

SKRIPSI

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



STEVEN SUTANA

NPM: 2012730046

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

UNDERGRADUATE THESIS

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



STEVEN SUTANA

NPM: 2012730046

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

LEMBAR PENGESAHAN

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

STEVEN SUTANA

NPM: 2012730046

Bandung, «tanggal» «bulan» «tahun»

Menyetujui,

Pembimbing Utama

Pembimbing Pendamping

Pascal Alfadian, M.Com.
Ketua Tim Penguji

«pembimbing pendamping/2»
Anggota Tim Penguji

«penguji 1»

«penguji 2»

Mengetahui,

Ketua Program Studi

Mariskha Tri Aditia, PDEng

PERNYATAAN

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

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

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

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

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

Meterai

Steven Sutana
NPM: 2012730046

ABSTRAK

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

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

ABSTRACT

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

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

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

KATA PENGANTAR

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

Bandung, «bulan» «tahun»

Penulis

DAFTAR ISI

KATA PENGANTAR	xv
DAFTAR ISI	xvii
DAFTAR GAMBAR	xviii
DAFTAR TABEL	xix
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	3
2.1 Play Framework	3
2.1.1 Struktur Play Framework	3
2.1.2 Body Parsers	6
2.1.3 Internationalization	6
2.2 OpenLayers	7
2.2.1 Bing Maps	8
2.2.2 Draw Poin	8
2.3 Zurb Foundation	8
2.3.1 Sistem Grid	9
2.4 Chrome DevTools	9
2.4.1 Elements	10
2.4.2 Network	10
2.4.3 Sources	13
2.4.4 Timeline	13
2.4.5 Profile	14
DAFTAR REFERENSI	17
A THE PROGRAM	19
B THE SOURCE CODE	21

DAFTAR GAMBAR

2.1	Struktur Play Framework	4
2.2	Contoh <i>Routes</i>	4
2.3	Interaksi <i>Body Parsers</i> dengan <i>Request</i>	6
2.4	Sistem <i>grid</i> kosong sebelum memulai desain.	9
2.5	Contoh pembagian sistem <i>grid</i>	9
2.6	Panel Elements	10
2.7	Panel Network	11
2.8	Contoh Header	11
2.9	Contoh peninjauan sumber daya tersedia	12
2.10	Contoh peninjauan sumber daya tidak tersedia	12
2.11	Contoh Response	12
2.12	Contoh Cookies	13
2.13	Panel Sources dengan menyalakan Conditional <i>breakpoints</i>	13
2.14	Panel Timeline saat melakukan pencarian rute	14
2.15	Contoh CPU <i>profiler</i>	14
2.16	Contoh Heap <i>profiler</i>	15
A.1	Interface of the program	19

DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Pemanasan global, padatnya lalu lintas, dan tingginya harga bahan bakar merupakan masalah umum yang sering dijumpai pada kehidupan sehari-hari. Masalah ini dapat dipecahkan dengan menggunakan kendaraan umum. Tetapi masih banyak orang yang enggan untuk menggunakan kendaraan umum karena susahnya penggunaan kendaraan umum. Penggunaan kendaraan umum dapat dipermudah dengan adanya KIRI. Peran KIRI sangat sederhana, yaitu memberitahu dimana lokasi sekarang dan kemana lokasi tujuan, lalu KIRI akan memberitahu bagaimana cara sampai ke lokasi tujuan dengan menggunakan kendaraan umum [1].

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

Type safety [3] adalah fitur keamanan untuk mencegah kesalahan tipe data. Kesalahan tipe data dapat disebabkan oleh perbedaan tipe untuk konstanta program, variabel, dan fungsi. Sebagai contoh tipe data yang dibutuhkan berupa Float tetapi dalam program tipe data yang dimasukkan berupa Integer. Beberapa bahasa pemrograman terdapat fitur *type safety*.

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

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

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

1.2 Rumusan Masalah

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

1.3 Tujuan

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

1.4 Batasan Masalah

1. Play Framework yang digunakan adalah versi 2.4.3.
2. Kode KIRI yang sudah ada diambil dari Github pascalalfadian[5].

1.5 Metode Penelitian

Berikut adalah metode penelitian yang digunakan dalam pembuatan skripsi ini:

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

1.6 Sistematika Penulisan

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

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

BAB 2

LANDASAN TEORI

2.1 Play Framework

2.1.1 Struktur Play Framework

Play Framework [4] merupakan *framework* untuk aplikasi web dengan menggunakan bahasa Java dan Scala. Play Framework tidak sepenuhnya menggunakan bahasa Java, tetapi ada juga bahasa Scala. Terdapat bahasa Scala bukan berarti harus mempelajari bahasa Scala karena dalam Play 2 dilengkapi dengan Java API yang komplit, memberikan opsi untuk memilih bahasa pemrograman yang cocok. Play Framework mempunyai antarmuka yang sederhana, nyaman, fleksibel, dan kuat. Beberapa fitur utama yang membuat Play Framework produktif dan penggunaan yang nyaman:

