     var startInput = $('#startInput');
6|     var finishInput = $('#finishInput');
7|     var temp = startInput.val();
8|     startInput.val(finishInput.val());
9|     finishInput.val(temp);
10|    coordinates['start'] = null;
11|    coordinates['finish'] = null;
12|    if (startInput.val() != '' && finishInput.val() != '') {
13|        findRouteClicked();
14|    }
15| }

```

### Tombol Reset

Pengguna dapat melakukan pemilihan tempat dari awal dan mengulang tampilan peta. Pertama kali yang dilakukan adalah mencari pada dokumen dengan *id* resetbutton dan memanggil *method click* dengan *parameter* fungsi resetScreen seperti pada kode listing 3.15.

Listing 3.15: *Method* untuk memanggil fungsi JavaScript ketika tombol *reset* ditekan

```

1| ...
2| $('#resetbutton').click(resetScreen);
3| ...

```

Fungsi resetScreen berisi berbagai fungsi seperti pada kode listing 3.16, yaitu:

Listing 3.16: Fungsi JavaScript resetScreen

```

1| function resetScreen() {
2|     clearRoutingResultsOnTable();
3|     clearRoutingResultsOnMap();
4|     clearAlerts();
5|     clearStartFinishMarker();
6|     $.each(['start', 'finish'], function(sfIndex, sfValue) {
7|         var placeInput = $('#' + sfValue + 'Input');
8|         placeInput.val('');
9|         placeInput.prop('disabled', false);
10|        $('#' + sfValue + 'Select').addClass('hidden');
11|    });
12| }

```

#### 1. Fungsi clearRoutingResultsOnMap

Fungsi untuk menghapus pada resultVectorSource yang merupakan berbagai *marker* pada peta sebagai hasil pencarian rute dan memperbarui peta. Kode dapat dilihat pada kode listing 3.17.

Listing 3.17: Fungsi JavaScript untuk menghapus hasil pencarian rute pada peta

```

1|     function clearRoutingResultsOnMap() {
2|         resultVectorSource.clear();
3|         updateRegion(region, false);
4|     }

```

#### 2. Fungsi clearRoutingResultsOnTable

Fungsi untuk menghapus tampilan tabel sebagai hasil pencarian rute yang akan ditampilkan pada pengguna. Kode dapat dilihat pada kode listing 3.18.

Listing 3.18: Fungsi JavaScript untuk menghapus tampilan tabel

```

1|     function clearRoutingResultsOnTable() {
2|         $('.tabs').remove();
3|         $('.tabs-content').remove();
4|     }

```

### 3. Fungsi clearAlerts

Fungsi untuk menghapus *alerts* sebagai tanda yang akan ditampilkan kepada pengguna, seperti sedang melakukan pencarian rute atau masalah koneksi. Kode dapat dilihat pada kode listing 3.19.

Listing 3.19: Fungsi JavaScript untuk menghilangkan *alerts*

