

SKRIPSI

**PENGEMBANGAN APLIKASI PENCARI
RUTE KENDARAAN UMUM
UNTUK WINDOWS PHONE**



YOHAN

NPM: 2011730048

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2014**

UNDERGRADUATE THESIS

**DEVELOPMENT APPLICATION PUBLIC TRANSPORT
ROUTE SEARCH FOR WINDOWS PHONE**



YOHAN

NPM: 2011730048

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

2014

ABSTRAK

Transportasi menjadi bagian yang tidak terpisahkan dalam kehidupan manusia saat penelitian ini dibuat. Namun saat ini banyak orang yang lebih memilih menggunakan kendaraan pribadi dibanding kendaraan umum. Maraknya penggunaan kendaraan pribadi menimbulkan masalah pemanasan global, kemacetan, dan harga bahan bakar minyak yang tinggi.

Pesatnya perkembangan teknologi membuat perangkat bergerak kian digemari di Indonesia. Salah satu sistem operasi yang meramaikan industri perangkat bergerak adalah Windows Phone. Windows Phone merupakan sistem operasi buatan Microsoft. Salah satu yang membedakan sistem operasi Windows Phone dengan sistem operasi lain adalah antarmuka yang Microsoft sebut Metro. Saat penelitian ini dilakukan sistem operasi Windows Phone adalah versi 8.

Perangkat lunak yang berhasil dibangun pada penelitian ini menggunakan bahasa C# untuk Windows Phone. Untuk mencari rute angkutan umum penulis memanfaatkan Kiri API dan memanfaatkan *web service* untuk mendapatkan rute yang diinginkan pengguna.

Kata-kata kunci: Rute, Kendaraan Umum, Windows Phone

ABSTRACT

Transportation becomes an integral part of human life when this thesis was made. But today many people who prefer to use private transport compared to public transport. Widespread use of private vehicles cause problems of global warming, congestion, and fuel prices are high.

The rapid development of technology makes it increasingly popular mobile devices in Indonesia. One of the operating system that enliven the mobile industry is Windows Phone. Windows Phone is Microsoft operating system. One of the differentiating Windows Phone operating system with other operating systems is a Microsoft interface called Metro. Currently this research system Windows Phone operating is version 8.

The software successfully constructed in this study using the language C# for Windows Phone. To find a public transport route Left writers utilize the web service API and utilize to get the desired route users.

Keywords: Route, Public Transport, Windows Phone

DAFTAR ISI

DAFTAR ISI	ix
DAFTAR GAMBAR	xii
DAFTAR TABEL	xiv
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	2
1.4 Batasan Masalah	2
1.5 Metode Penelitian	2
1.6 Sistematika Penulisan	3
2 DASAR TEORI	5
2.1 Windows Phone	5
2.1.1 Lingkungan Kerja	5
2.1.2 XAML	6
2.1.3 Kontrol terhadap Ponsel	6
2.1.4 Siklus Hidup Aplikasi	10
2.1.5 Peta di Windows Phone	12
2.1.6 Lokasi	17
2.1.7 Memanfaatkan Sumber Data	20
2.1.8 Thread	22
2.2 Kiri API	23
2.2.1 <i>Web Service</i> Penentuan Rute	24
2.2.2 <i>Web Service</i> Pencarian Lokasi	25
2.2.3 <i>Web Service</i> Menemukan Transportasi Terdekat	26
3 ANALISIS	29
3.1 Analisis Aplikasi Sejenis	29
3.2 Analisis Aplikasi	32
3.2.1 Kebutuhan Aplikasi	32
3.2.2 Analisis Kontrol yang Dipakai	32
3.2.3 Analisis Terhadap Siklus Hidup Aplikasi	33
3.2.4 Analisis Peta	34
3.2.5 Analisis Pemanfaatan Sumber Data	36
3.2.6 Analisis Kiri API	36
3.2.7 Diagram <i>Use Case</i> dan Skenario	41
3.2.8 Kelas Diagram	43
4 PERANCANGAN	45
4.1 Diagram <i>Sequence</i>	45

4.2	Perancangan Kelas	49
4.2.1	Kelas <i>PhoneApplicationPage</i>	50
4.2.2	Kelas <i>MainPage</i>	52
4.2.3	Kelas <i>City</i>	54
4.2.4	Kelas <i>BackgroundWorker</i>	54
4.2.5	Kelas <i>Geocoordinate</i>	54
4.2.6	Kelas <i>Geolocator</i>	55
4.2.7	Kelas <i>Geoposition</i>	55
4.2.8	Kelas <i>LocationFinder</i>	55
4.2.9	Kelas <i>Map</i>	56
4.2.10	Kelas <i>HttpClient</i>	57
4.2.11	Kelas <i>JsonConvert</i>	57
4.2.12	Kelas <i>Protocol</i>	57
4.2.13	Kelas <i>RootObjectSearchPlace</i>	59
4.2.14	Kelas <i>SearchResult</i>	59
4.2.15	Kelas <i>RootObjectFindRoute</i>	59
4.2.16	Kelas <i>RoutingResult</i>	60
4.2.17	Kelas <i>Route</i>	60
4.3	Perancangan Antar Muka	62
4.3.1	Antarmuka Kelas <i>MainPage</i>	62
4.3.2	Antarmuka Kelas <i>Map</i>	63
4.3.3	Antarmuka Kelas <i>Route</i>	64
5	IMPLEMENTASI DAN PENGUJIAN APLIKASI	67
5.1	Implementasi	67
5.1.1	Perangkat Keras untuk Implementasi	67
5.1.2	Perangkat Lunak untuk Implementasi	67
5.1.3	Hasil Implementasi	68
5.2	Pengujian	70
5.2.1	Lingkungan Pengujian	71
5.2.2	Pengujian Fungsional	72
5.2.3	Pengujian Eksperimental	73
5.2.4	Perbandingan Aplikasi Pencari Rute di Android dengan Aplikasi Pencari Rute di Windows Phone	73
6	KESIMPULAN DAN SARAN	77
6.1	Kesimpulan	77
6.2	Saran	77
DAFTAR REFERENSI	79	
A	KODE PROGRAM KELAS MAINPAGE	81
B	KODE PROGRAM KELAS MAP	89
C	KODE PROGRAM KELAS ROUTE	91
D	KODE PROGRAM KELAS CITY	99
E	KODE PROGRAM KELAS LOCATIONFINDER	101
F	KODE PROGRAM KELAS PROTOCOL	105

DAFTAR GAMBAR

2.1	Hirarki navigasi	7
2.2	<i>TextBlock</i> , <i>TextBox</i> dan <i>PasswordBox</i>	8
2.3	Gambar kontrol pada Windows Phone	9
2.4	Antarmuka <i>ListBox</i>	10
2.5	Gambar siklus hidup aplikasi[1]	10
2.6	Tampilan peta pada Windows Phone	12
2.7	<i>cartographic</i>	13
2.8	Keluaran Toolkit Pushpin pada peta [2]	14
2.9	Tampilan polyline pada Peta	15
3.1	Tampilan utama aplikasi Public Transport	29
3.2	Menunjuk lokasi pada peta	30
3.3	Memberikan daftar nama tempat dan nama jalan terkait	30
3.4	Tampilan rute kendaraan umum dalam bentuk daftar	31
3.5	Tampilan rute kendaraan umum di peta	31
3.6	Tampilan <i>pushpin</i> untuk angkot	35
3.7	Tampilan <i>pushpin</i> untuk jalan kaki	35
3.8	Diagram <i>use case</i>	42
3.9	Diagram Kelas	44
4.1	Diagram <i>sequence</i> Mendapatkan Kordinat perangkat	45
4.2	Diagram <i>sequence</i> Mendapatkan Kordinat pada Peta	46
4.3	Diagram <i>sequence</i> Mendapatkan Rute	48
4.4	Diagram Kelas	49
4.5	Kelas City	49
4.6	Kelas JsonConvert	49
4.7	Kelas LocationFinder	50
4.8	Kelas Map	50
4.9	<i>Package</i> WindowsPhone	50
4.10	Kelas Protocol	51
4.11	Kelas Route	51
4.12	Kelas SearchPlace	51
4.13	Kelas FindRoute	52
4.14	Antarmuka <i>MainPage</i>	62
4.15	Antarmuka Daftar Tempat	63
4.16	Antarmuka <i>Map</i>	64
4.17	Antarmuka <i>Route</i>	64
4.18	Antarmuka Rute dalam bentuk Daftar	65
5.1	Gambar antarmuka kelas <i>MainPage</i>	68
5.2	Gambar antarmuka <i>Splash</i> di kelas <i>MainPage</i>	69
5.3	Gambar antarmuka <i>list</i> asal dan <i>list</i> tujuan di kelas <i>MainPage</i>	69
5.4	Gambar antarmuka kelas <i>Map</i>	70

5.5	Gambar antarmuka menunggu di kelas Route	70
5.6	Gambar antarmuka kelas Route	71
5.7	Gambar antarmuka <i>list</i> di kelas Route	71

DAFTAR TABEL

2.1	Properti kelas Map	15
2.2	Properti <i>Polyline Class</i>	17
2.3	Kelas pada <i>Namespace Geolocator</i>	19
2.4	Properti pada <i>Geocoordinate</i>	19
2.5	Tabel parameter layanan penentuan rute	24
2.6	Tabel parameter layanan pencarian lokasi	25
2.7	Tabel parameter layanan menemukan transportasi terdekat	26
3.1	Skenario mandapatkan lokasi untuk masukan lokasi asal dan lokasi tujuan	41
3.2	Skenario memasukan lokasi asal dan lokasi tujuan pada <i>TextBox</i>	42
3.3	Skenario menunjuk lokasi asal dan lokasi tujuan pada peta	43
3.4	Skenario memilih alamat	43
3.5	Skenario menampilkan rute kendaraan umum	43
5.1	Tabel Hasil pengujian fungsionalitas terhadap pengguna	72
5.2	Tabel Hasil pengujian fungsionalitas	75
5.3	Tabel perbedaan	76

¹

BAB 1

²

PENDAHULUAN

³ Pada Bab satu akan dibahas pendahuluan dari penelitian yang dilakukan. Bab satu terbagi
⁴ dalam delapan *sub bab*, yaitu *latar belakang, rumusan masalah, tujuan, batasan masalah,*
⁵ *ruang lingkup masalah, metode penelitian, teknik pengumpulan data, dan sistematika penu-*
⁶ *lisan.*

⁷ **1.1 Latar Belakang**

⁸ Transportasi menjadi bagian yang penting bagi manusia di saat penelitian ini dilakukan.
⁹ Ada dua jenis transportasi bagi seseorang yaitu kendaraan umum dan kendaraan pribadi.
¹⁰ Kenyataannya pada saat penelitian ini dilakukan banyak yang lebih memilih kendaraan
¹¹ pribadi dibanding kendaraan umum. Maraknya penggunaan kendaraan pribadi dan penam-
¹² bahan jalur kendaraan yang tidak sebanding banyaknya kendaraan menimbulkan kemacetan.
¹³ Maraknya penggunaan kendaraan pribadi dikarenakan kurang nyamannya kendaraan umum
¹⁴ dan kesulitan dalam menentukan kendaraan umum yang harus dinaiki. Banyaknya rute ken-
¹⁵ daraan umum membuat orang kebingungan dalam memilih kendaraan umum menuju lokasi
¹⁶ yang diinginkan. Seseorang cenderung malas untuk bertanya dan mencari rute yang efisien.
¹⁷ Karena hal tersebut, seseorang lebih memilih menggunakan kendaraan pribadi ketimbang
¹⁸ kendaraan umum.

¹⁹ Ide pembuatan aplikasi yang memudahkan seseorang dalam menentukan rute kendara-
²⁰ an umum sudah lebih dulu ada yang dikenal dengan nama Kiri. Kiri dibuat dengan latar
²¹ belakang

²² tiga masalah besar yaitu pemanasan global, kemacetan, dan harga bahan bakar minyak yang
²³ tinggi¹. Meskipun Kiri pertama dibuat di web tetapi Kiri dapat dimanfaatkan untuk pen-
²⁴ carian kendaraan selain di web. Pemanfaatan Kiri tersebut dalam mencari rute kendaraan
²⁵ umum dengan menggunakan Kiri API.