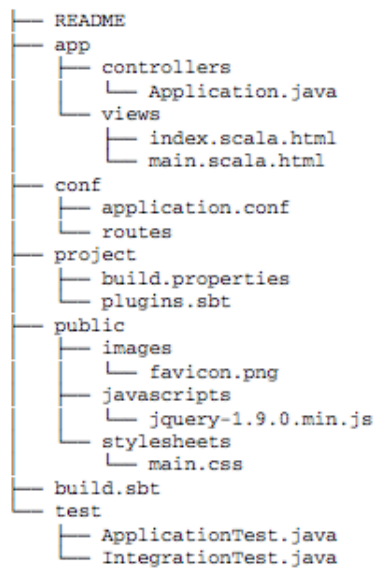
1. Penggunaan Play Framework sederhana.
2. Konfigurasi skema URL aplikasi deklaratif.
3. Pemetaan type-safe ¹
4. Play Framework menyediakan contoh sintaks *type-safe*.
5. Arsitektur yang mencakup teknologi HTML5.
6. Kode langsung aktif berubah ketika memuat kembali halaman web.
7. Fitur *full-stack web-framework*,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

¹bahasa pemrograman untuk mencegah tipe data yang salah.

² <http://docs.oracle.com/javase/7/docs/technotes/guides/net/http-keepalive.html>, diakses 30 September 2015

³ http://www.technologyuk.net/computing/software_development/event_driven_programming.shtml, diakses 30 September 2015

Play *Framework* memiliki struktur yang dapat dilihat pada gambar 2.1.



Gambar 2.1: Struktur Play Framework

Conf. Play

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



Gambar 2.2: Contoh *Routes*

Public Play

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

App. Play

Direktori *app* merupakan direktori utama pada aplikasi. Direktori *app* berisi kode aplikasi dan berbagai kebutuhan untuk menyusun aplikasi, seperti sumber file Java dan file *template*. Contoh direktori *app* pada saat pertama kali membuat aplikasi Play dapat dilihat pada gambar 2.1. Dalam *controller*, terdapat file *Application.java* yang berisi kode Java untuk memuat halaman web. *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 figur 2.1.1. *Method* pada *Controller* menghasilkan hasil berupa *Result* yang berupa *view* dengan memberi parameter "Hello World". *Method* pada *controller* dan *view* dihubungkan melalui pendefinisian pada *routes*. Sebagai contoh pada figur 2.1.1, *method ok* membangun *HTTP response* yang mengandung *response body* sebagai hasil dari *template list*. *Method ok* menerima *parameter* berupa "Hello World". Hasil *Results* dapat berupa banyak *method*, misalnya *ok* untuk mengembalikan kode *HTTP Response* 200, *notFound* untuk mengembalikan kode *HTTP Response* 404 atau tidak ada halaman yang dituju, *badRequest* adalah *Result* dengan kode *HTTP Response* 400 atau adanya kesalahan masukan pengguna, *internalServerError* adalah *Result* dengan kode *HTTP Response* 500 atau adanya kesalahan pada server, *status* adalah *Result* dengan kode *HTTP Response* yang dapat ditentukan sendiri beserta pesannya, *redirect* adalah *Result* untuk mengalihkan halaman. Contoh berbagai *Result* terdapat pada figur 2.1.1.

```

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"></script>
11 |   </head>
12 |   <body>
13 |     @title
14 |   </body>
15 | </html>

```

```

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

```

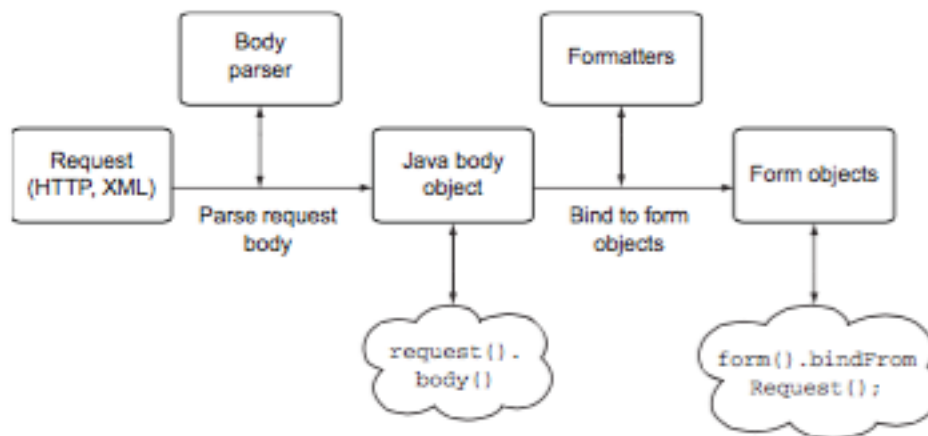
```

1 | Result ok = ok("Hello world!");
2 |
3 | Result notFound = notFound();
4 | Result pageNotFound = notFound("<h1>Page not found</h1>").as("text/html");
5 |
6 | Result badRequest = badRequest(views.html.form.render(formWithErrors));
7 |
8 | Result oops = internalServerError("Oops");
9 |
10 | Result anyStatus = status(488, "Strange response type");
11 |
12 | Result redirect = redirect("/user/home");

```

2.1.2 Body Parsers

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



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

2.1.3 Internationalization

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

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

Untuk mendukung beberapa bahasa dalam Play, membuat kunci pesan yang dipetakan pesan sesungguhnya pada file pesan seperti pada figur 2.1.3.

```
1| welcome = Welcome!  
2| users.login = Log in  
3| shop.thanks = Thank you for your order
```

Play harus mengetahui bahasa apa saja yang ada pada aplikasi. Untuk itu, daftarkan bahasa pada `application.conf` seperti pada figur 2.1.3.

```
1| application.langs="en,en-US,nl"
```

Setelah itu, buat file pada direktori `conf` dengan nama `messages.LANG` dimana `LANG` adalah bahasa yang digunakan. Sebagai contoh untuk file bahasa Prancis akan disimpan pada `conf/messages.fr` dengan isinya

.