```
1 function clearAlerts() {
2     $('.alert-box').remove();
3 }
```

### 4. Fungsi clearRoutingResultsOnMap

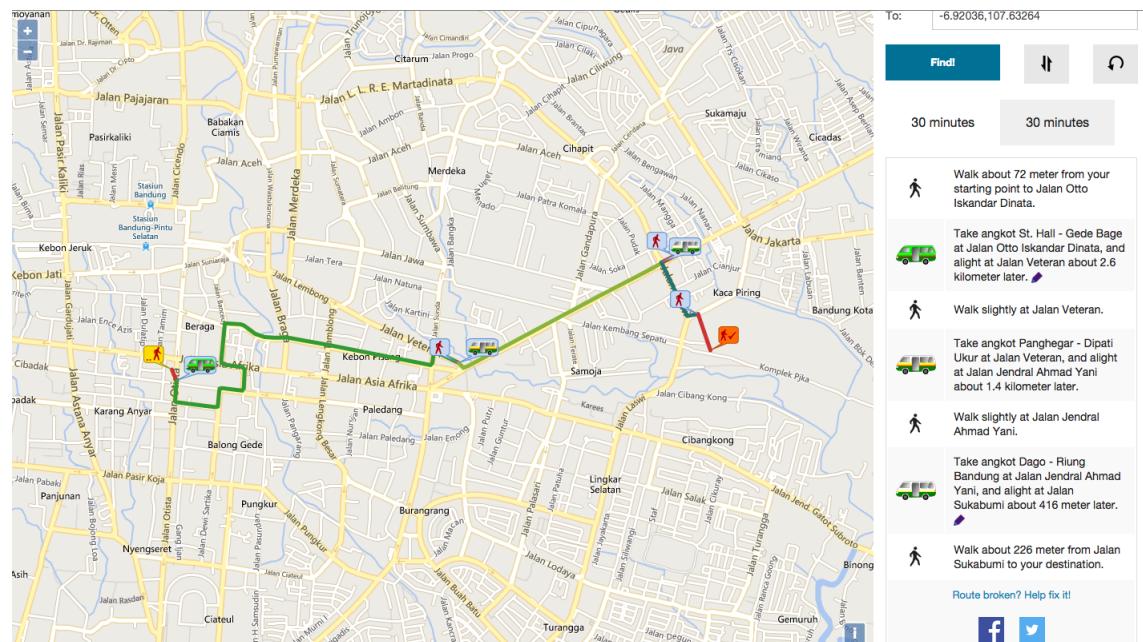
Fungsi untuk menghapus *markers* tempat awal dan tujuan, lalu menghapus fitur pada inputVectorSource. Kode dapat dilihat pada kode listing 3.20.

Listing 3.20: Fungsi JavaScript untuk menghilangkan *alerts*

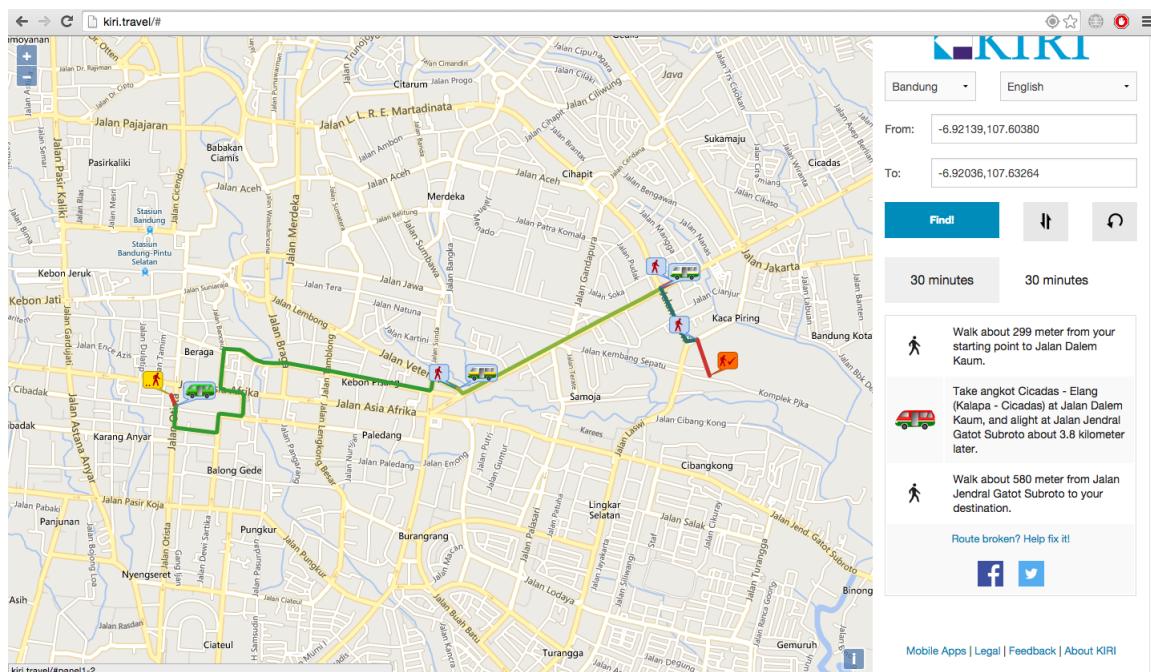
```
1 function clearStartFinishMarker() {
2     if (markers['start'] != null) {
3         markers['start'] = null;
4     }
5     if (markers['finish'] != null) {
6         markers['finish'] = null;
7     }
8     inputVectorSource.clear();
9 }
```

### Tombol Find

Pengguna dapat mencari rute untuk sampai ke tujuan (Gambar 3.8). Pengguna dapat memilih rute alternatif yang sudah disediakan KIRI jika ada (Gambar 3.9).



Gambar 3.8: Contoh Pencarian Rute pada KIRI



Gambar 3.9: Contoh Rute Alternatif pada KIRI

Hal yang pertama kali dilakukan adalah melakukan validasi jika salah satu textfield kosong, maka akan langsung membatalkan proses dan memberi alert kepada pengguna. Jika tidak kosong, maka akan memunculkan alert ‘mohon menunggu’. Setelah itu melakukan pengecekan apakah isi dari textfield merupakan format yang benar untuk garis lintang dan bujur. Jika formatnya benar, maka dimasukkan ke dalam array coordinates dan melakukan penambahan completedLatLon yang berfungsi untuk mengetahui apakah kedua textfield sudah benar formatnya untuk dilakukan pencarian rute. Jika formatnya tidak benar, maka melakukan pengecekan array coordinates kosong atau tidak. Jika kosong, maka melakukan pencarian pilihan tempat yang akan menjadi tempat sugesti yang diberikan kepada pengguna. Jika tidak kosong, maka memanggil fungsi checkCoordinatesThenRoute dengan parameter berupa coordinates. Terakhir, melakukan pengecekan completedLatLon jika isinya sama dengan dua, maka memanggil fungsi checkCoordinatesThenRoute. Kode dapat dilihat pada kode listing 3.21.

Listing 3.21: Fungsi JavaScript untuk ketika tombol *find* ditekan

```

1 /**
2  * A function that will be called when find route button is clicked
3  * (or triggered by another means)
4 */
5 function findRouteClicked() {
6     // Validate
7     var cancel = false;
8     $.each(['start', 'finish'], function(sfIndex, sfValue) {
9         if ($('#' + sfValue + 'Input').val() === '') {
10             cancel = true;
11             return;
12         }
13     });
14     if (cancel) {
15         showAlert(messageFillBoth, 'alert');
16         return;
17     }
18
19     clearAlerts();
20     clearRoutingResultsOnTable();
21     showAlert(messagePleaseWait, 'secondary');
22 }
```

```

23 var completedLatLon = 0;
24 $.each(['start', 'finish'], function(sfIndex, sfValue) {
25     var placeInput = $('#' + sfValue + 'Input');
26     var placeSelect = $('#' + sfValue + 'Select');
27     if (isLatLng(placeInput.val())) {
28         coordinates[sfValue] = placeInput.val();
29         completedLatLon++;
30     } else {
31         if (coordinates[sfValue] == null) {
32             // Coordinates not yet ready, we do a search place
33             protocol.searchPlace(
34                 placeInput.val(),
35                 region,
36                 function(result) {
37                     placeSelect.empty();
38                     placeSelect.addClass('hidden');
39                     if (result.status != 'error') {
40                         if (result.searchresult.length > 0) {
41                             $.each(result.searchresult, function(index, value) {
42                                 var placeSelect = $('#' + sfValue + 'Select');
43                                 placeSelect
44                                     .append($('</option>')
45                                         .attr('value', value['location'])
46                                         .text(value['placename']));
47                                 placeSelect.removeClass('hidden');
48                             });
49                             coordinates[sfValue] = result.searchresult[0]['location'];
50                             checkCoordinatesThenRoute(coordinates);
51                         } else {
52                             clearSecondaryAlerts();
53                             clearRoutingResultsOnMap();
54                             showAlert(placeInput.val() + messageNotFound, 'alert');
55                         }
56                     } else {
57                         clearSecondaryAlerts();
58                         clearRoutingResultsOnMap();
59                         showAlert(messageConnectionError, 'alert');
60                     }
61                 });
62             } else {
63                 // Coordinates are already available, skip searching
64                 checkCoordinatesThenRoute(coordinates);
65             }
66         }
67     });
68     if (completedLatLon == 2) {
69         checkCoordinatesThenRoute(coordinates);
70     }
71 }

```

Fungsi checkCoordinatesThenRoute melakukan pengecekan jika array coordinates tempat awal dan tempat tujuan tidak kosong maka melakukan protocol.findRoute. Jika hasil results sama dengan ‘ok’, maka menunjukkan rute pencarian. Jika hasil result bukan ‘ok’, maka akan menampilkan alert ‘gangguan koneksi’. Kode dapat dilihat pada kode listing 3.22.

Listing 3.22: Fungsi JavaScript checkCoordinatesThenRoute

