

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

STEVEN SUTANA—2012730046

1 Data Skripsi

Pembimbing utama/tunggal: **Pascal Alfadian**

Pembimbing pendamping: -

Kode Topik : **PAS3902**

Topik ini sudah dikerjakan selama : **1** semester

Pengambilan pertama kali topik ini pada : Semester **39** - **Ganjil 15/16**

Pengambilan pertama kali topik ini di kuliah : **Skripsi 1**

Tipe Laporan : **B** - Dokumen untuk reviewer pada presentasi dan **review Skripsi 1**

2 Detail Perkembangan Penggerjaan Skripsi

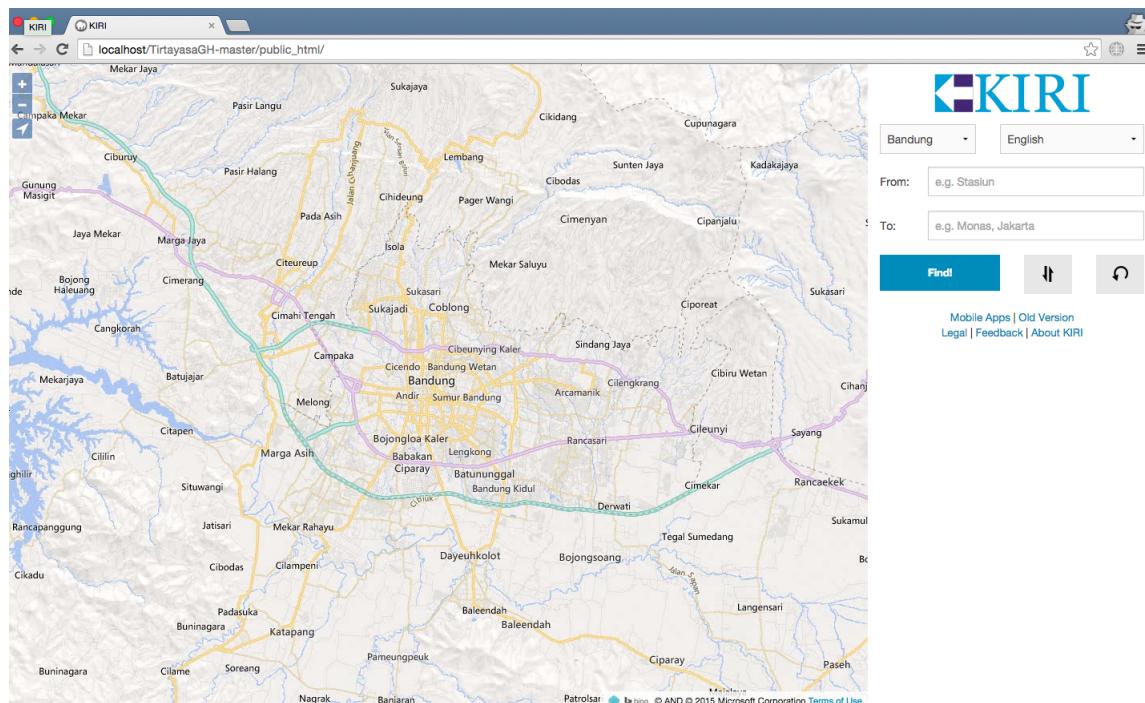
Detail bagian pekerjaan skripsi sesuai dengan rencana kerja/laporan perkembangan terakhir :

1. Memahami dan melakukan analisa kode KIRI yang sudah ada.

status : Ada sejak rencana kerja skripsi.

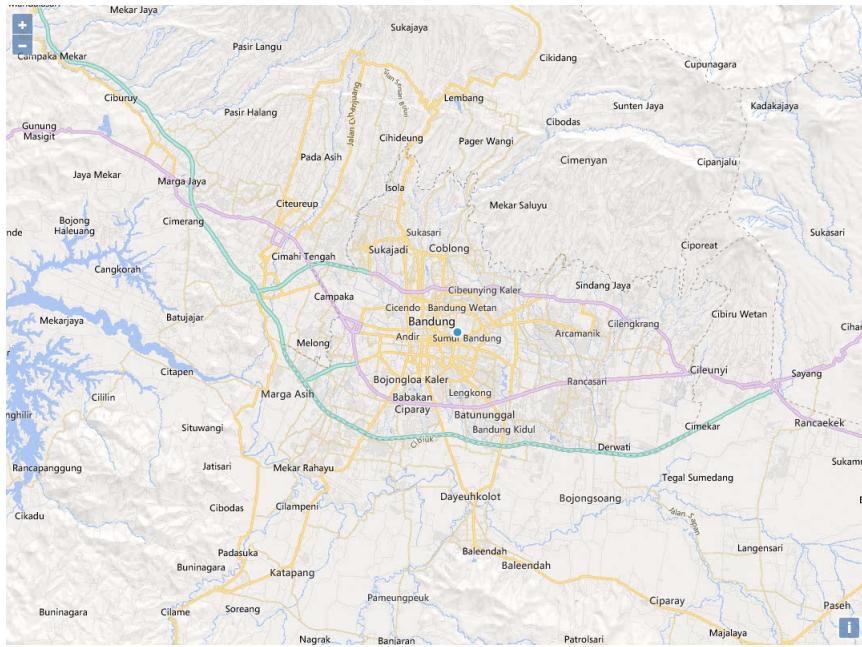
hasil :

Pada halaman utama KIRI (dapat dilihat pada gambar 1), terdapat beberapa bagian yaitu:



Gambar 1: Halaman Utama KIRI

Peta Peta pada KIRI (Gambar 2) berfungsi untuk menampilkan peta pada pengguna berdasarkan kota pengguna dan juga menentukan tempat asal dan tujuan pengguna dengan melakukan klik pada peta.



Gambar 2: Peta pada KIRI

KIRI menggunakan OpenLayers yang berbasis JavaScript untuk memuat peta pada halaman *web*. Pertama melakukan deklarasi peta yang digunakan menggunakan BingMaps. Penggunaan BingMaps membutuhkan dua *parameter*, yaitu *key* yang merupakan kunci untuk menggunakan BingMaps dan *imagerySet* yang merupakan tipe peta pada BingMaps. dan tipe peta pada BingMaps tersebut seperti pada kode listing 1.