2.2 OpenLayers

OpenLayers [6] merupakan *library* yang memiliki performa tinggi dan fitur yang dikemas untuk kebutuhan menampilkan peta menggunakan JavaScript. Dalam pengembangan aplikasi yang menggunakan fitur peta, tugas yang paling penting dan utama adalah membuat peta tersebut. Peta menjadi inti untuk menambahkan dan menampilkan data. Fitur yang terdapat pada OpenLayers adalah:

- *Tiled Layers*

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

- *Vector Layers*

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

- Cepat dan Siap untuk Perangkat *Mobile*

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

- Mudah menyesuaikan peta dan *cutting edge*

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

Modul OpenLayers yang dipakai dalam penelitian ini adalah:

- Bing Maps untuk menampilkan peta menggunakan Bing. KIRI menggunakan *map provider* Bing Maps sebagai peta pada halaman utama KIRI.

- Draw untuk menggambar poin pada peta. Saat peta KIRI menangkap *event mouseclick*, muncul poin yang berupa kustom gambar pada peta KIRI sebagai asal tempat dan tujuan.

2.2.1 Bing Maps

Peta yang digunakan merupakan Bing Maps dari Microsoft. Jika ingin menggunakan Bing Maps pada Openlayers, harus mendapatkan kunci yang didapat dari Bing Maps serta memilih tipe peta yang akan ditampilkan. Sebagai contoh pada figur 2.2.1.

```

1      var mapLayer = new ol.layer.Tile(
2      {
3          source : new ol.source.BingMaps(
4          {
5              key : 'AuV7xXD6_UMiQ5BLoZr0xkpjLpzWqMT55772Q8XtLIQeuDebHPKiNXSlZXxEr1GA',
6              imagerySet : 'Road'
7          })
8      });

```

2.2.2 Draw Poin

Pada peta KIRI, pengguna dapat memilih tempat awal dan tujuan dengan memasukkan nama tempat atau memilih tempat pada peta. Pada Openlayers, disediakan fitur untuk menangani kursor klik pengguna. Pertama menentukan vektor gambar yang akan ditampilkan pada peta seperti pada figur 2.2.2. Setelah itu, menggambar pada peta seperti pada figur ??.

```

1      var source = new ol.source.Vector({wrapX: false});
2      var vector = new ol.layer.Vector({
3          source: source,
4          style: new ol.style.Style({
5              fill: new ol.style.Fill({
6                  color: 'rgba(255, 255, 255, 0.2)'
7              }),
8              stroke: new ol.style.Stroke({
9                  color: '#ffcc33',
10                 width: 2
11             }),
12             image: new ol.style.Circle({
13                 radius: 7,
14                 fill: new ol.style.Fill({
15                     color: '#ffcc33'
16                 })
17             })
18         })
19     });

1      draw = new ol.interaction.Draw({
2          source: source,
3          type: /** @type {ol.geom.GeometryType} */ (value),
4          geometryFunction: geometryFunction,
5          maxPoints: maxPoints
6      });

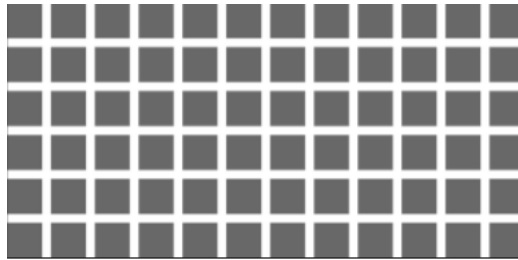
```

2.3 Zurb Foundation

Zurb Foundation [7] merupakan sebuah *toolkit* yang membantu untuk desain dan mengembangkan sebuah halaman *web*. Zurb Foundation menggunakan sistem *grid*, banyak komponen CSS serta JavaScript.

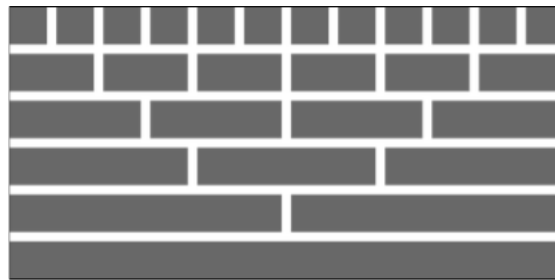
2.3.1 Sistem Grid

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



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

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



Gambar 2.5: Contoh pembagian sistem *grid*.

Komponen Foundation adalah kode CSS yang membantu untuk desain dan merepresentasikan konten halaman *web*. Dengan menggunakan komponen Foundation, tidak perlu desain sendiri dari awal. CSS pada Foundation juga dapat dikustomisasi sesuai dengan kebutuhan. Jika ingin menggunakan komponen Foundation, kita hanya perlu menggunakan tag class pada elemen yang diinginkan. Contoh penggunaan komponen Foundation dapat dilihat pada figur 2.5.