²⁶ Pesatnya perkembangan teknologi sekarang ini mendorong perkembangan perangkat ber-
²⁷ gerak (*mobile*). Perangkat bergerak kian digemari orang-orang terutama di Indonesia. Salah
²⁸ satu yang menarik perhatian adalah Windows Phone 8 yang dibuat Microsoft. Antarmuka
²⁹ Windows Phone 8 yang disebut *Metro* cukup menarik dan mudah digunakan.

³⁰ Berdasarkan hal tersebut, penulis mencoba mengembangkan aplikasi Pencarian Rute
³¹ Kendaraan Umum di Windows Phone dalam tugas akhir ini. Aplikasi yang penulis kem-
³² bangan akan memungkinkan pengguna menemukan rute kendaraan umum untuk sampai di
³³ tujuan. Untuk memudahkan pengguna, penulis akan menampilkan dalam dua bentuk yaitu

¹<http://static.kiri.travel/en-about/>

1 peta dan daftar.

2 **1.2 Rumusan Masalah**

3 Sehubung dengan latar belakang diatas timbul permasalahan sebagai berikut:

- 4 ● Bagaimana membuat aplikasi di Windows Phone?
- 5 ● Bagaimana mengintegrasikan Kiri API dengan aplikasi pencari rute kendaraan umum
6 di Windows Phone?

7 **1.3 Tujuan**

8 Berdasarkan rumusan masalah pada sub bab 1.2, maka tujuan dari pembuatan tugas akhir
9 ini adalah:

- 10 ● Mempelajari cara pembuatan perangkat lunak di Windows Phone lalu mengembangkan
11 aplikasi yang akan dibuat.
- 12 ● Membuat aplikasi di Windows Phone yang memanfaatkan Kiri API.

13 **1.4 Batasan Masalah**

14 Ruang lingkup pengembangan aplikasi Pencari Rute Kendaraan untuk Windows Phone ini
15 dibatasi hal berikut:

- 16 ● Aplikasi ini akan berjalan di sistem operasi Windows Phone 8.
- 17 ● Aplikasi ini membutuhkan koneksi internet.
- 18 ● Aplikasi ini akan menampilkan rute jalur angkot, bus umum dan travel di tiga kota
19 besar yaitu Bandung, Jakarta, dan Surabaya.

20 **1.5 Metode Penelitian**

21 Metode Penelitian yang penulis gunakan dalam membuat tugas akhir ini adalah sebagai
22 berikut:

- 23 ● Melakukan studi pustaka mengenai XAML, kontrol dan navigasi di Windows Phone,
24 Peta di Windows Phone, GPS di Windows Phone dan Kiri API.
- 25 ● Melakukan analisis terhadap aplikasi lain yang menggunakan Kiri API.
- 26 ● Melakukan analisis terhadap dasar teori untuk pembangunan aplikasi Pencarian Rute
27 Kendaraan Umum untuk Windows Phone.
- 28 ● Melakukan perancangan aplikasi Pencarian Rute Kendaraan Umum untuk Windows
29 Phone.
- 30 ● Implementasi dari aplikasi Pencarian Rute Kendaraan Umum untuk Windows Phone.

- 1 • Menguji aplikasi Pencarian Rute Kendaraan Umum untuk Windows Phone.
- 2 • Membuat kesimpulan.

3 1.6 Sistematika Penulisan

4 Bab 1 membahas latar belakang masalah, rumusan masalah, tujuan penulisan tugas
5 akhir, batasan masalah, ruang lingkup masalah, metode penelitian, dan teknik pengumpulan
6 data tugas akhir ini.

7 Bab 2 membahas tentang teori-teori yang akan digunakan dalam tugas akhir ini. Bahasan
8 yang dijelasakan pada bab ini adalah Windows Phone dan Kiri API. Teori Windows Phone
9 yang dijelaskan meliputi lingkungan kerja, XAML, kontrol terhadap ponsel, siklus hidup
10 aplikasi, peta di Windows Phone, lokasi, dan memanfaatkan sumber data. Teori Kiri API
11 yang dijelaskan meliputi *web service* penentuan rute, *web service* pencarian lokasi, dan *web*
12 *service* menemukan transportasi terdekat.

13 Bab 3 membahas tentang analisis pembangunan aplikasi Pencarian Rute Kendaraan
14 Umum untuk Windows Phone. Pada Bab 3 akan dibahas mengenai analisis kebutuhan apli-
15 kasi, analisis kontrol yang dipakai, analisis terhadap siklus hidup aplikasi, analisis terhadap
16 siklus hidup aplikasi, analisis peta, analisis memanfaatkan sumber data, analisis Kiri API,
17 diagram *use case*, dan diagram kelas.

¹

BAB 2

²

DASAR TEORI

³ Bab ini berisi dasar teori dari pembangunan Aplikasi Pencarian Rute Kendaraan Umum
⁴ untuk Windows Phone. Beberapa teori yang dibahas dalam bab ini adalah XAML, kontrol
⁵ terhadap ponsel, siklus hidup Windows Phone, peta, lokasi , pemanfaatan sumber data, dan
⁶ Kiri API.

⁷ **2.1 Windows Phone**

⁸ Windows Phone merupakan sistem operasi untuk perangkat bergerak yang dikembangkan
⁹ Microsoft. Pengembangan aplikasi Windows Phone membutuhkan Windows Desktop
¹⁰ sebagai media pengembangan. Bahasa pemrograman yang digunakan untuk membuat
¹¹ perangkat lunak di Windows Phone yaitu C# atau Visual Basic^[2].

¹² Pada sub bab [2.1.1](#) sampai [2.1.7](#) akan membahas pemrograman di Windows Phone.
¹³ Pembahasan akan dimulai dengan apa itu Windows Phone dan fitur di Windows Phone
¹⁴ yang akan digunakan dalam pembangunan perangkat lunak Pencarian Rute Kendaraan di
¹⁵ Windows Phone.

¹⁶ **2.1.1 Lingkungan Kerja**

¹⁷ Microsoft .NET framework merupakan sebuah perangkat lunak yang dibangun untuk
¹⁸ membantu dalam pembangunan aplikasi di Windows, Windows Phone, Windows Server,
¹⁹ and Microsoft Azure^[1]. Microsoft .NET framework terdiri dari runtime bahasa umum dan
²⁰ perpustakaan kelas .NET Framework, yang meliputi kelas, interface, dan jenis nilai yang
²¹ mendukung berbagai teknologi. Microsoft .NET Framework menyediakan lingkungan yang
²² mudah dikelola, pengembangan yang disederhanakan, dan integrasi dengan berbagai bahasa
²³ pemrograman, termasuk Visual Basic dan Visual C#.

²⁴ Seperti yang telah disebutkan ada dua bahasa pemrograman dalam .NET Framework
²⁵ yang dipakai dalam pembangunan aplikasi di Windows Phone 8 yaitu Visual Basic dan
²⁶ Visual C#.

²⁷ Untuk masalah kehandalan keduanya menawarkan kehandalan yang baik. Kelebihan dari
²⁸ Visual Basic adalah bahasa pemrograman berorientasi objek yang kuat dan memiliki banyak
²⁹ pengembangan fitur di inheritance, polymorphism, interfaces, and overloading^[1]. Kelebihan
³⁰ dari C# yang merupakan pengembangan dari C/C++ adalah sederhana, modern, aman dan
³¹ berorientasi objek^[1]. Satu hal yang dirasakan penulis adalah kenyamanan ketika memilih
³² bahasa .NET tersebut. Akan lebih mudah bagi developer yang menggunakan Visual Basic
³³ 6.0 untuk menggunakan Visual

- 1 Basic .NET. Tetapi bagi developer yang menggunakan C++ atau java sebelumnya akan
- 2 lebih mudah menggunakan C#.

3 2.1.2 XAML

4 *Extensible Application Markup Language* (XAML) merupakan bahasa deklaratif yang
5 dipakai untuk membuat antarmuka aplikasi. XAML merupakan bahasa yang digunakan
6 untuk membuat antarmuka di Windows Phone 8[2]. Pada dasarnya penggunaan XAML
7 sama dengan HTML pada pembuatan antarmuka web. XAML dapat menginisialisasi objek
8 dan mengatur properti untuk menunjukkan hubungan antar objek.

9 Untuk aturan penulisan sintak XAML didasarkan pada XML. Setiap XAML Windows
10 Runtime menggunakan konvensi bahasa XAML dan ditulis pada *namespace* yang ditandai
11 dengan *prefix* x sebagai elemen paling atas. Setelah itu di baris ke dua dimulai dengan
12 xmlns diikuti titik dua, lalu nama dari *namespace*, diikuti tanda sama dengan dan *path*
13 yang merepresentasikan *namespace*. *Prefix* x pada XAML mengandung beberapa struktur
14 program yang sering kita gunakan yaitu :

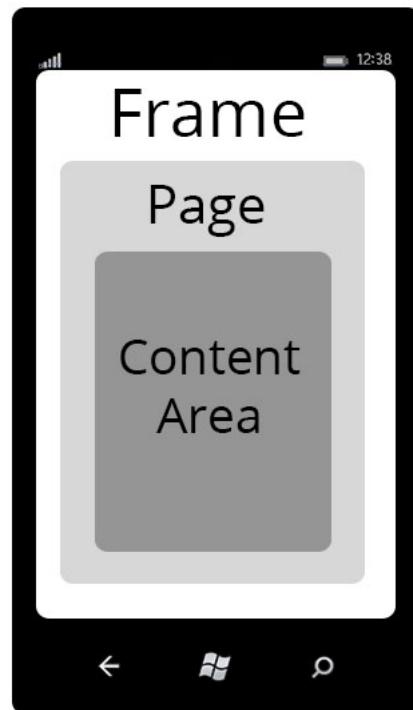
- 15 • x:Key : sebuah nama unik untuk menunjuk referensi ke suatu resource atau berkas
16 lain. Nilai ini dapat dipanggil kembali untuk menggunakan resource tersebut.
- 17 • x:Class : menunjukkan nama kelas.
- 18 • x:Name : menunjukkan nama sebuah obyek dan untuk membedakan antar obyek yang
19 satu dengan obyek yang lain.
- 20 • x:Uid : mengidentifikasi elemen objek dalam XAML. Elemen objek merupakan objek
21 yang dapat melakukan kontrol terhadap kelas atau elemen lain yang ditampilkan di
22 desain antarmuka.

23 2.1.3 Kontrol terhadap Ponsel

24 Maksud dari kontrol terhadap ponsel adalah pengaturan tata letak terhadap antarmuka
25 di Windows Phone[1]. Windows Phone 8 menyediakan banyak set kontrol yaitu tata letak,
26 tombol, kontrol masukan untuk mendapatkan informasi sampai ke menu.

27 2.1.3.1 Navigasi

28 Aplikasi yang dibuat di Windows Phone didasarkan pada model halaman. Model ha-
29 laman adalah pengguna berpindah dari satu halaman ke halaman lain dengan konten yang
30 berbeda dan *frame* sebagai pengontrolnya. Setiap antarmuka aplikasi dibungkus dengan
31 *frame*. *Frame* ini yang akan melakukan kontrol terhadap aplikasi dan memungkinkan per-
32 pindahan dari satu halaman ke halaman lain. Sedangkan halaman merupakan pembungkus
33 dari elemen di dalamnya saja. Untuk lebih jelas mengenai *frame*, halaman, dan area konten
34 dapat dilihat pada gambar 2.1.



Gambar 2.1: Hirarki navigasi

2.1.3.2 Kontrol Tata Letak

Kontrol tata letak merupakan penampung pada antarmuka Windows Phone untuk objek di antarmuka dan kontrol yang lain (tombol radio, *textbox*, dan lain-lain). Kontrol tata letak digunakan untuk meletakan objek-objek di layar. Ketika pertama membuat aplikasi Windows Phone maka tata letak dasar sebagai penampung akan langsung dibuat berikut panel judul dan panel konten. Selanjutnya untuk penambahan kontrol tata letak yang lain dapat ditambahkan di panel konten.