Listing 1: Deklarasi peta BingMaps

```
var mapLayer = new ol.layer.Tile(
{
    source : new ol.source.BingMaps(
    {
        key : 'AuV7xD6_UMiQ5BLoZr0xkpjLpzWqMT55772Q8XtLIQeuDebHPKiNXSlZXxE',
        imagerySet : 'Road'
    })
});
```

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 2.

Listing 2: Objek *ol.source.Vector*

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

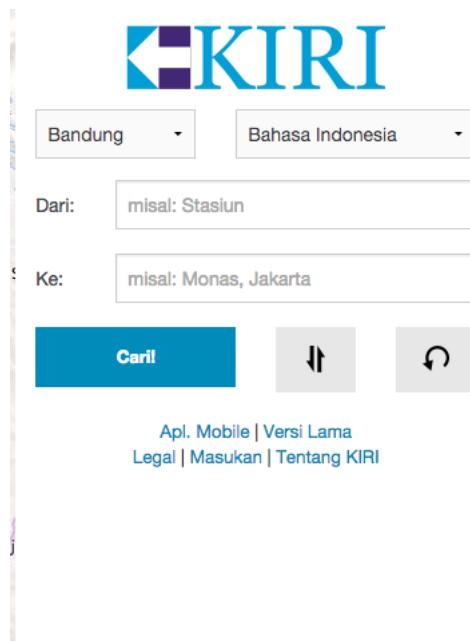
Listing 3: Instansiasi peta

```
var map = new ol.Map(
{
    ...
});
```

```
layers : [ mapLayer, new ol.layer.Vector({source: inputVectorSource}),
target : 'map'
});
```

Form Samping

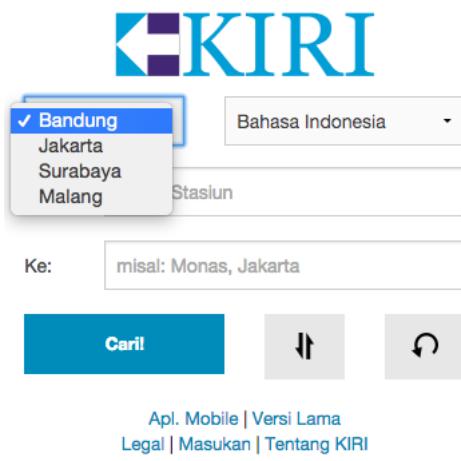
Form yang terdapat pada halaman utama KIRI (Gambar 3) terdiri dari:



Gambar 3: Form pada KIRI

Dropdown Menu Kota

Dropdown yang berfungsi untuk memilih kota yang akan ditampilkan pada peta (Gambar 4).



Gambar 4: Dropdown Menu Kota pada KIRI

Melakukan deklarasi variabel regioninfos sebagai *associated array* pada file constants.php. Setiap kota direpresentasikan sebagai proto_region_KOTA dimana KOTA adalah kota yang ada pada KIRI. Setiap proto_region_KOTA terdapat *lat* sebagai garis lintang, *lon* sebagai garis bujur dan *zoom* sebagai tingkat *zoom* untuk memperbarui peta. Proto_region_KOTA juga terdapat *name* untuk

menampilkan pilihan kota dan *searchplace_regex* untuk pencarian rute pada pilihan kota seperti pada kode listing 4.

Listing 4: Deklarasi variabel regioninfos

```
..
/** Different parameters for different regions. */
$regioninfos = array(
    $proto_region_bandung => array(
        'lat' => -6.91474,
        'lon' => 107.60981,
        'radius' => 17000,
        'zoom' => 12,
        'searchplace_regex' => ', *(bandung|bdg)$',
        'name' => 'Bandung'
    ),
    $proto_region_jakarta => array(
        'lat' => -6.21154,
        'lon' => 106.84517,
        'radius' => 15000,
        'zoom' => 11,
        'searchplace_regex' => ', *(jakarta|jkt)$',
        'name' => 'Jakarta'
    ),
    $proto_region_surabaya => array(
        'lat' => -7.27421,
        'lon' => 112.71908,
        'radius' => 15000,
        'zoom' => 12,
        'searchplace_regex' => ', *(surabaya|sby)$',
        'name' => 'Surabaya'
    ),
    $proto_region_malang => array(
        'lat' => -7.9812985,
        'lon' => 112.6319264,
        'radius' => 15000,
        'zoom' => 13,
        'searchplace_regex' => ', *(malang|mlg)$',
        'name' => 'Malang'
    )
);
..
```

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 5

Listing 5: Menampilkan pilihan kota kepada pengguna

..

```

<select class="fullwidth" id="regionselect">
    <?php
        foreach ($regioninfos as $key => $value) {
            print "<option value=\"$key\"";
            if ($key == $region) {
                print " selected";
            }
            print ">" . $value[ 'name' ] . "</option>\n";
        }
    ?>
</select>
..

```

Untuk memperbarui peta, KIRI menggunakan fungsi JavaScript dengan menerima dua parameter, yaitu newRegion dan updateCookie. Pertama membuat *cookie* dengan kunci region, lalu membuat variabel *point* dengan mengubah String menjadi LonLat dari titik tengah peta yang dituju. Untuk memperbarui peta dengan mengatur titik tengah pada peta yaitu memanggil *method setCenter* yang menerima parameter ol.proj.transform yang berisi garis lintang dan bujur serta kode dari Sistem dan Transformasi Koordinat. Setelah itu, mengatur tingkat *zoom* dengan memanggil *method setZoom* dengan parameter berupa tingkat *zoom* dari peta yang dituju. Kode dapat dilihat pada kode listing 6

Listing 6: Fungsi JavaScript untuk memperbarui peta

```

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

```

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 7.

Listing 7: Fungsi JavaScript untuk mengubah String menjadi *array* Float

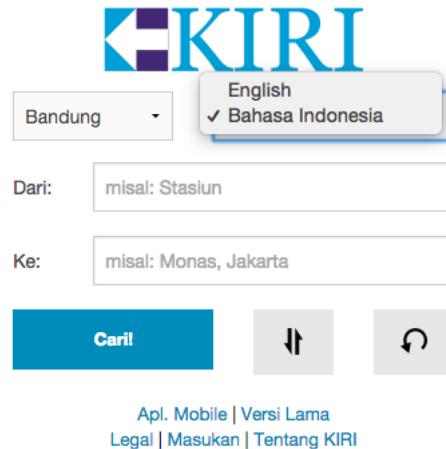
```

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

```

Dropdown Menu Bahasa

Dropdown yang berfungsi untuk memilih bahasa yang akan digunakan pada KIRI (Gambar 5).