1| ``

2.4 Chrome DevTools

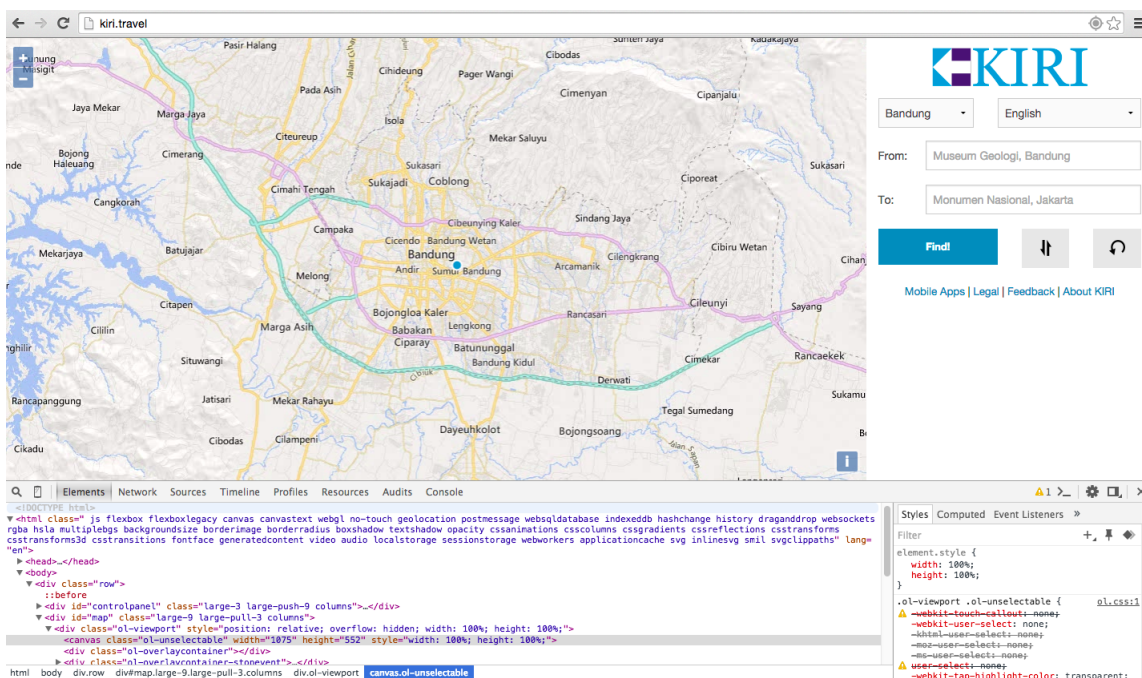
Chrome DevTools[8] merupakan perangkat untuk memperhatikan dan melakukan *debugging* halaman web yang terdapat pada *browser* Google Chrome. DevTools dapat digunakan secara efisien untuk memeriksa tampilan, mengatur *breakpoints* JavaScript, dan optimasi kode. DevTools dapat diakses dengan melakukan klik kanan pada halaman web lalu klik periksa elemen. DevTools disusun dalam beberapa panel *task-orientated*. Beberapa panel tersebut adalah:

1. **Elements**, untuk memeriksa, melihat, dan mengubah tampilan halaman web.
2. **Network**, untuk memantau aktivitas jaringan pada halaman web secara *real-time*.

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

2.4.1 Elements

Panel Elements dapat memperlihatkan struktur halaman web dalam bentuk *Document Object Model* (DOM), dan dapat mengubah elemen DOM dengan cepat. DOM adalah struktur logis dokumen serta cara dokumen diakses dan diubah ⁴. Sebagai contoh pada gambar 2.6, pemeriksaan elemen akan memperlihatkan dua bagian, yaitu DOM dan CSS yang digunakan pada DOM.