Terdapat 3 jenis panel yang digunakan untuk menangani tata letak yaitu *Grid*, *StackPanel*, dan *Canvas*. Perlu diperhatikan bahwa setiap halaman hanya memiliki satu jenis panel. Berikut adalah 3 jenis panel pada Windows Phone:

- *StackPanel* merupakan panel yang memposisikan element menjadi 1 baris dan beberapa elemen di setiap halaman diposisikan horizontal atau vertical saja.
- *Grid* merupakan panel yang mendukung tata letak yang rumit. Panel ini memposisikan elemen di baris dan kolom mana saja di setiap halaman.
- *Canvas* memposisikan elemen sebagai absolut kordinat. Jadi setiap elemen di dalam *canvas* dapat diposisikan spesifik sesuai kordinat x dan y.

Kode untuk mengatur jenis panel pada Windows Phone dapat dilihat pada [listing 2.1](#).

Listing 2.1: Kode tata letak grid

```
18 | <Grid x:Name="ContentPanel">
19 | </Grid>
```

1 2.1.3.3 Kontrol Terhadap Teks

2 Kontrol terhadap teks akan menampilkan konten yang memiliki tipe *String*. Terdapat
 3 berbagai macam kontrol terhadap teks di Windows Phone yaitu *TextBlock*, *TextBox* dan
 4 *PasswordBox*. Ketiga macam kontrol tersebut dibedakan berdasarkan tujuannya. Berikut
 5 keterangan, gambar 2.2, dan kode pada *listing 2.2* kontrol teks.

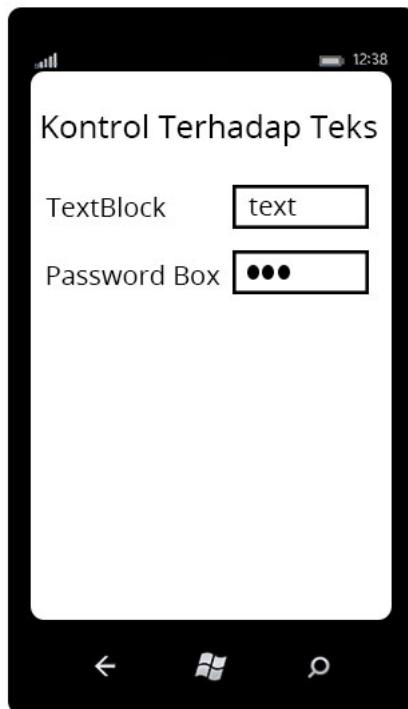
- 6
- *TextBlock* merupakan tempat menaruh potongan teks yang hanya bisa dilihat.

7

 - *TextBox* biasanya digunakan untuk teks masukan yang pendek. Tapi bisa juga dipakai
 8 untuk masukan yang banyak dan beberapa baris.

9

 - *PasswordBox* biasanya digunakan untuk masukan yang bersifat rahasia. Karakter
 10 yang dimasukan langsung disamarkan menjadi bentuk titik.



Gambar 2.2: *TextBlock*, *TextBox* dan *PasswordBox*

Listing 2.2: Kode untuk menampilkan *TextBlock* dan *TextBox*

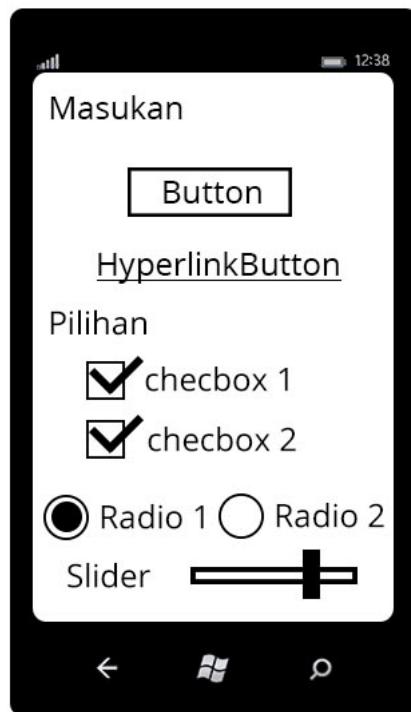
```
11 | <TextBlock x:Name="TextBlock1" Text="TextBlock"/>
12 | <TextBox x:Name="TextBox1" Text="TextBox"/>
```

13 2.1.3.4 Tombol dan Kontrol Pilihan

14 Tombol memungkinkan pengguna untuk berpindah halaman. Sedangkan kontrol pilihan
 15 memudahkan dalam memilih. Berikut keterangan, gambar 2.3, dan kode pada *listing 2.3*
 16 tombol dan kontrol pilihan.

- 17
- *Button* merupakan kontrol yang dipakai pengguna untuk mengaktifkan *event* tekan.

- 1 • *HyperlinkButton* merupakan kontrol yang menampilkan tautan. Jika *HyperlinkButton* ditekan maka akan berpindah ke halaman yang akan dituju.
- 2
- 3 • *CheckBox* merupakan kontrol yang memungkinkan pengguna memilih beberapa item.
- 4
- 5 • *RadioButton* merupakan kontrol yang memungkinkan pengguna memilih satu pilihan dari beberapa pilihan.
- 6
- 7 • *Slider* merupakan kontrol yang memungkinkan pengguna memilih nilai kisaran dari jalur yang sudah disediakan.



Gambar 2.3: Gambar kontrol pada Windows Phone

Listing 2.3: Kode untuk menampilkan *TextBlock*, *TextBox*, dan *PasswordBox*

```
8 | <Button x:Name="find" Content="Button"/>
```

9 2.1.3.5 Kontrol Daftar

10 Kontrol yang dipakai untuk menampilkan daftar dari beberapa *item*. Berikut keterangan,
11 gambar 2.4, dan kode pada listing 2.4 kontrol daftar.

- 12 • *ListBox* akan menampilkan daftar *item*. Daftar ini dapat dipilih dengan cara ditekan.
- 13 • *LongListSelector* dipakai untuk mengelompokan, menampilkan, dan melakukan penggulungan terhadap daftar yang panjang.
- 14



Gambar 2.4: Antarmuka *ListBox*

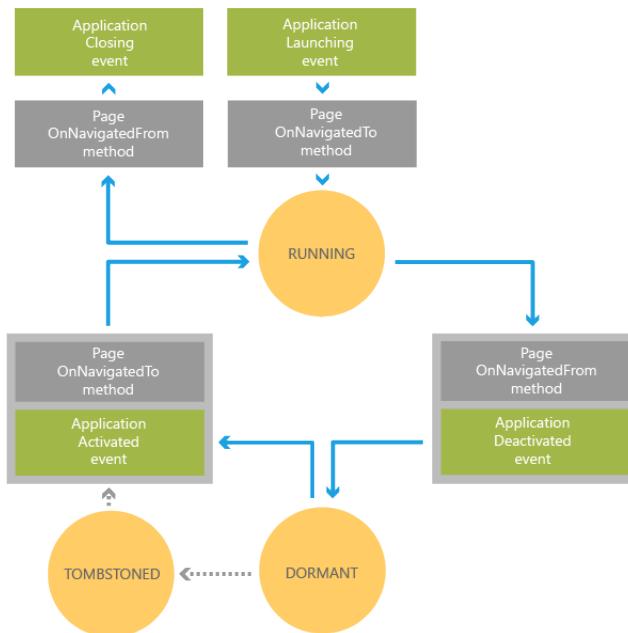
Listing 2.4: Kode untuk menampilkan *listBox*

```

1 <ListBox>
2     <ListBoxItem Content="Item 1" />
3     <ListBoxItem Content="Item 2" />
4     <ListBoxItem Content="Item 3" />
5     <ListBoxItem Content="Item 4" />
6     <ListBoxItem Content="Item 5" />
7 </ListBox>
```

8 2.1.4 Siklus Hidup Aplikasi

9 Siklus hidup aplikasi merupakan waktu mulai dari aplikasi dijalankan sampai dengan
10 aplikasi diberhentikan dari memori. Siklus hidup aplikasi penting diketahui agar pengguna
11 tidak kecewa (dalam aplikasi ini yaitu kecewa karena aplikasi keluar saat layar perangkat
12 dimatikan) menggunakan aplikasi serta memastikan sumber daya tersedia (dalam aplikasi ini
13 yaitu sumber daya GPS). Seringkali pengguna tidak berhati-hati dalam menggunakan apli-
14 kasi, maka dari itu penulis harus memahami kapan aplikasi harus diaktifkan, ditangguhkan,
15 atau bahkan dinonaktifkan karena sudah tidak digunakan. Gambar 2.5 adalah ilustrasi dari
16 siklus hidup pada Windows Phone.



Gambar 2.5: Gambar siklus hidup aplikasi[1]

17 Sesuai gambar 2.5 lingkaran melambangkan keadaan aplikasi, persegi panjang menun-
18 jukan peristiwa aplikasi atau tingkat peristiwa di halaman. Berikut ini adalah keterangan
19 untuk siklus hidup Windows Phone pada gambar 2.5.

20 • *The Launching Event*

21 Merupakan tampilan awal saat aplikasi dipilih yang memberitahukan pengguna bah-

1 wa aplikasi sedang dijalankan. *Event* ini akan dipanggil ketika aplikasi di jalankan
2 pertama kali. *Event* ini akan berjalan di belakang (*background processing*) ketika apli-
3 kasi ditutup sementara atau sedang berada pada keadaan *Dormant* atau *Tombstoned*
4 menjadi *running*.

5 ● *Running*

6 Setelah dijalankan, aplikasi akan masuk ke keadaan *running*. Hal ini akan terus ber-
7 langsung sampai pengguna berpindah ke halaman depan, atau mundur melewati ha-
8 laman utama aplikasi. Aplikasi keluar dari keadaan *running* jika perangkat di kunci.
9 Keadaan *running* masih dapat terjadi saat perangkat di kunci dengan menonaktifkan
10 *idle detection* pada aplikasi.

11 ● *method OnNavigatedFrom*

12 Merupakan *method* yang dipanggil ketika berpindah ke halaman lain aplikasi. Ketika
13 *method* ini dipanggil maka aplikasi akan menyimpan keadaan dari halaman sebelum
14 ditinggalkan. Hal tersebut dibutuhkan agar halaman tersebut bisa dikembalikan ke
15 keadaan sebelum ditinggalkan saat pengguna ingin kembali ke halaman tersebut. Pe-
16 manggilan dilakukan ketika berpindah antara halaman di aplikasi atau ketika berpin-
17 dah aplikasi.

18 ● *The Deactivated Event*

19 *Event* ini akan terjadi ketika pengguna berpindah aplikasi dan menekan tombol "start"
20 atau menjalankan aplikasi lain. Untuk penanganan *deactivated event*, aplikasi harus
21 menyimpan data sebelumnya, sehingga data sebelumnya dapat dikembalikan suatu
22 saat. Windows Phone 8 juga mendukung sistem pengembalian data dengan *State*
23 *Object*. *State Object* akan digunakan untuk menyimpan keadaan aplikasi sebelum
24 aplikasi dinonaktifkan.

25 ● *Dormant*

26 Keadaan ini akan terjadi setelah *deactivated event*. Pada keadaan ini, semua *thread*
27 aplikasi akan dihentikan dan tidak ada proses yang terjadi, tetapi kondisi aplikasi
28 tetap utuh di memori. Tetapi jika sistem operasi membutuhkan memori yang lebih
29 besar maka aplikasi yang dalam keadaan *Dormant* akan menjadi *Tombstone* untuk
30 membebaskan memori.

31 ● *Tombstoned*

32 Aplikasi yang masuk ke keadaan *Tombstoned* akan dihentikan, namun sistem operasi
33 akan menyimpan informasi aplikasi pada saat aplikasi berada di keadaan *deactivated*.