```

1 /**
2  * Check if coordinates are complete. If yes, then start routing.
3  * @param coordinates the coordinates to check.
4  */
5 function checkCoordinatesThenRoute(coordinates) {
6     if (coordinates['start'] != null && coordinates['finish'] != null) {
7         protocol.findRoute(
8             coordinates['start'],
9             coordinates['finish'],
10            locale,
11            function(results) {
12                if (results.status === 'ok') {
13                    showRoutingResults(results);
14                } else {
15                    clearSecondaryAlerts();
16                    showAlert(messageConnectionError, 'alert');
17                }
18            });
19    }
20 }

```

### 3.1.3 Internationalization

Penggunaan Internationalization (i18n) pada PHP dengan cara deklarasi semua variabel yang akan digunakan pada proses i18n terlebih dahulu, misalnya buat file dengan nama tirtayasa\_en.php untuk Bahasa Inggris dan tirtayasa\_id.php untuk Bahasa Indonesia. Pada setiap file tersebut, masukkan *script* PHP untuk menentukan teks yang keluar pada halaman *web* seperti pada kode listing 3.23 dan kode listing 3.24.

Listing 3.23: Script PHP untuk Bahasa Inggris

```

1 <?php
2
3     $index_about_kiri = "About KIRI";
4     $index_apps = "Mobile Apps";
5     $index_advanced_ = "Advanced ...";
6     $index_buylticket = "BUY TICKET";
7     $index_connectionerror = 'Connection problem';
8 ?>

```

Listing 3.24: Script PHP untuk Bahasa Indonesia

```

1 <?php
2     $index_about_kiri = "Tentang KIRI";
3     $index_apps = "Apl. Mobile";
4     $index_advanced_ = "Lanjut ...";
5     $index_buylticket = "BUY TICKET";
6     $index_connectionerror = 'Gangguan koneksi';
7 ?>

```

Setelah itu, masukkan *script* PHP pada *tag* HTML yang ingin diubah saat dilakukan i18n. Adanya *script* PHP pada tag HTML, maka teks akan berubah jika dilakukan i18n seperti pada kode listing 3.25.

Listing 3.25: Script PHP untuk Internationalization

```

1 ...
2     <label for="startInput" class="inline"><?php print $index_from; ?></label>
3     <label for="finishInput" class="inline"><?php print $index_to; ?></label>
4     <a href="#" class="small button expand" id="findbutton"><strong><?php print $index_find; ?></strong></
5         a>
...

```

## 3.2 Analisis Sistem Usulan

### 3.2.1 Form Samping

#### Dropdown Menu Kota

Untuk menampilkan pilihan kota pada Play Framework , menampilkan semua pilihan kota yang terdapat pada KIRI. Kode dapat dilihat pada kode listing 3.26

Listing 3.26: Menampilkan pilihan kota kepada pengguna

```

1 ...
2 <select class="fullwidth" id="regionselect">
3     <option value="bdg">Bandung</option>
4     <option value="jkt">Jakarta</option>
5     <option value="sby">Surabaya</option>
6     <option value="mlg">Malang</option>
7 </select>
8 ...

```

Untuk memperbarui peta menggunakan Play Framework , membuat fungsi JavaScript update-Map dengan parameter array Float sebagai garis bujur dan garis lintang dan zoom sebagai tingkat zoom pada peta. Kode dapat dilihat pada kode listing 3.27.

Listing 3.27: Fungsi JavaScript untuk memperbarui peta

```

1 var updateMap = function(arrLonLat ,zoom){
2
3     map.getView().setCenter(ol.proj.transform(arrLonLat , 'EPSG:4326' , 'EPSG:3857'));
4     map.getView().setZoom(zoom);
5 };

```

Untuk menambahkan aksi ketika memilih opsi *dropdown* kota, pertama melakukan pencarian pada dokumen dengan *id* regionselect. Handler tersebut berfungsi ketika mengganti pilihan pada dropdown, memasukkan koordinat yang tepat untuk opsi kota dan memanggil fungsi updateMap untuk memperbarui peta. Kode dapat dilihat pada kode listing 3.28.

Listing 3.28: Fungsi JavaScript untuk menambahkan *handler* ketika mengganti *dropdown* kota

```

1 var regionSelect = document.getElementById("regionselect");
2
3 var handlerRegion = function () {
4     var regionValue = regionSelect.value;
5     if(regionValue=="bdg")
6     {
7         arrLonLat = [107.60981,-6.91474];
8         updateMap(arrLonLat,12);
9     } else if(regionValue=="jkt")
10    {
11        arrLonLat = [106.84517,-6.21154];
12        updateMap(arrLonLat,11);
13    } else if(regionValue=="sby")
14    {
15        arrLonLat = [112.71908,-7.27421];
16        updateMap(arrLonLat,12);
17    } else
18    {
19        arrLonLat = [112.6319264,-7.9812985];
20        updateMap(arrLonLat,13);
21    }
22}
23
24};
25 regionSelect.addEventListener("change",handlerRegion , false);

```

## Dropdown Menu Bahasa

Untuk menampilkan pilihan bahasa, menggunakan *tag* HTML *option*. Pada bagian ini. Kode dapat dilihat pada ??

Listing 3.29: Menampilkan pilihan bahasa kepada pengguna

```

1 ...
2 <select class="fullwidth" id="localeselect">
3     <option value="en">English</option>
4     <option value="id">Bahasa Indonesia</option>
5 </select>
6 ...

```

Ketika memilih *dropdown* bahasa, pertama melakukan pencarian pada dokumen dengan *id* localeselect. Setelah itu menambahkan *handler* yang berisi melakukan pengecekan isi dari *dropdown* bahasa, lalu memanggil *method* JavaScript *window.location.replace* dengan *parameter* URL baru yang sesuai dengan bahasa. Kode dapat dilihat pada kode listing 3.30.

Listing 3.30: Fungsi JavaScript untuk Internationalization

