

SKRIPSI

**PORTING APLIKASI BERBASIS PHP KE PLAY
FRAMEWORK (STUDI KASUS: KIRI *FRONT-END*)**



STEVEN SUTANA

NPM: 2012730046

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2016**

UNDERGRADUATE THESIS

**PORING APPLICATION-BASED 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
2016**

LEMBAR PENGESAHAN

PORING APLIKASI BERBASIS PHP KE PLAY FRAMEWORK (STUDI KASUS: KIRI *FRONT-END*)

STEVEN SUTANA

NPM: 2012730046

Bandung, «tanggal» Mei 2016

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 APLIKASI BERBASIS PHP KE 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» Mei 2016

Meterai

Steven Sutana
NPM: 2012730046

ABSTRAK

KIRI merupakan sebuah aplikasi yang membantu pengguna dalam menggunakan kendaraan umum. Saat ini, KIRI dikembangkan dalam bahasa PHP. Namun bahasa PHP tidak cocok untuk proyek besar. Masalah yang sering dijumpai pada bahasa PHP adalah tidak ada deklarasi tipe variabel.

Bahasa pemrograman Java merupakan bahasa pemrograman yang *statically-typed*, yang berarti semua varibel harus dideklarasikan terlebih dahulu sebelum digunakan. Dengan menggunakan Java, kesalahan tipe data dapat dicegah. Oleh karena itu, dibuatlah KIRI dalam bahasa Java dengan menggunakan Play Framework.

Pengujian aplikasi KIRI dilakukan dengan membandingkan perbedaan waktu eksekusi PHP dengan Play Framework dari masing-masing kasus. Berdasarkan hasil pengujian, aplikasi KIRI dengan menggunakan Play Framework dapat berjalan dengan baik dan bahkan lebih cepat waktu eksekusinya dalam memuat konten dinamis dibandingkan dengan aplikasi KIRI dengan menggunakan PHP.

Kata-kata kunci: KIRI Travel, Teknik Informatika UNPAR, Play Framework, Chrome DevTools, Zurb Foundation, OpenLayers

ABSTRACT

KIRI is an application that helps the user in using public transport. Currently, KIRI developed in PHP language. However, the PHP language is not suitable for large projects. The common problem there is no variable type declaration.

The Java programming language is statically-typed, which means that all variables must first be declared before they can be used. By using Java, data type errors can be prevented. Therefore, KIRI made in Java language using Play Framework.

The testing of KIRI is done by comparing execution time between PHP and Play Framework of each case. The results of KIRI testing prove that KIRI made using Play Framework can run well and even faster when load dynamic content compared to KIRI using PHP.

Keywords: KIRI Travel, Teknik Informatika UNPAR, Play Framework, Chrome DevTools, Zurb Foundation, OpenLayers

Teknik Informatika UNPAR dan diri sendiri

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas berkat yang diberikan kepada penulis sehingga dapat menyelesaikan tugas akhir dengan judul **Porting PHP menjadi Play Framework (Studi Kasus : KIRI *Front-End*)** dengan baik dan tepat waktu. Penulis juga berterima kasih kepada pihak-pihak yang telah memberikan dukungan dan bantuan kepada penulis dalam menyelesaikan tugas akhir ini, yaitu:

1. Orang tua dan keluarga yang selalu memberikan dukungan kepada penulis.
2. Bapak Pascal Alfadian sebagai dosen pembimbing yang telah membimbing penulis hingga dapat menyelesaikan tugas akhir ini.
3. Teman-teman Teknik Informatika UNPAR angkatan 2012 yang telah berbagi ilmu kepada penulis.
4. Pihak-pihak lain yang belum disebutkan, yang berperan dalam penyelesaian tugas akhir ini.

Akhir kata, penulis berharap agar tugas akhir ini dapat bermanfaat bagi pembaca yang hendak melakukan penelitian dan pengembangan yang terkait dengan tugas akhir ini.

Bandung, Mei 2016

Penulis

DAFTAR ISI

KATA PENGANTAR	xv
DAFTAR ISI	xvii
DAFTAR GAMBAR	xix
DAFTAR TABEL	xx
1 PENDAHULUAN	1
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
2 LANDASAN TEORI	5
2.1 Java Database Connectivity (JDBC)	5
2.1.1 Injeksi SQL	5
2.2 Play Framework	7
2.2.1 Struktur Play Framework	7
2.2.2 Web Service	10
2.2.3 JSON	10
2.2.4 Konfigurasi JDBC	11
2.2.5 Internationalization	11
2.2.6 Logging	12
2.3 OpenLayers	12
2.3.1 Bing Maps	13
2.3.2 Draw Poin	13
2.4 Zurb Foundation	14
2.4.1 Sistem Grid	14
2.5 Chrome DevTools	15
2.5.1 Elements	15
2.5.2 Network	16
2.5.3 Sources	19
2.5.4 Timeline	19
2.5.5 Profile	20
3 ANALISIS	23
3.1 Analisis Sistem Kini	23
3.1.1 Peta	24
3.1.2 Form Samping	25
3.1.3 Internationalization	35

3.1.4	Pencarian Rute	36
3.2	Analisis Sistem Usulan	42
3.2.1	Peta	42
3.2.2	Form Samping	42
3.2.3	Internationalization	44
3.2.4	Pencarian Rute	45
3.3	Analisis Use Case	45
3.3.1	Skenario Use Case	46
3.4	Analisis Activity Diagram	48
3.5	Analisis Kelas	50
4	PERANCANGAN	53
4.1	Diagram Kelas Rinci	53
4.2	Perancangan Antarmuka	61
4.2.1	Peta	62
4.2.2	Form Samping	62
5	IMPLEMENTASI DAN PENGUJIAN	65
5.1	Implementasi	65
5.1.1	Lingkungan Implementasi	65
5.1.2	Hasil Implementasi	65
5.2	Pengujian	67
5.2.1	Pengujian Fungsional	67
5.2.2	Pengujian Eksperimental	68
6	KESIMPULAN DAN SARAN	71
6.1	Kesimpulan	71
6.2	Saran	71
DAFTAR REFERENSI		73
A	KODE PROGRAM PHP	75
B	KODE PROGRAM <i>Controller</i>	81
C	KODE PROGRAM <i>Models</i>	91
D	KODE PROGRAM <i>View</i>	95

DAFTAR GAMBAR

2.1	Struktur Play Framework [1]	8
2.2	Contoh <i>Routes</i> [1]	8
2.3	Sistem <i>grid</i> kosong sebelum memulai desain.	14
2.4	Contoh pembagian sistem <i>grid</i>	15
2.5	Panel Elements	16
2.6	Panel Network	17
2.7	Contoh Header	17
2.8	Contoh peninjauan sumber daya tersedia	18
2.9	Contoh peninjauan sumber daya tidak tersedia	18
2.10	Contoh Response	18
2.11	Contoh Cookies	19
2.12	Panel Sources dengan menyalakan Conditional <i>breakpoints</i>	19
2.13	Panel Timeline saat melakukan pencarian rute	20
2.14	Contoh CPU <i>profiler</i>	20
2.15	Contoh Heap <i>profiler</i>	21
3.1	Halaman Utama KIRI	23
3.2	Peta pada KIRI	24
3.3	Form pada KIRI	25
3.4	Dropdown Menu Kota pada KIRI	26
3.5	Dropdown Menu Bahasa pada KIRI	28
3.6	Input User(Nama Tempat)	29
3.7	Input User(Klik pada peta)	30
3.8	Contoh Pencarian Rute pada KIRI	33
3.9	Contoh Rute Alternatif pada KIRI	33
3.10	Use Case Diagram KIRI	46
3.11	Activity Diagram KIRI	49
3.12	Diagram Kelas Analisis KIRI	50
4.1	Diagram Kelas Rinci	61
4.2	Halaman Utama KIRI	62
4.3	Form pada KIRI	63
4.4	Contoh Pencarian Rute pada KIRI	64
5.1	Halaman Utama KIRI	66
5.2	Contoh Pencarian Rute pada KIRI	66
5.3	Contoh Rute Alternatif pada KIRI	67
5.4	Grafik Perbandingan Kecepatan	69

DAFTAR TABEL

5.1 Tabel Pengujian Fungsional	68
5.2 Tabel Pengujian Eksperimental	68

BAB 1

PENDAHULUAN

1.1 Latar Belakang

KIRI [2] adalah sebuah aplikasi yang membantu pengguna dalam menggunakan kendaraan umum. Peran KIRI sangat sederhana, yaitu jika diberitahu di mana lokasi sekarang dan kemana lokasi tujuan, lalu KIRI akan memberitahu bagaimana cara sampai ke lokasi tujuan dengan menggunakan kendaraan umum.

Kode KIRI [3] menggunakan bahasa PHP. Bahasa PHP [4] 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 deklarasi tipe variabel. Tipe variabel pada PHP ditentukan pada saat *runtime* tergantung pada konteks apa variabel tersebut digunakan. Hal ini memicu banyaknya kesalahan tipe data dalam aplikasi.

Bahasa pemrograman Java [5] merupakan bahasa pemrograman yang *statically-typed*, yang berarti semua varibel harus dideklarasikan terlebih dahulu sebelum digunakan. Tipe data variabel menentukan nilai yang dapat dimasukkan ke variabel tersebut dan operasi yang dapat dilakukan pada variabel tersebut. Dengan menggunakan Java, kesalahan tipe data dapat dicegah. Kesalahan tipe data dapat disebabkan oleh perbedaan tipe untuk konstanta program, variabel, dan fungsi. Sebagai contoh tipe data yang dibutuhkan berupa Integer tetapi dalam program tipe data yang dimasukkan berupa String.

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[1].

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 yang dibuat dalam bahasa 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

Rumusan masalah yang akan dibahas pada skripsi ini adalah:

1. Bagaimana memahami dan menganalisis kode KIRI yang sudah ada?
2. Bagaimana melakukan porting kode KIRI dalam bahasa PHP menjadi Play Framework (Java) ?
3. Bagaimana perbandingan performa KIRI dalam bahasa PHP dengan KIRI dalam bahasa Java dari segi kecepatan?

1.3 Tujuan

Tujuan dengan membahas skripsi ini adalah:

1. Memahami dan menganalisis kode KIRI.
2. Melakukan porting kode KIRI dalam bahasa PHP menjadi Play Framework (Java).
3. Membandingkan performa KIRI dalam bahasa PHP dengan KIRI dalam bahasa Java dari segi kecepatan.

1.4 Batasan Masalah

Beberapa batasan dengan skripsi ini adalah:

1. Play Framework yang digunakan adalah versi 2.4.3.
2. Kode KIRI yang digunakan adalah versi commit ‘b650bfa’ yang tersedia di Github pascalalfadian[3].

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.
4. Bab 4: Perancangan, yaitu membahas mengenai perancangan aplikasi, meliputi diagram kelas rinci beserta deskripsi kelas dan fungsinya dan perancangan antarmuka aplikasi.
5. Bab 5: Implementasi dan Pengujian, yaitu membahas mengenai implementasi dan pengujian aplikasi, meliputi lingkungan implementasi, hasil implementasi, pengujian fungsional, dan pengujian eksperimental.
6. Bab 6: Kesimpulan dan Saran, yaitu berisi kesimpulan dari hasil pembangunan aplikasi berserta saran untuk pengembangan berikutnya.

BAB 2

LANDASAN TEORI

2.1 Java Database Connectivity (JDBC)

JDBC API [6] adalah standar industri untuk konektivitas basisdata-independen antara bahasa pemrograman Java dan berbagai basisdata. JDBC API menyediakan API untuk akses basisdata berbasis SQL. JDBC API memungkinkan untuk melakukan tiga hal:

- Membangun koneksi dengan basisdata atau mengakses sumber data.
- Mengirim *statement* SQL.
- Mengolah hasil dari *statement* SQL.

2.1.1 Injeksi SQL

Injeksi SQL [7] adalah sebuah teknik yang menyalahgunakan sebuah celah keamanan yang terjadi dalam lapisan basisdata sebuah aplikasi. Celah ini terjadi ketika masukan pengguna tidak disaring secara benar dari karakter-karakter pelolos bentukan string yang diimbuhkan dalam *statement* SQL. Hal ini adalah sebuah contoh dari sebuah kategori celah keamanan yang lebih umum yang dapat terjadi setiap kali sebuah bahasa pemrograman atau skrip ditambahkan di dalam bahasa pemrograman yang lain. Bentuk-bentuk celah keamanan injeksi SQL:

- Karakter-karakter pelolos yang tidak disaring secara benar.

Bentuk injeksi SQL ini terjadi ketika masukan pengguna tidak disaring dari karakter-karakter pelolos dan kemudian diteruskan ke dalam sebuah *statement* SQL. Baris kode berikut menggambarkan celah keamanan:

```
1| statement := "SELECT * FROM pengguna WHERE nama = '" + namaPengguna + "';"
```

Jika variabel **namaPengguna** dirangkai sedemikian rupa oleh pengguna yang bermaksud buruk, *statement* SQL tersebut bisa melakukan lebih daripada yang pengarangnya maksudkan.

Sebagai contoh, mengeset variabel **namaPengguna** sebagai:

```
1|      'a' or 't'='t'
```

menjadikan *statement* SQL ini oleh bahasa yang memuatnya:

```
1|      SELECT * FROM pengguna WHERE nama = 'a' or 't'='t';
```

Jika kode ini akan digunakan dalam sebuah prosedur untuk melakukan otentikasi, maka contoh ini dapat dipakai untuk memaksakan pemilihan sebuah nama pengguna yang sah karena evaluasi *statement* SQL tersebut akan selalu bernilai benar.

Nilai `namaPengguna` berikut ini pada *statement* di atas akan menyebabkan dihapusnya tabel “pengguna”:

```
1|     a';DROP TABLE pengguna; SELECT * FROM data WHERE nama LIKE '%
```

Masukan ini menjadikan *statement* akhir SQL sebagai berikut:

```
1|     SELECT * FROM pengguna WHERE nama = 'a';DROP TABLE pengguna;
```

- Penanganan tipe data yang tidak benar.

Bentuk injeksi SQL ini terjadi ketika sebuah unsur masukan pengguna tidak diperiksa batasan-batasan tipenya. Ini dapat terjadi ketika sebuah unsur numerik akan digunakan dalam sebuah *statement* SQL, tetapi pemrogram tidak melakukan pemeriksaan untuk memastikan bahwa masukan pengguna adalah numerik. Sebagai contoh:

```
1|     statement := "SELECT * FROM data WHERE id = " + variabel_a + ";"
```

Terlihat jelas dari *statement* ini, `variabel_a` menjadi sebuah nomor yang berhubungan dengan unsur “id”. Namun begitu, jika pada kenyataannya sebuah string, maka pengguna akhir dapat memanipulasi *statement* tersebut sesukanya, dan karena itu mengabaikan kebutuhan akan karakter-karakter pelolos. Sebagai contoh, mengeset `variabel_a` sebagai

```
1|     1;DROP TABLE pengguna
```

akan menghapus tabel “pengguna” dari basis data karena hasil akhir SQL-nya akan menjadi sebagai berikut:

```
1|     SELECT * FROM data WHERE id = 1;DROP TABLE pengguna;
```

- Celah keamanan dalam server basis data.

Terkadang celah keamanan dapat berada dalam perangkat lunak server basis data itu sendiri, seperti yang terjadi pada fungsi-fungsi `real_escape_chars()` di server MySQL.

Prepared Statement

Prepared Statement adalah fitur yang digunakan untuk mengeksekusi *statement* basisdata berulang kali dengan tingkat efisiensi yang tinggi. Biasanya digunakan dengan *statement* SQL seperti `query` atau `update`, *statement* mengambil bentuk *template* yang diganti dengan nilai-nilai konstan tertentu yang diganti selama setiap eksekusi. Dengan menggunakan Prepared Statement, waktu eksekusi untuk *statement* SQL berkurang dan *statement* SQL dikirim ke basisdata langsung, dimana dilakukan kompilasi. Akibatnya, Prepared Statement berisi tidak hanya sebuah *statement* SQL, tetapi *statement* SQL yang telah dilakukan pre-kompilasi.

JDBC menyediakan kelas `PreparedStatement` untuk mendukung Prepared Statement. Hal pertama yang dilakukan adalah menyiapkan koneksi untuk *database*. Memanggil objek `DB` yang disediakan oleh Play Framework untuk mendapatkan koneksi ke *database*. Dengan memanggil objek `DB`, koneksi ke database sudah disediakan sesuai dengan konfigurasi database. Untuk menggunakan `PreparedStatement`, pertama membuat objek yang berupa `PreparedStatement` dengan menyiapkan *statement* SQL tetapi nilai yang ingin dimasukkan ke *statement* SQL diberi ‘?’ terlebih dahulu. Kelas `PreparedStatement` mempunyai beberapa *method* dengan rincian:

- `public void setString(int parameterIndex, String x)`

Berfungsi untuk menentukan nilai yang ingin dimasukkan.

Parameter:

- **parameterIndex** Indeks yang ingin dimasukkan nilainya pada *statement* SQL.
- **String x** Nilai yang ingin dimasukkan.

Kembalian: Tidak ada.

- **public void executeUpdate()**

Berfungsi untuk eksekusi statement SQL yang merupakan statement SQL manipulasi data.

Kembalian: Tidak ada.

- **public ResultSet executeQuery()**

Berfungsi untuk eksekusi statement SQL.

Kembalian: Objek ResultSet yang merupakan objek yang dibuat dari *statement* SQL.Setelah selesai dengan koneksi *database*, koneksi *database* harus ditutup.

Contoh kode PreparedStatement dapat dilihat pada kode listing 2.1.

Listing 2.1: Contoh PreparedStatement

```

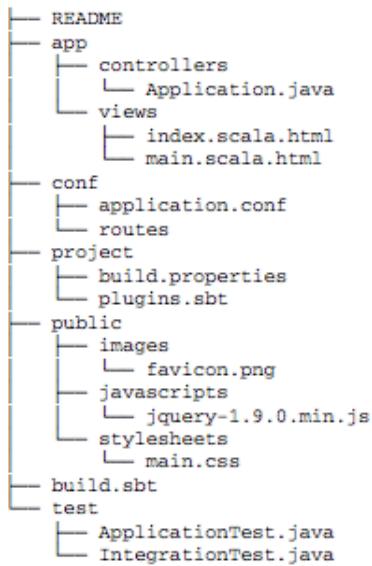
1 Connection connection = DB.getConnection();
2 try {
3     java.sql.PreparedStatement stmt = connection.prepareStatement(
4         "INSERT INTO cache(type, cacheKey, cacheValue) VALUES (?, ?, ?)");
5     stmt.setString(1, type);
6     stmt.setString(2, key);
7
8     stmt.setString(3, String.valueOf(value));
9     stmt.executeUpdate();
10    connection.close();
11 } catch (Exception e) {
12     log_error("Warning: Can't store cache "+e.getMessage());
13 }
```

2.2 Play Framework

2.2.1 Struktur Play Framework

Play Framework [1] merupakan *framework* untuk aplikasi web dengan menggunakan bahasa Java dan Scala. Play Framework tidak sepenuhnya menggunakan bahasa Java, tetapi juga bahasa Scala. Terdapat bahasa Scala bukan berarti harus mempelajari bahasa Scala karena dalam Play 2 dilengkapi dengan Java API yang lengkap, memberikan opsi untuk memilih bahasa pemrograman yang cocok. Play Framework mempunyai antarmuka yang sederhana, nyaman, fleksibel, dan kuat. Referensi [1] 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.

Play Framework memiliki struktur yang dapat dilihat pada gambar 2.1.



Gambar 2.1: Struktur Play Framework [1]

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 GET dan akan memanggil *method* list pada kelas Products di controllers.

Gambar 2.2: Contoh *Routes* [1]

Direktori public

Direktori *public* mengandung semua sumber daya yang disediakan langsung tanpa melalui proses lebih dahulu. Direktori *public* biasanya mengandung file gambar, *stylesheets*, JavaScript, dan halaman statis HTML. Contoh direktori *public* dapat dilihat pada gambar 2.1. Di contoh tersebut, file *images/favicon.png* dapat diakses melalui URL `@routes.Assets.versioned("images/favicon.png")`.

Direktori app.

Direktori *app* merupakan direktori utama pada aplikasi. Direktori *app* berisi kode aplikasi dan berbagai kebutuhan untuk menyusun aplikasi, seperti kode 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*. 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 2.2. *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 2.3, *method ok* membangun HTTP *response* yang mengandung *response body* sebagai hasil dari *template list*. *Method ok* menerima parameter berupa “Hello World”. Pada [1], hasil *Result* memiliki tipe **static**, tetapi pada Play Framework 2.4 **static** dihilangkan.