34 ● *The Activated Event*

35 *Event* ini dipanggil ketika aplikasi meninggalkan keadaan *Dormant* atau *Tombstoned*.
36 Operasi ini dilakukan pada latar belakang.

37 ● *The OnNavigatedTo Method*

38 *method* ini dipanggil ketika pengguna berpindah ke halaman yang sebelumnya diting-
39 galkan. *method* ini akan memeriksa keadaan aplikasi dan memulihkannya jika keadaan
40 sebelumnya pernah disimpan.

1 ● *The Closing Event*

2 Event ini akan tercapai ketika pengguna berpindah mundur keluar dari halaman utama.
3 Pada kasus ini, aplikasi akan dihentikan dan tidak ada keadaan yang disimpan.

4 **2.1.5 Peta di Windows Phone**

5 Peta yang dipakai di Windows Phone adalah *Windows Phone Maps*. Windows Phone
6 menawarkan beberapa pilihan dalam tampilan peta mulai dari *cartographic*, pencahayaan
7 dan pandangan. Selain tampilan pada sub bab ini akan dibahas mengenai mendapatkan
8 lokasi, petunjuk arah, *MapPolyline* dan *Pushpin*[1].

9 **2.1.5.1 Penambahan Peta Ke Aplikasi**

10 Untuk penambahan peta pada Windows Phone menggunakan kontrol peta. Kontrol peta
11 merupakan bagian dari *library* Windows Phone. Dengan begitu untuk dapat menggunakan-
12 nya perlu direferensikan. Untuk dapat menggunakannya juga harus ditambah *capability*
13 ID_CAP_MAP. Setelah hal tersebut dilakukan barulah peta dapat ditampilkan. Berikut
14 gambar 3.5, kode XAML pada *listing 2.5*, dan kode program pada *listing 2.6* peta.



Gambar 2.6: Tampilan peta pada Windows Phone

Listing 2.5: Menampilkan peta dengan nama MyMap dari XAML

```
15 | <Controls:Map x:Name="MyMap"/>
```

Listing 2.6: Menampilkan peta dengan nama MyMap dari kode program

```
16 | public void mapFrom()
17 | {
18 |     InitializeComponent();
19 |     Map MyMap = new Map();
20 |     ContentPanel.Children.Add(MyMap);
21 | }
```

22 **2.1.5.2 Tampilan Peta di Windows Phone**

23 Dalam tampilannya ada beberapa hal yang perlu diperhatikan agar pengguna merasa
24 nyaman saat melihat peta di Windows Phone. Beberapa tampilan yang bisa ditampilkan
25 dibuat untuk hal yang berbeda-beda. Berikut akan dibahas menentukan pusat dan tingkat
26 zoom, *cartographic*, warna dan tampilan peta.

- Menentukan pusat peta berarti menentukan titik tengah sebagai pandangan awal di peta. Untuk penentuan titik tengah dibutuhkan 2 nilai yaitu *latitude* dan *longitude*. Sedangkan *zoom* merupakan properti untuk mengatur seberapa dekat atau jauh pandangan yang akan ditampilkan di peta. *Zoom* memiliki nilai yang bisa diatur dari satu hingga dua puluh. Kode untuk mengatur titik tengah peta dan tingkat *zoom* dapat dilihat pada *listing 2.7* dan *listing 2.8*.

Listing 2.7: Mengatur tingkat zoom dari XAML

```
8 | <Controls:Map x:Name="MyMap" ZoomLevel="10" Margin="-25,0,-16,0" />
```

Listing 2.8: Mengatur tingkat zoom dari dari kode program

```
9 | public mapFrom()
10| {
11|     InitializeComponent();
12|     Map MyMap = new Map();
13|
14|     //Mengatur titik tengah peta
15|     MyMap.Center = new GeoCoordinate(47.6097, -122.3331);
16|
17|     //mengatur tingkat zoom
18|     MyMap.ZoomLevel = 10;
19|     ContentPanel.Children.Add(MyMap);
20| }
```

- Cartographic* peta di Windows Phone merupakan cara pandang dalam melihat dan menerjemahkan peta. Terdapat empat jenis *cartographic*, yaitu:

- *Road*: Tampilan normal 2 dimensi.
- *Aerial*: Tampilan peta yang diambil dari foto di udara.
- *Hybrid*: Tampilan Aerial yang digabung dengan jalan dan label.
- *Terrain*: Menampilkan gambar fisik bumi termasuk ketinggian dan air.

Gambar 2.7: *cartographic*

- Mode warna yang disediakan Windows Phone ada dua yaitu terang dan gelap. Secara *default* mode pada peta di Windows Phone adalah terang.
- Tampilan pada Peta di Windows Phone dapat berubah karena hasil diputar, dimiringkan, ditarik, dan diturunkan. Berikut beberapa hal yang dapat diatur sebagai tampilan di peta.
 - *Heading* merupakan representasi dari derajat secara geometri. Derajat ini didefinisikan dalam 0 sampai 360 yang dipakai untuk memutar peta. Contoh, 0 atau 360 ke arah utara, 90 ke arah barat, 180 ke arah selatan, dan 270 derajat ke arah timur.

- 1 – *Pitch* merupakan derajat kemiringan dari peta dari sudut pengguna. Contoh,
 2 *Pitch* = 0 berarti melihat dari atas ke bawah sedangkan *Pitch* = 45 berarti
 3 melihat dari samping ke bawah dengan sudut 45 derajat.

4 2.1.5.3 Pushpin ke Peta

5 *Pushpin* merupakan elemen yang dapat ditempatkan pada peta secara spesifik dan bisa
 6 dipakai untuk interaksi pada peta. Peta tidak mendukung langsung penggunaan *pushpin*
 7 karena merupakan elemen dari *MapOverlay* (bagian/lapisan terpisah dari peta). Untungnya
 8 di Windows Phone memiliki Windows Phone 8 *Toolkit* yang memiliki set objek agar dapat
 9 menggunakan *pushpin* pada peta di Windows Phone. Contoh keluaran *pushpin* dapat dilihat
 10 pada Gambar 2.8 dan kode untuk menampilkannya dapat dilihat pada *listing 2.9*.



Gambar 2.8: Keluaran Toolkit Pushpin pada peta [2]

Listing 2.9: Kode untuk menampilkan pushpin

```
11 MapOverlay overlay = new MapOverlay
12 {
13     GeoCoordinate = map.Center,
14     Content = new Border
15     {
16         BorderBrush = new SolidColorBrush(Colors.FromArgb(120, 255, 0, 0)),
17         Child = new TextBlock(){Text="Pushpin"},
18         BorderThickness = new Thickness(1),
19         Background = new SolidColorBrush(Colors.FromArgb(120, 255, 0, 0)),
20         Width = 80,
21         Height = 60
22     }
23 };
24 MapLayer layer = new MapLayer();
25 layer.Add(overlay);
26
27 map.Layers.Add(layer);
```

28 2.1.5.4 Polyline pada Peta

29 Dalam menentukan arah dibutuhkan dua titik yaitu titik awal dan titik tujuan. Tentu
 30 saja arah tersebut butuh ditandai dengan garis. *Polyline* merupakan tentetan garis lurus
 31 yang saling terhubung satu sama lain. Dengan *polyline* arah pada peta dapat ditandai
 32 dengan warna maupun tebal atau tipisnya garis. Contoh keluaran *polyline* dapat dilihat
 33 pada Gambar 2.9 dan kode untuk menampilkannya dapat dilihat pada *listing 2.10*.

Listing 2.10: Kode untuk menampilkan polyline

```
34 MapPolyline line = new MapPolyline();
35 line.StrokeColor = Colors.Red;
```



Gambar 2.9: Tampilan polyline pada Peta

```

1   line.StrokeThickness = 10;
2   line.Path.Add(new GeoCoordinate(-6.8619546, 107.614441));
3   line.Path.Add(new GeoCoordinate(-6.908693, 107.611185));

```

2.1.5.5 Namespace Control Map

Namespace merupakan nama yang dipakai untuk mengatur kelas-kelas. Windows Phone 8 sudah menyediakan namespace bawaan untuk mengatur peta. Namespace yang disediakan adalah *Maps.Controls*. Namespace ini yang berisi kelas-kelas yang paling sering digunakan untuk mengatur peta pada Windows Phone. Agar dapat menggunakan kelas pada namespace tersebut perlu ditambahkan namespace dan capabilities. Namespace yang harus ditambahkan pada baris awal XAML adalah *Microsoft.Phone.Maps.Controls*. Selanjutnya ada penambahan capabilities *ID_CAP_MAP*. Penambahan capabilities ditambahkan pada *WMAppManifest.xml*.

2.1.5.6 Kelas Map

Merupakan kelas yang mewakili kontrol map.

Berikut properti yang dapat digunakan pada kelas ini.

Nama	Deskripsi
<i>CartographicMode</i>	Mengatur dan mendapatkan tipe dari peta.
<i>Center</i>	Mengatur dan mendapatkan titik tengah pada peta.
<i>ColorMode</i>	Mengatur dan mendapatkan mode warna peta.
<i>Heading</i>	Mengatur dan mendapatkan arah pandang peta.
<i>Height</i>	Mengatur dan mendapatkan tinggi.
<i>LandmarksEnabled</i>	Indikasi apakah bangunan 3D ditampilkan.
<i>Name</i>	Mengatur dan mendapatkan nama untuk identifikasi objek.
<i>PedestrianFeaturesEnabled</i>	Indikasi fitur pejalan kaki ditampilkan.
<i>Pitch</i>	Mengatur dan mendapatkan derajat kemiringan peta.
<i>Tag</i>	Mengatur dan mendapatkan nilai objek.
<i>TileSources</i>	Mendapatkan koleksi lapisan lantai.
<i>Width</i>	Mengatur dan mendapatkan lebar.
<i>ZoomLevel</i>	Mengatur dan mendapatkan tingkat zoom pada peta.

Tabel 2.1: Properti kelas Map

15

Berikut *method* yang dapat digunakan pada kelas ini.

16

- 1 ● *SetView(LocationRectangle)*
- 2 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai wilayah geografis. *method* ini tidak mengembalikan nilai.
- 4 ● *SetView(GeoCoordinate, Double)*
- 5 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah dan tingkat zoom. *method* ini tidak mengembalikan nilai.
- 7 ● *SetView(LocationRectangle, MapAnimationKind)*
- 8 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai region geografis dan animasi. *method* ini tidak mengembalikan nilai.
- 10 ● *SetView(LocationRectangle, Thickness)*
- 11 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai region geografis dengan batas tertentu. *method* ini tidak mengembalikan nilai.
- 13 ● *SetView(GeoCoordinate, Double, MapAnimationKind)*
- 14 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah, tingkat zoom, dan animasi. *method* ini tidak mengembalikan nilai.
- 16 ● *SetView(GeoCoordinate, Double, Double)*
- 17 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah, tingkat zoom, dan *heading*. *method* ini tidak mengembalikan nilai.
- 19 ● *SetView(LocationRectangle, Thickness, MapAnimationKind)*
- 20 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai wilayah geografis dengan batas tertentu, dan animasi. *method* ini tidak mengembalikan nilai.
- 22 ● *SetView(GeoCoordinate, Double, Double, MapAnimationKind)*
- 23 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah, tingkat zoom, *heading*, dan animasi. *method* ini tidak mengembalikan nilai.
- 25 ● *SetView(GeoCoordinate, Double, Double, Double)*
- 26 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah, tingkat zoom, *heading*, *pitch*. *method* ini tidak mengembalikan nilai.
- 28 ● *SetView(GeoCoordinate, Double, Double, Double, MapAnimationKind)*
- 29 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah, tingkat zoom, *heading*, *pitch*, dan animasi. *method* ini tidak mengembalikan nilai.
- 31 ● *UpdateLayout*
- 32 *method* yang memastikan semua posisi objek turunan mengikuti tata letak.

33 **2.1.5.7 Polyline Class**

34 Merupakan kelas yang dipakai untuk menggambarkan garis lurus yang saling terhubung.
35 Kelas ini tergabung ke dalam *namespace Microsoft.Phone.Controls*.