```

1 ...
2 var localeSelect = document.getElementById("localeselect");
3
4 var handlerLocale = function(){
5     var localeValue = localeSelect.value;
6     if(localeValue=="en")
7     {
8         window.location.replace("http://localhost:9000");
9     }

```

```

10     else if(localeValue == "id")
11     {
12       window.location.replace("http://localhost:9000/id");
13     }
14   };
15 };
16 localeSelect.addEventListener("change", handlerLocale, false);
17 ..

```

## Textfield

Pada Play Framework , *Textfield* pada KIRI dibuat agar dapat dilakukan proses i18n, seperti pada kode listing 3.31 untuk *textfield* tempat asal dan kode listing 3.32 untuk *textfield* tempat tujuan.

Listing 3.31: Menampilkan *textfield* tempat awal kepada pengguna

```

1 ..
2 <div class="small-2 columns">
3   <label for="startInput" class="inline">@Messages.get("from")</label>
4 </div>
5 <div class="small-10 columns">
6   <input type="text" id="startInput" value=""
7   placeholder="@Messages.get("ph_from")">
8 </div>
9 ..

```

Listing 3.32: Menampilkan *textfield* tempat tujuan kepada pengguna

```

1 ..
2 <div class="small-2 columns">
3   <label for="finishInput" class="inline">@Messages.get("to")</label>
4 </div>
5 <div class="small-10 columns">
6   <input type="text" id="finishInput" value=""
7   placeholder="@Messages.get("ph_to")">
8 </div>
9 ..

```

Pada Play Framework , hal yang diubah adalah pengambilan gambar pada folder ‘assets/images. Kode dapat dilihat pada kode listing 3.33.

Listing 3.33: Membuat *event* klik pada peta

```

1 //for map click event
2 map.on('click', function(event) {
3   if (startInput.value === '')
4   {
5     markers['start'] = new ol.Feature({
6       geometry: new ol.geom.Point(event.coordinate)
7     })
8     markers['start'].setStyle(new ol.style.Style({
9       image: new ol.style.Icon({
10         src: '/assets/images/start.png',
11         anchor: [1.0, 1.0]
12       })
13     }));
14     inputVectorSource.addFeature(markers['start']);
15     startInput.value = latLngToString(ol.proj.transform(event.coordinate, 'EPSG:3857', 'EPSG:4326'));
16   }else if (finishInput.value === '') {
17     markers['finish'] = new ol.Feature({
18       geometry: new ol.geom.Point(event.coordinate)
19     })
20     markers['finish'].setStyle(new ol.style.Style({
21       image: new ol.style.Icon({
22         src: 'assets/images/finish.png',
23         anchor: [0.0, 1.0]
24       })
25     }));
26     inputVectorSource.addFeature(markers['finish']);
27     finishInput.value = latLngToString(ol.proj.transform(event.coordinate, 'EPSG:3857', 'EPSG:4326'));
28   }
29 });

```

## Tombol Swap

Pengguna dapat menukar isi dari *textfield* tempat asal dan tujuan. Pertama kali yang dilakukan adalah mencari pada dokumen dengan *id* swapbutton dan menambahkan fungsi seperti pada kode listing 3.34. Fungsi tersebut berisi menukar isi dari *textfield* tempat asal dan tempat tujuan seperti pada kode listing 3.14.

Listing 3.34: *Method* untuk memanggil fungsi JavaScript ketika tombol *swap* ditekan

```
1 ..  
2 var swapBtn = document.getElementById("swapbutton");  
3  
4 swapBtn.addEventListener("click", swapInput);  
5 ..
```

Listing 3.35: Fungsi JavaScript untuk menukar isi *textfield* tempat asal dan tujuan

```
1 /**  
2  * Swap the inputs  
3  */  
4 function swapInput() {  
5     var temp = startInput.value;  
6  
7     startInput.value = finishInput.value;  
8     finishInput.value = temp;  
9  
10    coordinates['start'] = null;  
11    coordinates['finish'] = null;  
12  
13    if(startInput.value != "" && finishInput.value != ""){  
14        findRouteClicked();  
15    }  
16}  
17 }
```

## Tombol Reset

Pertama kali yang dilakukan adalah mencari pada dokumen dengan *id* resetbutton dan menambahkan fungsi reset ketika tombol reset ditekan seperti pada kode listing 3.36.

Listing 3.36: *Method* untuk memanggil fungsi JavaScript ketika tombol *reset* ditekan

```
1 ..  
2 var resetBtn = document.getElementById("resetbutton");  
3 resetBtn.addEventListener("click", reset);  
4 ..
```

Fungsi reset berisi berbagai fungsi seperti pada kode listing 3.16, yaitu:

Listing 3.37: Fungsi JavaScript reset

```
1 /**  
2  * Reset Screen  
3  */  
4 function reset() {  
5     clearRoutingResultsOnTable();  
6     clearRoutingResultsOnMap();  
7     clearAlerts();  
8     clearStartFinishMarker();  
9  
10    var x = ['start', 'finish'];  
11    for (x in markers, function(sfIndex, sfValue) {  
12        var placeInput = '#' + sfValue + 'Input';  
13        placeInput.val('');  
14        placeInput.prop('disabled', false);  
15        '#' + sfValue + 'Select'.addClass('hidden');  
16    });  
17}
```

### 1. Fungsi clearRoutingResultsOnMap

Fungsi untuk menghapus pada resultVectorSource yang merupakan berbagai *marker* pada peta sebagai hasil pencarian rute dan memperbarui peta. Kode dapat dilihat pada kode listing 3.38.

Listing 3.38: Fungsi JavaScript untuk menghapus hasil pencarian rute pada peta

```

1  function clearRoutingResultsOnMap() {
2      resultVectorSource.clear();
3      arrLonLat = [107.60981, -6.91474];
4      updateMap(arrLonLat, 12);
5  }

```

### 2. Fungsi clearRoutingResultsOnTable

Fungsi untuk menghapus tampilan tabel sebagai hasil pencarian rute yang akan ditampilkan pada pengguna. Pertama melakukan pencarian pada dokumen dengan *id* tabs dan tabs-content. Lalu, menghapus pada dokumen dengan *id* tersebut. Kode dapat dilihat pada kode listing 3.39.

Listing 3.39: Fungsi JavaScript untuk menghapus tampilan tabel