Listing 2.2: Contoh View (index.scala.html)

```

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 2.3: Contoh Controller

```

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

```

Beberapa alternatif hasil Results 2.3, 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.

2.2.2 Web Service

Play Framework menyediakan *library* untuk memanggil layanan HTTP dari luar aplikasi Play yang disebut *library WS*. Library WS memanggil layanan HTTP secara *asynchronous*. Tahap penting yang dilakukan Library WS adalah:

1. Melakukan permintaan *web service*, dan
2. Memproses respon yang didapat.

Pertama menentukan hasil respon yang ingin didapat memiliki tipe data apa, seperti String atau JSON. Setelah itu, menentukan URL halaman web yang ingin diproses. Lalu dapat menambahkan query parameter (jika ada). Query parameter menerima dua input, yaitu key dan value. Terakhir menerima respon dengan melakukan pemetaan dari F.Promise menjadi WSResponse. Kelas F merupakan kumpulan fungsi pembantu dalam Play Framework .

2.2.3 JSON

Dalam aplikasi KIRI, data yang disediakan maupun data yang didapat dari Web Service berupa JSON. Play menggunakan Jackson, sebuah library JSON yang mengubah objek ke atau dari JSON.

Menulis JSON

Untuk menulis keluaran JSON, Play menggunakan ObjectNode. ObjectNode dapat menulis JSON dengan menerima dua parameter, yaitu *key* dan *value*. Jika ingin memasukkan *value* dengan tipe Object, maka *value* tersebut harus diubah terlebih dahulu menjadi format JSON yang benar. Contoh dapat dilihat pada kode listing 2.4.

Listing 2.4: Contoh menulis JSON menggunakan ObjectNode

```

1  ObjectNode objectNode = Json.newObject();
2  objectNode.put(Constants.PROTO_STATUS, Constants.proto_status_ok);
3
4
5  ArrayList<ArrayList<Object>> route_output = new ArrayList<ArrayList<Object>>();
6  objectNode.put(Constants.PROTO_STEPS, Json.toJson(route_output));

```

Mendapatkan JSON

Untuk mendapatkan JSON, pertama menguraikan String menjadi JsonNode, lalu mendapatkan *value* dari *key* dengan memanggil *method* get(index) atau findPath(index) dimana *index* tersebut adalah *key* yang ingin dicari. Contoh dapat dilihat pada kode listing 2.5.

Listing 2.5: Contoh mendapatkan JSON menggunakan JsonNode

```
1| JsonNode json_result = Json.parse(result);
2| json_result.findPath("status").textValue();
3|
```

2.2.4 Konfigurasi JDBC

Play Framework mempunyai konfigurasi untuk menghubungkan basisdata dengan JDBC. Pertama menambahkan javaJdbc dan “mysql” % “mysql-connector-java” % “5.1.37” pada libraryDependencies pada file build.sbt. Lalu menambahkan:

1. db.default.driver=com.mysql.jdbc.Driver
Merupakan driver basisdata yang digunakan.
2. db.default.url=“jdbc:mysql://localhost/Skripsi”
Merupakan url dari basisdata yang digunakan dengan basisdata Skripsi.
3. db.default.username=“root”
Username yang digunakan pada basisdata.
4. db.default.password=“root”
Password yang digunakan pada basisdata.

Untuk mendapatkan konektivitas JDBC dapat dilakukan dengan cara:

Listing 2.6: Contoh mendapatkan konektivitas JDBC

```
1| Connection connection = DB.getConnection();
```

2.2.5 Internationalization

Pengguna aplikasi mungkin berasal dari beda negara dan bahasa, juga punya format yang berbeda untuk tanggal, angka, dan waktu. Kombinasi dari bahasa dan 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 2.7 dan 2.8

Listing 2.7: 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 2.8: 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 2.9. Pada [1], konfigurasi bahasa dalam satu String. Tetapi pada Play Framework 2.4 konfigurasi bahasa dalam bentuk array of String.

Listing 2.9: Konfigurasi Bahasa i18n

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

2.2.6 Logging

Logging dapat membantu dalam penggerjaan aplikasi KIRI. Kita dapat memonitor, melihat kesalahan program, dan memperbaiki program. Play menyediakan API untuk logging yang dapat diakses melalui objek Logger. Contoh logging dapat dilihat pada kode listing 2.10.

Listing 2.10: Contoh logging menggunakan objek Logger

```
1 |     Logger.debug("mode search json : " + json_output.toString());
```

2.3 OpenLayers

OpenLayers [8] 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 merupakan hal penting pada aplikasi KIRI 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.

2.3.1 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 2.11.

Listing 2.11: Penggunaan Bing Maps pada Openlayers

```

1 var mapLayer = new ol.layer.Tile(
2 {
3   source : new ol.source.BingMaps(
4     {
5       key : 'AuV7xD6_UMiQ5BLoZr0xkpjLpzWqMT55772Q8XtLIQuDebHPKiNXSIZXxEr1GA',
6       imagerySet : 'Road'
7     }
8 ));
```

2.3.2 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 2.12. Setelah itu, menggambar pada peta seperti pada figur 2.13.

Listing 2.12: 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 2.13: Menambahkan vektor pada peta

```

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

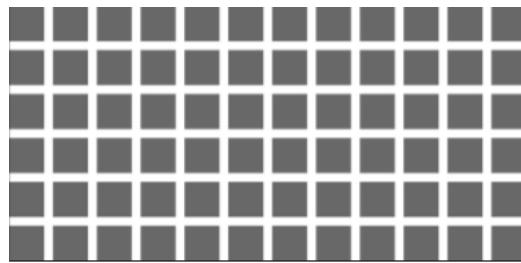
```

2.4 Zurb Foundation

Zurb Foundation [9] merupakan sebuah *toolkit* yang membantu untuk desain dan mengembangkan sebuah halaman *web*. Zurb Foundation menggunakan sistem *grid*, banyak komponen CSS serta JavaScript.

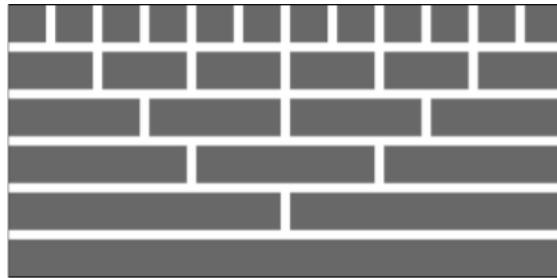
2.4.1 Sistem Grid

Sistem *grid* pada Zurb Foundation adalah semacam *spreadsheet*, graf atau tabel yang digunakan untuk mengatur tampilan HTML. Gambaran sistem *grid* dapat dilihat pada gambar 2.3.



Gambar 2.3: 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 2.4.



Gambar 2.4: 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 2.14.

Listing 2.14: Menggunakan kelas yang sudah disediakan dari Foundation

```
1| 
```

2.5 Chrome DevTools

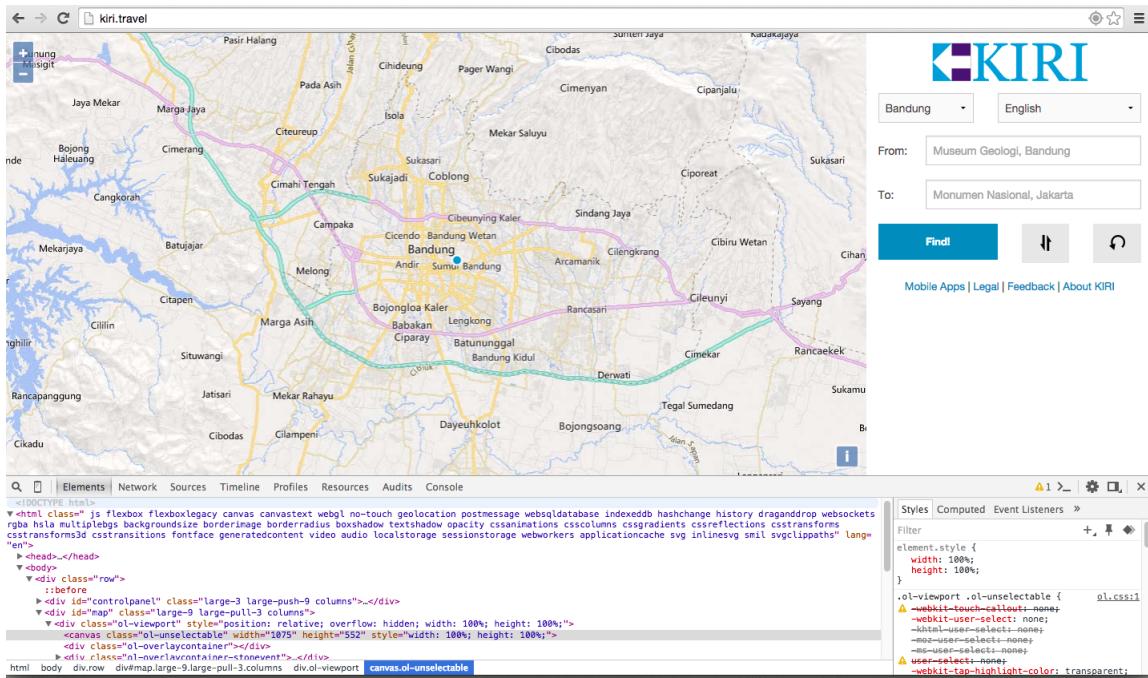
Chrome DevTools[10] merupakan perangkat untuk memperhatikan dan melakukan *debugging* halaman web yang terdapat pada *browser* Google Chrome. DevTools dapat digunakan secara efisien untuk memeriksa tampilan, mengatur *breakpoints* JavaScript, dan optimasi kode. DevTools dapat diakses dengan melakukan 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.

2.5.1 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 diubah¹. Sebagai contoh pada gambar 2.5, pemeriksaan elemen akan memperlihatkan dua bagian, yaitu DOM dan CSS yang digunakan pada DOM.



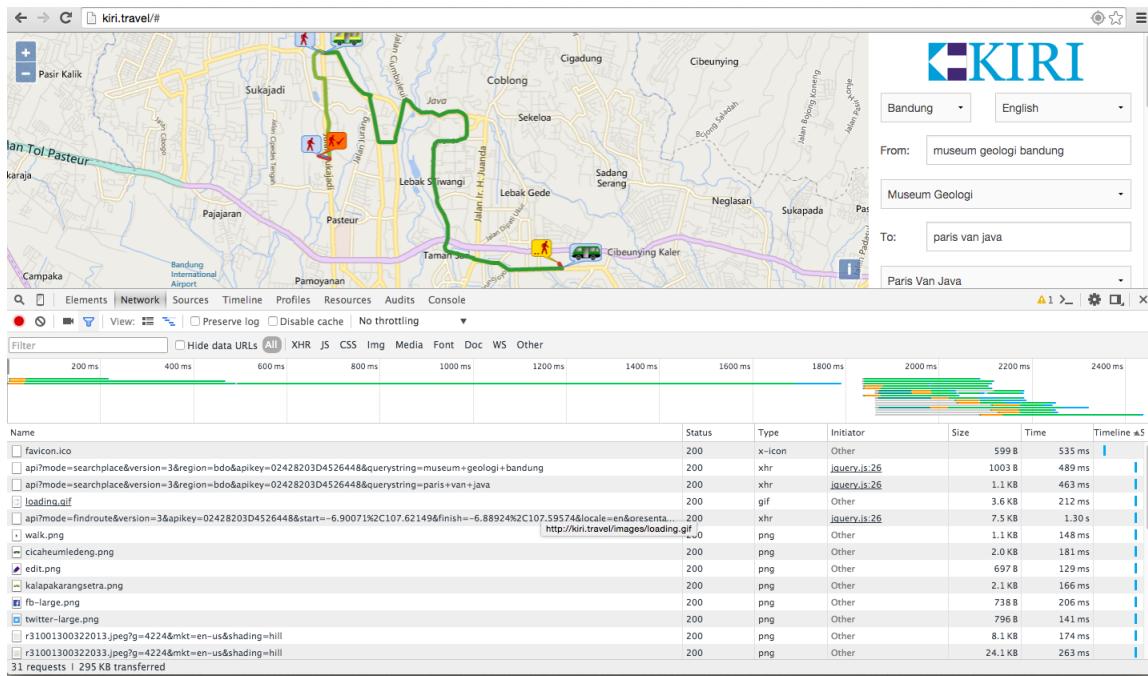
Gambar 2.5: Panel Elements

2.5.2 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 2.6, 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.

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



Gambar 2.6: 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.



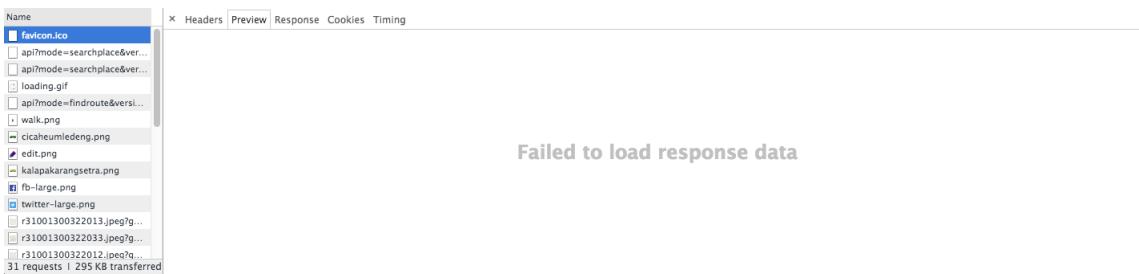
Gambar 2.7: Contoh Header

- **Preview**

Preview menampilkan peninjauan sumber daya jika sumber daya tersebut tersedia. Gambar 2.8 menunjukkan adanya peninjauan sumber daya, sedangkan gambar 2.9 menunjukkan tidak ada peninjauan sumber daya.



Gambar 2.8: Contoh peninjauan sumber daya tersedia



Gambar 2.9: Contoh peninjauan sumber daya tidak tersedia

• Response

Response menampilkan respon dari sumber daya yang dipilih. Gambar 2.10 menunjukkan respon dari sumber daya.



Gambar 2.10: 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 2.11 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

Gambar 2.11: Contoh Cookies

2.5.3 Sources

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

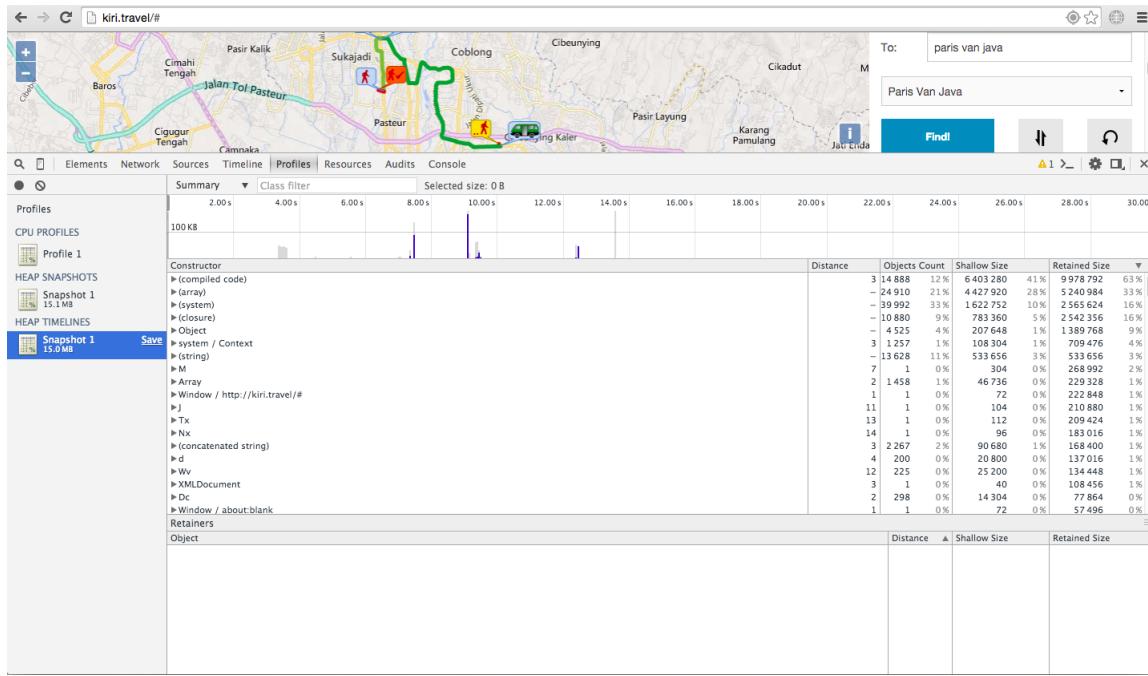
Gambar 2.12: Panel Sources dengan menyalakan Conditional *breakpoints*

2.5.4 Timeline

Panel Timeline memberikan gambaran lengkap waktu yang dibutuhkan semua sumber daya yang dibutuhkan ketika memuat dan menggunakan halaman web. Sebagai contoh pada gambar 2.13, panel Timeline memberikan gambaran lengkap ketika melakukan pencarian rute.

²Terdapat 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*

- Heap *profiler* menunjukkan distribusi memori oleh JavaScript dan DOM yang berhubungan pada halaman web. Gambar 2.15 menunjukkan disribusi memori.



Gambar 2.15: Contoh Heap *profiler*

- JavaScript *profiler* menunjukkan dimana waktu eksekusi dihabiskan pada skrip.

BAB 3

ANALISIS

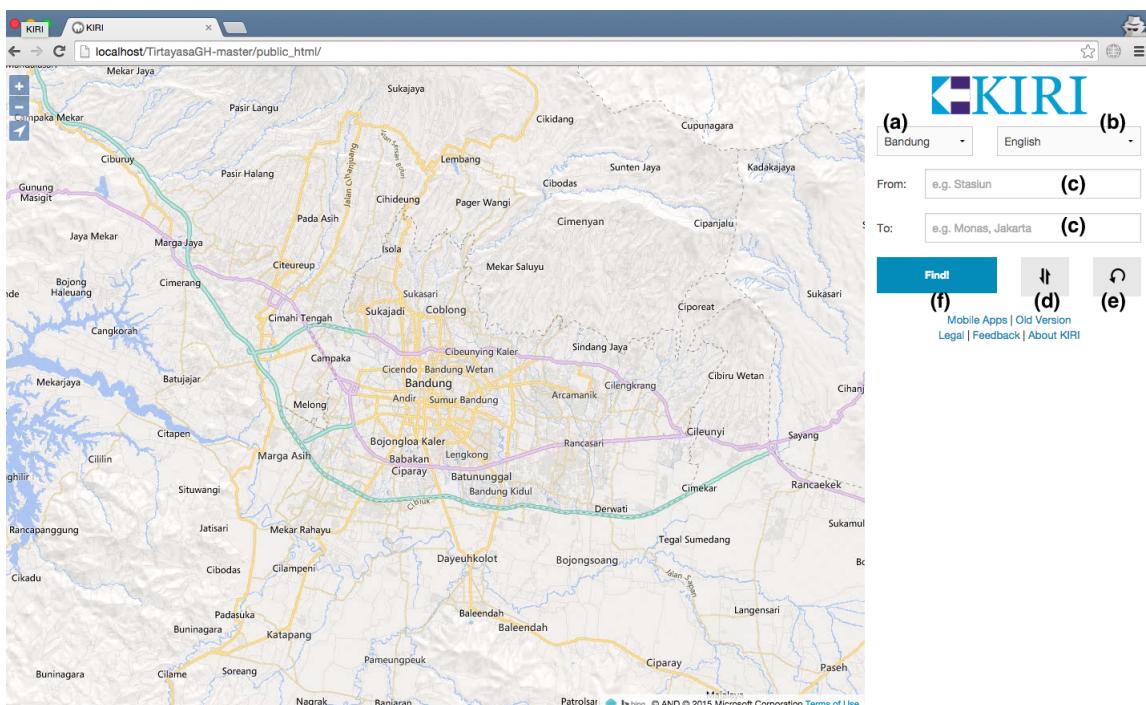
3.1 Analisis Sistem Kini

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

1. Peta

2. Form Samping yang terdiri dari beberapa bagian, yaitu:

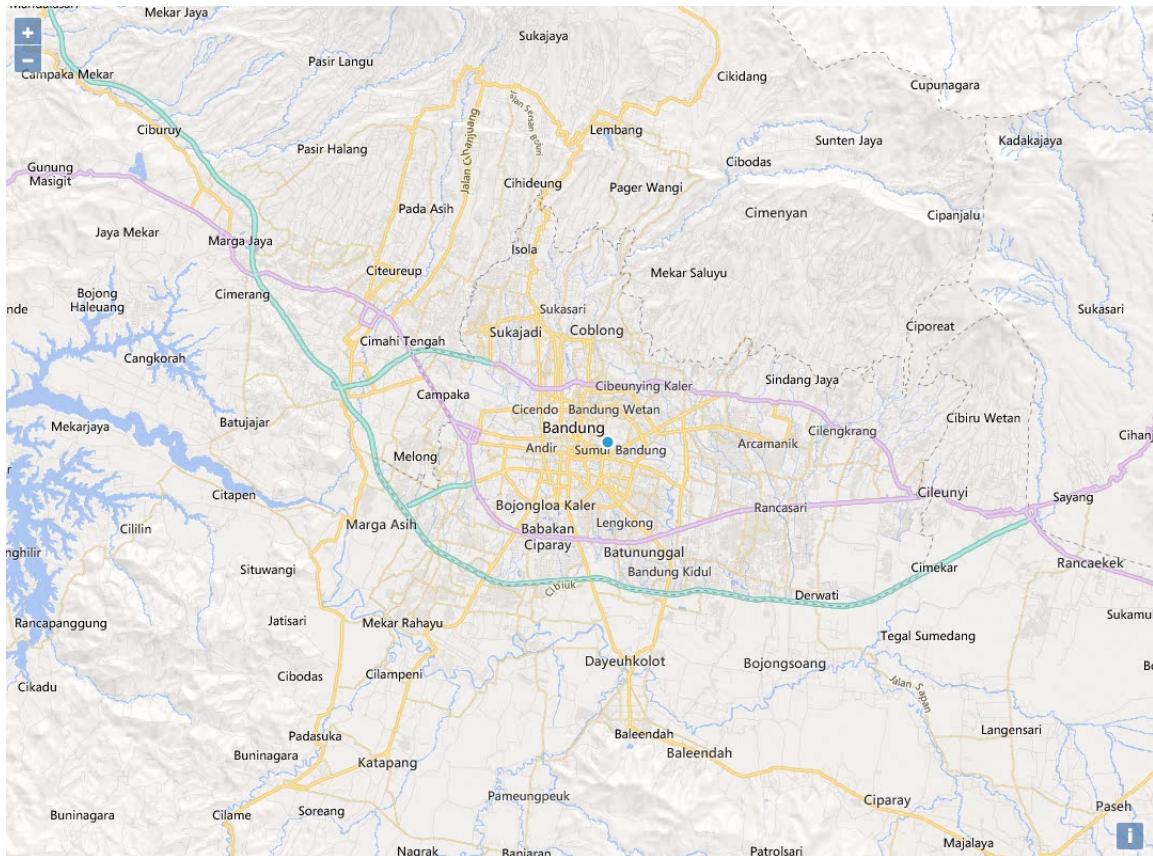
- (a) Dropdown Menu Kota, untuk memilih kota tujuan untuk pencarian rute.
- (b) Dropdown Menu Bahasa, untuk memilih bahasa yang akan ditampilkan pada aplikasi.
- (c) Textfield “From” & “To”, untuk menerima masukan pengguna.
- (d) Tombol Swap, untuk menukar isi masukan pengguna antara Textfield “From” dan Textfield “To”.
- (e) Tombol Reset, untuk menyetel ulang peta dan masukan pengguna.
- (f) Tombol Find, untuk mencari pencarian rute.



Gambar 3.1: Halaman Utama KIRI

3.1.1 Peta

Peta pada aplikasi KIRI berfungsi untuk menerima masukan pengguna berupa klik, menampilkan titik start dan finish, serta menampilkan hasil pencarian rute.



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. Contoh deklarasi peta dapat dilihat 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 : 'AuV7xD6_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

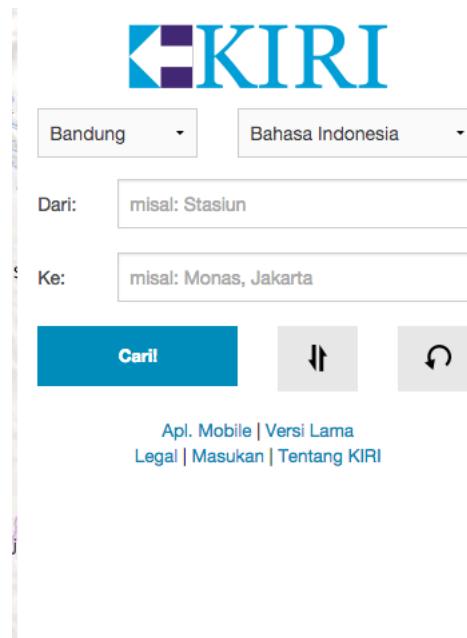
```

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

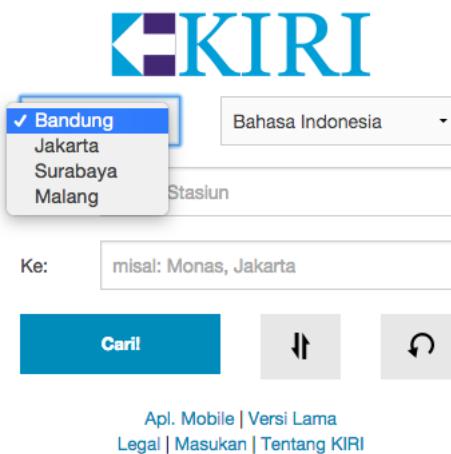
Form yang terdapat pada halaman utama KIRI (Gambar 3.3) terdiri dari:



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 variabel proto_region_KOTA dimana KOTA adalah kota yang ada pada KIRI. Setiap proto_region_KOTA merupakan array dengan indeks:

1. *lat* sebagai garis lintang,
2. *lon* sebagai garis bujur,
3. *zoom* sebagai tingkat *zoom* untuk memperbarui peta,
4. *name* untuk menampilkan nama kota,
5. *searchplace_regex* untuk pencarian rute pada pilihan kota.

Kode dapat dilihat 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(
        'lat' => -7.9812985,

```

```

30     'lon' => 112.6319264,
31     'radius' => 15000,
32     'zoom' => 13,
33     'searchplace_regex' => ', *(malang|mlg)$',
34     'name' => 'Malang',
35   )
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 format garis lintang dan garis bujur 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  */

```

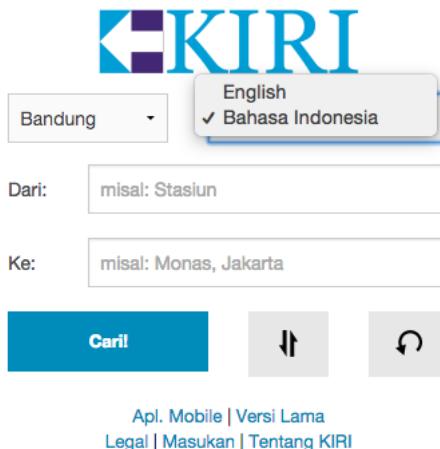
```

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* sama dengan `id` atau `locale` sama dengan `en`. Lalu menyimpan *cookies* dengan data `locale`. 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 ...