Gambar 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 8

Listing 8: Menampilkan pilihan bahasa kepada pengguna

```
...
<select class="fullwidth" id="localeselect">
    <option value="en">English</option>
    <option value="id"
        <?php if ($locale == $proto_locale_indonesia) print " selected"; ?>
        Indonesia</option>
</select>
...
```

Ketika memilih *dropdown* bahasa, memanggil *method* JavaScript dengan *parameter* berupa fungsi yang berisi menambahkan URL dengan *query locale=id* atau *locale=en*. Kode dapat dilihat pada kode listing 9.

Listing 9: Fungsi JavaScript untuk Internationalization

```
...
// Event handlers
var localeSelect = $('#localeselect');
localeSelect.change(function() {
    // IE fix: when window.location.origin is not available
    if (!window.location.origin) {
        window.location.origin = window.location.protocol + "//" + window.lo...
```

Textfield

Textfield pada KIRI menggunakan PHP agar dapat dilakukan proses Internationalization, seperti pada kode listing 10 untuk *textfield* tempat asal dan kode listing 11 untuk *textfield* tempat tujuan. Textfield pada KIRI dapat menerima dua masukan pengguna, yaitu:

Listing 10: Menampilkan *textfield* tempat awal kepada pengguna

```
..  
<div class="small-2 columns">  
    <label for="startInput" class="inline"><?php print $index_from; ?></label>  
</div>  
<div class="small-10 columns">  
    <input type="text" id="startInput" value=""  
        placeholder="<?php print $index_placeholder_start; ?>">  
</div>  
..
```

Listing 11: Menampilkan *textfield* tempat tujuan kepada pengguna

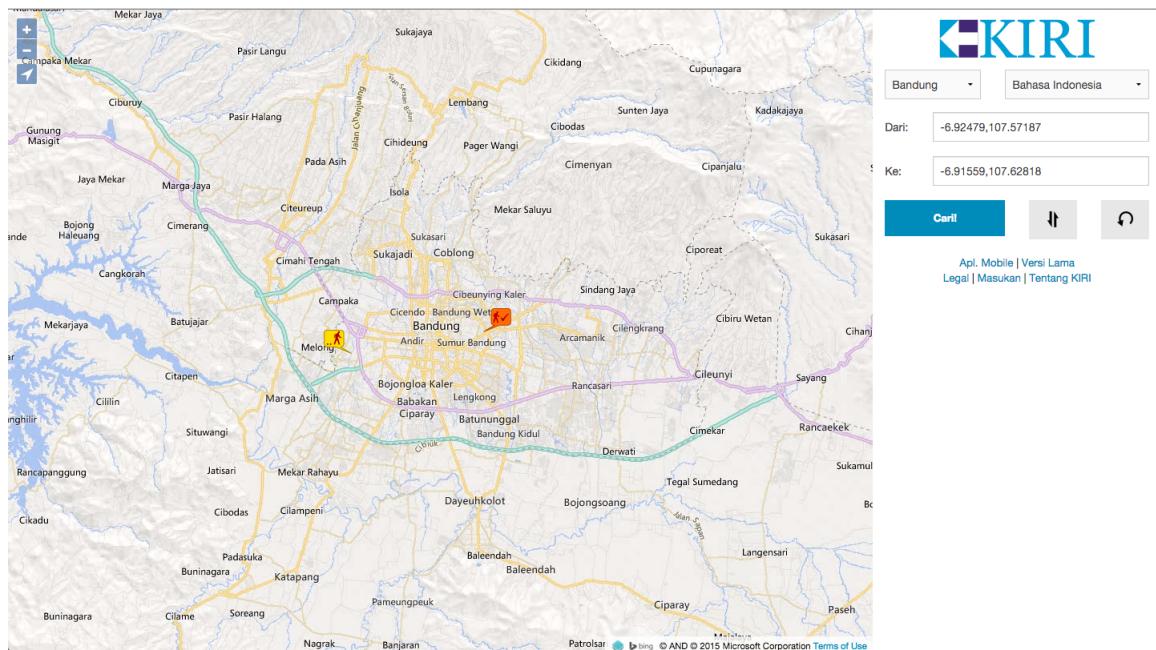
```
..  
<div class="small-2 columns">  
    <label for="finishInput" class="inline"><?php print $index_to; ?></label>  
</div>  
<div class="small-10 columns">  
    <input type="text" id="finishInput" value=""  
        placeholder="<?php print $index_placeholder_finish; ?>">  
</div>  
..
```

- (a) **Textfield dengan Masukan Nama Tempat**, pengguna dapat memasukkan nama tempat asal dan tujuan (Gambar 6)

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

Gambar 6: Input User(Nama Tempat)

- (b) **Textfield dengan Masukan Klik Peta**, pengguna memasukkan koordinat tempat asal dan tujuan dengan klik pada peta. Dengan melakukan klik pada peta, *textfield* tempat asal dan tujuan akan secara otomatis terisi oleh koordinat masing-masing tempat (Gambar 7).



Gambar 7: Input User(Klik pada peta)

Agar peta dapat diklik, maka memanggil method `on` dengan parameter ‘click’ dan fungsi yang akan diimplementasikan ketika melakukan klik pada peta. Isi fungsi tersebut adalah pertama melakukan pengecekan apabila textfield tempat asal atau tempat tujuan kosong, maka membuat geometry yang merupakan objek `ol.geom.Point` dengan parameter koordinat pada peta yang diklik oleh pengguna, lalu membuat marker dengan gambar `start.png` bila textfield tempat asal kosong atau `finish.png` bila textfield tempat tujuan kosong. Setelah itu, menambahkan fitur marker ke `inputVectorSource` yang akan ditampilkan pada peta dan menulis koordinat pada textfield tempat asal atau tempat tujuan. Kode dapat dilihat pada kode listing 12.