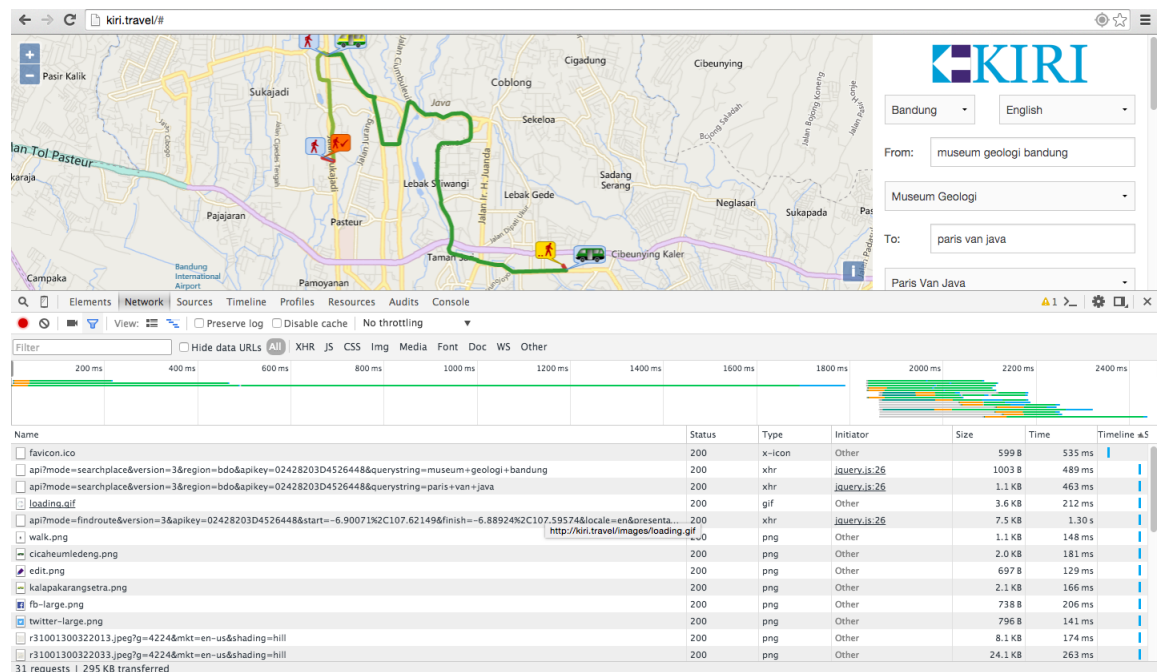
Gambar 2.6: Panel Elements

2.4.2 Network

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

⁴<http://www.w3.org/TR/DOM-Level-2-Core/introduction.html>, diakses 2 Oktober 2015

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



Gambar 2.7: Panel Network

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

- **Header**

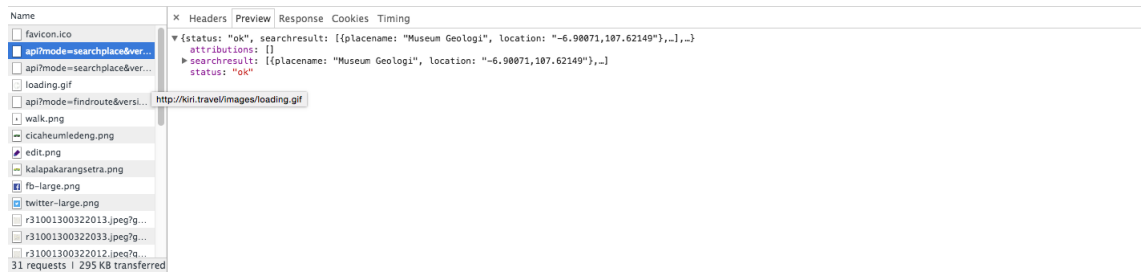
Header menampilkan *request URL*, *request method*, *status code*, *response headers*, *request headers*, dan *query string parameters* beserta nilainya.



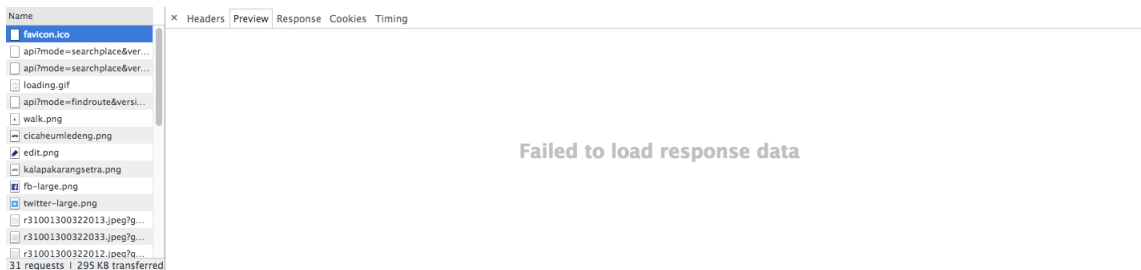
Gambar 2.8: Contoh Header

• Preview

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



Gambar 2.9: Contoh peninjauan sumber daya tersedia



Gambar 2.10: Contoh peninjauan sumber daya tidak tersedia

• Response

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



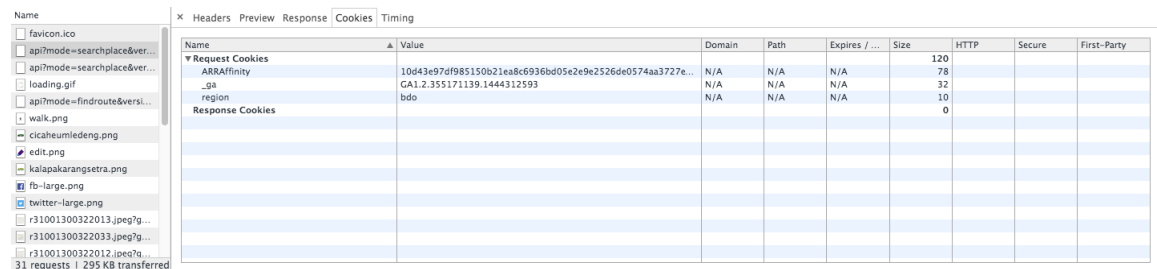
Gambar 2.11: Contoh Response

• Cookies

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

- **Name**, nama *cookie*.
- **Value**, nilai *cookie*.
- **Domain**, asal *cookie*.

- **Path**, URL *cookie*.
- **Expires / Max-Age**, batas habis *cookie*.
- **Size**, ukuran *cookie*.
- **HTTP**
- **Secure**
- **First-Party**