Berikut properti yang dapat digunakan pada kelas ini.

Nama	Deskripsi
<i>Dispatcher</i>	Mendapatkan objek yang terkait.
<i>Path</i>	Mengatur dan mendapatkan kumpulan nilai <i>GeoCoordinates</i> yang membuat <i>polyline</i> .
<i>StrokeColor</i>	Mengatur dan mendapatkan warna garis.
<i>StrokeDashed</i>	Mengatur dan mendapatkan nilai untuk menggambar <i>polyline</i> pustus-putus.
<i>StrokeThickness</i>	Mengatur dan mendapatkan lebar garis untuk menggambar <i>polyline</i> .

Tabel 2.2: Properti *Polyline Class*

- 1
- 2 Berikut *method* yang dapat digunakan pada kelas ini.
- 3 • *CheckAccess*
4 *method* yang menentukan bisa atau tidaknya pemanggilan *thread* untuk mengakses
5 objek.
- 6 • *ClearValue*
7 *method* yang akan membersihkan nilai lokal
- 8 • *Finalize*
9 *method* yang dipakai untuk melakukan pembersihan pada sumber daya yang tidak
10 terpakai sebelum objek dihancurkan.

11 2.1.5.8 *Pushpin Class*

12 Merupakan kelas yang dipakai untuk menggambarkan elemen terpisah diatas peta. Mes-
13 kipun pushpin merupakan bawaan pada peta untuk menunjuk suatu lokasi tetapi *pushpin*
14 dari peta tidak dapat diubah-ubah. *Pushpin* pada Windows Phone 8 dapat dibuat sesuai
15 kebutuhan. Namun ada cara lain dengan menambahkan Windows Phone Toolkit. Windows
16 Phone Toolkit mempunyai komponen untuk menggambar pushpin diatas peta.

17 2.1.6 Lokasi

18 Aplikasi di Windows Phone 8 dapat memanfaatkan lokasi di mana perangkat berada.
19 Aplikasi dapat melacak lokasi sesaat pengguna atau pelacakan selama periode tertentu.
20 Data lokasi perangkat berasal dari berbagai sumber termasuk *Global Positioning System*
21 (*GPS*), *Wireless Fidelity* (*Wi-Fi*), dan jaringan seluler. Ada 2 set API berbeda yang dapat
22 dimanfaatkan di Windows Phone yaitu *Runtime Location API* dan *.NET Location API*.
23 Windows Phone *Runtime Location* memiliki keunggulan fitur yang banyak sedangkan *.NET*
24 *Location* direkomendasikan jika aplikasi ditargetkan pada Windows Phone 7.1 dan Windows
25 Phone 8[1].

26 Hal yang perlu diperhatikan dalam menentukan layanan lokasi adalah penangkap GPS,
27 Wi-Fi, dan jaringan seluler. Perangkat tersebut berfungsi sebagai penyedia data lokasi de-
28 ngan berbagai tingkat akurasi dan konsumsi daya. Perangkat diatas juga berkomunikasi
29 langsung untuk memutuskan sumber mana yang digunakan untuk menentukan lokasi per-
30 angkat berdasarkan ketersediaan data lokasi dan prasyarat yang ditentukan aplikasi. Lapisan

1 diatas penyedia data lokasi tersebut adalah pengelola antarmuka. Aplikasi akan mengunakan
2 antarmuka tersebut untuk memulai dan menghentikan layanan lokasi, mengatur tingkat
3 akurasi, dan menerima data lokasi.

4 Karena pengguna dapat berpindah tempat untuk menuju tempat yang lain, maka pelacakan
5 lokasi harus dilakukan terus menerus. Pelacakan lokasi secara terus menerus ini dapat
6 dilakukan di depan maupun di belakang aplikasi Windows Phone 8. Pelacakan aplikasi di
7 depan akan memungkinkan aplikasi melacak lokasi pengguna sekaligus melakukan perbaruan
8 antarmuka. Jika pelacakan lokasi di belakang aplikasi maka tidak ada perubahan pada antarmuka
9 namun pelacakan dilakukan secara terus menerus. Pelacakan yang terus menerus di
10 belakang aplikasi akan membuat keadaan aplikasi cepat dipulihkan dari keadaan *Dormant*.

11 2.1.6.1 Mendapatkan Posisi Pengguna

12 Di Windows Phone 8 telah ada *GeoCoordinate class* yang dapat digunakan untuk mengetahui posisi pengguna. *Geolocator class* dari *Windows.Devices.Geolocation* akan mengembalikan posisi saat ini. Untuk menggunakan *Geolocator*, perlu menghidupkan *ID_CAP_LOCATION* di *|properties| WMAppManifest.xml*. Dalam mendapatkan posisi perlu diperhatikan status dari GPS karena membutuhkan waktu dari awal pengaktifan hingga mendapatkan lokasi pengguna secara akurat. Untuk lebih jelas mengenai status posisi dapat dilihat pada nilai status dibawah ini.

- 19 • *Ready* : Jika lokasi tersedia.
- 20 • *Initializing* : Jika status penangkap GPS belum memiliki cukup satelit untuk mendapatkan posisi yang akurat.
- 22 • *NoData* : Data lokasi belum tersedia. Status ini muncul jika aplikasi sedang memanggil *GetGeopositionAsync()* atau *register*.
- 24 • *Disable* : Status mengindikasikan tidak diperbolehkannya pengaksesan lokasi.
- 25 • *NotInitialized* : Data lokasi belum tersedia. Status ini muncul jika aplikasi belum memanggil *GetGeopositionAsync()* atau *register*.
- 27 • *NotAvailable* : Jika sensor arah mata angin dan lokasi tidak tersedia.

28 2.1.6.2 Namespace Geolocator

29 *Namespace* merupakan nama yang dipakai untuk mengatur kelas-kelas. Windows Phone
30 8 sudah menyediakan *namespace* bawaan untuk mengakses lokasi. *Namespace* yang disediakan
31 adalah *namespace geolocator*. *Namespace* ini akan mengakses lokasi geografis dari
32 perangkat dan mendukung pelacakan lokasi dari waktu ke waktu. Agar dapat menggunakan
33 kelas pada *namespace* tersebut perlu ditambahkan *namespace* dan *capabilities*. *Namespace* yang
34 harus ditambahkan pada baris awal XAML adalah **Windows.Device.Geolocator**.
35 Selanjutnya ada penambahan *capabilities* *ID_CAP_LOCATION*. Penambahan *capabilities*
36 ditambahkan pada *WMAppManifest.xml*. Kelas yang diatur oleh *namespace geolocator*
37 dapat dilihat pada tabel 2.3[3].

Kelas	Deskripsi
<i>Geocoordinate</i>	Berisi informasi untuk mengidentifikasi lokasi geografis.
<i>Geolocator</i>	Mendukung dalam pengaksesan lokasi perangkat.
<i>Geoposition</i>	Memberikan data lokasi beserta <i>latitude</i> dan <i>longitude</i> atau data alamat.

Tabel 2.3: Kelas pada *Namespace Geolocator*

1 2.1.6.3 *Geocoordinate*

2 Geocoordinate adalah kelas yang menunjukkan lokasi sebagai kordinat geografis. Kelas
 3 ini hanya menyediakan properti yang hanya bisa dibaca. Kelas ini menyediakan properti
 4 yang ditunjukan pada tabel 2.4.

Properti	Deskripsi
<i>Altitude</i>	Ketinggian lokasi dalam satuan meter.
<i>Heading</i>	Arah menghadap perangkat dalam satuan derajat yang relative terhadap mata angin utara.
<i>Latitude</i>	Garis lintang dalam satuan derajat.
<i>Longitude</i>	Garis bujur dalam satuan derajat.
<i>Point</i>	Lokasi dari <i>Geocoordinate</i> .
<i>Speed</i>	Kecepatan dalam satuan meter per detik.

Tabel 2.4: Properti pada *Geocoordinate*

5 2.1.6.4 *Geolocator*

6 *Geolocator* merupakan kelas yang mendukung pengaksesan terhadap lokasi.

7 Berikut *method* yang disediakan *Geolocator*:

- 8 • *public IAsyncOperation<Geoposition> GetGeopositionAsync()*
 9 Operator await diatas dimaksudkan untuk meminta posisi lokasi terus menerus sampai
 10 selesai dan menunda tugas yang lain.
 11 *GetGeopositionAsync()* merupakan bawaan kelas *Geolocator* akan meminta data lokasi
 12 dan menanganinya sampai selesai. Kembalian dari *GetGeopositionAsync()* adalah
 13 objek *Geoposition*.

14 Berikut Properti yang disediakan kelas Geolocator:

- 15 • *public PositionStatus LocationStatus { get; }*
 16 Merupakan properti dari kelas *geolocator* untuk mendapatkan status posisi dengan
 17 mengembalikan kelas *PositionStatus*. Status pada kelas *PositionStatus* adalah *Ready*,
 18 *Initializing*, *NoData*, *Disable*, *NotInitialized*, dan *NotAvailable*.

- 19 • *public PositionAccuracy DesiredAccuracy { get; set; }*
 20 Properti yang digunakan untuk mengatur dan mendapatkan tingkat akurasi. Untuk
 21 tingkat akurasi dapat dipilih tingkat *High* untuk tingkat akurasi tinggi dan dipilih
 22 tingkat *Default* untuk menghemat daya. Keluaran dari properti ini adalah tipe data
PositionAccuracy.

- 1 • *public Nullable<uint> DesiredAccuracyInMeters { get; set; }*

2 Sama seperti properti *DesiredAccuracy* diatas tetapi dalam satuan meter. Keluaran
3 dari properti ini adalah tipe data *uint*.

- 4 • *public uint ReportInterval { get; set; }*

5 Merupakan properti untuk mendapatkan selang waktu pembaruan lokasi. Properti ini
6 mengeluarkan tipe data unit.

7 **2.1.6.5 Geoposition**

8 *Geoposition* merupakan kelas yang memuat lokasi (*latitude* dan *longitude*). Berikut
9 Properti yang disediakan kelas *Geoposition*:

- 10 • *public CivicAddress CivicAddress { get; }*

11 Data alamat sipil yang terkait dengan lokasi geografis.

- 12 • *public Geocoordinate Coordinate { get; }*

13 Data latitude dan longitude yang terkait lokasi geografis.

14 **2.1.7 Memanfaatkan Sumber Data**

15 Hal yang penting dari sebuah aplikasi adalah informasi. Windows Phone 8 memiliki
16 kemampuan dalam menghubungkan aplikasi dengan sumber data lainnya. Memanfaatkan
17 sumber data ada dua cara yaitu yang lokal atau berada di perangkat dan *web service*. *Web*
18 *Service* merupakan *method* komunikasi antara dua perangkat melalui jaringan.

19 Sebelum data dapat dikirim antar perangkat perlu dilakukan *Serialization*. *Serialization*
20 disini merupakan proses mentransformasikan objek ke format yang bisa dengan mudah di-
21 kirim melewati jaringan atau disimpan di database. Formatnya disini berupa string yang
22 direpresentasikan sebagai objek di XML atau JSON(Javascript Object Notation). Ada beberapa
23 objek yang dapat melakukan serialisasi, tetapi yang akan dibahas penulis disini hanya
24 serialisasi JSON[2].

25 Banyak *web service* yang mengembalikan data dalam format JSON. JSON memiliki
26 format *data-interchange* yang ringan [4]. Karena hal tersebut JSON mudah diurai dan di-
27 hasilkan oleh mesin. Kurung kurawal mengindikasikan objek, kurung siku berarti array, dan
28 properti berupa nama dan nilai pasangan yang dipisahkan oleh titik dua. JSON format
29 memiliki ukuran data yang kecil dan baik untuk penggunaan perangkat bergerak. Untuk
30 contoh format JSON dapat dilihat di bagian Kiri API pada Bab dua ini karena Kiri API
31 menggunakan format JSON. Serialisasi menggunakan *DataContractJsonSerializer* membuat
32 serialisasi mudah untuk menerjemahkan form String JSON ke objek yang dapat langsung
33 digunakan. *DataContractJsonSerializer* memakai *WriteObject()* untuk serialisasi and *Read-
34 ableObject()* untuk de-serialisasi.