Listing 12: Membuat *event* klik pada peta

```
...
// Map click event
map.on('click', function(event) {
    if ($('#startInput').val() === '') {
        markers['start'] = new ol.Feature({
            geometry: new ol.geom.Point(event.coordinate)
        })
        markers['start'].setStyle(new ol.style.Style({
            image: new ol.style.Icon({
                src: 'images/start.png',
                anchor: [1.0, 1.0]
            })
        }));
        inputVectorSource.addFeature(markers['start']);
        $('#startInput').val(latLngToString(ol.proj.transform(event.coor
    } else if ($('#finishInput').val() === '') {
        markers['finish'] = new ol.Feature({
            geometry: new ol.geom.Point(event.coordinate)
        })
    }
});
```

```

        markers[ 'finish' ].setStyle(new ol.style.Style({
            image: new ol.style.Icon({
                src: 'images/finish.png',
                anchor: [ 0.0, 1.0 ]
            })
        }));
        inputVectorSource.addFeature(markers[ 'finish' ]);
        $('#finishInput').val(latLngToString(ol.proj.transform(event.coord
    }
});

..

```

Tombol Swap

Pengguna dapat menukar isi dari *textfield* tempat asal dan tujuan. Pertama kali yang dilakukan adalah mencari pada dokumen dengan *id* swapbutton dan memanggil *method click* dengan *parameter* fungsi swapInput seperti pada kode listing 13. Fungsi swapInput berisi melakukan pencarian pada dokumen dengan id startInput dan finishInput. Melakukan penampungan sementara dengan mengambil isi dari *textfield* tempat asal, lalu mengubah isi dari *textfield* tempat asal dengan tujuan dan mengubah isi dari *textfield* tempat tujuan dengan isi dari penampungan sementara. Setelah itu, jika kedua *textfield* ada isinya, melakukan pencarian rute. Kode dapat dilihat pada kode listing 14.

Listing 13: *Method* untuk memanggil fungsi JavaScript ketika tombol *swap* ditekan

```

...
$('#swapbutton').click(swapInput);
...
```

Listing 14: Fungsi JavaScript untuk menukar isi *textfield* tempat asal dan tujuan

```

/**
 * Swap the inputs
 */
function swapInput() {
    var startInput = $('#startInput');
    var finishInput = $('#finishInput');
    var temp = startInput.val();
    startInput.val(finishInput.val());
    finishInput.val(temp);
    coordinates[ 'start' ] = null;
    coordinates[ 'finish' ] = null;
    if (startInput.val() != '' && finishInput.val() != '') {
        findRouteClicked();
    }
}
```

Tombol Reset

Pengguna dapat melakukan pemilihan tempat dari awal dan mengulang tampilan peta. Pertama kali yang dilakukan adalah mencari pada dokumen dengan *id* resetbutton dan memanggil *method click* dengan *parameter* fungsi resetScreen seperti pada kode listing 15.

Listing 15: *Method* untuk memanggil fungsi JavaScript ketika tombol *reset* ditekan

```
..  
$( '#resetbutton' ). click( resetScreen );  
..
```

Fungsi `resetScreen` berisi berbagai fungsi seperti pada kode listing 16, yaitu:

Listing 16: Fungsi JavaScript `resetScreen`

```
function resetScreen() {  
    clearRoutingResultsOnTable();  
    clearRoutingResultsOnMap();  
    clearAlerts();  
    clearStartFinishMarker();  
    $.each([ 'start' , 'finish' ] , function(sfIndex , sfValue) {  
        var placeInput = $( '#' + sfValue + 'Input' );  
        placeInput . val( '' );  
        placeInput . prop( 'disabled' , false );  
        $( '#' + sfValue + 'Select' ) . addClass( 'hidden' );  
    });  
}
```

(a) **Fungsi `clearRoutingResultsOnMap`**

Fungsi untuk menghapus pada `resultVectorSource` yang merupakan berbagai *marker* pada peta sebagai hasil pencarian rute dan memperbarui peta. Kode dapat dilihat pada kode listing 17.

Listing 17: Fungsi JavaScript untuk menghapus hasil pencarian rute pada peta

```
function clearRoutingResultsOnMap() {  
    resultVectorSource . clear();  
    updateRegion( region , false );  
}
```

(b) **Fungsi `clearRoutingResultsOnTable`**

Fungsi untuk menghapus tampilan tabel sebagai hasil pencarian rute yang akan ditampilkan pada pengguna. Kode dapat dilihat pada kode listing 18.

Listing 18: Fungsi JavaScript untuk menghapus tampilan tabel

```
function clearRoutingResultsOnTable() {  
    $( '.tabs' ) . remove();  
    $( '.tabs-content' ) . remove();  
}
```

(c) **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 19.

Listing 19: Fungsi JavaScript untuk menghilangkan *alerts*

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

(d) Fungsi clearRoutingResultsOnMap

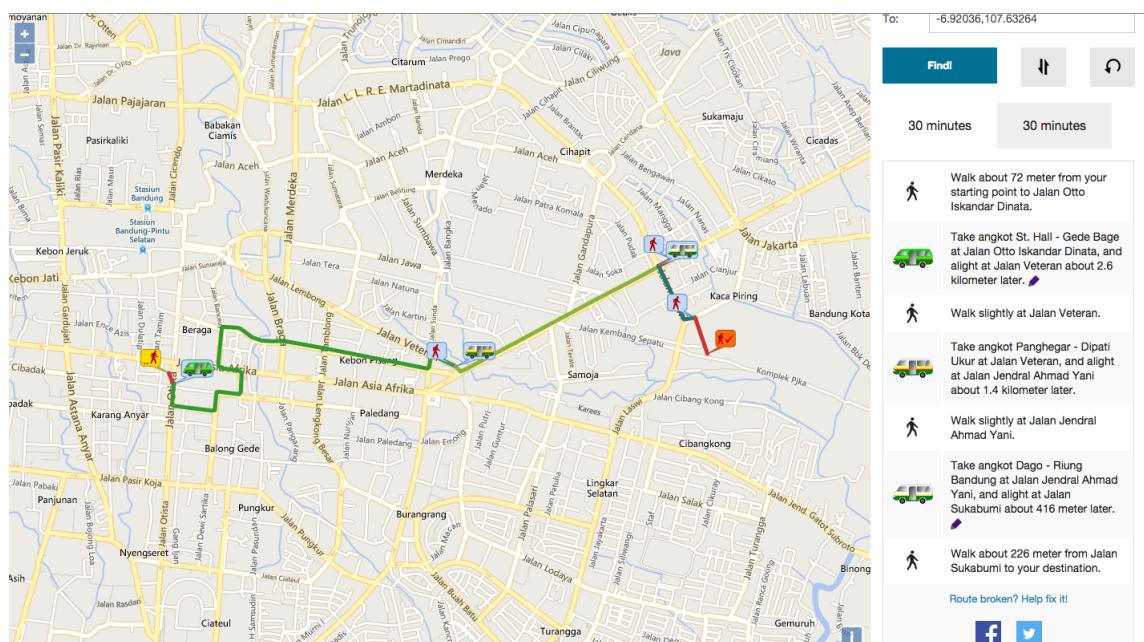
Fungsi untuk menghapus *markers* tempat awal dan tujuan, lalu menghapus fitur pada input *VectorSource*. Kode dapat dilihat pada kode listing 20.