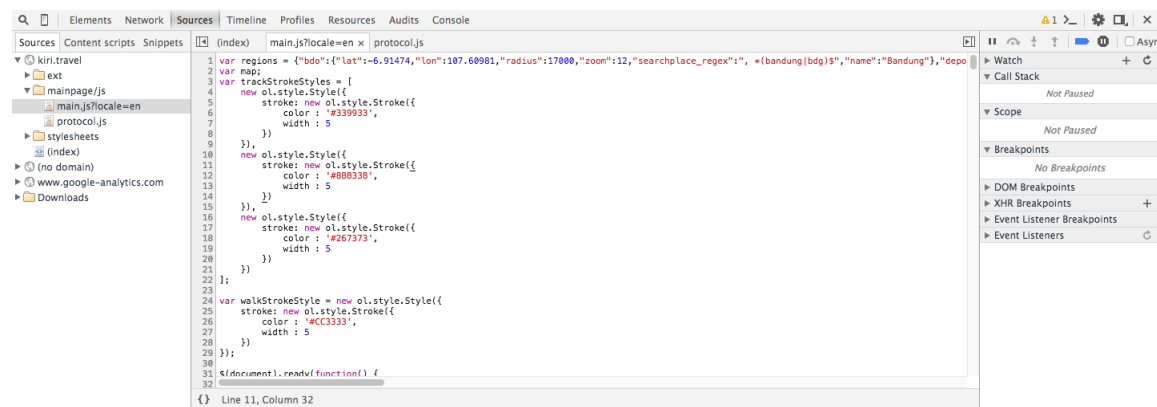


Name	Value	Domain	Path	Expires / ...	Size	HTTP	Secure	First-Party
Request Cookies								
ARRAffinity	10d43e97df985150b21ea8c6936bd05e2e9e2526de0574aa3727e...	N/A	N/A	N/A	120			
.ga	CA1.2.355171139.1444312593	N/A	N/A	N/A	78			
region	bdo	N/A	N/A	N/A	32			
Response Cookies								
					10			
					0			

Gambar 2.12: Contoh Cookies

2.4.3 Sources

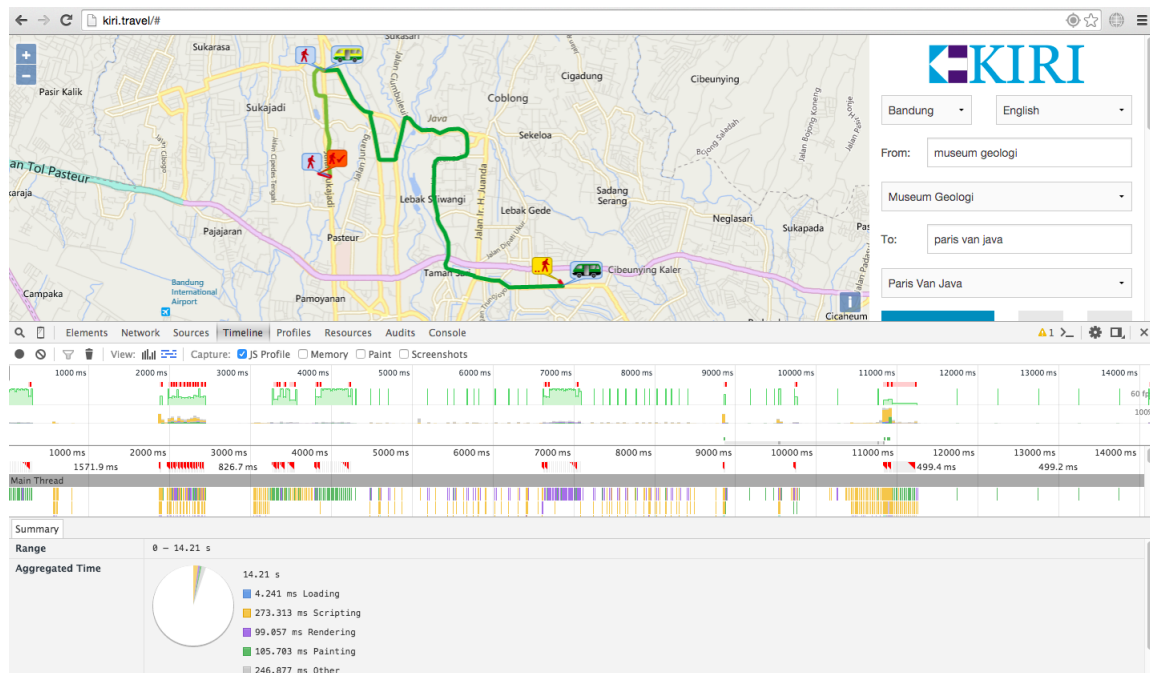
Panel Sources memungkinkan untuk melakukan *debugging* JavaScript dengan menggunakan *breakpoints* ⁵. Pengembang membutuhkan alat *debugging* untuk menemukan penyebab masalah dan memperbaikinya dengan cepat.