35 **2.1.7.1 HttpClient**

36 Merupakan Kelas yang dipakai untuk mengirim permintaan HTTP dan menerima kembali
37 HTTP dari *Uniform Resource Identifier*(URI) yang dapat diidentifikasi. *Uniform Re-
38 source Identifier*(URI) merupakan urutan kompak karakter yang mengidentifikasi sumber
39 daya abstrak dan fisik [5]. Berikut *method* yang disediakan kelas *HttpClient*.

1 ● *DeleteAsync(Uri)*

2 *method* yang dipakai untuk mengirimkan permintaan DELETE ke URI yang spesifik
3 sebagai operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memung-
4 kinkan aplikasi untuk melanjutkan pekerjaan selagi *method* ini dipanggil². *method* ini
5 membutuhkan parameter URI sebagai tujuan dari permintaan. Sedangkan kembali-
6 annya berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek
7 tersebut memiliki 2 parameter yaitu hasil berupa pesan kembalian dari http dan ke-
8 majuan dari data yang dikirim.

9 ● *GetAsync(Uri)*

10 *method* yang dipakai untuk mengirimkan permintaan GET ke URI yang spesifik seba-
11 gi operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan
12 aplikasi untuk melanjutkan pekerjaan selagi *method* ini dipanggil². *method* ini mem-
13 butuhkan parameter URI sebagai tujuan dari permintaan. Sedangkan kembaliannya
14 berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut
15 memiliki 2 parameter yaitu hasil berupa pesan kembalian dari http dan kemajuan dari
16 data yang dikirim.

17 ● *GetAsync(Uri,HttpCompletionOption)*

18 *method* yang dipakai untuk mengirimkan permintaan GET ke URI yang spesifik seba-
19 gi operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan
20 aplikasi untuk melanjutkan pekerjaan selagi *method* ini dipanggil². *method* ini mem-
21 butuhkan parameter URI sebagai tujuan dari permintaan dan nilai tambahan yang
22 dimaksukan sebagai indikasi operasi dianggap selesai. Sedangkan kembaliannya ber-
23 upa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut
24 memiliki 2 parameter yaitu hasil berupa pesan kembalian dari http dan kemajuan dari
25 data yang dikirim.

26 ● *GetBufferAsync(Uri)*

27 *method* yang dipakai untuk mengirimkan permintaan GET ke URI yang spesifik seba-
28 gi operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan
29 aplikasi untuk melanjutkan pekerjaan selagi *method* ini dipanggil². *method* ini mem-
30 butuhkan parameter URI sebagai tujuan dari permintaan. Sedangkan kembaliannya ber-
31 upa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut
32 memiliki 2 parameter yaitu hasil berupa pesan kembalian yang dikirimkan secara
33 buffer(disimpan dalam memori) dan kemajuan dari data yang dikirim.

34 ● *GetInputStreamAsync(Uri)*

35 *method* yang dipakai untuk mengirimkan permintaan GET ke URI yang spesifik seba-
36 gi operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan
37 aplikasi untuk melanjutkan pekerjaan selagi *method* ini dipanggil². *method* ini mem-
38 butuhkan parameter URI sebagai tujuan dari permintaan. Sedangkan kembaliannya ber-
39 upa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut
40 memiliki 2 parameter yaitu hasil berupa pesan kembalian yang dikirimkan secara
41 stream(langsung sesuai waktu) dan kemajuan dari data yang dikirim.

42 ● *GetStringAsync(Uri)*

1 *method* yang dipakai untuk mengirimkan permintaan *GET* ke URI yang spesifik sebagai operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan aplikasi untuk melanjutkan pekerjaan selagi *method* ini dipanggil². *method* ini membutuhkan parameter URI sebagai tujuan dari permintaan. Sedangkan kembalinya berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut memiliki 2 parameter yaitu hasil berupa pesan kembalian dalam bentuk string dan kemajuan dari data yang dikirim.

8 • *PostAsync(Uri)*

9 *method* yang dipakai untuk mengirimkan permintaan *POST* ke URI yang spesifik sebagai operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan aplikasi untuk melanjutkan pekerjaan selagi *method* ini dipanggil². *method* ini membutuhkan parameter URI sebagai tujuan dari permintaan. Sedangkan kembalinya berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut memiliki 2 parameter yaitu hasil berupa pesan kembalian dari http dan kemajuan dari data yang dikirim.

16 • *SendRequestAsync(HttpRequestMessage)*

17 *method* yang dipakai untuk mengirimkan permintaan HTTP sebagai operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan aplikasi untuk melanjutkan pekerjaan selagi *method* ini dipanggil². *method* ini membutuhkan parameter pesan dari permintaan. Sedangkan kembalinya berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut memiliki 2 parameter yaitu hasil berupa pesan kembalian dari http dan kemajuan dari data yang dikirim.

23 • *SendRequestAsync(HttpRequestMessage, HttpCompletionOption)*

24 *method* yang dipakai untuk mengirimkan permintaan HTTP sebagai operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan aplikasi untuk melanjutkan pekerjaan selagi *method* ini dipanggil². *method* ini membutuhkan parameter pesan dari permintaan dan nilai tambahan yang dimaksukan sebagai indikasi operasi dianggap selesai. Sedangkan kembalinya berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut memiliki 2 parameter yaitu hasil berupa pesan kembalian dari http dan kemajuan dari data yang dikirim.

31 **2.1.7.2 Json.NET**

32 Json.NET merupakan *library* untuk melakukan *serialize* dan *deserialize* terhadap objek dalam .NET. *Library* ini memiliki 2 kelas yaitu kelas JsonConvert dan kelas JsonSerializer. Berikut merupakan keterangan dari dua kelas tersebut.

35 •

36 **2.1.8 Thread**

37 Thread merupakan entitas dalam suatu proses yang dapat dijalankan untuk eksekusi sebuah operasi.

²<http://msdn.microsoft.com/en-us/library/ms734701%28v=vs.110%29.aspx>

1 **2.1.8.1 *BackgroundWorker***

2 *BackgroundWorker* merupakan kelas dari Windows Phone yang dapat mengeksekusi
3 operasi dalam thread terpisah. Berikut *method* yang disediakan kelas *BackgroundWorker*.

- 4 • *RunWorkerAsync()*

5 *method* yang digunakan untuk memulai eksekusi operasi di latar belakang.

6 Berikut *event* yang disediakan kelas *BackgroundWorker*.

- 7 • *DoWork*

8 *event* yang terjadi ketika *method RunWorkerAsync()* dipanggil.

- 9 • *RunWorkerCompleted*

10 *event* yang terjadi ketika operasi di latar belakang selesai dilakukan, dibatalkan, atau
11 mencapai *exception*.

12 **2.2 Kiri API**

13 API atau *Application Programming Interface* merupakan aturan yang dikodekan secara
14 spesifik yang dapat digunakan untuk komunikasi antar aplikasi. Jadi API disini memfasilitasi
15 untuk pemanggilan fungsi-fungsi tertentu diluar aplikasi itu sendiri. Pemanfaatan Kiri API
16 dilakukan dengan menggunakan JSON atau *JavaScript Object Notation* format.

17 Pemanfaatan Kiri API dengan melakukan permintaan dengan parameter *POST* atau *GET*
18 lalu Kiri akan mengembalikan hasil dalam format JSON. Permintaan tersebut dikirimkan
19 ke URL atau *Uniform Resource Locator*. Berikut URL yang disediakan Kiri Api.

- 20 • <http://preview.kiri.travel/handle.php>

21 Merupakan URL untuk uji coba. Untuk kemampuannya juga menurut dokumentasi
22 Kiri API masih tidak stabil.

- 23 • <http://kiri.travel/handle.php>

24 Merupakan URL produksi. Ini merupakan URL yang direkomendasikan untuk mena-
25 ngani permintaan pengguna.

26 Untuk setiap permintaan membutuhkan *API key* yang didapat dengan mendaftar[6]. Peng-
27 gunaan API memungkinkan pengaksesan di mana saja dengan menggunakan koneksi inter-
28 net. Pada sub bab 2.2.1 sampai sub bab 2.2.3 penulis akan membahas beberapa layanan
29 Kiri API.

30 Berikut langkah-langkah untuk mendapatkan *API key*.

- 31 • Masuk ke situs dev.kiri.travel.

- 32 • Register dengan memasukan alamat email, nama, dan nama perusahaan.

- 33 • Password akan dikirimkan ke alamat email. Tentunya password akan dibuat otomatis
34 oleh pihak Kiri.

- 35 • Login dengan menggunakan password yang dikirim ke alamat email.

- 36 • Setelah berhasil login, di menu utama pilih *API Keys Managements*.

- 1 • Pilih tombol Add lalu masukan deskripsi penggunaan *API key*.
- 2 • *API key* didapat dan dapat digunakan.

3 2.2.1 Web Service Penentuan Rute

4 *Web service* penentuan rute merupakan layanan Kiri API yang digunakan untuk men-
 5 dapatkan langkah perjalanan dari lokasi asal ke lokasi tujuan. Parameter dan keterangan
 6 untuk layanan ini dapat dilihat pada tabel 2.5.

<i>version</i>	2	Memberitahukan bahwa layanan yang dipakai adalah protokol veris 2
<i>mode</i>	"findroute"	Mengintruksikan layanan untuk mencari rute
<i>locale</i>	"en" or "id"	Bahasa yang digunakan untuk balasan
<i>start</i>	lat,lng	Titik awal <i>latitude</i> dan <i>longitude</i>
<i>finish</i>	lat,lng	Titik akhir <i>latitude</i> dan <i>longitude</i>
<i>presentation</i>	"mobile" or "desktop"	Menentukan tipe presentasi untuk keluaran. Contoh, jika tipe presentasi "mobile", maka link "tel:" akan ditambahkan di hasil.
<i>apikey</i>	16-digit hexadecimals	<i>API key</i> yang digunakan

Tabel 2.5: Tabel parameter layanan penentuan rute

7 Format kembalian layanan penentuan rute dapat dilihat pada *listing 2.11*:

Listing 2.11: Kode kembalian pencarian rute

```

8 {
9   "status": "ok" or "error"
10  "routingresults": [
11    {
12      "steps": [
13        [
14          "walk" or "none" or others,
15          "walk" or vehicle_id or "none",
16          ["lat_1,lon_1", "lan_2,lon_2", ... "lat_n,lon_n"],
17          "human readable description, dependant on locale",
18          URL for ticket booking or null (future)
19        ],
20        [
21          "walk" or "none" or others,
22          "walk" or vehicle_id or "none",
23          ["lat_1,lon_1", "lan_2,lon_2", ... "lat_n,lon_n"],
24          "human readable description, dependant on locale",
25          URL for ticket booking or null (future)
26        ],
27        ...
28      ],
29      "traveltimes": any text string, null if and only if route is not found.
30    },
31    {
32      "steps": [ ... ],
33      "traveltimes": "..."
34    },
35    {
36      "steps": [ ... ],
37      "traveltimes": "..."
38    },
39  ...
40 }

```

41 Berikut maksud dari *listing 2.11*:
 42 Ketika pencarian rute sukses dilakukan maka status akan memberitahukan "ok" seperti
 43 di baris 2. Selanjutnya setiap langkah dari posisi awal ke posisi tujuan akan ditampung
 44 di elemen *array* untuk menampung langkah. Berikut keterangan dari setiap *array* yang
 45 menampung langkah.