Listing 20: Fungsi JavaScript untuk menghilangkan *alerts*

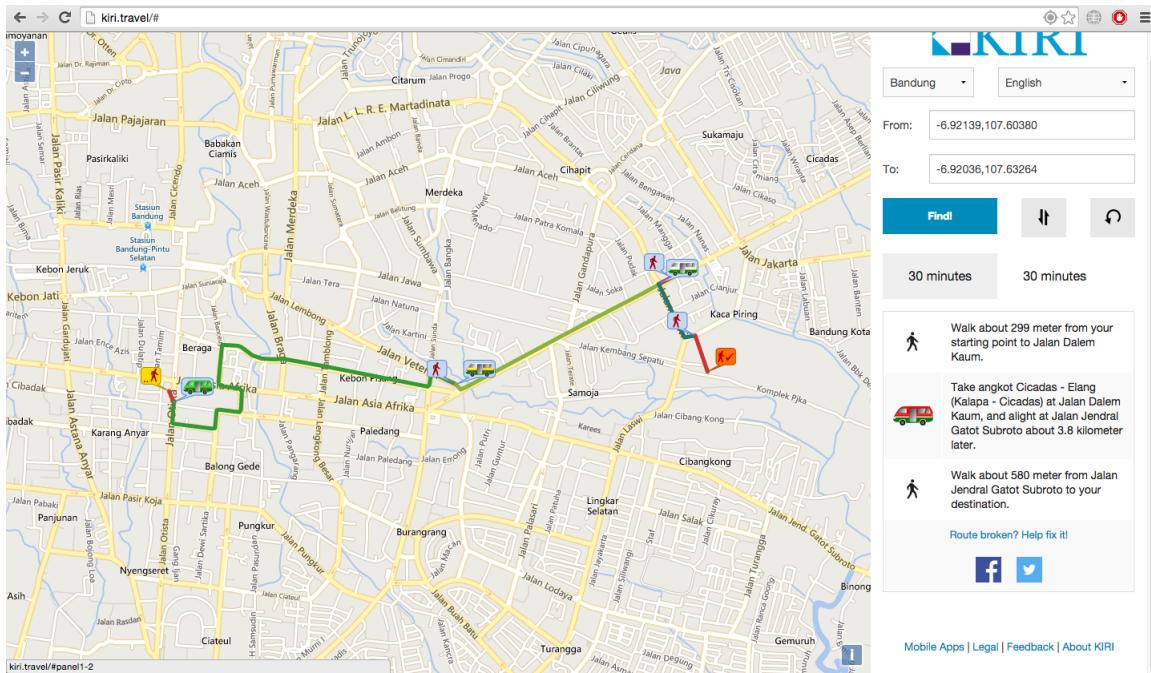
```
function clearStartFinishMarker() {
    if (markers[ 'start' ] != null) {
        markers[ 'start' ] = null;
    }
    if (markers[ 'finish' ] != null) {
        markers[ 'finish' ] = null;
    }
    inputVectorSource . clear ();
}
```

Tombol Find

Pengguna dapat mencari rute untuk sampai ke tujuan (Gambar 8). Pengguna dapat memilih rute alternatif yang sudah disediakan KIRI jika ada (Gambar 9).



Gambar 8: Contoh Pencarian Rute pada KIRI



Gambar 9: Contoh Rute Alternatif pada KIRI

Hal yang pertama kali dilakukan adalah melakukan validasi jika salah satu textfield kosong, maka akan langsung membatalkan proses dan memberi alert kepada pengguna. Jika tidak kosong, maka akan memunculkan alert ‘mohon menunggu’. Setelah itu melakukan pengecekan apakah isi dari textfield merupakan format yang benar untuk garis lintang dan bujur. Jika formatnya benar, maka dimasukkan ke dalam array coordinates dan melakukan penambahan completedLatLon yang berfungsi untuk mengetahui apakah kedua textfield sudah benar formatnya untuk dilakukan pencarian rute. Jika formatnya tidak benar, maka melakukan pengecekan array coordinates kosong atau tidak. Jika kosong, maka melakukan pencarian pilihan tempat yang akan menjadi tempat sugesti yang diberikan kepada pengguna. Jika tidak kosong, maka memanggil fungsi checkCoordinatesThenRoute dengan parameter berupa coordinates. Terakhir, melakukan pengecekan completedLatLon jika isinya sama dengan dua, maka memanggil fungsi checkCoordinatesThenRoute. Kode dapat dilihat pada kode listing 21.

Listing 21: Fungsi JavaScript untuk ketika tombol *find* ditekan

```
/*
 * A function that will be called when find route button is clicked
 * (or triggered by another means)
 */
function findRouteClicked() {
    // Validate
    var cancel = false;
    $.each(['start', 'finish'], function(sfIndex, sfValue) {
        if ($('#' + sfValue + 'Input').val() === '') {
            cancel = true;
            return;
        }
    });
    if (cancel) {
        showAlert(messageFillBoth, 'alert');
        return;
    }
}
```

```

}

clearAlerts();
clearRoutingResultsOnTable();
showAlert(messagePleaseWait, 'secondary');

var completedLatLon = 0;
$.each(['start', 'finish'], function(sfIndex, sfValue) {
    var placeInput = $('#' + sfValue + 'Input');
    var placeSelect = $('#' + sfValue + 'Select');
    if (isLatLng(placeInput.val())) {
        coordinates[sfValue] = placeInput.val();
        completedLatLon++;
    } else {
        if (coordinates[sfValue] == null) {
            // Coordinates not yet ready, we do a search place
            protocol.searchPlace(
                placeInput.val(),
                region,
                function(result) {
                    placeSelect.empty();
                    placeSelect.addClass('hidden');
                    if (result.status != 'error')
                        if (result.searchresults)
                            $.each(result.searchresults, function(index, place) {
                                placeSelect.append($('').attr('value', place.lat + ',' + place.lng).text(place.name));
                            });
                    coordinates[sfValue] = result.lat + ',' + result.lng;
                    checkCoordinatesThenRoute(coordinates);
                }
            );
        } else {
            clearSecondaryAlerts();
            clearRoutingResultsOnMap();
            showAlert(messageConn);
        }
    }
    // Coordinates are already available, skip searching
    checkCoordinatesThenRoute(coordinates);
});