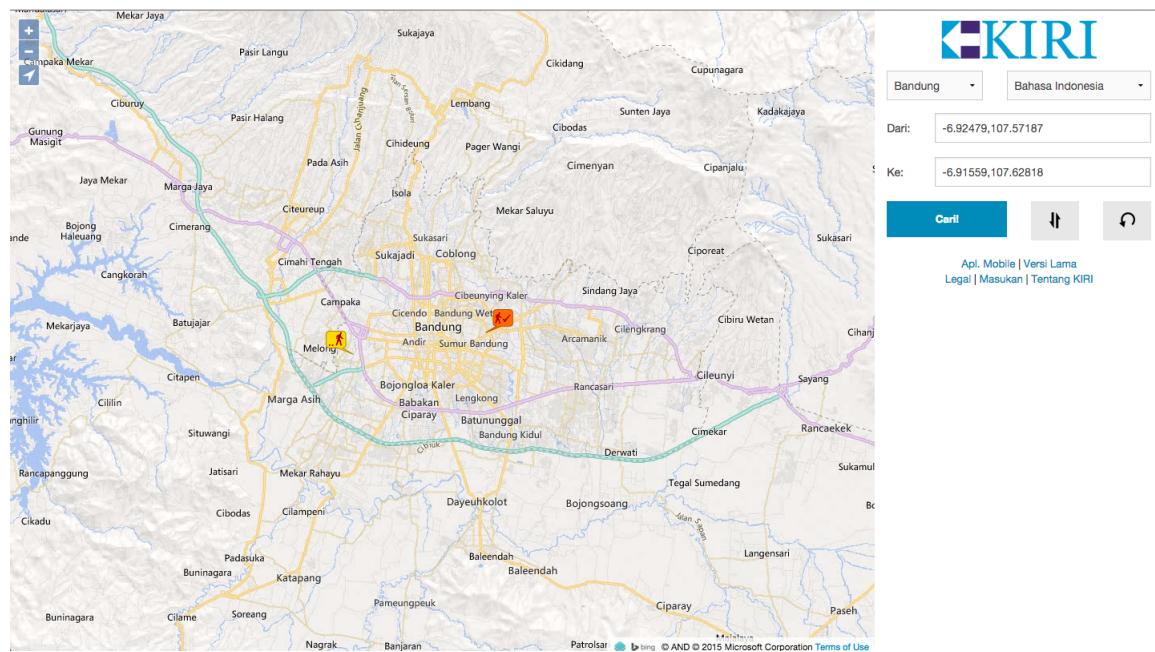
```

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

Dari:	<input type="text" value="Stasiun"/>
Ke:	<input type="text" value="Paris van Java"/>

Gambar 3.6: Input User(Nama Tempat)

2. **Textfield dengan Masukan Klik Peta**, pengguna memasukkan koordinat tempat asal dan tujuan dengan melakukan 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 geometri 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         $('#startInput').val(latLngToString(ol.proj.transform(event.coordinate, 'EPSG:3857', 'EPSG:4326')));
16     } else if ($('#finishInput').val() === '') {
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: 'images/finish.png',
23                 anchor: [0.0, 1.0]
24             })
25         }));
26         inputVectorSource.addFeature(markers['finish']);
27         $('#finishInput').val(latLngToString(ol.proj.transform(event.coordinate, 'EPSG:3857', 'EPSG:4326')));
    }
})

```

```

28    }
29 });
30 ..

```

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 mengisi 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 menukar isi dari *textfield* tempat asal dan tujuan. 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 mengisi 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 hasil pencarian rute pada peta dan memperbarui peta sesuai dengan `region` yang dipilih. 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 clearStartFinishMarker

Fungsi untuk menghapus *marker* 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*

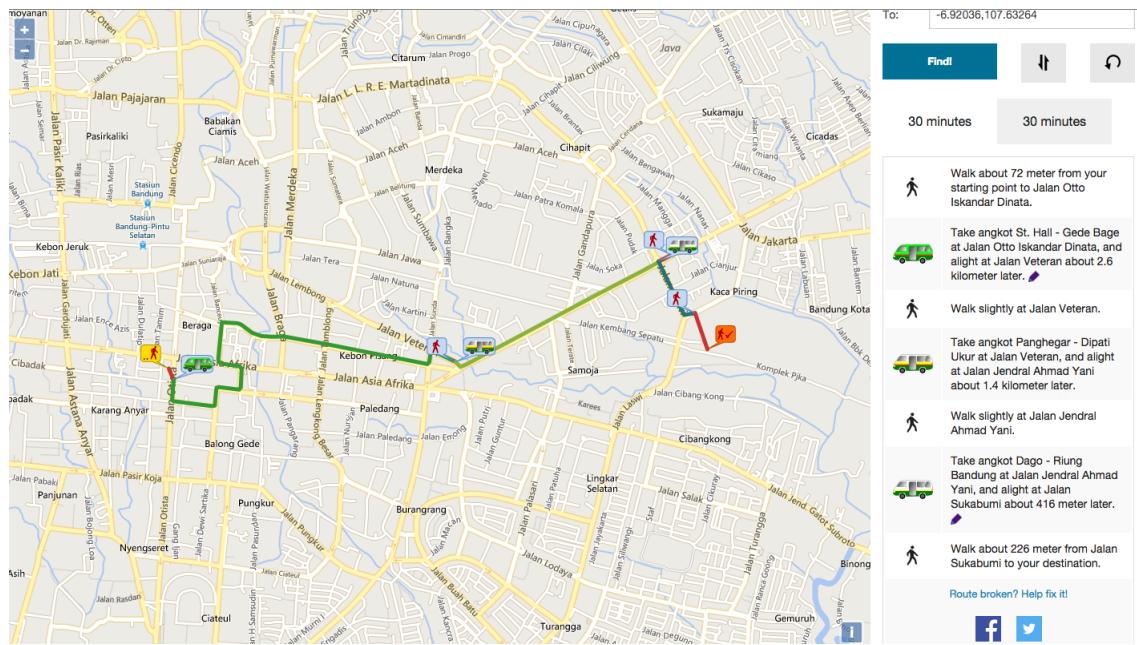
```

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|

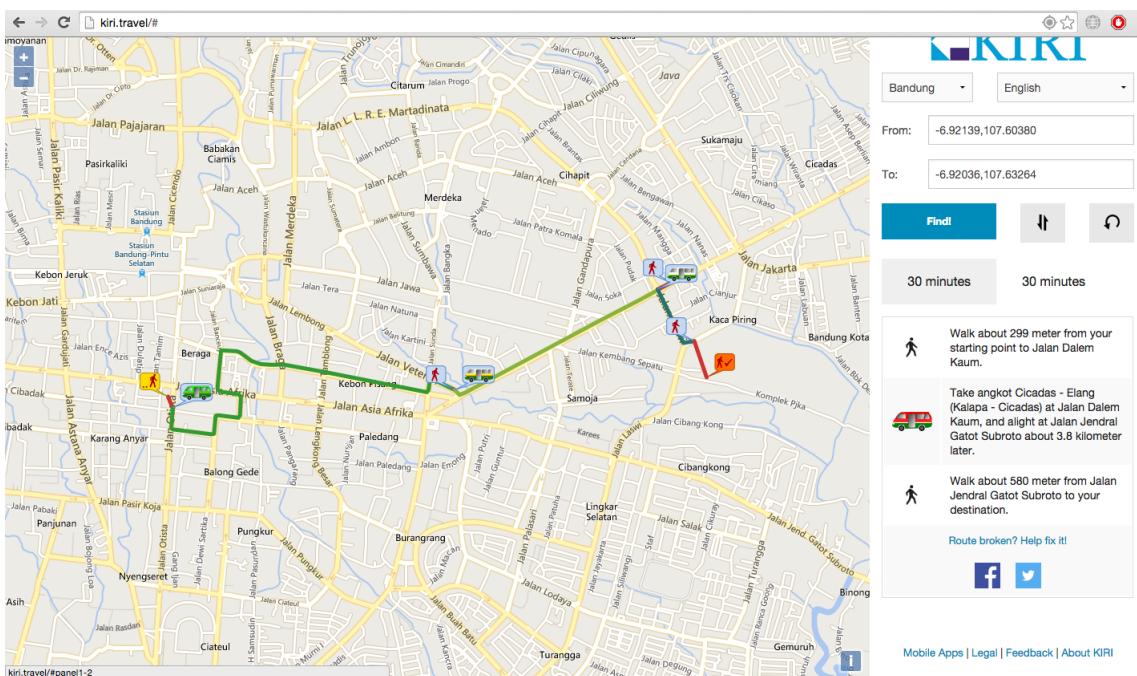
```

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 peringatan kepada pengguna. Jika tidak kosong, maka akan memunculkan peringatan ‘mohon menunggu’ karena sedang dilakukan pencarian rute. 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 jumlah `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     clearAlerts();
19     clearRoutingResultsOnTable();
20     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>' + value['location'] + '</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`. `protocol.findRoute` akan melakukan pencarian rute dengan koordinat tempat awal dan tujuan. Jika hasil `results` dari `protocol.findRoute` sama dengan ‘ok’, maka menunjukkan rute pencarian. Jika hasil result bukan ‘ok’, maka akan menampilkan peringatan ‘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 dilakukan dengan cara deklarasi semua variabel yang akan digunakan pada proses i18n terlebih dahulu, misalnya buat file dengan nama `tirtaya-sa_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 ...
```

Untuk menentukan file mana yang akan digunakan dalam proses i18n, jangan lupa untuk memuat file yang diinginkan seperti pada kode listing 3.26.

Listing 3.26: Script PHP untuk memuat file i18n

```

1 ...
2 require_once("../etc/locale/tirtayasa_$locale.php");
3 ...
```

3.1.4 Pencarian Rute

Pencarian rute pada KIRI dilakukan di *server side*, yaitu pada *file* handle.php. Langkah awal pada handle.php dapat dijelaskan sebagai berikut:

1. Memuat *file* constants.php untuk menyiapkan data yang akan dipakai pada halaman *web*.
2. Memuat utils.php untuk fungsi manipulasi data, koneksi basisdata, mengirimkan pencatatan, dan melakukan validasi.
 - Fungsi `init_mysql` untuk memuat koneksi basisdata dengan memasukkan *host*, *username*, *password*, dan basisdata yang digunakan pada MySQL.
 - Fungsi `log_statistic` untuk pencatatan data pada basisdata.
 - Fungsi `die_nice` untuk mengirimkan pesan kesalahan.
 - Fungsi `check_apikey_validity` untuk pengecekan apikey yang digunakan ada atau tidak.
 - Fungsi `retrieve_from_post` untuk mengambil data pada URL dengan memanggil method `\$_POST` dengan parameter *key* yang digunakan.
 - Fungsi `retrieve_from_get` untuk mengambil data pada URL dengan memanggil method `\$_GET` dengan parameter *key* yang digunakan.
 - Fungsi `retrieve_from_cache` untuk mengambil data pada basisdata *cache* dengan parameter ‘*type*’ dan ‘*cacheKey*’.
 - Fungsi `put_to_cache` untuk memasukkan data pada basisdata *cache* dengan parameter ‘*type*’ , ‘*cacheKey*’, dan ‘*cacheValue*’.
3. `start_working` untuk menyiapkan header pada PHP.
 - Fungsi `header('Content-Type: application/json')` untuk menghasilkan *output* berupa JSON.
 - Fungsi `header('Cache-control: no-cache')` digunakan pada HTTP/1.1 dan fungsi `header('Pragma: no-cache')` digunakan pada HTTP/1.0. Keduanya berfungsi untuk mencegah pengguna menyimpan *cache response*.

4. `init_mysql` untuk koneksi pada basisdata menggunakan MySQL.
5. Mendapatkan data `mode`, `version`, dan `apikey` dari *method POST* masing-masing dengan *parameter* ‘mode’, ‘version’, dan ‘apikey’.
6. Jika data `version` sama dengan `null`, maka isi data `version` dengan 1.
7. Melakukan pengecekan `apikey` apakah ada di basisdata atau tidak. Jika ada, cek lagi apakah ada pengecualian IP komputer pada basisdata.

Mode Findroute

Bagian ini terletak pada baris 14-181 dari handle.php (kode A.1). Langkah pertama yang dilakukan adalah memasukkan data `start`, `finish`, dan `locale` dengan mendapatkan *method POST* masing-masing dengan *parameter* ‘start’, ‘finish’, dan ‘locale’. Tetapi sebelum dimasukkan, memanggil *method addslashes* yang berfungsi jika ada karakter spesial seperti ‘ atau ” tidak dianggap sebagai suatu String yang berbeda. Setelah itu melakukan pengecekan untuk lokalisasi. Jika ada, memuat `file` (lokalisasi Bahasa Inggris atau lokalisasi Bahasa Indonesia). Setelah itu melakukan pengecekan variabel `presentation`, ada dua yaitu *mobile* atau *desktop*. Melakukan pengecekan version yang digunakan:

- Jika `version` lebih besar sama dengan 2, maka memasukkan data `count` dengan 1. Jika `presentation` sama dengan ‘`mobile`’, maka memasukkan data `count` dengan jumlah `array alternatives`. Setelah itu melakukan pengulangan sebanyak `count` yang berisi:
Menambahkan *query URL* <http://newmenjangan.cloudapp.net:8000> dengan:
 - “?start=x” dan “finish=y”, dimana x adalah koordinat tempat asal dan y adalah koordinat tempat tujuan.
 - `mw`, `wm`, dan `pt` masing-masing dengan nilai dari `array alternative` satu persatu dengan indeks ‘`mw`’, ‘`wm`’, dan ‘`pt`’.
 - memanggil fungsi `file_get_content` yang akan mengembalikan FALSE jika gagal dan mengembalikan JSON pesan kesalahan. Jika berhasil akan mendapatkan String dengan *URL* <http://newmenjangan.cloudapp.net:8000>, pembacaan maksimal sebanyak 51200 *bytes*.
 - memasukkan ke `array results` yang mempunyai indeks ‘`result`’ dengan nilai `true`.
- Jika `version` kurang dari 2, langsung akses *URL* <http://newmenjangan.cloudapp.net:8000> dengan *query* “?start=x” dan “finish=y”, dimana x adalah koordinat tempat asal dan y adalah koordinat tempat tujuan.

Langkah selanjutnya adalah pengulangan sejumlah data `results` menjadi `result` dengan nilai yang dimasukkan ke data `dummy` yang berisi:

- Membuat data `steps` yang berisi data `result` yang telah dipisahkan per baris.
- Melakukan pengulangan setiap data `steps` dan dimasukkan ke `step` yang berisi:

- Membuang spasi yang ada pada data `step`. Jika `step` sudah tidak ada isinya, melanjutkan ke langkah selanjutnya.
- Jika `step` sama dengan ‘`none`’, melakukan pengecekan pada banyaknya data `results`. Jika banyak data `results` bukan satu berarti ada rute alternatif dan akan melanjutkan ke langkah selanjutnya. Tetapi jika banyak data `results` adalah satu, maka tidak ada rute alternatif.
- Mendaftarkan data `means`, `means_detail`, `route`, `distance`, dan `nearbyplaceids` yang didapat dari data `step` yang telah dipisahkan ‘/’. Jika salah satu data kosong, akan menutup koneksi SQL dan mengeluarkan pesan kesalahan.
- Inisialisasi data `points` yang didapat dari data `route` yang sudah dipisahkan spasi, inisialisasi data `from` dari data `points` dengan indeks 0 dan inisialisasi data `to` dari data `points` dengan indeks terakhir.
- Jika nilai data `points` ada yang sama dengan ‘`start`’, maka mengganti data `points` tersebut dengan data `start`. Jika nilai data `points` ada yang sama dengan ‘`finish`’, maka mengganti data `points` tersebut dengan data `finish`.
- Memasukkan data `humanized_from` dengan memanggil fungsi `humanize_point` dengan parameter `from` dan `humanized_to` dengan memanggil fungsi `humanize_point` dengan parameter `to`.
- Jika `means` sama dengan ‘`walk`’, melakukan pengecekan apakah `humanized_from` sama dengan `humanized_to`. Jika `presentation` tidak sama dengan `mobile`, maka memasukkan `message_walk_samestreet` ke variabel `humanreadable`. Mengganti `from` yang ada di `humanreadable` dengan nilai `humanized_from`, mengganti `distance` yang ada di `humanreadable` dengan nilai `distance` yang dipresentasikan sesuai format wilayah pengguna. Lalu menghitung waktu tempuh yg dibutuhkan.
- Jika `means` tidak sama dengan ‘`walk`’, melakukan pencarian pada basisdata `tracks` digabung dengan `tracktypes` dengan kolom nama jalur, tipe jalur yang digunakan, `URL` untuk melakukan pemesanan (jika XTrans), kecepatan transportasi, dan informasi internal jalur dengan syarat:
 1. `trackTypeId` sama dengan `means` pada tabel `tracktypes` dan `tracks`.
 2. `trackid` sama dengan `means_detail` pada tabel `tracks`.

Jika ada hasil dari pencarian basisdata, maka:

1. Memasukkan nilai `trackname` dari tabel `tracks` ke `readable_track_name`.
2. Memasukkan nilai `nama` dari tabel `tracktypes` ke `track_type_name`.
3. Memasukkan nilai `speed` dari tabel `tracktypes` ke data `speed` yang sudah dijadikan format angka.

Lalu melanjutkan pengecekan apakah `presentation` sama dengan ‘`mobile`’, maka menambahkan data `humanized_from` dengan ‘%fromicon’ dan `humanized_to` dengan ‘%toicon’. Memasukkan `message_angkot` ke data `humanreadable`. Memperbarui nilai `humanreadable` dengan mengganti:

* %from dengan nilai dari `humanized_from`.

- * %to dengan nilai dari `humanized_to`.
- * %distance dengan nilai dari `distance` yang dipresentasikan sesuai format wilayah pengguna.
- * %trackname dengan nilai dari `readable_track_name`.
- * %tracktype dengan nilai dari `track_type_name`.

Setelah itu, menghitung waktu tempuh dari data `distance` dibagi dengan data `speed`. Melakukan pengecekan nilai URL dari tabel `tracktypes` dan nilai `extraParameters` dari tabel `tracktypes` kosong tidak. Jika tidak kosong maka menambahkan data `booking_url` dengan menambahkan nilai URL dari tabel `tracktypes` dan nilai `extraParameters` dari tabel `tracktypes`. Lalu melakukan pengecekan nilai `internalinfo` dari tabel `tracks` dimulai dengan `angkotwebid:`. Jika dimulai dengan `angkotwebid:`, maka mendapatkan `array token` dengan memisahkan ‘:’ pada `internalInfo`. Memasukkan data `editor_url` dengan awalan `URL` angkotwebid, `token` dengan indeks kedua, dan akhiran `URL` angkotwebid.

- Jika `humanreadable` tidak sama dengan `null`, maka memasukkan `route_output` dengan isi array dari `means`, `means_detail`, `points`, `humanreadable`, `booking_url`, dan `editor_url`.
- Memasukkan `routing_result` dengan indeks `steps` dengan nilai dari `route_output`.
- Memasukkan `routing_result` dengan indeks `traveltime` dengan nilai jam dan menit dari `travel_time` yang sudah dilakukan i18n.
- Memasukkan data `routing_result` ke dalam `array routing_results`.

Setelah itu, memasukkan data ke tabel statistik dengan `parameter`:

- `apikey` yang digunakan.
- `FINDROUTE` merupakan tipe statistik yang dimasukkan.
- `$start/$finish/sizeof($results)` merupakan keterangan statistik dimana `$start` adalah titik koordinat tempat asal, `$finish` adalah titik koordinat tempat tujuan, dan jumlah dari data `$results`.

Lalu menutup koneksi ke basisdata. Jika `version` tidak sama dengan `null` dan `version` lebih besar sama dengan 2, maka print hasil yang sudah di `encode` menjadi JSON, yaitu:

- `status` dengan nilai ‘ok’
- `routingresults` dengan nilai `routing_results` yang merupakan hasil pencarian rute.

Jika `version` sama dengan `null` dan `version` kurang dari 2, maka print hasil yang sudah di `encode` menjadi JSON, yaitu:

- `status` dengan nilai ‘ok’
- `routingresult` dengan nilai `routing_results` dengan indeks ke nol dan memiliki `key` berupa `steps` yang merupakan hasil pencarian rute.

- **traveltimes** dengan nilai **routing_results** dengan indeks ke nol dan memiliki **key** berupa **traveltimes** yang merupakan waktu tempuh.

Mode Search

Bagian ini terletak di baris 183-280 dari handle.php (kode A.1). Langkah pertama yang dilakukan adalah memasukkan data **querystring**, **apikey**, dan **region** dengan mendapatkan *method* POST masing-masing dengan *parameter* ‘**querystring**’, ‘**apikey**’, dan ‘**region**’. Jika **region** sama dengan **null**, maka isi **region** dengan ‘**bandung**’. Melakukan pengulangan sejumlah data **regioninfos** dengan nilai yang dimasukkan ke data **value** yang berisi:

- Mencari dalam **querystring** ada *pattern* ‘/searchplace’ dimana ‘**searchplace**’ diambil dari **value** dengan **key** ‘**searchplace_regex**’ dan tampilkan pada *array* **matches**. Jika terdapat *pattern* tersebut, maka memasukkan data **region** dengan nilai **key**, memperbarui **querystring** dengan mengambil panjang kalimat dari indeks nol sampai dengan indeks ditemukannya *pattern* tersebut.
- Mengubah **querystring** menjadi format *URL*.
- Memasukkan data **cached_searchplace** dengan mengambil data dari tabel **cache** dengan tipe **searchplace** dan **key** **region/querystring**. Jika **cached_searchplace** tidak sama dengan **null**, maka membuat JSON untuk *output* dengan:
 - **status** dengan nilai ‘ok’
 - **searchresult** dengan nilai **cached_searchplace**
 - **attributions** dengan nilai **null**

Jika **cached_searchplace** sama dengan **null**, memasukkan data **lat**, ‘\verb!lon!', dan **radius** yang didapati dengan indeks **region** dan \textit{key} ‘**lat**', ‘**lon**', dan ‘**radius**'. Memasukkan \verb!result! dengan hasil akses *URL* dari data **full_url**'. Jika \verb!result! sama dengan FALSE atau jumlah **result** lebih besar dari **maximum_http_response_size**, maka mengeluarkan pesan kesalahan. Setelah itu, memasukkan **json_result** dengan mengubah format JSON menjadi String dari data **result**. Jika **json_result** dengan indeks ‘**status**’ adalah ‘OK’ atau ‘ZERO_RESULTS’, maka menyiapkan *array* **search_result**. Jika hasilnya ‘ZERO_RESULTS’, maka mengeluarkan catatan kesalahan dan memperbarui data **size** menjadi 0. Jika hasilnya ‘OK’, maka mengambil angka minimal antara banyaknya data **json_result** dengan indeks ‘**results**’ dan **search_maxresult**. Melakukan pengulangan dari nol sampai dengan **size** yang dinisialkan dengan **i**:

- Memasukkan data **current_venue** dengan **json_result** indeks **results** dengan **key** **i**.
- Memasukkan data **search_result** dengan indeks **i** dan **key** **placename** yang isinya adalah **current_venue** dengan indeks **name**.
- Memasukkan data **search_result** dengan indeks **i** dan **key** ‘**location**’ yang isinya adalah String yang sudah diformat ‘%.latlonlf%.latlonlf’ yang nilainya diambil dari **current_venue** dengan indeks ‘**geometry**’ dan ‘**location**’ dengan **key** ‘**lat**’ dan ‘**lng**’.

- Menyiapkan `json_output` yang merupakan *array* dari:
 - * `status` dengan nilai ‘ok’
 - * `searchresult` dengan nilai `search_result` yang merupakan hasil pencarian rute
 - * `attributions` dengan nilai `null`
- Mengirim catatan dengan `apikey` yang digunakan.
- Memasukkan ke tabel `cache` dengan tipe ‘`cache_searchplace`’ dan `key region/querystring` yang berisi JSON `search_result`.
- Mengeluarkan hasil JSON `json_output`.

Mode Reporterror

Bagian ini terletak di baris 282-284 dari handle.php (kode A.1). Langkah pertama yang dilakukan adalah mendapatkan `errorcode` dari *method POST* dengan *parameter* ‘`errorcode`’. Mencatat kesalahan dan dimasukkan ke dalam *file*. Mengeluarkan JSON dengan:

- `status` dengan nilai ‘ok’

dan memberhentikan eksekusi program.

Mode Nearbytransport

Bagian ini terletak di baris 286-313 dari handle.php (kode A.1). Langkah pertama yang dilakukan adalah mendapatkan `start` dari *method POST* dengan *parameter* ‘`routestart`’. Jika `version` lebih besar sama dengan 2, maka membuat data `lines` dengan mendapatkan String dari *URL* <http://newmenjangan.cloudapp.net:8000> dengan query `/?start='start'`. Setelah mendapatkan String tersebut, lakukan pemisahan baris. Melakukan pengulangan sebanyak `lines` yang berisi:

- Melakukan pemisahan dari ‘\’ dari `line`, lalu mendaftarkan menjadi `tracktypeid`, `trackid`, dan `distance`.
- Membuat data `result` dengan eksekusi perintah SQL, yaitu mendapatkan \verb!trackname! dari tabel `tracks` dimana kolom `trackid` dan `tracktypeid` sesuai dengan data `tracktypeid` dan `trackid`.
- Membuat data `row` dengan mengambil hasil dari data `result` dan melakukan pengulangan yang berisi:
 - Membuat data `trackName` dengan mengambil `row` dengan indeks nol.
 - Membuat data `nearby_result` yang merupakan *array* dari `trackTypeId`, `trackId`, `trackName`, dan `distance`.
- Menyortir *array* `nearby_result` berdasarkan fungsi `nearby_result_compare`. Fungsi `nearby_result_com` membandingkan jarak yang diambil dari *array* dengan indeks ketiga.
- Memasukkan data ke tabel statistik dengan *parameter*:
 - `apikey` yang digunakan.

- NEARBYTRANSPORTS merupakan tipe statistik yang dimasukkan.
- \$start merupakan keterangan statistik dimana \$start adalah titik koordinat tempat asal dan jumlah dari data \$results.
- Mengeluarkan JSON dengan:
 - status dengan nilai ‘ok’
 - nearbytransports dengan nilai nearby_result

Fungsi humanize_point

Tujuan dari fungsi ini adalah merubah lokasi dalam bentuk *point* menjadi nama lokasi tempat dari *point* tersebut. Fungsi ini menerima parameter **location** yang merupakan String. Fungsi ini berisi sebagai berikut:

- Melakukan deklarasi variabel global untuk digunakan.
- Melakukan pengecekan data **location**. Jika **location** sama dengan ‘start’ maka mengembalikan **message_start**. Jika **location** sama dengan ‘finish’, maka mengembalikan **message_finish**.
- Jika bukan keduanya, maka memanggil fungsi **mysqli_escape_string**.
- Melakukan pengecekan pada tabel *cache* ada atau tidak dengan *key cache_geocoding* dan ber nilai sesuai dengan parameter **location** atau tidak dan dimasukkan ke data **cached_geocode**. Jika ada nilainya, langsung mengembalikan nilai tersebut.
- Jika tidak ada, maka akan menambahkan *query ?key=\$gmaps_server_key&latlng=\$location&sensor=false* pada URL <https://maps.googleapis.com/maps/api/geocode/json> dimana **gmaps_server_key** adalah *key* untuk API Google dan *latlng* adalah garis lintang dan garis bujur. Jika berhasil akses dan mendapatkan **status** sama dengan ‘OK’, maka akan menambahkan lokasi suggesi pada pengguna. Jika **status** tidak sama dengan ‘OK’, maka akan menampilkan pesan kesalahan.

3.2 Analisis Sistem Usulan

3.2.1 Peta

Peta pada sistem usulan menggunakan OpenLayers yang sama dengan sistem kini.

3.2.2 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 3.27

Listing 3.27: Menampilkan pilihan kota kepada pengguna