Gambar 2.13: Panel Sources dengan menyalakan Conditional *breakpoints*

2.4.4 Timeline

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

⁵Terdapat 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*

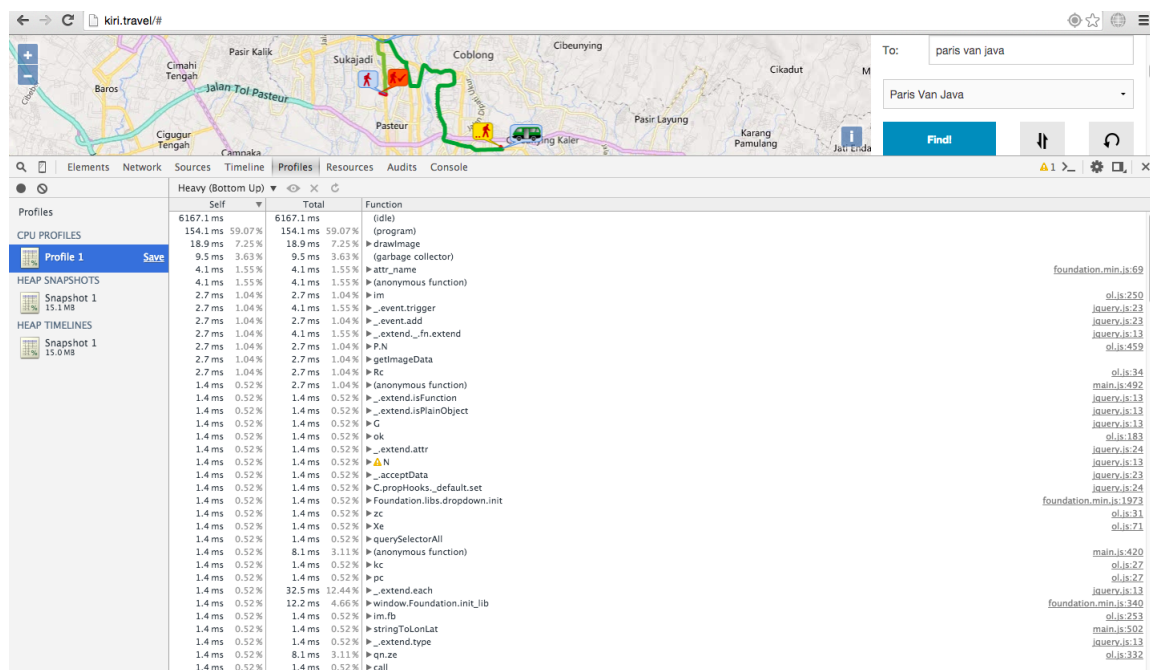


Gambar 2.14: Panel Timeline saat melakukan pencarian rute

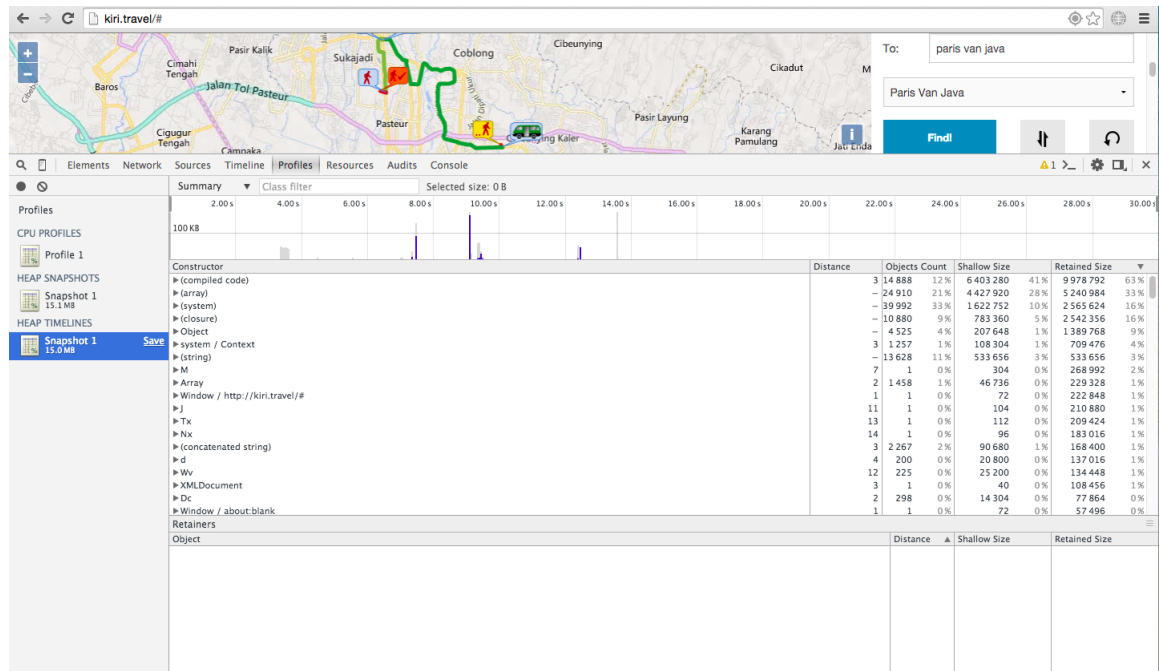
2.4.5 Profile

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

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

Gambar 2.15: Contoh CPU *profiler*

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

Gambar 2.16: Contoh Heap *profiler*

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

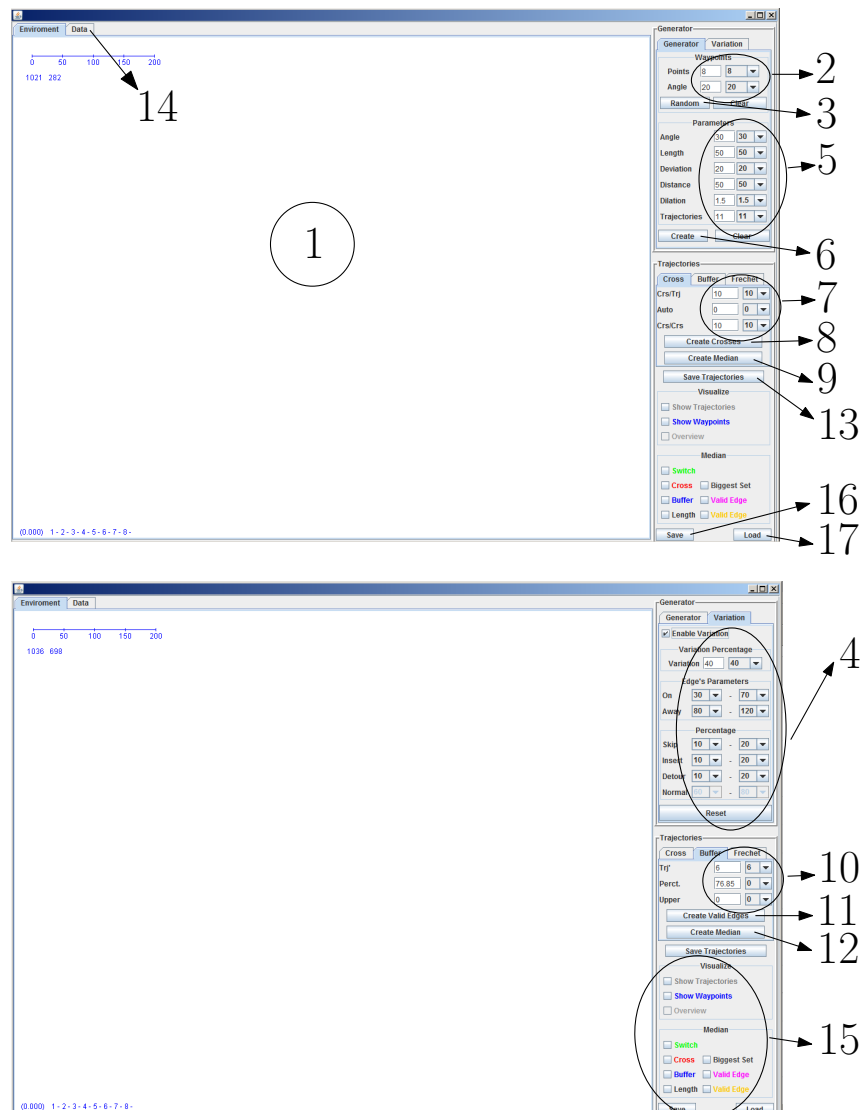
DAFTAR REFERENSI

- [1] Pascal Alfadian, “KIRI.” <http://static.kiri.travel/>, 2014. [Online; diakses 28 September 2015].
- [2] The PHP Group, “php.” <http://php.net>, 2015. [Online; diakses 28 September 2015].
- [3] Anonymous, “Type Safety.” https://en.wikipedia.org/wiki/Type_safety#Type-safe_and_type-unsafe_languages, 2015. [Online; diakses 27 Oktober 2015].
- [4] N. Leroux and S. D. Kaper, *Play for Java*. Manning Publications Co., 2014.
- [5] Pascal Alfadian, “TirtayasaGH.” <https://github.com/pascalalfadian/TirtayasaGH>, 2014. [Online; diakses 28 September 2015].
- [6] P. A. Santiago, *OpenLayers Cookbook*. Packt Publishing, 2012.
- [7] A. D. Patterson, *Getting Started with Zurb Foundation 4*. Packt Publishing, 2013.
- [8] Google, “Chrome DevTools.” <https://developers.google.com/web/tools/chrome-devtools/index?hl=en#learnmore>, 2015. [Online; diakses 2 Oktober 2015].

LAMPIRAN A

THE PROGRAM

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



Gambar A.1: Interface of the program

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

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

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

LAMPIRAN B

THE SOURCE CODE

Listing B.1: MyFurSet.java

```
1
2 import java.util.ArrayList;