```

```
        }
    }
});  
if (completedLatLon == 2) {  
    checkCoordinatesThenRoute(coordinates);  
}  
}  
}
```

Fungsi `checkCoordinatesThenRoute` melakukan pengecekan jika array `coordinates` tempat awal dan tempat tujuan tidak kosong maka melakukan `protocol.findRoute`. Jika hasil `results` sama dengan ‘ok’, maka menunjukkan rute pencarian. Jika hasil `result` bukan ‘ok’, maka akan menampilkan alert ‘gangguan koneksi’. Kode dapat dilihat pada kode listing 22.

Listing 22: Fungsi JavaScript checkCoordinatesThenRoute

```
/**  
 * Check if coordinates are complete. If yes , then start routing.  
 * @param coordinates the coordinates to check.  
 */  
function checkCoordinatesThenRoute(coordinates) {  
    if (coordinates[ 'start '] != null && coordinates[ 'finish '] != null) {  
        protocol.findRoute(  
            coordinates[ 'start '],  
            coordinates[ 'finish '],  
            locale,  
            function(results) {  
                if (results.status === 'ok') {  
                    showRoutingResults(results);  
                } else {  
                    clearSecondaryAlerts();  
                    showAlert(messageConnectionError , 'a  
                }  
            }));  
    }  
}
```

Internationalization

Penggunaan Internationalization (i18n) pada PHP dengan cara deklarasi semua variabel yang akan digunakan pada proses i18n terlebih dahulu, misalnya buat file dengan nama tirtayasa_en.php untuk Bahasa Inggris dan tirtayasa_id.php untuk Bahasa Indonesia. Pada setiap file tersebut, masukkan *script* PHP untuk menentukan teks yang keluar pada halaman *web* seperti pada kode listing 23 dan kode listing 24.

Listing 23: Script PHP untuk Bahasa Inggris

<?php

```
$index_about_kiri = "About KIRI";  
$index_apps = "Mobile Apps";  
$index_advanced_ = "Advanced . . .";  
$index_buvticket = "BUY TICKET";
```

```
$index_connectionerror = 'Connection problem';
?>
```

Listing 24: Script PHP untuk Bahasa Indonesia

```
<?php
    $index_about_kiri = "Tentang KIRI";
    $index_apps = "Apl. Mobile";
    $index_advanced_ = "Lanjut . . .";
    $index_buylticket = "BUY TICKET";
    $index_connectionerror = 'Gangguan koneksi';
?>
```

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 25.

Listing 25: Script PHP untuk Internationalization

```
...
<label for="startInput" class="inline"><?php print $index_from; ?></label>
<label for="finishInput" class="inline"><?php print $index_to; ?></label>
<a href="#" class="small button expand" id="findbutton"><strong><?php print $i
...

```

2. Melakukan studi literatur tentang metode yang berkaitan dengan kode PHP dan Java (Play Framework).

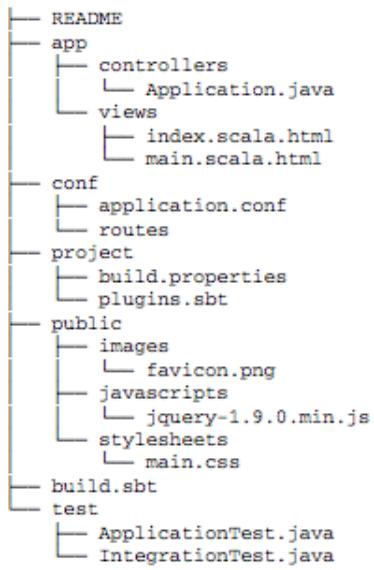
status : Ada sejak rencana kerja skripsi.

hasil :

Play Framework Play Framework [1] merupakan *framework* untuk aplikasi web dengan menggunakan bahasa Java dan Scala. Play Framework tidak sepenuhnya menggunakan bahasa Java, tetapi ada juga bahasa Scala. Terdapat bahasa Scala bukan berarti harus mempelajari bahasa Scala karena dalam Play 2 dilengkapi dengan Java API yang komplit, memberikan opsi untuk memilih bahasa pemrograman yang cocok. Play Framework mempunyai antarmuka yang sederhana, nyaman, fleksibel, dan kuat. Referensi yang digunakan membahas Play Framework 2.2, sedangkan versi Play Framework yang dipakai dalam penelitian ini adalah versi 2.4. Ada sedikit perbedaan sintaks yang akan dijelaskan pada bagian yang berbeda. Beberapa fitur utama yang membuat Play Framework produktif dan penggunaan yang nyaman:

- (a) Penggunaan Play Framework sederhana.
- (b) Konfigurasi skema URL aplikasi deklaratif.
- (c) Pemetaan type-safety.
- (d) Play Framework menyediakan contoh sintaks type-safety.
- (e) Arsitektur yang mencakup teknologi HTML5.
- (f) Kode langsung aktif berubah ketika memuat kembali halaman web.
- (g) Fitur *full-stack web-framework*, termasuk *persistence*, keamanan, dan *internationalization*. *Persistence* adalah ide yang menggunakan koneksi TCP yang sama untuk mengirim dan menerima beberapa HTTP *requests/responses* tanpa membuka TCP baru untuk setiap *requests/responses* dengan tujuan untuk meningkatkan kinerja HTTP.
- (h) Mendukung aplikasi *event-driven* dan dinamis.

Play Framework memiliki struktur yang dapat dilihat pada gambar 10.