```

1 ...
2 <select class="fullwidth" id="regionselect">
3     @for(regioninfo <- regioninfos) {
4         <option value=@regioninfo._1
5             @if(regioninfo._1 == region){
6                 selected
7             }
8             >@regioninfo._2.getName</option>
9     }
10 </select>
11 ...

```

Dropdown Menu Bahasa

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

Listing 3.28: Menampilkan pilihan bahasa kepada pengguna

```

1 ...
2 <select class="fullwidth" id="localeselect">
3     <option value="en">English</option>
4     <option value="id"
5         @if(locale == "id"){
6             selected
7         }
8         >Bahasa Indonesia</option>
9 </select>
10 ...

```

Textfield

Pada Play Framework , *Textfield* pada KIRI dibuat agar dapat dilakukan proses i18n, seperti pada kode listing 3.29 untuk *textfield* tempat asal dan kode listing 3.30 untuk *textfield* tempat tujuan.

Listing 3.29: 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 3.30: 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 3.31.

Listing 3.31: 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

Fungsi tombol Swap menggunakan sistem yang sudah ada karena menggunakan JavaScript.

Tombol Reset

Fungsi tombol Reset menggunakan sistem yang sudah ada karena menggunakan JavaScript.

Tombol Find

Fungsi tombol Find menggunakan sistem yang sudah ada karena menggunakan JavaScript.

3.2.3 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 3.32 dan kode listing 3.33.

Listing 3.32: 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 3.33: 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 metode untuk melakukan 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 3.34

Listing 3.34: 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></a>
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 3.35.

Listing 3.35: 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>

```

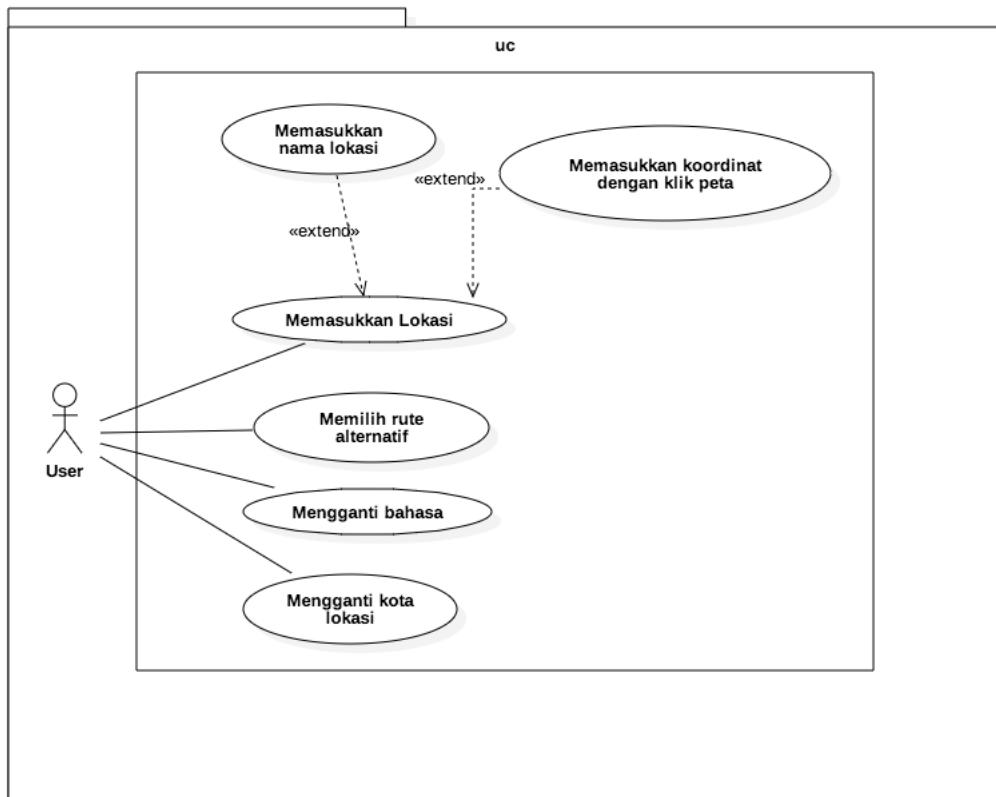
3.2.4 Pencarian Rute

Algoritma yang digunakan pada sistem usulan sama, tetapi ada beberapa struktur data yang berbeda yang akan dijelaskan sebagai berikut:

- Menyiapkan header pada aplikasi dengan menggunakan objek **response** dengan *method*:
 - Fungsi **setContentType** untuk menghasilkan *output* dengan tipe tertentu.
 - Fungsi **setHeader** untuk mengatur *header* pada aplikasi.
- *Array* pada Java menggunakan *HashMap* dan *ArrayList* karena array mengandung indeks dan *key* tertentu.
- Koneksi pada aplikasi menggunakan JDBC.
- Hasil eksekusi perintah SQL dimulai dari indeks ke satu, bukan nol.
- *StringBuilder* untuk menghasilkan output pada aplikasi.
- Melakukan format JSON dengan menggunakan *ObjectNode*.
- Melakukan *encode* JSON menggunakan *JsonNode*.
- Melakukan perhitungan matematika dengan menggunakan kelas *Math*.

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

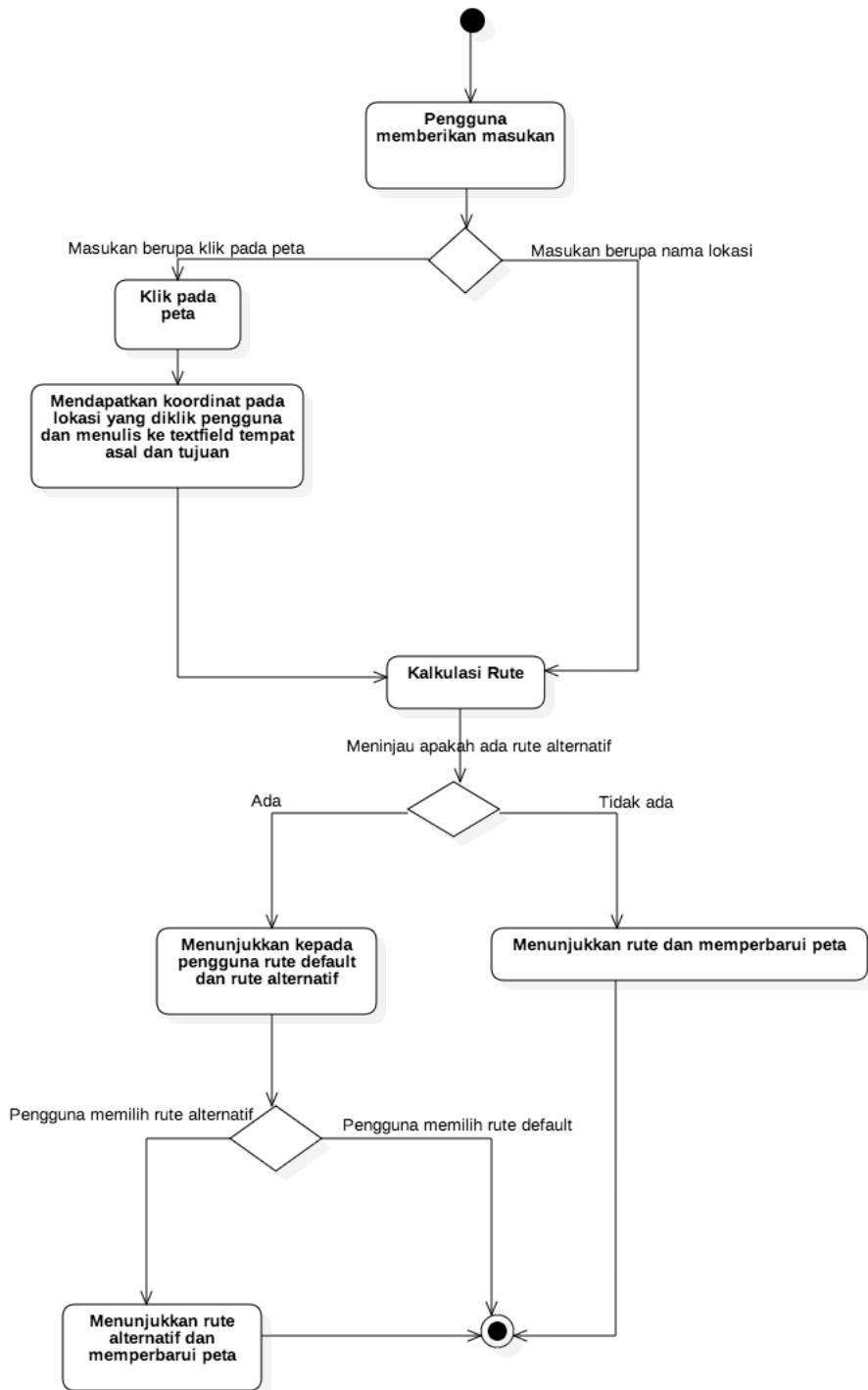
- Aktor: Pengguna
- Deskripsi: Memilih opsi lokasi kota yang akan ditampilkan pada peta dan pencarian rute pada kota tersebut.
- Kondisi awal: -
- Kondisi akhir: Menampilkan kepada pengguna dengan lokasi kota yang dipilih
- Skenario utama:

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:

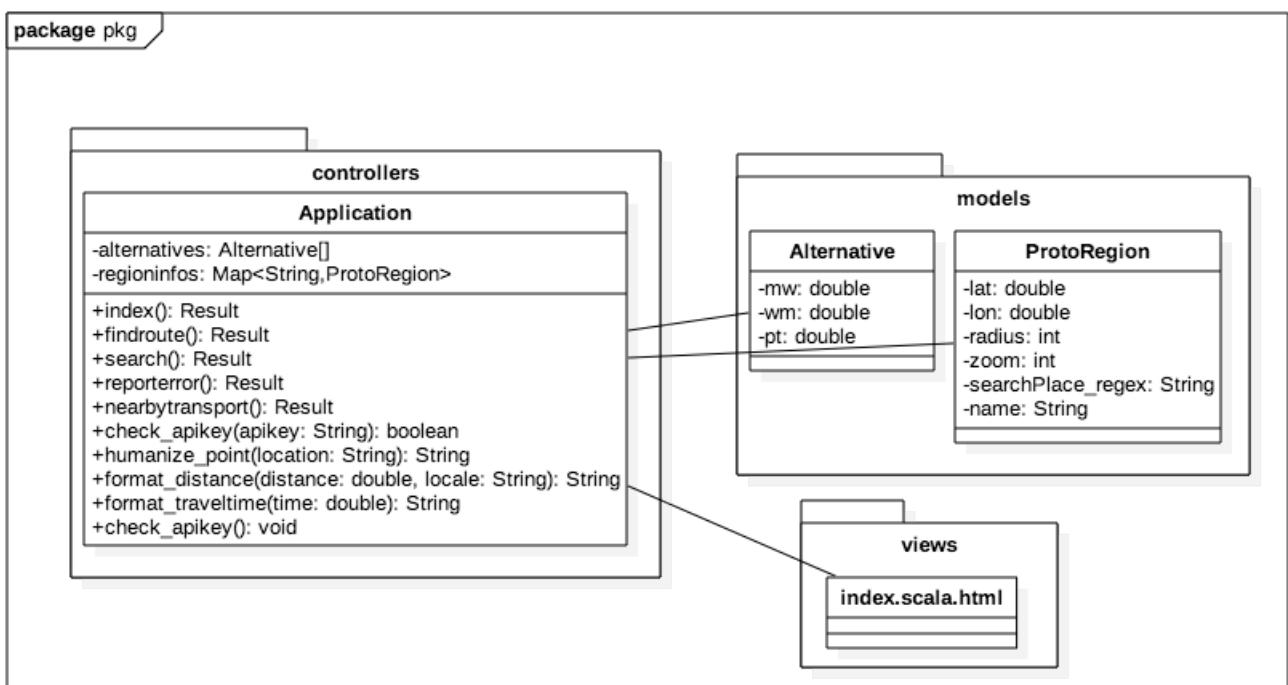
1. Pengguna memberikan masukan. Masukan dapat berupa dua, yaitu:
 - Masukan berupa nama lokasi.
 - Masukan berupa klik pada peta. Pengguna klik pada peta, kemudian sistem mendapatkan koordinat pada lokasi yang diklik pengguna dan menulis koordinat tersebut kepada

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.

3.5 Analisis Kelas

Diagram kelas analisis KIRI dapat dilihat pada [3.12](#).



Gambar 3.12: Diagram Kelas Analisis KIRI

Kelas-kelas yang ada pada KIRI dapat dijelaskan sebagai berikut:

1. Application
Kelas ini berfungsi untuk menghubungkan *model* dengan *view* dan berada pada *package* Controllers.

2. Alternative

Kelas untuk merepresentasikan alternatif rute yang akan dicari dan berada pada *package* Models.

3. ProtoRegion

Kelas untuk merepresentasikan detail kota yang disediakan KIRI dan berada pada *package* Models.

BAB 4

PERANCANGAN

Pada bab ini akan dijelaskan mengenai perancangan aplikasi meliputi diagram kelas rinci beserta deskripsi dan fungsinya, dan perancangan antarmuka.

4.1 Diagram Kelas Rinci

Diagram kelas rinci diperoleh dari hasil pengembangan diagram kelas analisis pada subbab 3.5. Diagram kelas rinci dapat dilihat pada Gambar 4.1.

Deskripsi kelas beserta fungsi dari diagram kelas rinci tersebut adalah sebagai berikut:

1. Application

Kelas ini merupakan kelas yang sudah disediakan Play Framework dan merupakan kelas Controller pada aplikasi KIRI. Atribut yang dimiliki kelas Application adalah:

- **DynamicForm dynamicForm:** objek DynamicForm yang berperan dalam mengambil query pada URL.
- **Comparator<ArrayList<String>> comparator:** objek Comparator yang berperan dalam fungsi untuk membandingkan String dalam ArrayList.
- **CacheApi cache:** objek CacheApi yang berperan dalam mengambil data *cache* yang telah dibuat oleh aplikasi KIRI.

Method-method yang dimiliki kelas ini merupakan *action method* dengan rincian sebagai berikut:

- **public Result index()**

Berfungsi untuk validasi *cache locale* dan *region* serta mengarahkan pengguna ke halaman utama KIRI.

Kembalian: Halaman utama KIRI.

- **public Result handle()**

Berfungsi untuk melakukan logikal data ketika pengguna mencari rute pada KIRI.

Kembalian: Halaman utama KIRI dengan hasil pencarian rute.

- **public String getFromMenjangan(String start,String finish,double mw,double wm,double pt)**

Berfungsi untuk mendapatkan hasil dari menjangan.

Parameter:

- **start** Lokasi awal pengguna.
- **finish** Lokasi akhir pengguna.
- **mw** Alternatif minimum walk.
- **wm** Alternatif walking multiplier.
- **pt** Alternatif penalty transfer.

Kembalian: Hasil dari layanan HTTP menjangan dengan query parameter.

- **public String getFromMenjangan(String start, String finish)**

Berfungsi untuk mendapatkan hasil dari menjangan.

Parameter:

- **start** Lokasi awal pengguna.
- **finish** Lokasi akhir pengguna.

Kembalian: Hasil dari layanan HTTP menjangan dengan query parameter.

- **public String getFromMenjangan(String start)**

Berfungsi untuk mendapatkan hasil dari menjangan.

Parameter:

- **start** Lokasi awal pengguna.

Kembalian: Hasil dari layanan HTTP menjangan dengan query parameter.

- **public String file_get_contents(String url)**

Berfungsi untuk mendapatkan hasil dari URL tertentu.

Parameter:

- **url** URL yang ingin didapatkan.

Kembalian: Hasil dari layanan HTTP dengan URL tertentu.

- **public String getPlacesAPI(String location, String radius, String querystring)**

Berfungsi untuk mendapatkan hasil JSON dari maps.googleapis.com.

Parameter:

- **location** Lokasi yang ingin dicari(garis lintang dan garis bujur).
- **radius** Radius dari lokasi yang ingin dicari.
- **querystring** Kata yang berkaitan dengan lokasi yang ingin dicari.

Kembalian: Hasil dari layanan HTTP menjangan dengan query parameter.

- **public String format_traveltime(double time)**

Berfungsi untuk melakukan format waktu.

Parameter:

- **time** Waktu yang ingin diubah formatnya.

Kembalian: Hasil waktu yang sudah diformat.

- **public String retrieve_data(String param)**

Berfungsi untuk mendapatkan data dari query URL.

Parameter:

- **param** Query yang ingin didapatkan.

Kembalian: Data pada query URL.

- **public boolean check_apikey(String apikey)**

Berfungsi untuk validasi apikey yang digunakan valid atau apikey yang digunakan ada batasan alamat IP.

Parameter:

- **apikey** Apikey yang ingin dicek.

Kembalian: Hasil pengecekan apikey dapat dipakai atau tidak. Jika tidak dapat dipakai, akan mengembalikan hasil pesan kesalahan.

- **public String humanize_point(String location)**

Berfungsi untuk mendapatkan nama lokasi tempat dari titik koordinat lokasi.

Parameter:

- **location** Titik koordinat lokasi.

Kembalian: Nama lokasi tempat.

- **public String format_distance(double distance, String locale)**

Berfungsi untuk melakukan format jarak berdasarkan lokalisasi.

Parameter:

- **distance** Jarak yang ingin diubah formatnya.
- **locale** Lokalisasi bahasa.

Kembalian: Hasil distance yang sudah diformat sesuai lokalisasi.

2. Constants

Kelas untuk mengisi variabel dengan konstanta dan berfungsi agar tidak menuliskan konstanta berulang kali dan membantu pengembangan aplikasi KIRI. Atribut yang dimiliki kelas Constants adalah:

- **Alternative[] ALTERNATIVES:** Objek Alternative dalam *array*.
- **Map<String, ProtoRegion> REGIONINFOS:** HashMap yang terdiri dari kunci berupa String dan berisi objek ProtoRegion.
- **String CACHE_GEOCODING:** Untuk mencari atau menyimpan pada basisdata cache dengan type geocoding.

- **String CACHE_SEARCHPLACE:** Untuk mencari atau menyimpan pada basisdata cache dengan type searchplace.
- **String PLACES_URL:** URL Google API untuk mencari nama lokasi tempat.
- **int SEARCH_MAXRESULT** Angka untuk menentukan batas maksimal hasil pencarian.
- **String GMAPS_SERVER_KEY:** Kunci agar dapat akses Google API.
- **String GMAPS_GEOCODE_URL:** URL Google API untuk mencari geocode lokasi tempat.
- **String ANGKOTWEBID_URL_PREFIX:** Awalan URL untuk <http://angkot.web.id>.
- **String ANGKOTWEBID_URL_SUFFIX:** Akhiran URL untuk <http://angkot.web.id>.
- **int SPEED_WALK:** Kecepatan jalan kaki.
- **String MENJANGAN_URL:** URL untuk mengakses <http://newmenjangan.cloudapp.net:8000>.
- **String PROTO_ATTRIBUTIONS:** Untuk menampilkan JSON dengan *key attributions*.
- **String PROTO_ERRORCODE:** Untuk mendapatkan nilai dari *query errorCode* pada protokol.
- **String PROTO_LOCALE:** Untuk mendapatkan nilai dari *query locale* pada URL.
- **String PROTO_LOCALE_INDONESIA:** Nilai dari *query locale* dengan isi ‘id’.
- **String PROTO_LOCALE_ENGLISH:** Nilai dari *query locale* dengan isi ‘en’.
- **String PROTO_LOCATION:** Untuk menyimpan pada HashMap dengan *key location*.
- **String PROTO_MESSAGE:** Untuk menampilkan JSON dengan *key message*.
- **String PROTO_MODE_FINDROUTE:** Untuk mendapatkan nilai dari *query mode* pada protokol.
- **String PROTO_MODE_REPORTERROR:** Untuk mendapatkan nilai dari *query mode* pada protokol.
- **String PROTO_MODE_SEARCH:** Untuk mendapatkan nilai dari *query mode* pada protokol.
- **String PROTO_MODE_NEARBYTRANSPORTS:** Untuk mendapatkan nilai dari *query mode* pada protokol.
- **String PROTO_NEARBYTRANSPORTS:** Untuk menampilkan JSON dengan *key nearbytransports*.
- **String PROTO_PLACENAME:** Untuk menampilkan JSON dengan *key placename*.
- **String PROTO_PRESENTATION_DESKTOP:** Untuk mengetahui apakah aplikasi dibuka di *desktop* atau tidak.