```

1  function clearRoutingResultsOnTable() {
2      var tabs = document.getElementById("tabs");
3      var tabsContent = document.getElementById("tabs-content");
4      tabs.remove();
5      tabsContent.remove();
6  }

```

### 3. Fungsi clearAlerts

Fungsi untuk menghapus *alerts* sebagai tanda yang akan ditampilkan kepada pengguna, seperti sedang melakukan pencarian rute atau masalah koneksi. Pertama melakukan pencarian dokumen dengan id alert-box, lalu menghapus pada dokumen. Kode dapat dilihat pada kode listing 3.40.

Listing 3.40: Fungsi JavaScript untuk menghilangkan *alerts*

```

1  function clearAlerts() {
2      var alertBox = document.getElementById("alert-box");
3      alertBox.remove();
4  }

```

## Tombol Find

Pada Play Framework , kode yang diganti hanya tampilan saja. Kode dapat dilihat pada kode listing 3.41.

Listing 3.41: Fungsi JavaScript untuk ketika tombol *find* ditekan

```

1 ...
2 <div class="small-6 columns">
3     <a href="#" class="small button expand" id="findbutton"><strong>@Messages.get("find")</strong></a>
4 </div>
5 ...

```

### 3.2.2 Internationalization

Penggunaan i18n pada Play Framework hampir sama dengan i18n pada PHP, pertama deklarasi semua variabel yang akan digunakan pada i18n, misal membuat file dengan nama messages.en untuk

Bahasa Inggris dan messages.id untuk Bahasa Indonesia. Pada setiap file tersebut, masukkan kunci beserta value untuk untuk menentukan teks yang keluar pada halaman web seperti pada kode listing 3.42 dan kode listing 3.43.

Listing 3.42: Script Play Framework untuk Bahasa Inggris

```

1 from = From:
2 ph_from = e.g. Stasiun
3 ph_to = e.g. Monas, Jakarta
4 find = Find!
5 to = To:
6 ...

```

Listing 3.43: Script Play Framework untuk Bahasa Indonesia

```

1 from = Dari:
2 ph_from = misal: Stasiun
3 ph_to = misal: Monas, Jakarta
4 find = Cari!
5 to = Ke:
6 ...

```

Pada Play Framework , ada dua cara untuk melakukan i18n dan sudah tersedia method untuk i18n, yaitu:

1. Memanggil *method* @Messages dengan parameter berupa String yang merupakan kunci dari file messages.LANG. Tetapi, dengan menggunakan cara ini, perlu dilakukan memuat ulang dua kali halaman *web*. Kode dapat dilihat pada 3.44

Listing 3.44: Script Play Framework untuk Internationalization

```

1 ...
2 <label for="startInput" class="inline">@Messages("from")</label>
3 <label for="finishInput" class="inline">@Messages("to")</label>
4 <a href="#" class="small button expand" id="findbutton"><strong>@Messages("find")</strong></
5 ...

```

2. Melakukan @import play.i18n.\_ pada *template view* dan menggunakan *method* Messages.get dengan parameter berupa String yang merupakan kunci dari file messages.LANG. Kode dapat dilihat pada 3.45.

Listing 3.45: Script Play Framework untuk Internationalization

```

1 @import play.i18n._
2 ...
3 <label for="startInput" class="inline">@Messages.get("from")</label>
4 <label for="finishInput" class="inline">@Messages.get("to")</label>
5 <a href="#" class="small button expand" id="findbutton"><strong>@Messages.get("find")</
   strong></a>

```

memanggil method @Messages dengan parameter berupa String yang merupakan kunci dari file messages.LANG. Dengan memanggil method @Messages ini, akan mengganti dengan value pada file messages.LANG seperti pada kode listing ??.

### 3.2.3 Tampilan

Tampilan pada Play Framework menggunakan format bahasa Scala. Tetapi penggunaan Scala pada Play Framework tidak mengharuskan mahir dalam menggunakan Scala. Play Framework menggunakan Scala tetapi ada ekstensi tambahan lagi yaitu HTML sehingga tidak berbeda jauh dalam penggunaannya dengan HTML biasa hanya dalam Scala, pedeklarasian variabel menggunakan notasi @, sedangkan pada PHP menggunakan notasi \$.

### 3.2.4 JavaScript dan stylesheet

Penggunaan *stylesheet* dan Javascript pada PHP hanya tinggal memasukkan direktori file yang hendak digunakan, seperti pada kode listing 3.46. Sedangkan pada Play Framework , penggunaan *stylesheet* dan Javascript harus didefinisikan terlebih dahulu pada routes seperti pada kode listing 3.47. Pada [4], untuk akses folder public method yang dipanggil adalah method **at** dan juga tidak disebutkan file mana yang akan diakses, sedangkan pada Play 2.4 method yang dipakai adalah method **versioned** dan menyebutkan file mana yang akan diakses, yaitu file:Asset. Setelah melakukan definisi pada routes, memasukkan stylesheet atau Javascript pada *view* dengan memanggil routes dengan direktori file yang hendak digunakan seperti pada kode listing 3.48.

Listing 3.46: Penggunaan *stylesheet* dan Javascript pada PHP

```
1 | <link rel="stylesheet" href="css/styleIndex.css" />
2 | <script src="foundation/js/vendor/modernizr.js"></script>
```

Listing 3.47: Routes untuk penggunaan *stylesheet* dan Javascript

```
1 | GET      /assets/*file           controllers.Assets.versioned(path="/public", file: Asset)
```

Listing 3.48: Penggunaan *stylesheet* dan Javascript pada Play Framework