Gambar 10: Struktur Play Framework

Direktori conf. Konfigurasi Play Framework terdapat pada direktori *conf*. Dalam direktori *conf*, terdapat file *application.conf* dan *routes*. File *application.conf* mengandung informasi data konfigurasi aplikasi, seperti *logging*, koneksi basis data, dan port berapa server berjalan. File *routes* menentukan *routes* aplikasi, yaitu pemetaan dari URL HTTP ke kode aplikasi. Setiap *routes* memiliki tiga bagian, yaitu HTTP *method*, URL *path*, dan *action method*. HTTP *method* merupakan metode yang dipakai dalam pengiriman HTTP. URL *path* adalah URL yang dipakai untuk mengakses halaman. *Action method* merupakan metode yang dipanggil ketika mengakses halaman pada URL *path*. Sebagai contoh dapat dilihat pada 11, HTTP *method* yang dipakai pada URL */list* adalah HTTP *method* GET dan akan memanggil *method* *list()* pada kelas *Products* di *controllers*.



Gambar 11: Contoh *Routes*

Direktori public Direktori *public* mengandung semua sumber daya yang disediakan langsung tanpa melalui proses terlebih dahulu. Direktori *public* biasanya mengandung file gambar, *stylesheets*, JavaScript, dan halaman statis HTML. Contoh direktori *public* dapat dilihat pada gambar 10.

Direktori app. Direktori *app* merupakan direktori utama pada aplikasi. Direktori *app* berisi kode aplikasi dan berbagai kebutuhan untuk menyusun aplikasi, seperti sumber file Java dan file *template*. Contoh direktori *app* pada saat pertama kali membuat aplikasi Play dapat dilihat pada gambar 10.

Dalam *controller*, terdapat file *Application.java* yang berisi kode Java untuk memuat halaman *view*. *Controller* adalah kelas untuk menerima HTTP *request* dan mengembalikan nilai dari HTTP *request* berupa *view*. Ada dua file *template*, yaitu *index.scala.html* dan *main.scala.html* yang berfungsi untuk menentukan halaman HTML yang akan dimuat. Semua konten yang dihasilkan di server dan dikirimkan ke klien seperti halaman HTML disebut *view*. *View* dapat menerima parameter yang didefinisikan pada

template view. Contoh view dengan menerima parameter berupa String dapat dilihat pada kode listing 26. *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 27, *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 dalam **static**, tetapi pada Play Framework 2.4 **static** dihilangkan.

Listing 26: Contoh View

```
@(title: String)

<!DOCTYPE html>

<html>
    <head>
        <title>@title</title>
        <link rel="stylesheet" media="screen" href="@routes.Assets.at("stylesheets/ma
        <link rel="shortcut_icon" type="image/png" href="@routes.Assets.at("images/fa
            <script src="@routes.Assets.at("javascripts/jquery-1.9.0.min.js")" typ
        </head>
        <body>
            @title
        </body>
    </html>
```

Listing 27: Contoh Controller

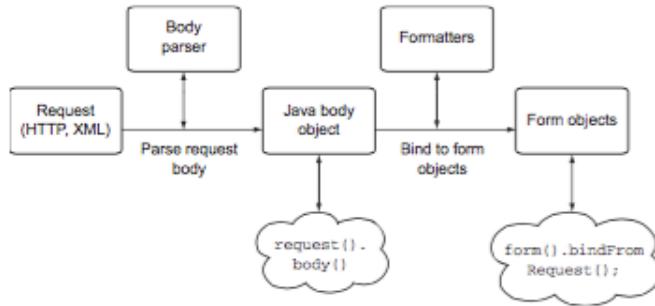
```
public Result index() {
    return ok(index.render("Hello_World"));
}
```

Beberapa alternatif hasil Results, yaitu:

- (a) **Method ok** Method yang berfungsi untuk mengembalikan Result dengan kode *HTTP Response* 200 atau sukses dalam mengirim **HTTP Request**.
- (b) **Method notFound** Method yang berfungsi untuk mengembalikan Result dengan kode *HTTP Response* 404 atau tidak ada halaman yang dituju.
- (c) **Method badRequest** Method yang berfungsi untuk mengembalikan Result dengan kode *HTTP Response* 400 atau adanya kesalahan masukan pengguna.
- (d) **Method internalServerError** Method yang berfungsi untuk mengembalikan Result dengan kode *HTTP Response* 500 atau adanya kesalahan pada server.
- (e) **Method status** Method yang berfungsi untuk mengembalikan Result dengan kode *HTTP Response* yang dapat ditentukan sendiri beserta pesannya.
- (f) **Method redirect** Method yang berfungsi untuk mengembalikan Result untuk mengalihkan halaman.

Body Parsers *Body parsers* bertugas untuk melakukan pemetaan *request body* menjadi objek. Setiap *action method* POST dan PUT mengandung *body*. Jumlah *body* dapat satu atau banyak, dan dapat berupa XML, JSON, data biner, atau dapat berupa apapun sesuai Content-Type pada *header request*. *Body parsers* akan menguraikan *body* menjadi objek Java. *Body parsers* mengubah *request* menjadi

objek yang dapat digunakan oleh komponen Play. Karena *body* JSON dan *body* XML berbeda pengurainya, Play menggunakan *body parsers* yang berbeda pula implementasinya. Berbeda Content-Type pada *header request*, *body parsers* spesifik dapat mengubah data yang masuk menjadi sesuatu yang dapat dimengerti oleh Play. Ilustrasi dapat dilihat pada gambar 12.



Gambar 12: Interaksi *Body Parsers* dengan *Request*

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

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

Untuk mendukung beberapa bahasa dalam Play, membuat kunci pesan yang dipetakan pesan sesungguhnya pada file pesan. Pesan ini dimuat dalam file messages.LANG dimana LANG merupakan bahasa yang dipakai. Messages.LANG disimpan dalam direktori conf. Sebagai contoh, terdapat dua bahasa yang dipakai pada aplikasi, yaitu Bahasa Inggris dan Bahasa Indonesia seperti pada kode listing 28 dan 29

Listing 28: Contoh messages.en untuk i18n

```

from = From:
ph_from = e.g. Stasiun
ph_to = e.g. Monas, Jakarta
find = Find!
to = To:

```

Listing 29: Contoh messages.id untuk i18n

```

from = Dari:
ph_from = misal: Stasiun

```

```
ph_to = misal: Monas, Jakarta
find = Cari!
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 30. Pada [1], konfigurasi bahasa dalam satu String. Tetapi pada Play Framework 2.4 konfigurasi bahasa dalam bentuk array of String.

Listing 30: Konfigurasi Bahasa i18n