- **String PROTO_PRESENTATION_MOBILE:** Untuk mengetahui apakah aplikasi dibuka di *mobile* atau tidak.
- **String PROTO_REGION:** Untuk mendapatkan nilai dari *query* URL **region**.
- **String PROTO_REGION_BANDUNG:** Region Bandung dengan kode ‘**bdo**’.
- **String PROTO_REGION_JAKARTA:** Region Jakarta dengan kode ‘**jkt**’.
- **String PROTO_REGION_SURABAYA:** Region Surabaya dengan kode ‘**sby**’.
- **String PROTO_REGION_MALANG:** Region Malang dengan kode ‘**mgl**’.
- **String PROTO_ROUTESTART:** Untuk mendapatkan nilai dari *query* URL **start**.
- **String PROTO_ROUTINGRESULT:** Untuk menampilkan JSON dengan *key* **routingresult**.
- **String PROTO_ROUTINGRESULTS:** Untuk menampilkan JSON dengan *key* **routingresults**.
- **String PROTO_SEARCH_QUERYSTRING:** Untuk mendapatkan nilai dari *query* URL **querystring**.
- **String PROTO_SEARCH_RESULT:** Untuk menampilkan JSON dengan *key* **searchresult**.
- **String PROTO_STATUS:** Untuk menampilkan JSON dengan *key* **status**.
- **String PROTO_STATUS_ERROR:** Untuk menampilkan JSON dengan nilai ‘**error**’.
- **String PROTO_STATUS_OK:** Untuk menampilkan JSON dengan nilai ‘**ok**’.
- **String PROTO_STEPS:** Untuk menampilkan JSON dengan *key* **steps**.
- **String PROTO_TRAVELTIME:** Untuk menampilkan JSON dengan *key* **traveltimes**.
- **String PROTOKD_POINT_FINISH:** Untuk pengecekan *array point* yang didapat dari URL <http://newmenjangan.cloudapp.net:8000> terdapat kata kunci ‘**finish**’.
- **String PROTOKD_POINT_START:** Untuk pengecekan *array point* yang didapat dari URL <http://newmenjangan.cloudapp.net:8000> terdapat kata kunci ‘**start**’.
- **String PROTOKD_RESULT_NONE:** Untuk pengecekan *array point* yang didapat dari URL <http://newmenjangan.cloudapp.net:8000> terdapat kata kunci ‘**none**’.
- **String PROTOKD_TRANSITMODE_WALK:** Untuk pengecekan *array point* yang didapat dari URL <http://newmenjangan.cloudapp.net:8000> terdapat kata kunci ‘**walk**’.

3. Alternative

Kelas yang berada pada *package* Models. Kelas ini untuk merepresentasikan alternatif dalam pencarian rute pada URL <http://newmenjangan.cloudapp.net:8000>. Kelas Alternative mempunyai atribut antara lain:

- **double mw:** Maximumwalk dari alternatif.
- **double wm:** Walkingmultiplier dari alternatif.
- **double pt:** Penaltrytransfer dari alternatif.

Method-method yang dimiliki kelas ini merupakan *action method* dengan rincian sebagai berikut:

- **public double getPt()**
Berfungsi untuk mendapatkan atribut pt.
Kembalian: Nilai pt.
- **public double getWm()**
Berfungsi untuk mendapatkan atribut wm.
Kembalian: Nilai wm.
- **public double getMw()**
Berfungsi untuk mendapatkan atribut mw.
Kembalian: Nilai mw.

4. ProtoRegion

Kelas yang berada pada *package* Models. Kelas ini untuk merepresentasikan kota dengan masing-masing mempunyai atribut sendiri. Kelas ProtoRegion mempunyai atribut antara lain:

- **double lat:** Titik lintang dari kota tertentu.
- **double lon:** Titik bujur dari kota tertentu.
- **int radius:** Radius dari kota tertentu.
- **int zoom:** Tingkat *zoom* dari kota tertentu.
- **String searchPlace_regex:** *Regex* dari kota tertentu guna dalam pencarian nama lokasi.
- **String name:** Nama dari kota tertentu.

Method-method yang dimiliki kelas ini merupakan *action method* dengan rincian sebagai berikut:

- **public double getLat()**
Berfungsi untuk mendapatkan atribut titik lintang.
Kembalian: Nilai titik lintang.
- **public double getLon()**
Berfungsi untuk mendapatkan atribut titik bujur.
Kembalian: Nilai titik bujur.
- **public int getZoom()**
Berfungsi untuk mendapatkan atribut tingkat *zoom*.
Kembalian: Nilai *zoom*.
- **public int getRadius()**
Berfungsi untuk mendapatkan atribut radius.
Kembalian: Nilai radius.
- **public String getSearchPlace_regex()**
Berfungsi untuk mendapatkan atribut *regex*.
Kembalian: Nilai *regex*.

- **public String getName()**

Berfungsi untuk mendapatkan atribut nama.

Kembalian: Nilai nama.

5. Utils

Kelas yang merupakan kelas pembantu untuk fungsi-fungsi dalam aplikasi KIRI. Kelas Utils mempunyai *Method-method* dengan rincian sebagai berikut:

- **public static void log_statistic(String verifier, String type, String additional_info)**

Berfungsi untuk memasukkan data dalam basisdata statistic.

Parameter:

- **verifier** Apikey yang mengisi data ke tabel.
- **type** Tipe statistic.
- **additional_info** Keterangan statistik.

. **Kembalian:** Tidak ada.

- **public static boolean domain_matches(String url, String filter)**

Berfungsi untuk pengecekan URL dengan filter sama atau tidak.

Parameter:

- **url** Apikey yang mengisi data ke tabel.
- **filter** Filter untuk URL.

. **Kembalian:** `true` jika URL sama dengan filter, `false` jika tidak sama.

- **public static int indexPregMatch(String regex, String content)**

Berfungsi untuk mengambil indeks kata tertentu dalam suatu kalimat jika ada.

Parameter:

- **regex** Kata yang ingin dicari.
- **content** Kalimat yang ingin dicari.

. **Kembalian:** Index kata tersebut dalam kalimat. Jika tidak ada akan mengembalikan -1.

- **public static ObjectNode die_nice(String message)**

Berfungsi untuk mengembalikan JSON dengan status error dan message.

Parameter:

- **message** Message yang ingin dituliskan.

. **Kembalian:** JSON dengan status error dan message.

- **public static ObjectNode well_done(String message)**

Berfungsi untuk mengembalikan JSON dengan status ok dan message(jika ada).

Parameter:

- **message** Message yang ingin dituliskan.

. **Kembalian:** JSON dengan status ok dan message.

- **public static String get_from_cache(String type, String key)**

Berfungsi untuk mendapatkan nilai cacheValue dari tabel cache dengan tipe dan key

tertentu.

Parameter:

- **type** Tipe cache.
- **key** Key dari cache.

. **Kembalian:** Nilai dari cacheValue.

• **public static void put_to_cache(String type, String key)**

Berfungsi untuk menyimpan nilai cacheValue ke tabel cache dengan tipe dan key tertentu.

Parameter:

- **type** Tipe cache.
- **key** Key dari cache.

. **Kembalian:** Tidak ada.

• **public static void log_error(String message)**

Berfungsi untuk melakukan log dengan pesan tertentu.

Parameter:

- **message** Pesan yang ingin dilog.

. **Kembalian:** Tidak ada.

• **public static String validateLocale(String locale)**

Berfungsi untuk melakukan validasi locale yang didapat dari cache.

Parameter:

- **locale** Input locale.

. **Kembalian:** Locale yang sudah divalidasi.

• **public static String validateRegion(String region)**

Berfungsi untuk melakukan validasi region yang didapat dari cache.

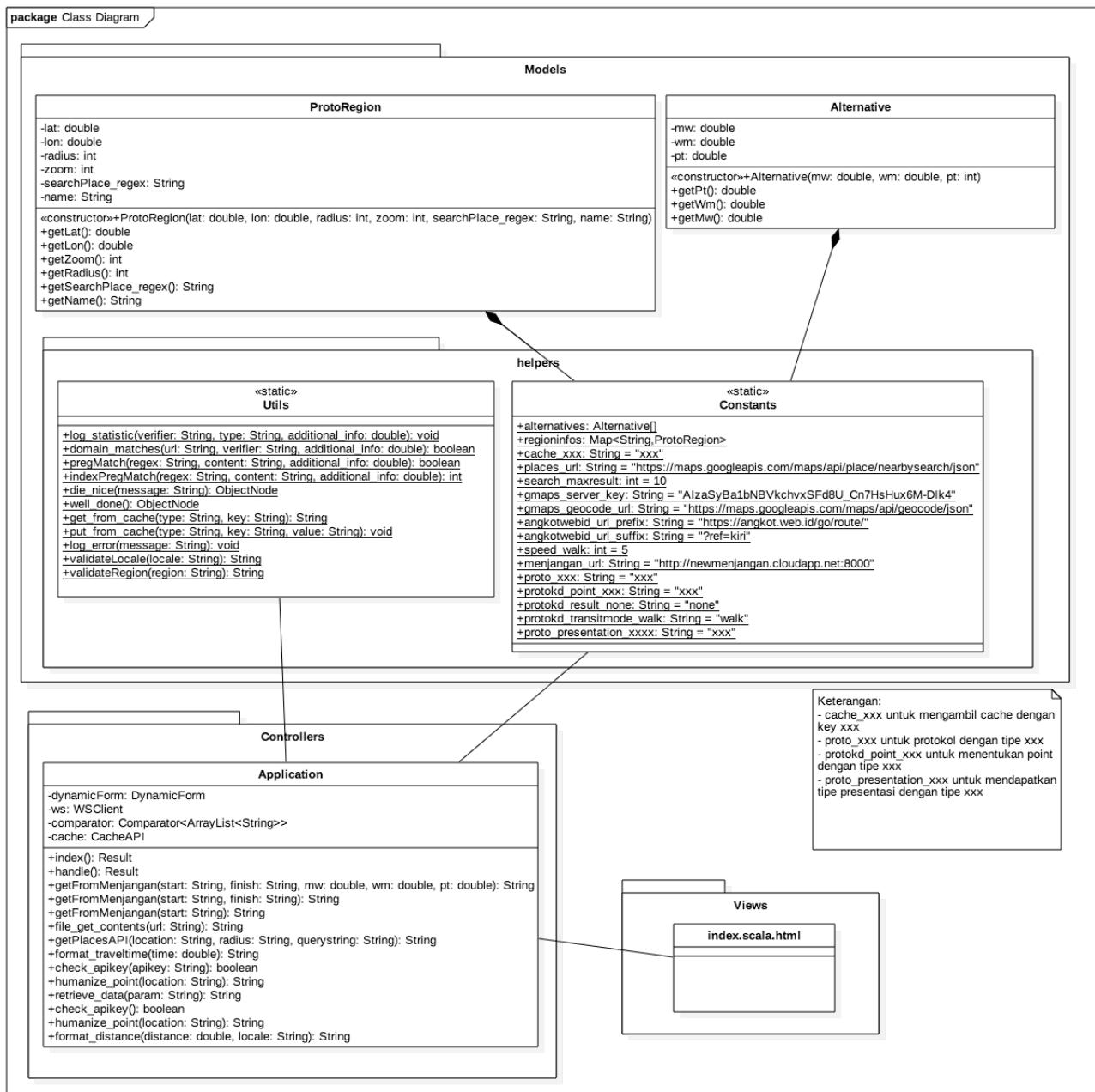
Parameter:

- **region** Input region.

. **Kembalian:** Region yang sudah divalidasi.

6. Index.scala.html

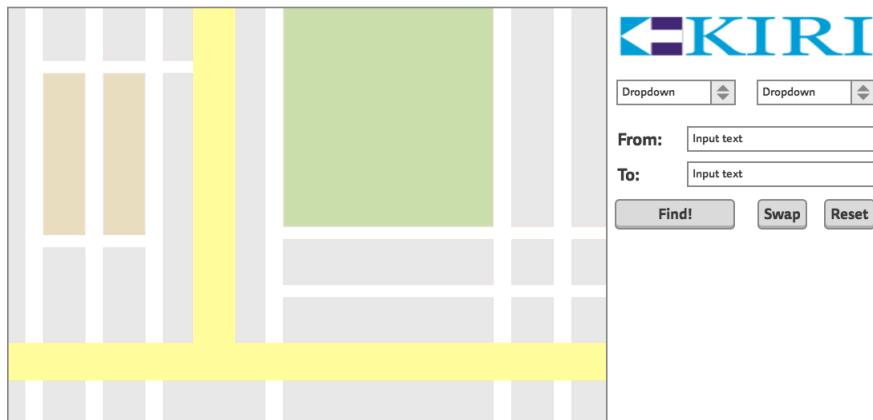
Kelas yang berada pada package Views dan berfungsi untuk tampilan pada aplikasi KIRI.



Gambar 4.1: Diagram Kelas Rinci

4.2 Perancangan Antarmuka

Untuk memenuhi kebutuhan interaksi antara pengguna dengan sistem, maka sebuah antarmuka mengikuti aplikasi KIRI yang lama. Antarmuka terdiri dari:



Gambar 4.2: Halaman Utama KIRI

Pada halaman utama KIRI (dapat dilihat pada gambar 4.2), terdapat beberapa bagian yaitu:

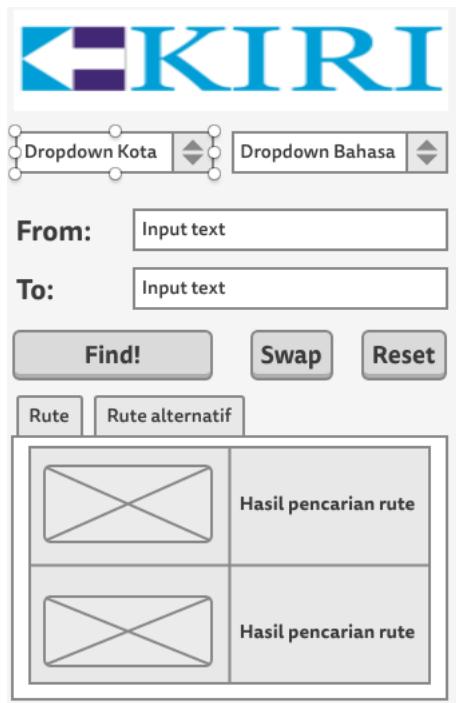
1. **Peta**
2. **Form Samping** yang terdiri dari beberapa bagian, yaitu:
 - (a) Dropdown Menu Kota
 - (b) Dropdown Menu Bahasa
 - (c) Textfield
 - (d) Tombol Swap
 - (e) Tombol Reset
 - (f) Tombol Find

4.2.1 Peta

Peta berfungsi untuk memudahkan pengguna dalam memilih lokasi awal dan lokasi akhir serta memberikan gambaran rute yang telah dicari oleh KIRI. KIRI menggunakan OpenLayers yang berbasis JavaScript untuk memuat peta pada halaman *web*.

4.2.2 Form Samping

Form yang terdapat pada halaman utama KIRI (Gambar 4.3) terdiri dari:



Gambar 4.3: Form pada KIRI

Dropdown Menu Kota

Dropdown yang berfungsi untuk memilih kota yang akan ditampilkan pada peta dan memperbarui peta.

Dropdown Menu Bahasa

Dropdown yang berfungsi untuk memilih bahasa yang akan digunakan pada aplikasi KIRI.

Textfield

Textfield pada KIRI agar pengguna dapat memasukkan *input* sendiri. Textfield pada KIRI dapat menerima dua masukan pengguna, yaitu:

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

Tombol Swap

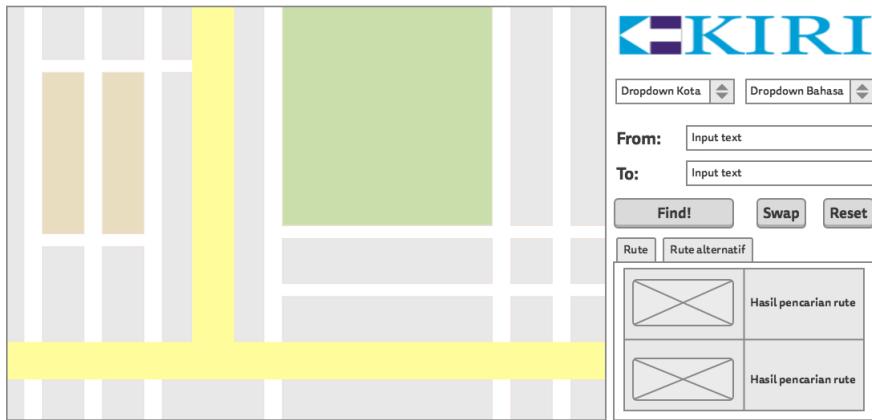
Pengguna dapat menukar isi dari *textfield* tempat asal dan tujuan.

Tombol Reset

Pengguna dapat melakukan pemilihan tempat dari awal dan mengulang tampilan peta.

Tombol Find

Pengguna dapat mencari rute untuk sampai ke tujuan (Gambar 4.4). Pengguna dapat memilih rute alternatif yang sudah disediakan KIRI jika ada.



Gambar 4.4: Contoh Pencarian Rute pada KIRI

BAB 5

IMPLEMENTASI DAN PENGUJIAN

Bab ini terdiri atas dua bagian, yaitu Implementasi Perangkat Lunak dan Pengujian Perangkat Lunak. Bagian implementasi berisi penjelasan lingkungan pengembangan perangkat lunak dan hasil implementasi. Sedangkan bagian pengujian berisi hasil pengujian fungsional dan eksperimental terhadap perangkat lunak yang telah dibangun.

5.1 Implementasi

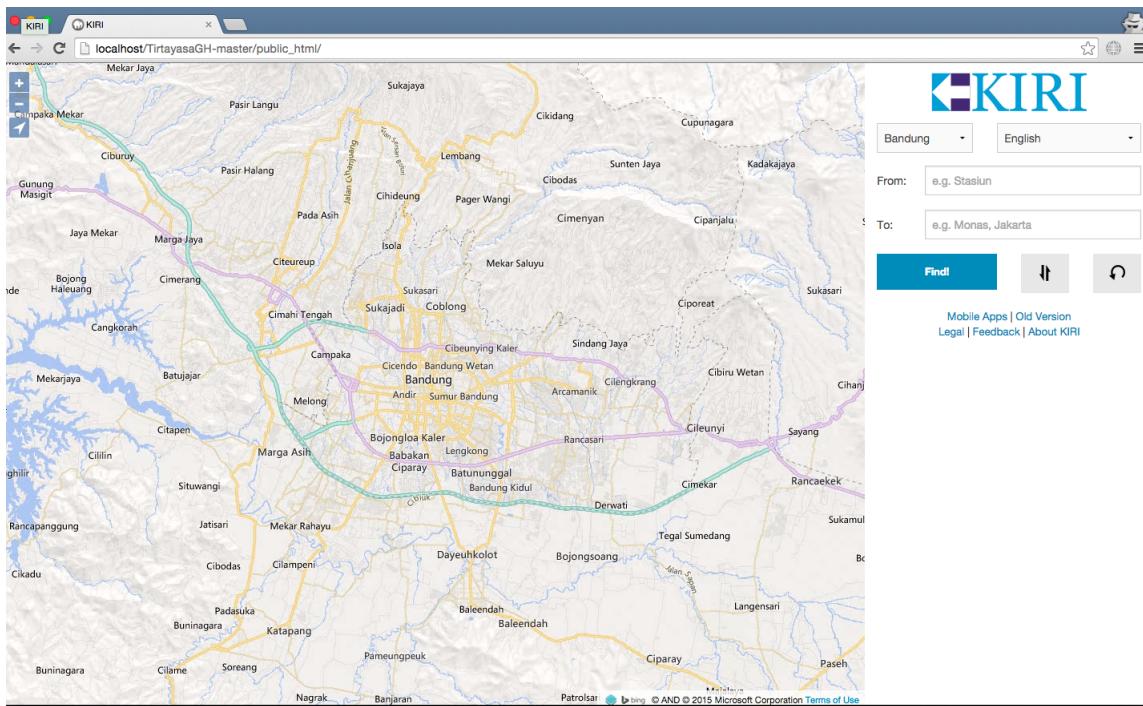
5.1.1 Lingkungan Implementasi

Implementasi perangkat lunak ini dilakukan di komputer dengan spesifikasi sebagai berikut:

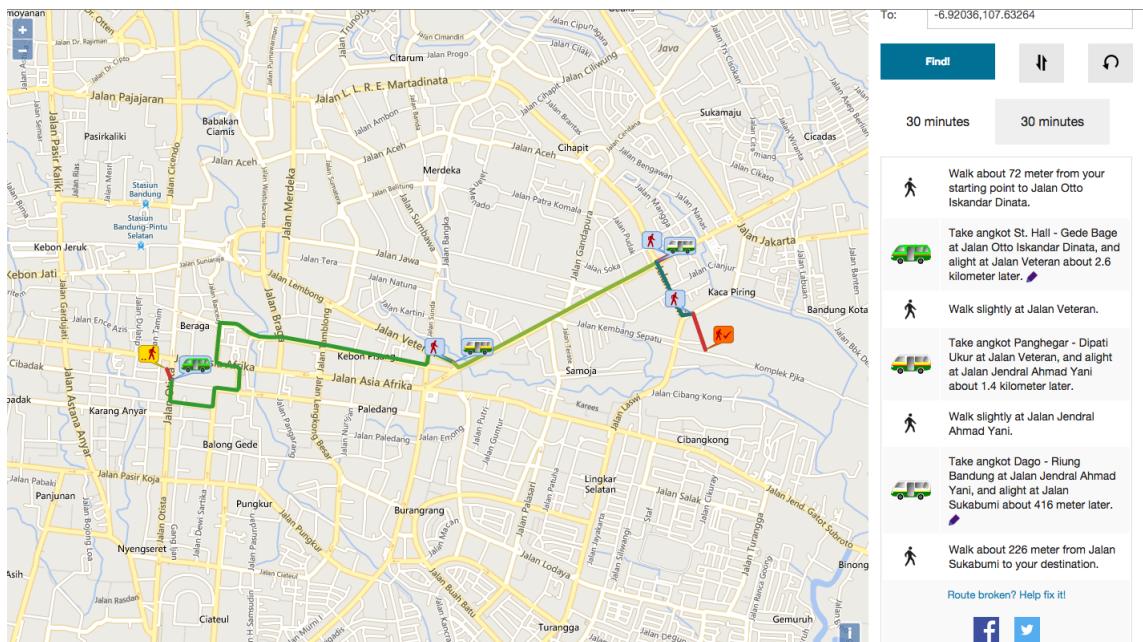
1. Processor: 1.6 GHz
2. RAM: 4 GB 1600 MHz DDR3
3. Sistem Operasi: OS X Yosemite version 10.10.5
4. Versi Java: 1.8.0_51

5.1.2 Hasil Implementasi

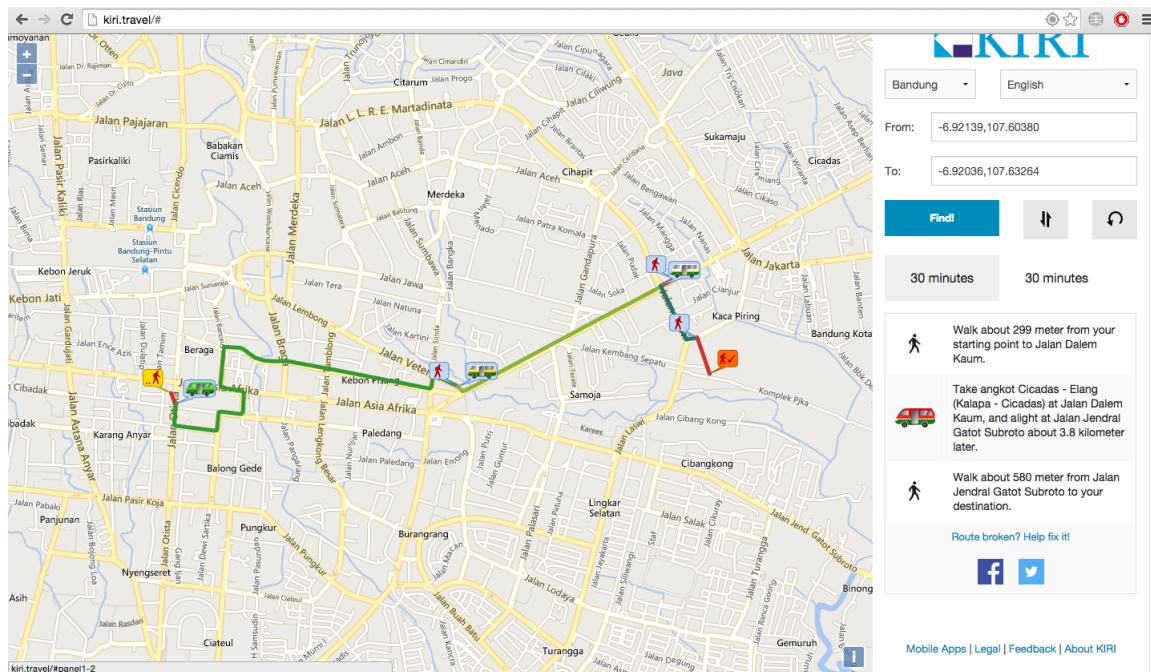
Hasil implementasi berupa aplikasi berbasis web yang menggunakan Play Framework. Aplikasi dapat diakses pada jaringan lokal dengan URL <http://localhost:9000>. Aplikasi KIRI terdiri dari satu halaman utama. Halaman utama KIRI, pengguna dapat melakukan pencarian rute KIRI dengan masukan nama tempat atau masukan berupa klik pada peta yang disediakan pada halaman utama KIRI. Setelah melakukan masukan untuk pencarian rute, muncul hasil pencarian rute dan/atau hasil alternatif rute yang ditampilkan pada halaman utama KIRI serta menggambarkan hasil rute pada peta yang disediakan aplikasi KIRI.