- Indeks ke 0 atau baris 7 pada *listing 2.11* dapat berisi "walk" atau "none" atau "others". Baris tersebut berarti jika "walk" untuk berjalan kaki, "none" jika rute tidak ditemukan dan "others" untuk menggunakan kendaraan.
- Indeks ke 1 atau baris 8 pada *listing 2.11* merupakan detail dari indeks 0. Artinya jika indeks 0 menyatakan "walk" berarti indeks 1 harus "walk", "none" berarti indeks 1 harus "none", dan selain itu menyatakan id kendaraan yang mana bisa dipakai untuk ditampilkan gambarnya.
- Indeks ke 2 atau baris 9 pada *listing 2.11* adalah deretan nilai tipe *String* yang berisi jalur dalam format "lat,lng". Maksud dari "lat,lng" disini adalah titik awal dan titik akhir dari setiap jalur yang dilewati.
- Indeks ke 3 atau baris 10 pada *listing 2.11* berisi bentuk yang akan ditampilkan kepada pengguna. Informasi yang disampaikan dapat berupa:
 - %fromicon = untuk menunjukkan ikon "from". Biasanya untuk mode presentasi di perangkat bergerak.
 - %toicon = untuk menunjukkan ikon "to". Biasanya untuk mode presentasi di perangkat bergerak.
- Indeks ke 4 atau baris 11 pada *listing 2.11* berisi URL untuk pemesanan tiket jika tersedia. Jika tidak tersedia akan bernilai *null*.

Kiri telah menyediakan gambar untuk setiap angkutan umum. Gambar tersebut dapat diakses di URL:

- [http://kiri.travel/images/means/\[means\]/\[means_details\].png](http://kiri.travel/images/means/[means]/[means_details].png)
- [http://kiri.travel/images/means/\[means\]/baloon/\[means_details\].png](http://kiri.travel/images/means/[means]/baloon/[means_details].png)

Nilai [means] dapat diambil dari indeks 0 nilai kembalian dan nilai [means_details] dapat diambil dari indeks 1 nilai kembalian.

2.2.2 Web Service Pencarian Lokasi

Merupakan layanan Kiri API yang digunakan untuk mencari lokasi beserta kordinat *latitude* dan *longitude*. Parameter dan keterangan untuk layanan ini dapat dilihat pada tabel *2.6*.

<i>version</i>	2	Memberitahukan bahwa layanan yang dipakai adalah protokol versi 2
<i>mode</i>	"searchplace"	Mengintruksikan layanan untuk mencari tempat
<i>region</i>	"cgk" or "bdo" or "sub"	Kota yang akan dicari tempatnya
<i>querystring</i>	teks apa saja dengan minimum text satu karakter	<i>Query string</i> yang akan dicari menggunakan layanan ini
<i>apikey</i>	16-digit heksadesimal	<i>API key</i> yang digunakan

Tabel 2.6: Tabel parameter layanan pencarian lokasi

Format kembalian dari layanan pencarian lokasi dapat dilihat pada *listing 2.12*.

Listing 2.12: Kode kembalian pencarian lokasi

```

1  {
2      "status": "ok" or "error"
3      "searchresult": [
4          {
5              "placename": "place_name"
6              "location": "lat,lon"
7          },
8          {
9              "placename": "place_name"
10             "location": "lat,lon"
11         },
12         ...
13     ]
14     "attributions": [
15         "attribution_1", "attribution_2", ...
16     ]
17 }

```

18 Berikut maksud dari *listing 2.12*:

19 Ketika pencarian lokasi sukses dilakukan maka status akan memberitahukan "ok" seperti di
20 baris 2. Selanjutnya akan ditampilkan hasil dari lokasi yang ada beserta atributnya. Berikut
21 keterangan dari format dari pencarian lokasi:

22 • "searchresult" (pada baris 4 sampai 7, 8 sampai 11, dan seterusnya) berisi array dari
23 tempat:

- 24 – placename: nama tempat
- 25 – location: latitude dan longitude dari tempat

26 • "attributions" berisi kumpulan nilai yang berisikan atribut tambahan untuk dimun-
27 culkan.

28 2.2.3 Web Service Menemukan Transportasi Terdekat

29 Merupakan Kiri API yang digunakan untuk menemukan rute transportasi terdekat sesuai
30 titik yang diinginkan pengguna. Parameter dan keterangan untuk layanan ini dapat dilihat
31 pada tabel 2.7.

<i>version</i>	2	Memberitahukan bahwa layanan yang dipakai adalah protokol versi 2
<i>mode</i>	"nearbytransports"	Menginstruksikan layanan untuk mencari rute transportasi terdekat
<i>start</i>	<i>latitude</i> dan <i>longitude</i>	Kota yang akan dicari tempatnya
<i>apikey</i>	16-digit hexadesimal	<i>API key</i> yang digunakan

Tabel 2.7: Tabel parameter layanan menemukan transportasi terdekat

32 Format kembalian layanan menemukan transportasi terdekat dapat dilihat pada *lis-*
33 *ting 2.13*.

Listing 2.13: Kode kembalian menemukan lokasi terdekat

```

34 {
35     "status": "ok" or "error"
36     "nearbytransports": [
37         [
38             "walk" or "none" or others,
39             "walk" or vehicle_id or "none",
40             text string,
41             decimal value
42         ],
43         [

```

```
1   "walk" or "none" or others,
2   "walk" or vehicle_id or "none",
3   text string,
4   decimal value
5   ],
6   ...
7 }
8 }
```

9 Berikut maksud dari *listing 2.13*:

10 Ketika pencarian rute sukses dilakukan maka status akan memberitahukan "ok" seperti di
11 baris 2. Selanjutnya akan diberikan array yang berisi transportasi terdekat yang diurutkan
12 dari yang terdekat ke yang terjauh. Berikut keterangan dari setiap array tersebut:

- 13 • Indeks ke 0 atau baris 5 pada *listing 2.13* dapat berisi "walk" atau "none" atau
14 "others". Artinya jika "walk" berarti berjalan kaki, "none" jika rute tidak ditemukan
15 dan "others" berarti menggunakan kendaraan.
- 16 • Indeks ke 1 atau baris 6 pada *listing 2.13* merupakan detail dari indeks 0. Artinya jika
17 indeks 0 "walk" berarti indeks 1 harus "walk", "none" berarti indeks 1 harus "none"
18 dan selain itu menyatakan id kendaraan yang mana bisa dipakai untuk ditampilkan
19 gambarnya.
- 20 • Indeks ke 2 atau baris 7 pada *listing 2.13* berisi nama kendaraan.
- 21 • Indeks ke 3 atau baris 8 pada *listing 2.13* berisi jarak dengan satuan kilometer.

1

BAB 3

2

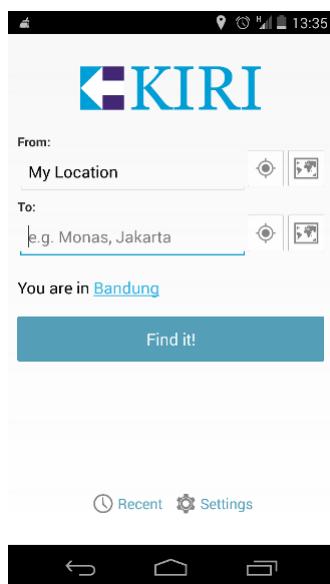
ANALISIS

- 3 Pada bab ini akan dibahas mengenai analisis aplikasi sejenis, analisis kebutuhan aplikasi,
4 analisi kontrol yang dipakai, analisis terhadap siklus hidup aplikasi, analisis peta, analisis
5 pemanfaatan sumber data, analisis Kiri API, diagram *Use Case*, dan diagram kelas.

6 **3.1 Analisis Aplikasi Sejenis**

7 Aplikasi sejenis yang penulis temui bernama Public Transport¹. Namun aplikasi Public
8 Transport tersebut hanya dapat dijalankan di sistem aplikasi android. Aplikasi Public
9 Transport ini memanfaatkan Kiri API. Aplikasi tersebut penggunaannya sederhana. Di ha-
10 laman awal pengguna dapat mengetikan lokasi awal dan tujuan. Selain dengan mengetik
11 pengguna juga dapat menunjuk lokasi pada peta. Setelah lokasi dipilih sistem akan me-
12 mastikan dengan memberi daftar nama jalan dan tempat terkait. Jika sudah memilih maka
13 sistem akan mengeluarkan hasil pencarian rute.

14 Berikut adalah tampilan dari aplikasi Public Transport (Gambar 3.1 sampai 3.5):



Gambar 3.1: Tampilan utama aplikasi Public Transport

15 Gambar 3.1 menunjukkan halaman utama aplikasi Public Transport. Di halaman ini
16 pengguna dapat memasukan lokasi asal dan lokasi tujuan. Cara memasukan lokasi ada 2
17 macam yaitu dengan mengetik dan menunjuk pada peta dengan mengetuk tombol peta.

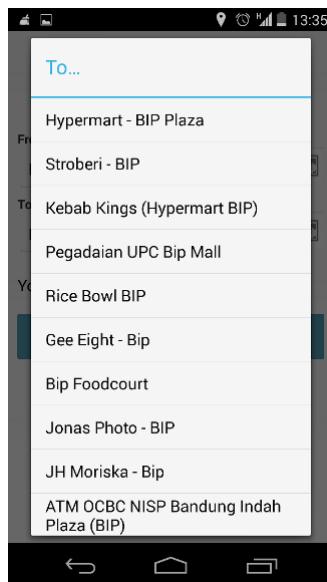
¹<https://play.google.com/store/apps/details?id=travel.kiri.smarttransportapp>

- 1 Bila pengguna ingin menunjuk lokasi pengguna berada dapat dilakukan dengan mengetuk tombol kordinat. Tersedia juga pelihan kota yang dapat dipilih oleh pengguna.



Gambar 3.2: Menunjuk lokasi pada peta

- 3 Gambar 3.2 jika pengguna sudah mengetahui lokasi namun tidak tahu nama lokasi.
- 4 Pada halaman ini pengguna diarahkan untuk menemukan lokasi pada peta dan mengetuk lokasi tersebut untuk memilihnya.



Gambar 3.3: Memberikan daftar nama tempat dan nama jalan terkait

- 6 Pada gambar 3.3 pengguna dapat memilih nama tempat terkait. Pemilihan didasarkan sesuai masukan pengguna untuk memastikan tempat asal maupun tempat tujuan. Jika nama tempat sudah jelas maka tidak akan ada halaman ini.

- 9 Pada gambar 3.4 menampilkan daftar rute kendaraan umum yang harus dinaiki beserta gambar untuk mempermudah pengguna. Selain itu disertakan juga jarak dan perkiraan waktu sampai di lokasi tujuan.

- 12 Pada gambar 3.5 menampilkan rute kendaraan umum dan jalur yang harus dilalui pada



Gambar 3.4: Tampilan rute kendaraan umum dalam bentuk daftar



Gambar 3.5: Tampilan rute kendaraan umum di peta

- 1 peta. Dengan cara ini pengguna dapat mengetahui posisi dan jalur yang harus dilalui.

3.2 Analisis Aplikasi

Applikasi akan dibuat menggunakan bahasa pemrograman C#. Applikasi yang digunakan untuk membangun Aplikasi Pencari Rute Kendaraan Umum untuk Windows Phone adalah Visual Studio Express 2012. Pada sub bab ini akan dibahas kebutuhan aplikasi, analisis kontrol yang dipakai, analisis terhadap siklus hidup aplikasi, analisis peta, analisis pemanfaatan sumber data, analisa Kiri API, diagram *use case*, dan diagram kelas dari aplikasi yang akan dibangun.

3.2.1 Kebutuhan Aplikasi

Dari hasil observasi penulis dalam menentukan lokasi asal dan lokasi tujuan ada dua cara. Kedua cara tersebut yaitu dengan menulis alamat atau tempat dan dengan menunjuk pada peta. Cara menuliskan alamat atau tempat yaitu dengan menuliskan alamat atau tempat pada tempat yang disediakan untuk asal dan tujuan. Cara menunjuk pada peta yaitu dengan mengetuk layar di posisi yang diinginkan. Kedua hal tersebut pada dasarnya sama saja tetapi ada faktor kemudahan pengguna dalam pemakaiannya. Jadi penulis menyediakan dua cara tersebut pada aplikasi yang penulis buat agar pengguna dapat memilih salah satunya.