```
...
# The application languages
# ~~~~
play.i18n.langs = [ "en", "id" ]
...
```

3. Merancang dan mengimplementasikan kode KIRI yang sudah ada menjadi Play Framework.

status : Ada sejak rencana kerja skripsi.

hasil :

Listing 31: Template *view* pada Play Framework

```
@import play.i18n._

<!DOCTYPE html>

<html>

    <head>
        <title>KIRI</title>
        <link rel="shortcut icon" type="image/png" href="@routes.Assets.versioned("img/favicon.png")"

        @*<link rel="stylesheet" href="@routes.Assets.versioned("css/app.css")"
        <link rel="stylesheet" href="@routes.Assets.versioned("css/bootstrap.css")"
        <link rel="stylesheet" href="@routes.Assets.versioned("css/font-awesome.css")"

        <script src="@routes.Assets.versioned("foundation/js/vendor.js")">
        <script src="http://openlayers.org/en/v3.11.1/build/ol.js">
        @*<script src="@routes.Assets.versioned("js/app.js")">
        @*<script src="http://openlayers.org/en/v3.10.1/build/ol.js">

    </head>
    <body>
        @Messages.get("hello")
        <div class="row">

            <div id="controlpanel" class="large-3 large-push-9">
                <div class="row center">
```

```


</div>

<div class="row">
  <div class="small-5 columns">
    <select class="fullwidth" id="regionSelect">
      <option value="bdg">Bandung</option>
      <option value="jkt">Jakarta</option>
      <option value="sby">Surabaya</option>
      <option value="mlg">Malang</option>
    </select>
  </div>
  <div class="small-7 columns">
    <select class="fullwidth" id="localeSelect">
      <option value="en">English</option>
      <option value="id">Bahasa Indonesia</option>
    </select>
  </div>
</div>

<div class="row">
  <div class="small-2 columns">
    <label for="startInput" class="inline-block">From:</label>
  </div>
  <div class="small-10 columns">
    <input type="text" id="startInput" value="123" placeholder="@Messages.get("ph_from")" />
  </div>
</div>

<div class="row">
  <div class="large-12 columns">
    <select id="startSelect" class="hidden">
      <option value="1">1</option>
      <option value="2">2</option>
      <option value="3">3</option>
    </select>
  </div>
  <div class="row">
    <div class="small-2 columns">
      <label for="finishInput" class="inline-block">To:</label>
    </div>
    <div class="small-10 columns">
      <input type="text" id="finishInput" value="456" placeholder="@Messages.get("ph_to")" />
    </div>
  </div>
</div>

<div class="row">
  <div class="large-12 columns">
    <select id="finishSelect" class="hidden">
      <option value="1">1</option>
      <option value="2">2</option>
      <option value="3">3</option>
    </select>
  </div>
</div>

```

```

<div class="row">
  <div class="small-6 columns">
    <a href="#" class="small button expand" data-bbox="541 111 998 151">
      </a>
  <div class="small-3 columns">
    <a href="#" class="small button second" data-bbox="578 178 998 218">
      
    </a>
  </div>
  <div class="small-3 columns">
    <a href="#" class="small button second" data-bbox="615 261 998 301">
      
    </a>
  </div>
</div>
<div class="row">
  <div class="large-12 columns" id="routinggrid" data-bbox="578 371 998 411">
    <div id="results-section-container"></div>
  </div>
</div>
<div class="row">
  <div class="large-12 columns">
    <footer>
      <a href="http://static.kiri.travel" data-bbox="658 498 998 538">
        <a href="http://classic.kiri.travel" data-bbox="658 521 998 561">
          <a href="http://static.kiri.travel" data-bbox="658 544 998 584">
            <a href="http://static.kiri.travel" data-bbox="658 567 998 607">
              <a href="http://static.kiri.travel" data-bbox="658 590 998 630">
                </a>
      </a>
    </footer>
    &nbsp;
  </div>
</div>
<div class="row">
  </div>
</div>

<div id="map" class="large-9 large-pull-3 columns" data-bbox="452 728 998 761">
</div>

<script src="@routes.Assets.versioned("foundation/js/vendor/modernizr.js")" type="text/javascript" data-bbox="452 798 998 861">
<script src="@routes.Assets.versioned("foundation/js/vendor/jquery.js")" type="text/javascript" data-bbox="452 844 998 884">
<script src="@routes.Assets.versioned("foundation/js/vendor/jquery.fitvids.js")" type="text/javascript" data-bbox="452 877 998 917">
<script src="@routes.Assets.versioned("foundation/js/vendor/jquery.fancybox.js")" type="text/javascript" data-bbox="452 910 998 950">
<script src="@routes.Assets.versioned("js/newprotocol.js")" type="text/javascript" data-bbox="452 884 998 924">
<script src="@routes.Assets.versioned("js/index.js")" type="text/javascript" data-bbox="452 917 998 957">

```

```
</body>
</html>
```

4. Melakukan pengujian dan eksperimen.

status : Ada sejak rencana kerja skripsi.

hasil :

5. Membuat dokumen skripsi.

status : Ada sejak rencana kerja skripsi.

hasil : Dokumen skripsi sudah ditulis sampai dengan bab 3.

Pustaka

- [1] N. Leroux and S. D. Kaper, *Play for Java*. Manning Publications Co., 2014.

3 Pencapaian Rencana Kerja

Persentase penyelesaian skripsi sampai dengan dokumen ini dibuat dapat dilihat pada tabel berikut :

1*	2*(%)	3*(%)	4*(%)	5*(%)	6*(%)
1	20	20			15
2	20	20			20
3	20		20	Mengimplementasikan kode KIRI menjadi Play Framework	5
4	20		20	Pengujian seluruh fitur KIRI pada Play Framework	
5	20	10	10		10
Total	100	50	50		50

Keterangan (*)

1 : Bagian pengerjaan Skripsi (nomor disesuaikan dengan detail pengerjaan di bagian 5)

2 : Persentase total

3 : Persentase yang akan diselesaikan di Skripsi 1

4 : Persentase yang akan diselesaikan di Skripsi 2

5 : Penjelasan singkat apa yang dilakukan di S1 (Skripsi 1) atau S2 (skripsi 2)

6 : Persentase yang sudah diselesaikan sampai saat ini

Bandung, 20/11/2015

Steven Sutana

Menyetujui,

Nama: Pascal Alfadian

Pembimbing Tunggal ‘