Gambar 5.1: Halaman Utama KIRI



Gambar 5.2: Contoh Pencarian Rute pada KIRI



Gambar 5.3: Contoh Rute Alternatif pada KIRI

5.2 Pengujian

5.2.1 Pengujian Fungsional

Pengujian fungsional dilakukan untuk mengetahui kesesuaian reaksi perangkat lunak dengan reaksi yang diharapkan berdasarkan aksi pengguna terhadap perangkat lunak. Pengujian ini dilakukan pada berbagai sistem yaitu Windows dan MacOS dengan hasil yang sama. Terdapat sembilan tes kasus yang diujikan, detail serta hasilnya dapat dilihat pada Tabel 5.1.

Tabel 5.1: Tabel Pengujian Fungsional

No.	Aksi Pengguna	Reaksi yang diharapkan	Reaksi Perangkat Lunak
1	Pengguna menjalankan aplikasi	Halaman utama ditampilkan	Sesuai
2	Pengguna memilih kota yang ingin ditampilkan	Memperbarui peta dengan kota yang dipilih	Sesuai
3	Pengguna memilih bahasa yang ingin ditampilkan	Memperbarui tampilan dengan bahasa yang dipilih	Sesuai
4	Pengguna melakukan klik pada peta	Marker muncul dari titik yang diklik oleh pengguna dan titik koordinat muncul pada <i>textfield</i> tempat asal dan/atau tujuan	Sesuai
5	Pengguna memilih tombol Swap	Masukan pengguna tempat awal dan tujuan ditukar	Sesuai
6	Pengguna memilih tombol Reset	Memperbarui halaman utama menjadi semula	Sesuai
7	Pengguna memilih tombol Find	Hasil pencarian rute muncul beserta alternatifnya (jika ada)	Sesuai, namun dengan urutan yang berbeda. Hal ini tidak mengubah arti pesan yang dikirim.
8	Pengguna memasukkan nama tempat pada <i>textfield</i> tempat asal dan/atau tujuan	Sugesti nama tempat muncul pada masing-masing <i>textfield</i> dan menampilkan hasil pencarian beserta alternatifnya (jika ada)	Sesuai
9	Pengguna memilih hasil pencarian rute alternatif	Hasil pencarian rute alternatif muncul dan memperbarui peta dengan hasil pencarian rute alternatif	Sesuai

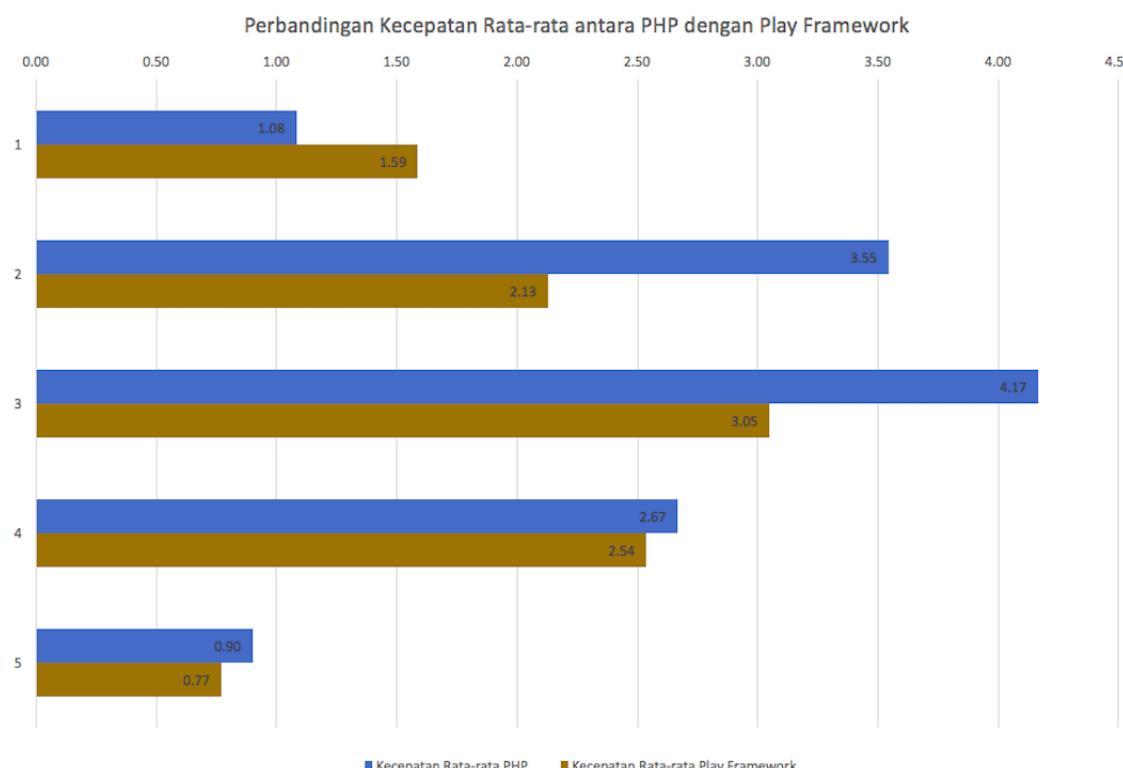
5.2.2 Pengujian Eksperimental

Pengujian eksperimental dilakukan untuk membandingkan perbedaan waktu eksekusi dari aplikasi KIRI (PHP) dengan aplikasi KIRI (Play Framework). Pengujian eksperimental mengambil tiga tes kasus dan masing-masing kasus diambil empat kali pengujian. Pengujian ini dilakukan pada sistem MacOS. Hasil pengujian dapat dilihat pada tabel 5.2 dan grafik 5.4.

Tabel 5.2: Tabel Pengujian Eksperimental

No.	Aksi	PHP				
		Waktu eksekusi pertama	Waktu eksekusi kedua	Waktu eksekusi ketiga	Waktu eksekusi keempat	Rata-rata
1.	Pengguna akses halaman utama KIRI	1.00	0.89	0.87	1.57	1.08
2.	Pengguna mencari rute dengan memasukkan nama tempat	4.12	3.97	3.55	2.54	3.55
3.	Pengguna mencari rute dengan klik pada peta	3.69	3.16	4.04	5.78	4.17
4.	Pengguna mengirimkan request dengan mode! sama dengan ‘findroute’	1.17	2.92	3.22	3.36	2.67
5.	Pengguna mengirimkan request dengan mode sama dengan ‘searchplace’	1.14	0.78	0.52	1.16	0.90

No.	Aksi	Play Framework				
		Waktu eksekusi pertama	Waktu eksekusi kedua	Waktu eksekusi ketiga	Waktu eksekusi keempat	Rata-rata
1.	Pengguna akses halaman utama KIRI	1.51	1.58	1.67	1.59	1.59
2.	Pengguna mencari rute dengan memasukkan nama tempat	2.08	1.15	2.54	2.74	2.13
3.	Pengguna mencari rute dengan klik pada peta	1.98	4.24	3.22	2.76	3.05
4.	Pengguna mengirimkan request dengan mode sama dengan ‘findroute’	3.63	2.15	2.96	1.41	2.54
5.	Pengguna mengirimkan request dengan mode sama dengan ‘searchplace’	0.72	0.1	1.38	0.88	0.77



Gambar 5.4: Grafik Perbandingan Kecepatan

Dari hasil dan grafik pengujian eksperimental, didapat kesimpulan bahwa waktu eksekusi Play Framework lebih cepat dalam memuat konten yang dinamis. Konten dinamis termasuk dengan struktur data dalam memuat konten dinamis tersebut. Sedangkan untuk konten statis, waktu eksekusi Play Framework kalah dibandingkan dengan PHP. Konten statis yang dimaksud adalah memuat konten tanpa melalui proses terlebih dahulu, misalnya gambar, JavaScript, dan Stylesheet pada aplikasi.

BAB 6

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Dari hasil *porting* aplikasi KIRI, didapatkan kesimpulan-kesimpulan sebagai berikut:

1. Berhasil memahami dan menganalisis kode KIRI dengan melakukan implementasi kode KIRI menjadi Play Framework (Java).
2. Berhasil melakukan porting kode KIRI dalam bahasa PHP menjadi Play Framework (Java).
3. Dari hasil pengujian eksperimental, waktu eksekusi KIRI menggunakan Play Framework dalam memuat konten dinamis lebih cepat, sedangkan dalam memuat konten statis, PHP lebih unggul.

6.2 Saran

Dari hasil penelitian termasuk kesimpulan yang didapat, berikut adalah beberapa saran untuk pengembangan:

1. Penelitian ini menggunakan layanan web untuk algoritma pencarian rute terpendek. Jika layanan web ini dalam keadaan tidak nyala, maka pencarian rute pada aplikasi KIRI tidak dapat berjalan. Sebaiknya algoritma pencarian rute dijalankan dalam aplikasi KIRI sehingga aplikasi KIRI dapat berjalan terus tanpa ada ketergantungan dari layanan web diluar KIRI.

DAFTAR REFERENSI

- [1] N. Leroux and S. D. Kaper, *Play for Java*. Manning Publications Co., 2014.
- [2] Pascal Alfadian, “KIRI.” <http://static.kiri.travel/>, 2014. [Online; diakses 28 September 2015].
- [3] Pascal Alfadian, “TirtayasaGH.” <https://github.com/pascalalfadian/TirtayasaGH>, 2014. [Online; diakses 28 September 2015].
- [4] The PHP Group, “php.” <http://php.net>, 2015. [Online; diakses 28 September 2015].
- [5] Oracle, “Primitive Data Types.” <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>, 2015. [Online; diakses 25 April 2016].
- [6] Oracle, “JDBC Overview.” <http://www.oracle.com/technetwork/java/overview-141217.html>, 2015. [Online; diakses 3 Maret 2016].
- [7] Anonymous, “Injeksi SQL.” https://id.wikipedia.org/wiki/Injeksi_SQL, 2015. [Online; diakses 3 Maret 2016].
- [8] P. A. Santiago, *OpenLayers Cookbook*. Packt Publishing, 2012.
- [9] A. D. Patterson, *Getting Started with Zurb Foundation 4*. Packt Publishing, 2013.
- [10] Google, “Chrome DevTools.” <https://developers.google.com/web/tools/chrome-devtools/index?hl=en#learnmore>, 2015. [Online; diakses 2 Oktober 2015].

LAMPIRAN A

KODE PROGRAM PHP

Listing A.1: handle.php

```

1 <?php
2 require_once '../etc/utils.php';
3 require_once '../etc/constants.php';
4 // We won't tell the error detail to the world. Ssssh!
5 $global_hush_hush = true;
6 start_working();
7 init_mysql();
8 $mode = retrieve_from_post($proto_mode);
9 $version = retrieve_from_post($proto_version, false);
10 $version = is_null($version) ? 1 : $version;
11 $apikey = retrieve_from_post($proto_apikey);
12 check_apikey($apikey);
13 if ($mode == $proto_mode_findroute) {
14     $start = addslashes(retrieve_from_post($proto_routestart));
15     $finish = addslashes(retrieve_from_post($proto_routefinish));
16     $locale = addslashes(retrieve_from_post($proto_locale));
17
18     if (file_exists("../etc/locale/tirtayasa_$locale.php")) {
19         require_once("../etc/locale/tirtayasa_$locale.php");
20     } else {
21         die_nice("Locale not found: $locale");
22     }
23     $presentation = addslashes(retrieve_from_post($proto_presentation));
24     if ($presentation != $proto_presentation_mobile && $presentation != $proto_presentation_desktop) {
25         die_nice("Presentation not understood: $presentation");
26     }
27     // Retrieve from menjangan server.
28     $results = array();
29     if ($version >= 2) {
30         $count = $presentation == $proto_presentation_mobile ? 1 : sizeof($alternatives);
31         for ($i = 0; $i < $count; $i++) {
32             $url = $menjangan_url . "?start=$start&finish=$finish";
33             $url .= '&' . $protokd_maximumwalk . '=' . $alternatives[$i][$protokd_maximumwalk];
34             $url .= '&' . $protokd_walkingmultiplier . '=' . $alternatives[$i][$protokd_walkingmultiplier];
35             $url .= '&' . $protokd_penaltytransfer . '=' . $alternatives[$i][$protokd_penaltytransfer];
36             $result = file_get_contents($url, NULL, NULL, 0, $maximum_http_response_size + 1);
37             if ($result == FALSE) {
38                 die_nice("There's an error while reading the menjangan response.");
39             }
40             $results[$result] = true;
41         }
42     } else {
43         $result = file_get_contents($menjangan_url . "?start=$start&finish=$finish", NULL, NULL, 0,
44             $maximum_http_response_size + 1);
45         if ($result == FALSE) {
46             die_nice("There's an error while reading the menjangan response.");
47         }
48         $results[$result] = true;
49     }
50     foreach ($results as $result=>$dummy) {
51         $travel_time = 0;
52         $route_output = array();
53         $steps = split("\n", $result);
54         foreach ($steps as $step) {
55             $step = trim($step);
56             if ($step == '') {
57                 // Could be the last line, ignore if empty.
58                 continue;
59             }
60             // Path is not found
61             if ($step == $protokd_result_none) {
62                 if (sizeof($results) == 1) {
63                     // There is not other alternative
64                     $route_output[] = array("none", "none", array($start, $finish), $message_routenotfound
65                         [$presentation], null, null);
66                     $travel_time = null;
67                     break;
68                 } else {
69                     // There is alternative, hence we just skip this step.
70                     continue 2;
71                 }
72             }
73             list($means, $means_detail, $route, $distance, $nearbyplaceids) = split("/", $step);

```

```

72     if (!isset($means) || !isset($route) || !isset($distance) || !isset($nearbyplaceids)) {
73         die_nice("Incomplete_response_in_this_line:$step");
74     }
75     $points = split(", ", $route);
76     $from = $points[0];
77     $to = $points[sizeof($points) - 1];
78     // Replace keywords with real location, then construct the detailed path
79     for ($i = 0, $size = sizeof($points); $i < $size; $i++) {
80         if ($points[$i] == $proto_kd_point_start) {
81             $points[$i] = $start;
82         }
83         if ($points[$i] == $proto_kd_point_finish) {
84             $points[$i] = $finish;
85         }
86     }
87
88     // Construct the human readable form of the walk
89     $humanized_from = humanize_point($from);
90     $humanized_to = humanize_point($to);
91     // Convert whole path to human readable form
92     if ($means == $proto_transitmode_walk) {
93         // Remove unnecessary information if not needed.
94         if ($humanized_from == $humanized_to) {
95             // When we're in mobile, skip this step (not really necessary)
96             if ($presentation == $proto_presentation_mobile) {
97                 $humanreadable = null;
98             } else {
99                 $humanreadable = $message_walk_samestreet;
100                $humanreadable = str_replace('%street', $humanized_from, $humanreadable);
101                $humanreadable = str_replace('%distance', format_distance($distance, $locale),
102                                            $humanreadable);
103            }
104        } else {
105            if ($presentation == $proto_presentation_mobile) {
106                $humanized_from .= '%fromicon';
107                $humanized_to .= '%toicon';
108            }
109            $humanreadable = $message_walk;
110            $humanreadable = str_replace('%from', $humanized_from, $humanreadable);
111            $humanreadable = str_replace('%to', $humanized_to, $humanreadable);
112            $humanreadable = str_replace('%distance', format_distance($distance, $locale),
113                                         $humanreadable);
114        }
115        $travel_time += $distance / $speed_walk;
116        $booking_url = null;
117        $editor_url = null;
118    } else {
119        // First the friendly name of the track
120        $means_detail = mysqli_escape_string($global_mysqli_link, $means_detail);
121        $result = mysqli_query($global_mysqli_link, "SELECT_tracks.trackname,tracktypes.name,
122                               tracktypes.url,tracks.extraParameters,tracktypes.speed,tracks.internalInfo
123                               FROM tracks
124                               JOIN tracktypes ON tracktypes.trackTypeId='{$means}' AND tracks.trackTypeId='{$means_detail}' AND tracks.trackId='{$means_detail}'") or
125        die_nice("Can't retrieve the track name from database: " . mysqli_error(
126            $global_mysqli_link));
127        if ($row = mysqli_fetch_row($result)) {
128            $readable_track_name = $row[0];
129            $track_type_name = $row[1];
130        } else {
131            die_nice("Can't find the track name and/or type with the given track id: $means_detail
132                    ");
133        }
134        // Construct the human readable form of the walk
135        if ($presentation == $proto_presentation_mobile) {
136            $humanized_from .= '%fromicon';
137            $humanized_to .= '%toicon';
138
139            $humanreadable = $message_angkot;
140            $humanreadable = str_replace('%from', $humanized_from, $humanreadable);
141            $humanreadable = str_replace('%to', $humanized_to, $humanreadable);
142            $humanreadable = str_replace('%distance', format_distance($distance, $locale),
143                                         $humanreadable);
144            $humanreadable = str_replace('%trackname', $readable_track_name, $humanreadable);
145            $humanreadable = str_replace('%tracktype', $track_type_name, $humanreadable);
146
147            $speed = intval($row[4]);
148            $travel_time += $distance / $speed;
149            if (!is_null($row[2]) && !is_null($row[3])) {
150                $booking_url = $row[2] . $row[3];
151            } else {
152                $booking_url = null;
153            }
154            if (startsWith($row[5], 'angkotwebid:')) {
155                $token = explode(':', $row[5]);
156                $editor_url = $angkotwebid_url_prefix . $token[1] . $angkotwebid_url_suffix;
157            } else {
158                $editor_url = null;
159            }
160            // compatibility patch for older 3rd party apps
161            if ($means == 'bdo_angkot' && intval($version) < 3) {
162                $means = 'angkot';
163            }
164        }
165        if (!is_null($humanreadable)) {
166            $route_output[] = array($means, $means_detail, $points, $humanreadable, $booking_url,
167                                   $editor_url);
168        }
169    }
170
171    $routing_result[$proto_steps] = $route_output;

```

```

162     $routing_result[$proto_traveltimes] = format_traveltimes($travel_time);
163     $routing_results[] = $routing_result;
164 }
165
166 //input log statistic
167 log_statistic("$apikey", "FINDROUTE", "$start/$finish/" . sizeof($results));
168
169 deinit_mysql();
170 if (!is_null($version) && $version >= 2) {
171     print json_encode(array(
172         $proto_status => $proto_status_ok,
173         $proto_routingresults => $routing_results
174     )));
175 } else {
176     print json_encode(array(
177         $proto_status => $proto_status_ok,
178         $proto_routingresult => $routing_results[0][$proto_steps],
179         $proto_traveltimes => $routing_results[0][$proto_traveltimes]
180     ));
181 }
182 } elseif ($mode == $proto_mode_search) {
183     $querystring = retrieve_from_post($proto_search_querystring);
184     $apikey = retrieve_from_post($proto_apikey);
185     $region = retrieve_from_post($proto_region, $version >= 2);
186     $region = is_null($region) ? $proto_region_bandung : $region;
187
188 // Check if there is region modifier from the query string
189 foreach ($regioninfos as $key => $value) {
190     if (preg_match('/' . $value['searchplace_regex'] . '/i', $querystring, $matches,
191         PREG_OFFSET_CAPTURE)) {
192         $region = $key;
193         $querystring = substr($querystring, 0, $matches[0][1]);
194         break;
195     }
196 }
197 $querystring = urlencode($querystring);
198 // The following place search hmethod uses Foursquare venue search service
199 // Note: to show "Search powered by Foursquare" before or at search result.
200 $cached_searchplace = get_from_cache($cache_searchplace, "$region/$querystring");
201 if (!is_null($cached_searchplace)) {
202     $json_output = array(
203         $proto_status => $proto_status_ok,
204         $proto_search_result => json_decode($cached_searchplace, true),
205         $proto_attributions => null
206     );
207     log_statistic("$apikey", "SEARCHPLACE", "$querystring/cache");
208 } else {
209     $city_lat = $regioninfos[$region]['lat'];
210     $city_lon = $regioninfos[$region]['lon'];
211     $city_radius = $regioninfos[$region]['radius'];
212 // pascal: switch to use 4sq
213 // $full_url = "$places_url?client_id=$foursq_client_id&client_secret=$foursq_client_secret&intent=browse&limit=10&v=20130611&ll=$city_lat,$city_lon&radius=$city_radius&query=$querystring";
214 // $full_url = "$places_url?key=$gmaps_server_key&location=$city_lat,$city_lon&radius=$city_radius&keyword=$querystring&types=establishment|route&sensor=true";
215 $result = file_get_contents($full_url, NULL, NULL, 0, $maximum_http_response_size + 1);
216 if ($result === FALSE) {
217     die_nice("There's an error while reading the places response ($full_url).");
218 }
219 // TODO this checking doesn't seem to work.
220 if (sizeof($result) > $maximum_http_response_size) {
221     die_nice("Data returned from $full_url is greater than the maximum.");
222 }
223
224 $json_result = json_decode($result, true);
225 if ($json_result['status'] == 'OK' || $json_result['status'] == 'ZERO_RESULTS') {
226     $search_result = array();
227     if ($json_result['status'] == 'ZERO_RESULTS') {
228         log_error("Place search not found: " . "$querystring");
229         $size = 0;
230     } else {
231         $size = min(sizeof($json_result['results']), $search_maxresult);
232     }
233     for ($i = 0; $i < $size; $i++) {
234         $current_venue = $json_result['results'][$i];
235         $search_result[$i][$proto_placename] = $current_venue['name'];
236         $search_result[$i][$proto_location] = sprintf(
237             '%.' . $latlon_precision . 'f,%.' . $latlon_precision . 'f',
238             $current_venue['geometry']['location']['lat'],
239             $current_venue['geometry']['location']['lng']
240         );
241     }
242 // pascal: switch to this for 4sq
243 // if ($json_result['meta']['code'] == '200') {
244 //     $search_result = array();
245 //     $size = min(sizeof($json_result['response']['venues']), $search_maxresult);
246 //     for ($i = 0; $i < $size; $i++) {
247 //         $current_4sq_venue = $json_result['response']['venues'][$i];
248 //         $search_result[$i][$proto_placename] = $current_4sq_venue['name'];
249 //         if (isset($current_4sq_venue['location']['address'])) {
250 //             $search_result[$i][$proto_placename] .= ' - ' . $current_4sq_venue['location']['address'];
251 //         }
252 //         $search_result[$i][$proto_location] = sprintf(
253 //             '%.' . $latlon_precision . 'f,%.' . $latlon_precision . 'f',
254 //             $current_4sq_venue['location']['lat'],
255 //             $current_4sq_venue['location']['lng']
256 //         );
257 }