```
1 | <link rel="stylesheet" href="@routes.Assets.versioned("css/styleIndex.css")" type="text/css">
2 | <script src="@routes.Assets.versioned("foundation/js/vendor/modernizr.js")"></script>
```

Untuk mendapatkan elemen View pada JavaScript PHP langsung memanggil **id** tujuan yang ada pada View sebagai parameter, seperti pada kode listing 3.49. Sedangkan pada Play Framework , mendapatkan elemen View pada JavaScript harus menggunakan method **document.getElementById** dengan parameter berupa **id** tujuan, seperti pada kode listing 3.50.

Listing 3.49: Mendapatkan elemen View pada JavaScript PHP

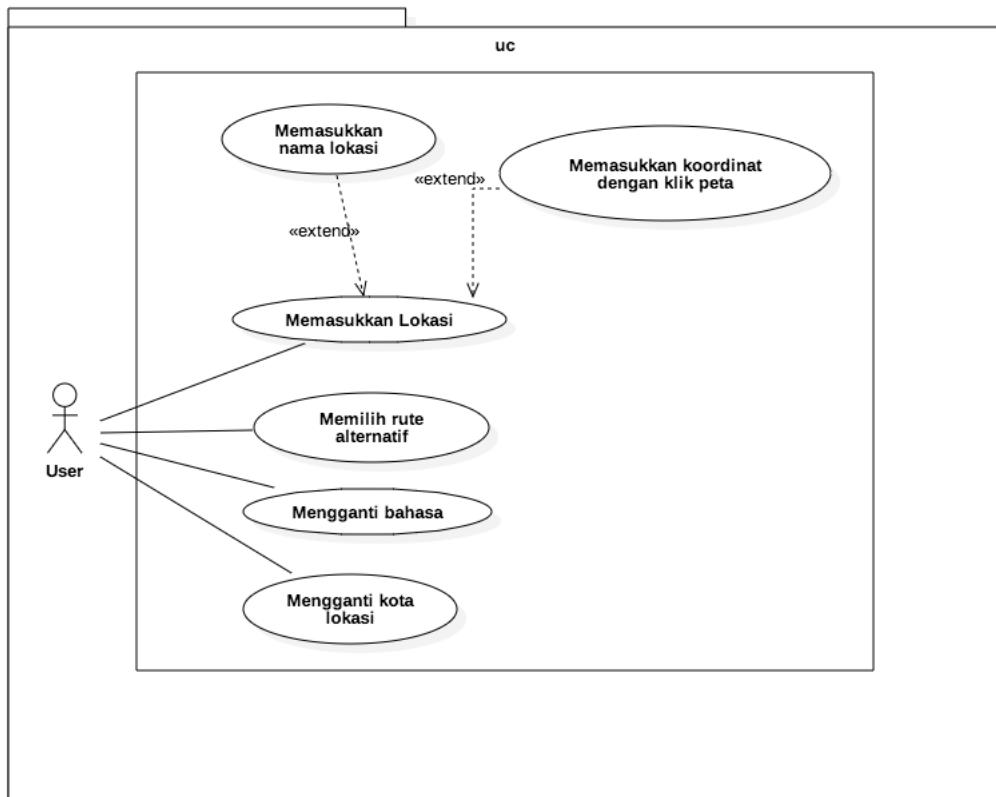
```
1 | $( '.tabs' ).remove();
2 | $( '.tabs-content' ).remove();
```

Listing 3.50: Mendapatkan elemen View pada JavaScript Play Framework