3 import java.util.Collections;
4 import java.util.HashSet;
5
6 /**
7  *
8  * @author Lionov
9  */
10
11 //class for set of vertices close to furthest edge
12 public class MyFurSet {
13     protected int id; //id of the set
14     protected MyEdge FurthestEdge; //the furthest edge
15     protected HashSet<MyVertex> set; //set of vertices close to furthest edge
16     protected ArrayList<ArrayList<Integer>> ordered; //list of all vertices in the set for each
17         trajectory
18     protected ArrayList<Integer> closeID; //store the ID of all vertices
19     protected ArrayList<Double> closeDist; //store the distance of all vertices
20     protected int totaltrj; //total trajectories in the set
21
22     /**
23      * Constructor
24      * @param id : id of the set
25      * @param totaltrj : total number of trajectories in the set
26      * @param FurthestEdge : the furthest edge
27      */
28     public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
29         this.id = id;
30         this.totaltrj = totaltrj;
31         this.FurthestEdge = FurthestEdge;
32         set = new HashSet<MyVertex>();
33         ordered = new ArrayList<ArrayList<Integer>>();
34         for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
35         closeID = new ArrayList<Integer>(totaltrj);
36         closeDist = new ArrayList<Double>(totaltrj);
37         for (int i = 0; i < totaltrj; i++) {
38             closeID.add(-1);
39             closeDist.add(Double.MAX_VALUE);
40         }
41     }
42
43     /**
44      * set a vertex into the set
45      * @param v : vertex to be added to the set
46      */
47     public void add(MyVertex v) {
48         set.add(v);
49     }
50
51     /**
52      * check whether vertex v is a member of the set
53      * @param v : vertex to be checked
54      * @return true if v is a member of the set , false otherwise
55      */
56     public boolean contains(MyVertex v) {
57         return this.set.contains(v);
58     }
59 }
```