```

```

256 //      );
257 //      if ($size == 0) {
258 //          log_error("Place search not found: \"$querystring\"");
259 //      }
260 $json_output = array(
261     $proto_status => $proto_status_ok,
262     $proto_search_result => $search_result,
263     $proto_attributions => null
264 );
265 );
266
267 //input log statistic
268 log_statistic("$apikey", "SEARCHPLACE", "$querystring/$size");
269 // Store to cache
270 put_to_cache($cache_searchplace, "$region/$querystring", json_encode($search_result));
271 deinit_mysql();
272
273 } else {
274 // switch to use 4sq
275 die_nice('Foursquare Place Search returned error: ' . $json_result['meta'][ 'code'] . " (for
this request: $full_url)", false);
276 die_nice('PlaceSearch returned error:' . $json_result[ 'status'] . " (for this request:
$full_url)", false);
277 }
278 }
279
280 print json_encode($json_output);
281 } elseif ($mode == $proto_mode_reporterror) {
282 $errorcode = retrieve_from_post($proto_errorcode);
283 log_error("Client reported error:$errorcode");
284 well_done();
285 } elseif ($mode == $proto_mode_nearbytransports) {
286 $start = retrieve_from_post($proto_routestart);
287 if ($version >= 2) {
288     $lines = explode("\n", file_get_contents($menjangan_url . "?start=$start", NULL, NULL, 0,
$maximum_http_response_size + 1));
289     $nearby_result = array();
290     foreach ($lines as $line) {
291         list($trackTypeId, $trackId, $distance) = explode("/", $line);
292         $result = mysqli_query($global_mysqli_link, "SELECT trackName FROM tracks WHERE trackId =
$trackId AND trackTypeId ='$trackTypeId'") or
293             die_nice("Can't get nearest track details:" . mysqli_error($global_mysqli_link));
294         while ($row = mysqli_fetch_array($result)) {
295             $trackName = $row[0];
296             $nearby_result[] = array (
297                 $trackTypeId,
298                 $trackId,
299                 $trackName,
300                 $distance
301             );
302         }
303     }
304     usort($nearby_result, "nearby_result_compare");
305     log_statistic($apikey, "NEARBYTRANSPORTS", " $start" . sizeof($results));
306     $json_output = array(
307         $proto_status => $proto_status_ok,
308         $proto_nearbytransports => $nearby_result
309     );
310     print json_encode($json_output);
311 } else {
312     die_nice("Nearby transit is not supported in version 1");
313 }
314 } else {
315     die_nice("Mode not understood: " . $mode . " ");
316 }
317 /**
318 * Replace a location point into a human readable form with most effort.
319 * This function requires MySQL connection to be active.
320 * For example:
321 * <ul>
322 * <li>'start' => 'your starting point'.
323 * <li>'xxx.xxx,yyy.yyy' => check in cache, or reverse geocode from google if miss
324 * </ul>
325 * @param string $location the original location constant
326 */
327 function humanize_point($location) {
328     global $global_mysqli_link;
329     global $protokd_point_start, $protokd_point_finish;
330     global $message_start, $message_finish;
331     global $gmaps_geocode_url, $gmaps_server_key, $maximum_http_response_size;
332     global $cache_geocoding;
333     if ($location == $protokd_point_start) {
334         return $message_start;
335     } else if ($location == $protokd_point_finish) {
336         return $message_finish;
337     } else {
338         $location = mysqli_escape_string($global_mysqli_link, $location);
339         $cached_geocode = get_from_cache($cache_geocoding, $location);
340         if (!is_null($cached_geocode)) {
341             return $cached_geocode;
342         } else {
343             // query from google.
344             $full_url = "$gmaps_geocode_url?key=$gmaps_server_key&latlng=$location&sensor=false";
345             $result = file_get_contents($full_url, NULL, NULL, 0, $maximum_http_response_size + 1);
346             if ($result == FALSE) {
347                 die_nice("There's an error while reading the geocoding response from $full_url.");
348             } // TODO this checking doesn't seem to work.
349             if (sizeof($result) > $maximum_http_response_size) {
350

```

```

351         die_nice("Data returned from full url is greater than the maximum.");
352     }
353     $json_response = json_decode($result, true);
354     if ($json_response === NULL) {
355         die_nice("Unable to retrieve JSON response from Google geocoding service.");
356     }
357     if ($json_response['status'] == 'OK') {
358         $bestguess = $location;
359         for ($i = 0; $i < count($json_response['results']); $i++) {
360             foreach ($json_response['results'][$i]['address_components'] as $component) {
361                 if (in_array('transit_station', $component['types']) || in_array('route',
362                     $component['types'])) {
363                     put_to_cache($cache_geocoding, $location, $component['long_name']);
364                     return $component['long_name'];
365                 }
366                 $bestguess = $component['long_name'];
367             }
368             log_error("Warning: can't find street name, use best guess for $location.");
369             put_to_cache($cache_geocoding, $location, $bestguess);
370             return $bestguess;
371         } else if ($json_response['status'] == 'ZERO_RESULTS') {
372             // If not found, return the coordinate.
373             log_error("Warning: can't find coordinate for $location.");
374             return $location;
375         } else {
376             die_nice("Problem while geocoding from Google reverse geocoding: " . $result);
377         }
378     }
379 }
380 /**
381 * A sorting comparison function to be used in nearby transports
382 * @param array $a an array, where index 3 is the distance
383 * @param array $b an array, where index 3 is the distance
384 * @return number as in usort() spec
385 */
386 function nearby_result_compare($a, $b) {
387     if ($a[3] > $b[3]) {
388         return +1;
389     } else if ($a[3] < $b[3]) {
390         return -1;
391     } else {
392         return 0;
393     }
394 }
395 /**
396 * Format a number into distance
397 * @param float $distance The distance
398 * @param string $locale The locale code 'en' or 'id'
399 */
400 function format_distance($distance, $locale) {
401     if (!is_numeric($distance)) {
402         die_nice("Distance is not a floating number: $distance");
403     }
404     if ($distance < 1) {
405         // Less than 1 km, show in meter
406         return floor($distance * 1000) . " meter";
407     } else {
408         // More than 1 km, show in km
409         $decimal = $locale == 'id' ? ',' : '.';
410         $fdist = floor($distance);
411         return $fdist . $decimal . floor(($distance - $fdist) * 10) . " kilometer";
412     }
413 }
414 /**
415 * Format numeric travel time to a human readable one.
416 * @param float $time the travel time in hour.
417 */
418 function format_traveltimes($time) {
419     global $message_hour, $message_min;
420     if (is_null($time)) {
421         return null;
422     } elseif ($time > 1) {
423         return round($time) . " $message_hour";
424     } else {
425         return 5 * ceil($time * 60 / 5) . " $message_min";
426     }
427 }
428 /**
429 * method for check if the api keys is in database.
430 * @param string $apikey provided by user to be checked.
431 */
432 function check_apikey ($apikey)
433 {
434     //check message with global hush_hush
435     global $global_hush_hush, $global_mysqli_link;
436     $global_hush_hush = false;
437     //check apikey: is this api key exist in database?
438     $apisqlquery = mysqli_query($global_mysqli_link, "SELECT apikeys.verifier , apikeys.ipFilter FROM
439     apikeys WHERE
440     apikeys.verifier = '$apikey'") or die_nice("failed to execute query on Apikey check.");
441     // if it exist, it must be found 1 row (because apiKeyOrDomain are unique)
442     if ($querry = mysqli_fetch_array($apisqlquery))
443     {
444         // check for ip filter
445         if ($_SERVER['REMOTE_ADDR'] != $querry['ipFilter'] && $querry['ipFilter'] != NULL)
446
447

```

```
448|     {
449|         die_nice("IP_address_is_not_accepted_for_this_API_key.");
450|     }
451| } else
452| {
453|     die_nice("API_key_is_not_recognized:$apikey.");
454| }
455| }
456| ?>
```

LAMPIRAN B

KODE PROGRAM *CONTROLLER*

Listing B.1: Application.java

```
1 package controllers;
2
3 import com.fasterxml.jackson.databind.JsonNode;
4 import com.fasterxml.jackson.databind.node.ArrayNode;
5 import com.fasterxml.jackson.databind.node.NullNode;
6 import com.fasterxml.jackson.databind.node.ObjectNode;
7 import models.*;
8 import models.helpers.Constants;
9 import models.helpers.Utils;
10 import play.cache.CacheApi;
11 import play.data.DynamicForm;
12 import play.data.Form;
13 import play.db.DB;
14 import play.i18n.Messages;
15 import play.libs.F;
16 import play.libs.Json;
17 import play.libs.ws.WS;
18 import play.libs.ws.WSResponse;
19 import play.mvc.*;
20
21 import views.html.*;
22
23 import javax.inject.Inject;
24 import java.sql.Connection;
25 import java.sql.ResultSet;
26 import java.sql.Statement;
27 import java.util.*;
28
29 public class Application extends Controller {
30
31     private DynamicForm dynamicForm;
32     @Inject
33     CacheApi cache;
34
35     private static final Comparator<ArrayList<String>> comparator = new Comparator<ArrayList<String>>() {
36         @Override
37         public int compare(ArrayList<String> o1, ArrayList<String> o2) {
38             //membandingkan array dengan indeks ke 3,yaitu jarak
39             return Double.compare(Double.parseDouble(o1.get(3)),Double.parseDouble(o2.get(3)));
40         }
41     };
42
43
44     public Result index() {
45         String locale = retrieve_data(Constants.PROTO_LOCALE);
46
47         Http.Cookie cookieLocale = request().cookie(Constants.PROTO_LOCALE);
48         if(locale==null){
49             if(cookieLocale!=null){
50                 locale = Utils.validateLocale(cookieLocale.value());
51             } else{
52                 locale = Constants.PROTO_LOCALE_ENGLISH;
53             }
54         } else{
55             //set default id
56             locale = Utils.validateLocale(locale);
57             response().setCookie(Constants.PROTO_LOCALE, locale);
58         }
59
60         //i18n
61         ctx().setTransientLang(locale);
62
63         String region = retrieve_data(Constants.PROTO_REGION);
64
65         Http.Cookie cookieRegion = request().cookie(Constants.PROTO_REGION);
66
67         if(region==null){
68             if(cookieRegion!=null){
69                 region = Utils.validateRegion(cookieRegion.value());
70             } else{
71                 region = Constants.PROTO_REGION_BANDUNG;
72             }
73         } else{//set default id
```

```

74     region = Utils.validateLocale(region);
75     response().setCookie(Constants.PROTO_REGION, region);
76 }
77
78 String regions = new String();
79
80 for (Map.Entry<String, ProtoRegion> iterator : Constants.REGIONINFOS.entrySet()) {
81     regions+= iterator.getKey()+"\{center:"+ iterator.getValue().getLat() + ","+ iterator.
82         getValue().getLon() + ",zoom:"+ iterator.getValue().getZoom() +"\},";
83 }
84
85 String message =
86     "var input_text=[], coordinates=[];\n" +
87     "input_text['start']=null;\n" +
88     "coordinates['start']=null;\n" +
89     "input_text['finish']=null;\n" +
90     "coordinates['finish']=null;\n" +
91
92     "var locale='"+ locale+"';\n" +
93     "var region='"+region+"';\n" +
94     "var messageBuyTicket='"+Messages.get("index_buylticket")+"';\n" +
95     "var messageConnectionError='"+Messages.get("index_connectionerror")+"';\n" +
96     "var messageFillBoth='"+Messages.get("index_fillboth")+"';\n" +
97     "var messageNotFound='"+Messages.get("index_notfound")+"';\n" +
98     "var messageOops='"+Messages.get("index_oops")+"';\n" +
99     "var messageOrderTicketHere='"+Messages.get("index_order_ticket_here")+"';\n" +
100    "var messagePleaseWait='<img src=\"/assets/images/loading.gif\" alt=\"...\"/>'+
101      '+Messages.get("index_pleasewait")+';\n" +
102    "var messageITake='"+Messages.get("index_itake")+"';\n" +
103    "var region='"+region+"';\n" +
104    "var regions={\n" +
105        regions +
106    "};\n" +
107    "$(document).foundation();";
108
109 return ok(index.render(locale, Constants.REGIONINFOS, message, region));
110
111 public Result handle()
112 {
113
114     response().setContent-Type("application/json");
115     response().setHeader("Cache-control", "no-cache");
116     response().setHeader("Pragma", "no-cache");
117 //initialize
118 try{
119
120     StringBuilder output = new StringBuilder();
121
122 //because java cant take string as an index, use map instead
123
124     Map<String, Boolean> results = new HashMap<String, Boolean>();
125
126     ArrayList<Map<String, JsonNode>> routing_results = new ArrayList<Map<String, JsonNode>>();
127
128     ArrayList<ArrayList<Object>> route_output = new ArrayList<ArrayList<Object>>();
129
130
131     String mode = retrieve_data("mode");
132     int version = Integer.parseInt(retrieve_data("version"));
133     if(version == 0){
134         version = 1;
135     }
136     String apikey = retrieve_data("apikey");
137
138     if(!check_apikey(apikey).equals("{}")){
139
140         return ok(check_apikey(apikey));
141     }
142
143     if(mode.equals(Constants.PROTO_MODE_FINDROUTE))
144     {
145         results = new HashMap<String, Boolean>();
146
147         String start = retrieve_data("start");
148         String finish = retrieve_data("finish");
149         String locale = retrieve_data("locale");
150
151
152 //localization
153     ctx().setTransientLang(locale);
154
155     String presentation = retrieve_data("presentation");
156
157     String result = null;
158     if(!presentation.equals(Constants.PROTO_PRESENTATION_MOBILE) && !presentation.equals(Constants.
159         .PROTO_PRESENTATION_DESKTOP))
160     {
161         Utils.log_error("Presentation not understood:"+ presentation);
162     }
163
164     if(version >=2)
165     {
166         int count = presentation.equals(Constants.PROTO_PRESENTATION_MOBILE)?1:Constants.
167             ALTERNATIVES.length;
168
169         for (int i = 0;i<count;i++){
170             result = getFromMenjangan(start, finish, Constants.ALTERNATIVES[i].getMw(), Constants.

```

```

169         ALTERNATIVES[ i ].getWm() , Constants.ALTERNATIVES[ i ].getPt() );
170     results.put( result , true );
171 }
172 } else // version <2
173 {
174     result = getFromMenjangan( start , finish );
175     results.put( result , true );
176 }
177 }
178 }
179
180 double travel_time = 0;
181 for ( Map.Entry<String , Boolean> iterator : results.entrySet() ) {
182     route_output = new ArrayList<ArrayList<Object>>();
183     travel_time = 0;
184
185     String[] steps = iterator.getKey().split( "\n" );
186     for ( String step: steps ) {
187
188         step = step.trim();
189         if( step.equals( "" ) ){
190             continue;
191         }
192
193         if( step.equals( Constants.PROTOKD_RESULT_NONE ) ){
194             if( results.size() == 1 ){
195                 ArrayList<Object> arrRouteOutput = new ArrayList<Object>();
196                 arrRouteOutput.add( "none" );
197                 arrRouteOutput.add( "none" );
198
199                 String arrStartFinish[] = new String[2];
200                 arrStartFinish[0] = start;
201                 arrStartFinish[1] = finish;
202
203                 arrRouteOutput.add( arrStartFinish );
204                 arrRouteOutput.add( Messages.get( "message_routenotfound_ " + presentation ) );
205                 arrRouteOutput.add( null );
206                 arrRouteOutput.add( null );
207
208                 route_output.add( arrRouteOutput );
209                 travel_time = 0;
210                 break;
211             } else{
212                 continue;
213             }
214         }
215
216         String[] stepSplit = step.split( "/" );
217         String means = stepSplit[0];
218         String means_detail = stepSplit[1];
219         String route = stepSplit[2];
220         String distance = stepSplit[3];
221
222         if( means.isEmpty() || route.isEmpty() || distance.isEmpty() ){
223             //die_nice
224             Utils.log_error( "Incomplete_response_in_this_line:" );
225         }
226
227         String[] points = route.split( "," );
228
229         String from = points[0];
230
231         String to = points[points.length - 1];
232
233         String booking_url = null , editor_url = null;
234
235         for ( int i = 0; i < points.length; i++ )
236         {
237
238             if( points[i].equals( Constants.PROTOKD_POINT_START ) )
239             {
240                 points[i] = start;
241             }
242
243             if( points[i].equals( Constants.PROTOKD_POINT_FINISH ) )
244             {
245                 points[i] = finish;
246             }
247         }
248
249
250         String humanized_from = "" , humanized_to = "";
251         try {
252             humanized_from = humanize_point( from );
253         } catch ( Exception e ) {
254             Utils.log_error( "humanized_from_error:" + e.getMessage() );
255             e.printStackTrace();
256         }
257
258         try {
259             humanized_to = humanize_point( to );
260         } catch ( Exception e ) {
261             Utils.log_error( "humanized_to_error:" + e.getMessage() );
262         }
263
264         String humanreadable = null;
265         if( means.equals( Constants.PROTOKD_TRANSITMODE_WALK ) )
266     {

```

```

267     if(humanized_from.equals(humanized_to))
268     {
269         if(!presentation.equals(Constants.PROTO_PRESENTATION_MOBILE))
270         {
271             humanreadable = Messages.get("message_walk_samestreet");
272             humanreadable = humanreadable.replaceAll("%street",humanized_from);
273             humanreadable = humanreadable.replaceAll("%distance",format_distance(
274                 Double.parseDouble(distance),locale));
275         }
276     }
277     {
278         if(presentation.equals(Constants.PROTO_PRESENTATION_MOBILE))
279         {
280             humanized_from += "";
281             humanized_to += "%toicon";
282         }
283         humanreadable = Messages.get("message_walk");
284         humanreadable = humanreadable.replaceAll("%from",humanized_from);
285         humanreadable = humanreadable.replaceAll("%to",humanized_to);
286         humanreadable = humanreadable.replaceAll("%distance",format_distance(Double.
287             parseDouble(distance),locale));
288     }
289 }
290
291 travel_time += Double.parseDouble(distance) / Constants.SPEED_WALK;
292 booking_url = null;
293 editor_url = null;
294
295 else
296 {
297     String readable_track_name = null,track_type_name = null;
298
299 double speed = 0;
300 java.sql.Connection
301         connection = DB.getConnection();
302 try {
303     Statement statement = connection.createStatement();
304
305     ResultSet result2 = statement.executeQuery("SELECT_tracks.trackname,_
306         tracktypes.name,_tracktypes.url,_tracks.extraParameters,_tracktypes.speed,
307         _tracks.internalInfo FROM_tracks JOIN_tracktypes ON_tracktypes.trackTypeId
308         =" +
309         means + 'AND_tracks.trackTypeId=' + means + 'AND_tracks.trackid='
310         "+ means_detail + "');");
311
312 while (result2.next()) {
313     //php starts from 0,jdbc start from 1
314
315     readable_track_name = result2.getString(1);
316     track_type_name = result2.getString(2);
317     speed = result2.getDouble(5);
318
319     if((result2.getString(3) != null && !result2.getString(3).isEmpty()) &&
320         result2.getString(4) != null && !result2.getString(4).isEmpty())
321     {
322         booking_url = result2.getString(3) + result2.getString(4);
323     }else{
324         booking_url = null;
325     }
326
327     if(result2.getString(6).startsWith("angkotwebid:"))
328     {
329         String [] token = result2.getString(6).split(":");
330         editor_url = Constants.ANGKOTWEBID_URL_PREFIX + token[1] + Constants.
331             ANGKOTWEBID_URL_SUFFIX;
332     }else{
333         editor_url = null;
334     }
335 }
336
337 connection.close();
338 } catch (Exception e) {
339     Utils.log_error("Can't retrieve the track name from database:" + e.getMessage
340     ());
341 }
342
343 if(presentation.equals(Constants.PROTO_PRESENTATION_MOBILE))
344 {
345     humanized_from += "%fromicon";
346     humanized_to += "%toicon";
347 }
348
349 humanreadable = Messages.get("message_angkot");
350 try{
351
352     humanreadable = humanreadable.replaceAll("%from",humanized_from);
353     humanreadable = humanreadable.replaceAll("%to",humanized_to);
354     humanreadable = humanreadable.replaceAll("%distance",format_distance(Double.
355         parseDouble(distance),locale));
356     humanreadable = humanreadable.replaceAll("%trackname",readable_track_name);
357     humanreadable = humanreadable.replaceAll("%tracktype",track_type_name);
358 }catch (Exception ex){
359     ex.printStackTrace();
360 }