```
1 | var tabs = document.getElementById("tabs").remove();
2 | var tabs_content = document.getElementById("tabs-content").remove();
```

## 3.3 Analisis Use Case

Diagram *use case* pada KIRI hanya mempunyai satu aktor, yaitu pengguna. Diagram *use case* dapat dilihat pada gambar 3.10.



Gambar 3.10: Use Case Diagram KIRI

Terdapat empat *use case*, yaitu:

1. **Memasukkan lokasi**, pengguna dapat memasukkan lokasi dengan memasukkan nama lokasi ataupun melakukan klik pada peta dan diubah menjadi koordinat.
2. **Memilih rute alternatif**, pengguna dapat memilih rute alternatif (jika ada) setelah proses pencarian rute.
3. **Mengganti bahasa**, pengguna dapat memilih bahasa yang ingin digunakan dengan bahasa yang disediakan, Bahasa Indonesia atau Bahasa Inggris.
4. **Mengganti lokasi kota**, pengguna dapat memilih lokasi kota mana yang ingin digunakan dengan pilihan kota yang disediakan, yaitu: Bandung, Jakarta, Surabaya, dan Malang.

### 3.3.1 Skenario Use Case

#### 1. Memasukkan lokasi

- Nama: Memasukkan lokasi
- Aktor: Pengguna
- Deskripsi: Memasukkan lokasi dengan memasukkan nama lokasi ataupun melakukan klik pada peta dan diubah menjadi koordinat.
- Kondisi awal: -

- Kondisi akhir: Pencarian rute dan rute alternatif (jika ada) yang akan ditampilkan pada pengguna berupa rute pada peta dan penjelasan rute.
- Skenario utama:

No	Aksi Aktor	Reaksi Sistem
1	Pengguna memasukkan lokasi	Sistem mendapatkan lokasi kemudian menampilkan hasil pencarian rute

- Eksepsi: Lokasi tidak ditemukan.

## 2. Memilih rute alternatif

- Nama: Memilih rute alternatif
- Aktor: Pengguna
- Deskripsi: Memilih rute alternatif.
- Kondisi awal: Pencarian rute sudah berhasil dan ada rute alternatif
- Kondisi akhir: Menampilkan kepada pengguna rute alternatif pada peta beserta penjelasannya.
- Skenario utama:

No	Aksi Aktor	Reaksi Sistem
1	Pengguna memilih rute alternatif	Sistem menampilkan pencarian rute alternatif pada peta dan penjelasannya

- Eksepsi: Tidak ada rute alternatif.

## 3. Mengganti bahasa

- Nama: Mengganti bahasa
- Aktor: Pengguna
- Deskripsi: Memilih opsi bahasa yang akan digunakan.
- Kondisi awal: -
- Kondisi akhir: Menampilkan kepada pengguna dengan bahasa yang dipilih
- Skenario utama:

No	Aksi Aktor	Reaksi Sistem
1	Pengguna memilih opsi bahasa	Sistem menampilkan tampilan dengan bahasa yang dipilih oleh pengguna

- Eksepsi: -

## 4. Mengganti lokasi kota

- Nama: Mengganti lokasi kota

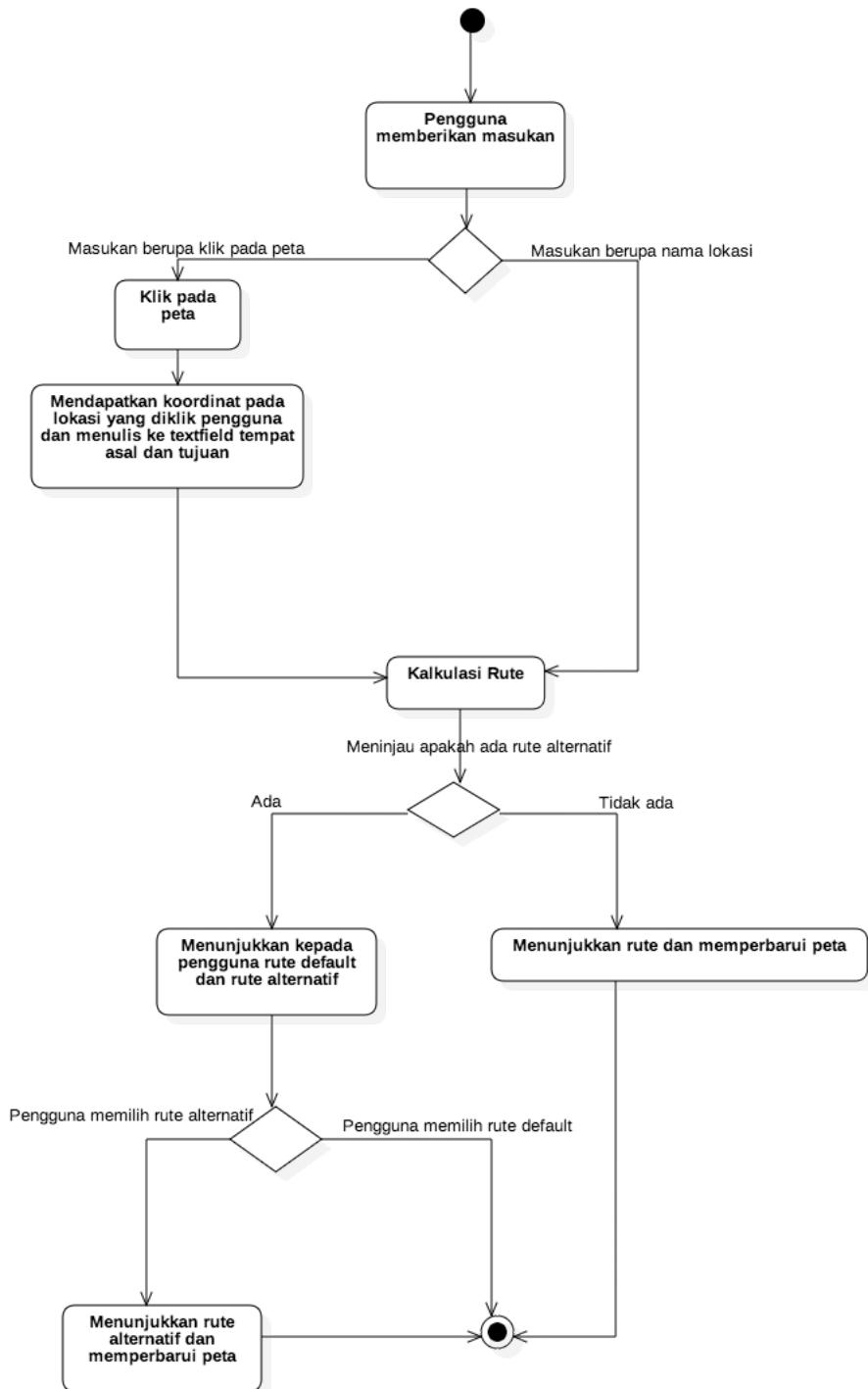
- Aktor: Pengguna
- Deskripsi: Memilih opsi lokasi kota yang akan ditampilkan pada peta dan pencarian rute pada kota tersebut.
- Kondisi awal: -
- Kondisi akhir: Menampilkan kepada pengguna dengan lokasi kota yang dipilih
- Skenario utama:

No	Aksi Aktor	Reaksi Sistem
1	Pengguna memilih opsi lokasi kota	Sistem menampilkan peta dengan lokasi kota yang dipilih dan menyiapkan pencarian rute pada kota tersebut

- Eksepsi: -

### 3.4 Analisis Activity Diagram

Diagram aktivitas KIRI dapat dilihat pada [3.11](#).



Gambar 3.11: Activity Diagram KIRI

Aktivitas-aktivitas yang ada pada KIRI dapat dijelaskan sebagai berikut:

1. Pengguna memberikan masukan. Masukan dapat berupa dua, yaitu:
  - Masukan berupa nama lokasi.
  - Masukan berupa klik pada peta. Pengguna klik pada peta, kemudian sistem mendapatkan koordinat pada lokasi yang diklik pengguna dan menulis koordinat tersebut kepada

*textfield* tempat asal atau tujuan.

2. Sistem melakukan proses pencarian rute dan melakukan pengecekan apakah ada rute alternatif atau tidak.
  - Jika tidak ada rute alternatif, maka menunjukkan rute dan memperbarui peta.
  - Jika ada rute alternatif, menunjukkan pada pengguna rute *default* dan rute alternatif.
3. Pengguna memilih rute default atau rute alternatif.
  - Jika pengguna memilih rute default, aktivitas selesai.
  - Jika pengguna memilih rute alternatif, maka sistem menunjukkan rute alternatif dan memperbarui peta.

## DAFTAR REFERENSI

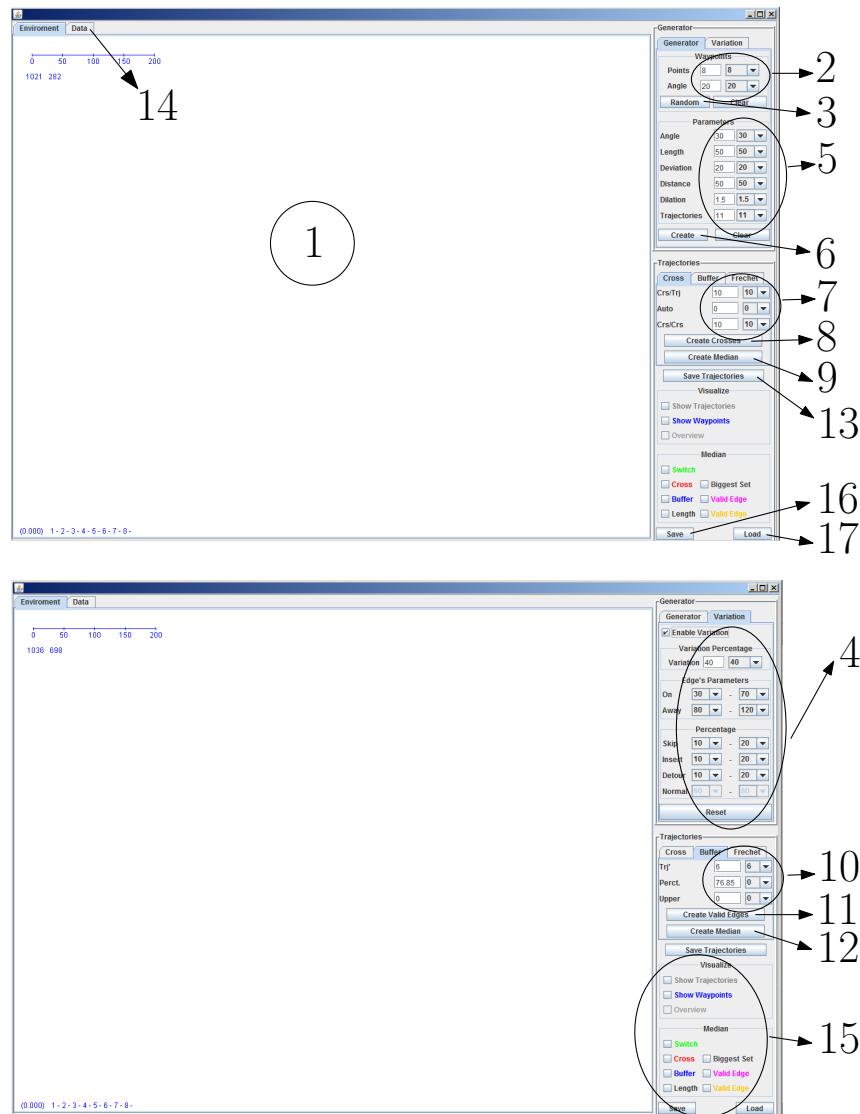
- [1] Pascal Alfadian, “KIRI.” <http://static.kiri.travel/>, 2014. [Online; diakses 28 September 2015].
- [2] Pascal Alfadian, “TirtayasaGH.” <https://github.com/pascalalfadian/TirtayasaGH>, 2014. [Online; diakses 28 September 2015].
- [3] The PHP Group, “php.” <http://php.net>, 2015. [Online; diakses 28 September 2015].
- [4] N. Leroux and S. D. Kaper, *Play for Java*. Manning Publications Co., 2014.
- [5] P. A. Santiago, *OpenLayers Cookbook*. Packt Publishing, 2012.
- [6] A. D. Patterson, *Getting Started with Zurb Foundation 4*. Packt Publishing, 2013.
- [7] Google, “Chrome DevTools.” <https://developers.google.com/web/tools/chrome-devtools/index?hl=en#learnmore>, 2015. [Online; diakses 2 Oktober 2015].



## LAMPIRAN A

### THE PROGRAM

The interface of the program is shown in Figure A.1:



Gambar A.1: Interface of the program

Step by step to compute the median trajectory using the program:

1. Create several waypoints. Click anywhere in the “Environment” area(1) or create them automatically by setting the parameters for waypoint(2) or clicking the button “Random”(3).

2. The “Variation” tab could be used to create variations by providing values needed to make them(4).
3. Create a set of trajectories by setting all parameters(5) and clicking the button “Create”(6).
4. Compute the median using the homotopic algorithm:
  - Define all parameters needed for the homotopic algorithm(7).
  - Create crosses by clicking the “Create Crosses” button(8).
  - Compute the median by clicking the “Compute Median” button(9).
5. Compute the median using the switching method and the buffer algorithm:
  - Define all parameters needed for the buffer algorithm(10).
  - Create valid edges by clicking the “Create Valid Edges”button(11).
  - Compute the median by clicking the “Compute Median”button(12).
6. Save the resulting median by clicking the “Save Trajectories” button(13). The result is saved in the computer memory and can be seen in “Data” tab(14)
7. The set of trajectories and its median trajectories will appear in the “Environment” area(1) and the user can change what to display by selecting various choices in “Visualize” and “Median” area(15).
8. To save all data to the disk, click the “Save”(16) button. A file dialog menu will appear.
9. To load data from the disk, click the “Load”(17) button.

# LAMPIRAN B

## THE SOURCE CODE

Listing B.1: MyFurSet.java