```

```

356     travel_time += Double.parseDouble(distance) / speed;
357
358     if(means.equals("bdo_angkot") && version < 3)
359     {
360         means = "angkot";
361     }
362 }
363 if(humanreadable != null && !humanreadable.isEmpty())
364 {
365
366     ArrayList<Object> arrRouteOutput = new ArrayList<Object>();
367     arrRouteOutput.add(means);
368     arrRouteOutput.add(means_detail);
369     arrRouteOutput.add(points);
370     arrRouteOutput.add(humanreadable);
371     arrRouteOutput.add(booking_url==null? NullNode.getInstance():booking_url);
372     arrRouteOutput.add(editor_url==null?NullNode.getInstance():editor_url);
373
374     route_output.add(arrRouteOutput);
375 }
376 }
377 }
378
379 Map<String , JsonNode> routing_result = new HashMap<String , JsonNode>();
380
381 routing_result.put(Constants.PROTO_STEPS,Json.toJson(route_output));
382 routing_result.put(Constants.PROTO_TRAVELTIME, Json.toJson(format_travelttime(travel_time)));
383
384 routing_results.add(routing_result);
385 }
386
387 Utils.log_statistic(apikey , "FINDROUTE" , start + "/" + finish + "/" + results.size());
388
389 ObjectNode objectNode = Json.newObject();
390 if(version >=2)
391 {
392     objectNode.put(Constants.PROTO_STATUS,Constants.PROTO_STATUS_OK);
393     objectNode.put(Constants.PROTO_ROUTINGRESULTS, Json.toJson(routing_results));
394
395 } else{
396     objectNode.put(Constants.PROTO_STATUS,Constants.PROTO_STATUS_OK);
397     objectNode.put(Constants.PROTO_ROUTINGRESULT, Json.toJson(routing_results.get(0).get(
398         Constants.PROTO_STEPS)));
399     objectNode.put(Constants.PROTO_TRAVELTIME, Json.toJson(routing_results.get(0).get(
400         Constants.PROTO_TRAVELTIME)));
401 }
402
403 String str = Json.stringify(objectNode);
404 output.append(str);
405
406 }else if(mode.equals(Constants.PROTO_MODE_SEARCH))
407 {
408
409     ObjectNode json_output = Json.newObject();
410
411     String querystring = retrieve_data(Constants.PROTO_SEARCH_QUERYSTRING);
412
413     String region = retrieve_data(Constants.PROTO_REGION);
414
415     region = region == null? Constants.PROTO_REGION_BANDUNG : region;
416
417     for (Map.Entry<String , ProtoRegion> iterator : Constants.REGIONINFOS.entrySet()) {
418         if( Utils.indexPregMatch(iterator.getValue().getSearchPlace_regex() ,querystring)!=-1){
419             region = iterator.getKey();
420             int matches = Utils.indexPregMatch(iterator.getValue().getSearchPlace_regex() ,
421                 querystring);
422             querystring = querystring.substring(0 ,matches);
423
424             break;
425         }
426     }
427
428     String cached_searchplace = Utils.get_from_cache(Constants.CACHE_SEARCHPLACE,region + "/" +
429     querystring);
430
431     if(cached_searchplace != null && !cached_searchplace.isEmpty()){
432         json_output.put(Constants.PROTO_STATUS,Constants.PROTO_STATUS_OK);
433         json_output.put(Constants.PROTO_SEARCH_RESULT,Json.toJson(Json.parse(cached_searchplace)));
434
435         json_output.put(Constants.PROTO_ATTRIBUTIONS,Json.toJson(NullNode.getInstance()));
436
437         Utils.log_statistic(apikey , "SEARCHPLACE" , querystring + "/cache");
438     }else
439     {
440         double city_lat = Constants.REGIONINFOS.get(region).getLat();
441         double city_lon = Constants.REGIONINFOS.get(region).getLon();
442         int city_radius = Constants.REGIONINFOS.get(region).getRadius();
443
444         String result = getPlacesAPI(city_lat+"."+city_lon ,Integer.toString(city_radius) ,
445             querystring);
446
447         JsonNode json_result = null;
448         try{
449
450             json_result = Json.parse(result);

```



```

543         list.add(trackId);
544         list.add(trackName);
545         list.add(distance);
546         nearby_result.add(list);
547     }
548
549
550     connection.close();
551 } catch (Exception e) {
552     Utils.log_error("Can't get nearest track details: " + e.getMessage());
553 }
554
555 }
556 Collections.sort(nearby_result, comparator);
557
558 Utils.log_statistic(apikey, "NEARBYTRANSPORTS", start + results.size());
559
560 ObjectNode json_output = Json.newObject();
561
562 json_output.put(Constants.PROTO_STATUS, Constants.PROTO_STATUS_OK);
563 json_output.put(Constants.PROTO_NEARBYTRANSPORTS, Json.toJson(nearby_result));
564
565
566 output.append(json_output.toString());
567
568 } else {
569     Utils.log_error("Nearby transit is not supported in version 1");
570 }
571
572 } else {
573     Utils.log_error("Mode not understood: " + mode);
574 }
575
576 return ok(output.toString());
577 } catch (Exception e) {
578     ObjectNode json_output = Json.newObject();
579
580     json_output.put(Constants.PROTO_STATUS, Constants.PROTO_STATUS_ERROR);
581     json_output.put(Constants.PROTO_MESSAGE, e.getMessage());
582
583     return internalServerError(Json.stringify(json_output));
584 }
585
586 }
587
588 }
589
590 }
591
592
593 public String getFromMenjangan(String start, String finish, double mw, double wm, double pt) {
594     F.Promise<String> promise = WS.url(Constants.MENJANGAN_URL)
595         .setQueryParameter("start", start)
596         .setQueryParameter("finish", finish)
597         .setQueryParameter("mw", Double.toString(mw))
598         .setQueryParameter("wm", Double.toString(wm))
599         .setQueryParameter("pt", Double.toString(pt))
600         .get()
601         .map(
602             new F.Function<WSResponse, String>() {
603                 public String apply(WSResponse response) {
604                     String result = response.getBody();
605                     return result;
606                 }
607             }
608         );
609
610
611     long timeout = 512000L; // 1 sec might be too many for most cases!
612     String result = promise.get(timeout);
613     return result;
614 }
615
616
617 public String getFromMenjangan(String start, String finish) {
618     F.Promise<String> promise = WS.url(Constants.MENJANGAN_URL)
619         .setQueryParameter("start", start)
620         .setQueryParameter("finish", finish)
621         .get()
622         .map(
623             new F.Function<WSResponse, String>() {
624                 public String apply(WSResponse response) {
625                     String result = response.getBody();
626                     return result;
627                 }
628             }
629         );
630
631
632     long timeout = 1000L; // 1 sec might be too many for most cases!
633     String result = promise.get(timeout);
634     return result;
635 }
636
637 public String getFromMenjangan(String start) {
638     F.Promise<String> promise = WS.url(Constants.MENJANGAN_URL)
639         .setQueryParameter("start", start)
640         .get()
641         .map(

```



```

740         }
741     } else{
742         objectNode = Utils.die_nice("API_key_is_not_recognized:" + apikey);
743     }
744 }
745
746 connection.close();
747 } catch (Exception e) {
748     objectNode = Utils.die_nice("failed_to_execute_query_on_Apikey_check." + e.getMessage());
749 }
750
751 output.append(objectNode.toString());
752 return output.toString();
753 }
754
755
756
757 public String humanize_point(String location) throws Exception {
758     if(location.equals(Constants.PROTOKD_POINT_START)){
759         return Messages.get("message_start");
760     }else if(location.equals(Constants.PROTOKD_POINT_FINISH)){
761
762         return Messages.get("message_finish");
763     }else{
764
765         String cached_geocode = Utils.get_from_cache(Constants.CACHE_GEOCODING,location);
766
767         if(cached_geocode != null && !cached_geocode.isEmpty()){
768             return cached_geocode;
769         }else{
770
771             String full_url = Constants.GMAPS_GEOCODE_URL + "?key=" + Constants.GMAPS_SERVER_KEY + "&
772                 latlng=" + location + "&sensor=false";
773             String result = file_get_contents(full_url);
774             JsonNode json_response = null;
775             try{
776
777                 json_response = Json.parse(result);
778             }catch(Exception e){
779                 Utils.log_error("ERROR_HUMANIZE_POINT:" + e.getMessage());
780             }
781
782             if(json_response.findPath("status").textValue().equals("OK")){
783                 String bestguess = location;
784                 ArrayNode arrayNode = (ArrayNode)json_response.withArray("results");
785                 String[] arr = {"transit_station","route"};
786                 for (int i = 0;i<arrayNode.size();i++){
787
788                     for (JsonNode component : json_response.findPath("results").get(0).get("address_components")){
789
790                         if(Json.stringify(component.findPath("types")).contains("transit_station") ||
791                             Json.stringify(component.findPath("types")).contains("route")){
792
793                             Utils.put_to_cache(Constants.CACHE_GEOCODING,location,component.findPath("long_name").textValue());
794                             return component.findPath("long_name").textValue();
795                         }
796                         bestguess = component.findPath("long_name").textValue();
797                     }
798
799                     Utils.log_error("Warning: can't find street name, use bestguess" + bestguess + " for "
800                         + location);
801                     Utils.put_to_cache(Constants.CACHE_GEOCODING,location,bestguess);
802                     return bestguess;
803
804                 }else if(json_response.get("status").equals("ZERO_RESULTS")){
805
806                     Utils.log_error("Warning: can't find coordinate for" + location);
807                     return location;
808                 }else{
809                     Utils.log_error("Problem while geocoding from Google reverse geocoding:");
810
811                     return "";
812                 }
813
814
815             }
816
817         }
818
819     public String format_distance(double distance ,String locale){
820
821
822         if(distance<1.0){
823             return (int) (Math.floor(distance*1000.0)) + "meter";
824         }else{
825             char decimal = locale == "id"? ',' : '.';
826             double fdist = Math.floor(distance);
827             return (int) (fdist + decimal + Math.floor((distance - fdist) * 10)) + "kilometer";
828         }
829
830     }
831
832 }

```


LAMPIRAN C

KODE PROGRAM *MODELS*

Listing C.1: Alternative.java

```
1 package models;
2
3 public class Alternative {
4     private double mw,wm,pt;
5
6
7     public Alternative(double mw, double wm, double pt) {
8         this.mw = mw;
9         this.wm = wm;
10        this.pt = pt;
11    }
12
13    public double getPt() {
14        return pt;
15    }
16
17
18    public double getWm() {
19        return wm;
20    }
21
22
23    public double getMw() {
24        return mw;
25    }
26
27 }
```

Listing C.2: ProtoRegion.java

```
1 package models;
2
3 public class ProtoRegion {
4
5     private double lat,lon;
6     private int radius,zoom;
7     private String searchPlace_regex ,name;
8
9     public ProtoRegion(double lat, double lon, int radius, int zoom, String searchPlace_regex ,String name)
10    {
11        this.lat = lat;
12        this.lon = lon;
13        this.radius = radius;
14        this.zoom = zoom;
15        this.searchPlace_regex = searchPlace_regex ;
16        this.name = name;
17    }
18
19    public double getLon() {
20        return lon;
21    }
22
23    public double getLat() {
24        return lat;
25    }
26
27    public int getZoom() {
28        return zoom;
29    }
30
31    public int getRadius() {
32        return radius;
33    }
34
35    public String getSearchPlace_regex() {
36        return searchPlace_regex ;
37    }
38
39    public String getName() {
40        return name;
41    }
42 }
```

Listing C.3: Constants.java

```

1 package models.helpers;
2
3 import models.Alternative;
4 import models.ProtoRegion;
5
6 import java.util.*;
7
8 public final class Constants {
9
10
11     public static final Alternative[] ALTERNATIVES;
12
13     public static final Map<String, ProtoRegion> REGIONINFOS;
14
15
16     // Cache types and expiry
17     public static final String CACHE_GEOCODING = "geocoding";
18     public static final String CACHE_SEARCHPLACE = "searchplace";
19
20     public static final String PLACES_URL = "https://maps.googleapis.com/maps/api/place/nearbysearch/json"
21         ;
22
23     /** Maximum number of search result. */
24     public static final int SEARCH_MAXRESULT = 10;
25     /** API key for server side apps. */
26     public static final String GMAPS_SERVER_KEY = "AIzaSyBalbNBVkhvxSFd8U_Cn7HsHux6M-DIk4";
27     /** URL for Google Maps" Reverse geocoding web service. */
28     public static final String GMAPS_GEOCODE_URL = "https://maps.googleapis.com/maps/api/geocode/json";
29
30     public static final String ANGKOTWEBID_URL_PREFIX = "https://angkot.web.id/go/route/";
31     public static final String ANGKOTWEBID_URL_SUFFIX = "?ref=kiri";
32
33     public static final int SPEED_WALK = 5;
34     public static final String MENJANGAN_URL = "http://newmenjangan.cloudapp.net:8000";
35
36     public static final String PROTO_ATTRIBUTIONS = "attributions";
37
38     public static final String PROTO_ERRORCODE = "errorcode";
39
40     public static final String PROTO_LOCALE = "locale";
41     public static final String PROTO_LOCALE_INDONESIA = "id";
42     public static final String PROTO_LOCALE_ENGLISH = "en";
43     public static final String PROTO_LOCATION = "location";
44     public static final String PROTO_MESSAGE = "message";
45
46     public static final String PROTO_MODE_FINDROUTE = "findroute";
47     public static final String PROTO_MODE_REPORTERROR = "reporterror";
48     public static final String PROTO_MODE_SEARCH = "searchplace";
49     public static final String PROTO_MODE_NEARBYTRANSPORTS = "nearbytransports";
50     public static final String PROTO_NEARBYTRANSPORTS = "nearbytransports";
51     public static final String PROTO_PLACENAME = "placename";
52     public static final String PROTO_PRESENTATION_DESKTOP = "desktop";
53     public static final String PROTO_PRESENTATION_MOBILE = "mobile";
54     public static final String PROTO_REGION = "region";
55     public static final String PROTO_REGION_BANDUNG = "bdo";
56     public static final String PROTO_REGION_JAKARTA = "cgk";
57     public static final String PROTO_REGION_SURABAYA = "sub";
58     public static final String PROTO_REGION_MALANG = "mlg";
59     public static final String PROTO_ROUTESTART = "start";
60     public static final String PROTO_ROUTINGRESULT = "routingresult";
61     public static final String PROTO_ROUTINGRESULTS = "routingresults";
62     public static final String PROTO_SEARCH_QUERYSTRING = "querystring";
63     public static final String PROTO_SEARCH_RESULT = "searchresult";
64     public static final String PROTO_STATUS = "status";
65     public static final String PROTO_STATUS_ERROR = "error";
66     public static final String PROTO_STATUS_OK = "ok";
67     public static final String PROTO_STEPS = "steps";
68     public static final String PROTO_TRAVELTIME = "traveltime";
69
70     // KalapaDago protocol constants
71     public static final String PROTOKD_POINT_FINISH = "finish";
72     public static final String PROTOKD_POINT_START = "start";
73     public static final String PROTOKD_RESULT_NONE = "none";
74     public static final String PROTOKD_TRANSITMODE_WALK = "walk";
75
76     static{
77
78         ALTERNATIVES = new Alternative[3];
79         ALTERNATIVES[0] = new Alternative(0.75, 1, 0.15);
80         ALTERNATIVES[1] = new Alternative(1, 0.75, 0.15);
81         ALTERNATIVES[2] = new Alternative(0.75, 1, 0.45);
82
83         REGIONINFOS = new HashMap<String, ProtoRegion>();
84
85         REGIONINFOS.put(PROTO_REGION_BANDUNG, new ProtoRegion(-6.91474, 107.60981, 17000, 12, ",,(bandung
86             | bdg)$", "Bandung"));
87         REGIONINFOS.put(PROTO_REGION_JAKARTA, new ProtoRegion(-6.21154, 106.84517, 15000, 11, ",,(jakarta
88             | jkt)$", "Jakarta"));
89         REGIONINFOS.put(PROTO_REGION_SURABAYA, new ProtoRegion(-7.27421, 112.71908, 15000, 12, ",,(surabaya|sby)$", "Surabaya"));
90         REGIONINFOS.put(PROTO_REGION_MALANG, new ProtoRegion(-7.9812985, 112.6319264, 15000, 13, ",,(malang|mlg)$", "Malang"));
91     }

```

Listing C.4: Utils.java

```

1 package models.helpers;
2
3 import com.fasterxml.jackson.databind.node.ObjectNode;
4 import play.db.DB;
5 import play.libs.Json;
6 import play.Logger;
7
8 import java.sql.Connection;
9 import java.sql.ResultSet;
10 import java.util.regex.Matcher;
11 import java.util.regex.Pattern;
12
13 public final class Utils {
14
15     public static void log_statistic(String verifier, String type, String additional_info) {
16         Connection connection = null;
17         try {
18             connection = DB.getConnection();
19             java.sql.PreparedStatement stmt = connection.prepareStatement(
20                 "INSERT INTO statistics_(verifier, _type, _additionalInfo) VALUES (?, ?, ?)");
21             stmt.setString(1, verifier);
22             stmt.setString(2, type);
23             stmt.setString(3, additional_info);
24             stmt.executeUpdate();
25
26             connection.close();
27         } catch (Exception e) {
28     }
29 }
30
31
32     public static ObjectNode die_nice(String message) {
33         ObjectNode obj = Json.newObject();
34         obj.put(Constants.PROTO_STATUS, Constants.PROTO_STATUS_ERROR);
35         obj.put(Constants.PROTO_MESSAGE, message);
36         return obj;
37     }
38
39     public static int indexPregMatch(String regex, String content) {
40         Pattern pattern = Pattern.compile(regex, Pattern.CASE_INSENSITIVE);
41         Matcher matcher = pattern.matcher(content);
42         int index = -1;
43         while (matcher.find()) {
44             index = matcher.start();
45         }
46         return index;
47     }
48
49
50     public static ObjectNode well_done(String message) {
51         ObjectNode obj = Json.newObject();
52         obj.put(Constants.PROTO_STATUS, Constants.PROTO_STATUS_OK);
53         if (message != null) {
54             obj.put(Constants.PROTO_MESSAGE, message);
55         }
56         return obj;
57     }
58
59     public static String get_from_cache(String type, String key) {
60
61         Connection connection = DB.getConnection();
62         StringBuilder output = new StringBuilder();
63         try {
64             java.sql.PreparedStatement stmt = connection.prepareStatement(
65                 "SELECT cacheValue FROM cache WHERE type=? AND cacheKey=?");
66             stmt.setString(1, type);
67             stmt.setString(2, key);
68
69             ResultSet result = stmt.executeQuery();
70             while (result.next()) {
71                 output.append(result.getString(1));
72             }
73
74
75             connection.close();
76         } catch (Exception e) {
77             log_error("Unable to retrieve from cache: " + e.getMessage());
78         }
79
80         return output.toString();
81     }
82
83     public static void put_to_cache(String type, String key, String value) {
84         Connection connection = DB.getConnection();
85         try {
86             java.sql.PreparedStatement stmt = connection.prepareStatement(
87                 "INSERT INTO cache(type, cacheKey, cacheValue) VALUES (?, ?, ?)");
88             stmt.setString(1, type);
89             stmt.setString(2, key);
90
91             stmt.setString(3, String.valueOf(value));
92             stmt.executeUpdate();
93             connection.close();
94         } catch (Exception e) {
95             log_error("Warning: Can't store cache "+e.getMessage());
96         }
97
98     }

```

```
99    }
100
101
102    public static void log_error(String message){
103        Logger.error(message);
104    }
105
106    public static String validateLocale(String locale){
107        if(locale.equals(Constants.PROTO_LOCALE_INDONESIA)){
108            return locale;
109        } else{
110            return Constants.PROTO_LOCALE_ENGLISH;
111        }
112    }
113
114    public static String validateRegion(String region){
115        if(Constants.REGIONINFOS.get(region) != null){
116            return region;
117        } else{
118            return Constants.PROTO_REGION_BANDUNG;
119        }
120    }
121
122
123
124
125
126
127 }
```

LAMPIRAN D

KODE PROGRAM *VIEW*

Listing D.1: index.scala.html

```
1 @import play.i18n.  
2 @(locale:String = "id",regioninfos:Map[String , models.ProtoRegion] , message:String , region:String)  
3 <!DOCTYPE html>  
4  
5 <html>  
6     <head>  
7         <title>KIRI</title>  
8         <link rel="shortcut_icon" type="image/png" href="@routes.Assets.versioned("images/favicon.ico")">  
9         <link rel="stylesheet" href="@routes.Assets.versioned("foundation/css/foundation.min.css")" type="text/css">  
10        <link rel="stylesheet" href="@routes.Assets.versioned("css/styleIndex.css")" type="text/css">  
11  
12        <script src="@routes.Assets.versioned("foundation/js/vendor/modernizr.js")"></script>  
13  
14        <link rel="stylesheet" href="http://openlayers.org/en/v3.11.1/css/ol.css" type="text/css">  
15        <script src="http://openlayers.org/en/v3.11.1/build/ol.js" type="text/javascript"></script>  
16  
17    </head>  
18    <body>  
19        <div class="row">  
20            <div id="controlpanel" class="large-3 large-push-9 columns">  
21                <div class="row_center">  
22                      
23                </div>  
24            </div>  
25            <div class="row">  
26                <div class="small-5 columns">  
27                    <select class="fullwidth" id="regionselect">  
28                        @for(regioninfo <- regioninfos) {  
29                            <option value=@regioninfo._1  
30                                @if(regioninfo._1 == region){  
31                                    selected  
32                                }  
33                            >@regioninfo._2.getName</option>  
34                        }  
35                    </select>  
36                </div>  
37                <div class="small-7 columns">  
38                    <select class="fullwidth" id="localeselect">  
39                        <option value="en">English</option>  
40                        <option value="id">  
41                            @if(locale == "id"){  
42                                selected  
43                            }  
44                            >Bahasa Indonesia</option>  
45                    </select>  
46                </div>  
47            </div>  
48            <div class="row">  
49                <div class="small-2 columns">  
50                    <label for="startInput" class="inline">@Messages.get("from")</label>  
51                </div>  
52                <div class="small-10 columns">  
53                    <input type="text" id="startInput" value=""  
54                        placeholder="@Messages.get("ph_from")">  
55                </div>  
56            </div>  
57            <div class="row">  
58                <div class="large-12 columns">  
59                    <select id="startSelect" class="hidden"></select>  
60                </div>  
61            </div>  
62            <div class="row">  
63                <div class="small-2 columns">  
64                    <label for="finishInput" class="inline">@Messages.get("to")</label>  
65                </div>  
66                <div class="small-10 columns">  
67                    <input type="text" id="finishInput" value="">  
68                </div>
```

```

73         placeholder="@Messages.get("ph_to")">
74     </div>
75 </div>
76 <div class="row">
77     <div class="large-12-columns">
78         <select id="finishSelect" class="hidden"></select>
79     </div>
80 </div>
81 <div class="row">
82     <div class="small-6-columns">
83         <a href="#" class="small-button-expand" id="findbutton"><strong>@Messages.get("find")</strong></a>
84     </div>
85     <div class="small-3-columns">
86         <a href="#" class="small-button-secondary-expand" id="swapbutton">
87             
88         </a>
89     </div>
90     <div class="small-3-columns">
91         <a href="#" class="small-button-secondary-expand" id="resetbutton">
92             
93         </a>
94     </div>
95 </div>
96 <div class="row">
97     <div class="large-12-columns" id="routingresults">
98         <div id="results-section-container"></div>
99     </div>
100 </div>
101 <div class="row">
102     <div class="large-12-columns">
103         <footer>
104             <a href="http://static.kiri.travel/$locale-apps">@Messages.get("apps")</a> |
105             <a href="http://classic.kiri.travel">@Messages.get("old")</a><br/>
106             <a href="http://static.kiri.travel/$locale-legal">@Messages.get("legal")</a> |
107             <a href="http://static.kiri.travel/$locale-feedback">@Messages.get("feedback")</a> |
108             <a href="http://static.kiri.travel/$locale-about">@Messages.get("about")</a>
109         </footer>
110         &nbsp;
111     </div>
112 </div>
113 <div class="row">
114     </div>
115 </div>
116
117 <div id="map" class="large-9-large-pull-3-columns"></div>
118 </div>
119
120
121 <script src="@routes.Assets.versioned("foundation/js/vendor/jquery.js")"></script>
122 <script src="@routes.Assets.versioned("foundation/js/vendor/fastclick.js")"></script>
123 <script src="@routes.Assets.versioned("foundation/js/foundation.min.js")"></script>
124 <script src="@routes.Assets.versioned("foundation/js/foundation/alert.js")"></script>
125 <script>
126     @Html(message)
127 </script>
128
129 <script src="@routes.Assets.versioned("js/newprotocol.js")"></script>
130 <script src="@routes.Assets.versioned("js/index.js")"></script>
131 <script>
132     (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
133         (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
134         m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
135     })(window,document,'script','//www.google-analytics.com/analytics.js','ga');
136
137     ga('create', 'UA-36656575-2', 'kiri.travel');
138     ga('require', 'displayfeatures');
139     ga('send', 'pageview');
140
141 </script>
142
143 </body>
144 </html>
```