```

1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.HashSet;
4
5 /**
6 * 
7 * @author Lionov
8 */
9
10 //class for set of vertices close to furthest edge
11 public class MyFurSet {
12     protected int id;                                //id of the set
13     protected MyEdge FurthestEdge;                   //the furthest edge
14     protected HashSet<MyVertex> set;                //set of vertices close to furthest edge
15     protected ArrayList<ArrayList<Integer>> ordered; //list of all vertices in the set for each
16         trajectory;
17     protected ArrayList<Integer> closeID;            //store the ID of all vertices
18     protected ArrayList<Double> closeDist;           //store the distance of all vertices
19     protected int totaltrj;                          //total trajectories in the set
20
21 /**
22 * Constructor
23 * @param id : id of the set
24 * @param totaltrj : total number of trajectories in the set
25 * @param FurthestEdge : the furthest edge
26 */
27 public MyFurSet(int id ,int totaltrj ,MyEdge FurthestEdge) {
28     this.id = id;
29     this.totaltrj = totaltrj;
30     this.FurthestEdge = FurthestEdge;
31     set = new HashSet<MyVertex>();
32     ordered = new ArrayList<ArrayList<Integer>>();
33     for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
34     closeID = new ArrayList<Integer>(totaltrj);
35     closeDist = new ArrayList<Double>(totaltrj);
36     for (int i = 0;i <totaltrj;i++) {
37         closeID.add(-1);
38         closeDist.add(Double.MAX_VALUE);
39     }
40 }
41
42 /**
43 * set a vertex into the set
44 * @param v : vertex to be added to the set
45 */
46 public void add(MyVertex v) {
47     set.add(v);
48 }
49
50 /**
51 * check whether vertex v is a member of the set
52 * @param v : vertex to be checked
53 * @return true if v is a member of the set , false otherwise
54 */
55 public boolean contains(MyVertex v) {
56     return this.set.contains(v);
57 }
58 }
```