Pada saat menuliskan lokasi atau tempat atau menunjuk langsung pada peta mungkin saja terjadi kesalahan. Kesalahan tersebut bisa saja disebabkan salah pengetikan atau nama tempat yang tidak ada. Maka dari itu dibutuhkan pemeriksaan terhadap masukan pengguna. Pemeriksaan tersebut dilakukan setelah pengguna memulai pencarian dengan menekan tombol "FIND".

Untuk hasil keluaran ada dua tipe seperti aplikasi peta lainnya. Kedua tipe tersebut adalah bentuk daftar dan bentuk peta. Bentuk daftar memudahkan dalam melihat tiap langkah rute. bentuk daftar memudahkan pengguna dalam melihat arah dan posisi lingkungan pada rute yang dipilih.

Aplikasi yang penulis bangun didasarkan pada kebutuhan sebagai berikut.

1. Pengguna dapat memasukan lokasi asal dan lokasi tujuan pada *TextBox* yang disediakan atau menunjuk langsung lokasi pada peta.
2. Mendapatkan lokasi terkait menurut lokasi yang dimasukan pengguna.
3. Menampilkan hasil rute angkutan umum dari lokasi asal ke lokasi tujuan.

3.2.2 Analisis Kontrol yang Dipakai

Dari kebutuhan yang telah disebutkan diatas penulis menyadari pentingnya kontrol yang harus dipakai. Kontrol yang dimaksud termasuk tata letak, teks, pilihan, dan daftar. Kebutuhan akan kontrol penting bukan hanya untuk kebutuhan tapi memudahkan pengguna.

Untuk kontrol tata letak penulis membayangkan pengaturan yang tertata rapih dan beberapa elemen dalam satu baris atau kolom. Tetapi juga penulis tidak mengharapkan penggunaan kontrol tata letak yang rumit. Dari hasil pengamatan penulis kontrol tata letak yang cocok adalah Grid. Kontrol tata letak ini penulis pilih karena mudah diposisikan sesuai baris dan kolom. Selain itu tampilan Grid akan menyesuaikan jika layar diputar dari posisi pemandangan ke posisi potret dan sebaliknya.

1 Kontrol terhadap teks juga diperlukan untuk aplikasi. Kebutuhan yang diperlukan adalah mengeluarkan potongan teks yang dapat dibaca dan tempat pengguna memasukan teks.
2 Untuk mengeluarkan teks yang dapat dilihat penulis akan menggunakan *TextBlock*. Te-
3 xtBlock digunakan untuk menampilkan tulisan "from" dan "to" pada halaman utama apli-
4 kasi. Untuk masukan pengguna terhadap aplikasi penulis akan menyediakan *TextBox* sebagai
5 tempat teks. *TextBox* digunakan sebagai masukan untuk lokasi asal dan lokasi tujuan.

6 Suatu aplikasi tentunya tidak hanya mempunyai satu halaman. Sama hal dengan aplikasi
7 yang penulis buat memiliki beberapa halaman yang mempunyai tugas berbeda. Karena hal
8 tersebut dibutuhkan kontrol untuk berpindah dari satu halaman ke halaman lain. Kontrol
9 yang dibutuhkan yaitu kontrol tombol. Kontrol tombol akan mengeksekusi *event click* yang
10 memungkinkan pindah halaman dan melakukan perintah. Kontrol tombol akan penulis
11 gunakan untuk berpindah ke halaman peta, menemukan lokasi pengguna, dan pencarian
12 rute. Pada Gambar ?? terdapat 5 tombol yaitu tombol map pada bagian from, tombol here
13 pada bagian from, tombol map pada bagian to, tombol here pada bagian to, dan tombol
14 find. Berikut kegunaan dari tombol-tombol tersebut.

- 16 • Tombol map pada bagian from
17 Tombol untuk berpindah dari halaman utama menuju halaman peta. Pada halaman
18 peta pengguna dapat menunjuk lokasi asal dan kembali lagi ke halaman utama. Saat
19 kembali ke halaman utama lokasi yang dipilih akan disimpan dan pada *TextBox* bagian
20 from akan tertulis "lokasi dari peta".
 - 21 • Tombol map pada bagian from
22 Tombol untuk mencari lokasi pengguna. Setelah tombol ini di tekan maka lokasi
23 pengguna akan disimpan dan pada bagian *TextBox* bagian from akan tertulis "here".
 - 24 • Tombol map pada bagian to
25 Tombol untuk berpindah dari halaman utama menuju halaman peta. Pada halaman
26 peta pengguna dapat menunjuk lokasi tujuan dan kembali lagi ke halaman utama.
27 Saat kembali ke halaman utama lokasi yang dipilih akan disimpan dan pada *TextBox*
28 bagian to akan tertulis "lokasi dari peta".
 - 29 • Tombol map pada bagian to
30 Tombol untuk mencari lokasi pengguna. Setelah tombol ini di tekan maka lokasi
31 pengguna akan disimpan dan pada bagian *TextBox* bagian to akan tertulis "here".
 - 32 • Tombol find Tombol ini akan mencari rute angkutan umum dan menampilkannya
33 pada halaman peta.
- 34 Pada aplikasi ini penulis akan menampilkan daftar tempat dan daftar rute angkutan
35 umum yang dipakai. Bentuk daftar digunakan penulis karena hasil tempat dan rute ang-
36 kutan umum akan banyak. Bentuk daftar yang dapat dipakai di Windows Phone adalah
37 menggunakan *ListBox*. *ListBox* akan menampilkan daftar tempat dan daftar rute satu per
38 satu menurun ke bawah.

39 **3.2.3 Analisis Terhadap Siklus Hidup Aplikasi**

40 Aplikasi pada Windows Phone memiliki siklus hidup yang dijelaskan pada bab 2.1.4.
41 Pengaturan aplikasi ini diatur sesuai konfigurasi awal sistem operasi Windows Phone. Tetapi

1 pengaturan ini dapat diatur sesuai kebutuhan aplikasi. Karena di aplikasi ini terdapat
2 keadaan yang berbeda dengan konfigurasi awal sistem operasi Windows Phone maka perlu
3 dilakukan pengaturan ulang siklus hidup.

4 Saat aplikasi dijalankan, pengguna memasukan lokasi asal dan lokasi tujuan. Setelah
5 memasukan lokasi pengguna akan mencari rute. Ketika rute berhasil ditemukan aplikasi akan
6 berada di keadaan Running. Tetapi ada kemungkinan pengguna berpindah aplikasi atau
7 mematikan layar untuk menghemat daya. Dalam kasus tersebut sistem operasi Windows
8 Phone akan menganggap aplikasi tidak aktif dan aplikasi akan masuk pada keadaan *dormant*.
9 Untuk menangani kasus tersebut maka penulis harus menyimpan keadaan dan informasi
10 saat sebelum aplikasi menjadi tidak aktif. Penanganan yang penulis akan lakukan adalah
11 menggunakan *method OnNavigatedFrom()*. Dengan *method* tersebut keadaan aplikasi akan
12 disimpan di memori.

13 Pada saat aplikasi masuk keadaan *Dormant* semua *thread* dan proses akan dihentikan.
14 Pada saat tersebut juga GPS Windows Phone akan terhenti dan tidak akan mengetahui
15 posisi pengguna. GPS akan kembali aktif mengetahui posisi pengguna jika pengguna masuk
16 ke aplikasi dan tentunya membutuhkan waktu untuk pelacakan lokasi. Tetapi Windows
17 Phone mendukung proses di belakang untuk pelacakan GPS selama keluar dari aplikasi
18 atau layar perangkat dimatikan. Maka dari itu aplikasi yang penulis buat akan mendukung
19 pengaksesan lokasi meskipun layar perangkat dimatikan atau berpindah aplikasi.

20 Ketika aplikasi sudah berada pada keadaan *Dormant* atau *Tombstoned*, pengguna masih
21 dapat memulihkan keadaan aplikasi saat aplikasi berada di keadaan *Running* sebelumnya.
22 Penanganan yang penulis akan lakukan untuk hal tersebut adalah menggunakan *method*
23 *OnNavigatedTo()*. Menggunakan *method* tersebut akan memulihkan informasi halaman pada
24 keadaan *Running* sebelumnya.

25 **3.2.4 Analisis Peta**

26 Untuk tampilan peta ada beberapa aspek yang perlu diperhatikan untuk memudahkan
27 pengguna. Aspek yang perlu diperhatikan adalah sebagai berikut.

- 28 • Pemetaan terhadap peta atau *cartographic* dan mode warna
- 29 • Tingkat *zoom*
- 30 • Menampilkan gambar dan keterangan angkutan umum menggunakan *pushpin*
- 31 • Menggambar rute pada peta menggunakan *polyline*

32 Untuk cara pandang peta terdapat 4 pandangan yang disediakan peta di Windows Phone
33 yaitu *Road*, *Aerial*, *Hybrid*, dan *Terrain*. Aplikasi ini adalah aplikasi pencari rute dan pan-
34 dangan lebih banyak diarahkan ke jalanan perkotaan. Kebutuhan pengguna adalah nama
35 jalan, kondisi jalan, dan kondisi sekitar. Dari dasar pandangan tersebut pandangan yang
36 penulis pilih untuk aplikasi ini adalah *Road*. Tambahan setelah mengatur pandangan peta
37 yaitu mengatur warna dan penulis akan menggunakan mode warna terang sesuai bawaan
38 peta di Windows Phone.

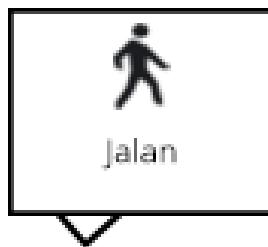
39 Tampilan awal peta di Windows Phone akan mengeluarkan peta dengan pandangan du-
40 nia. Karena aplikasi pencarian rute ini masih terbatas di Pulau Jawa, Indonesia terutama

1 Jawa Barat maka tingkat zoom harus diatur agar mengikuti lokasi pengguna dan di satu
 2 daerah saja. Jika pengguna berada di daerah Bandung maka tingkat zoom pada peta di-
 3 sesuaikan pada daerah tersebut. Tingkat zoom dapat dapat diatur dari kode dan XAML.
 4 Tingkat zoom yang penulis akan gunakan adalah 14. Tingkat zoom 14 akan menampilkan
 5 satu kota dengan jelas.

6 Di setiap kota ada satu angkutan umum yang banyak dipakai yaitu angkutan kota(angkot).
 7 Namun bagi yang baru pertama mengunjungi suatu daerah dan mencari angkot mungkin
 8 akan kesulitan membaca trayek dari angkot tersebut. Namun ada satu cara yang mudah
 9 untuk membedakan angkot di setiap rute yaitu dari warna dan coraknya. Agar dapat me-
 10 mudahkan pengguna dan menghindari pengguna dari kesalahan naik angkot maka Kiri API
 11 sudah menyediakan gambar angkot yang seusai dengan setiap rute. Gambar angkot terse-
 12 but akan ditempatkan di peta dengan suatu penampung beserta keterangannya. Salah satu
 13 teknik untuk menempatkan gambar tersebut adalah dengan membuat lapisan terpisah di
 14 atas peta tempat gambar tersebut. Untuk hal tersebut penulis akan memanfaatkan Pushpin
 15 sebagai lapisan terpisah untuk menaruh gambar dan keterangan angkutan umum. Berikut
 16 tampilan *pushpin* untuk angkot [3.6](#) dan *pushpin* untuk jalan kaki [3.7](#).



Gambar 3.6: Tampilan *pushpin* untuk angkot



Gambar 3.7: Tampilan *pushpin* untuk jalan kaki

17 Pencarian rute yang penulis gunakan untuk aplikasi yaitu dengan memakai Kiri API.
 18 Kiri API akan memberikan kembalian berupa titik-titik rute perjalanan dari lokasi asal ke
 19 lokasi tujuan. Karena hal itu penulis harus menggambar rute tersebut sesuai jalan pada
 20 peta. Untuk hal tersebut penulis akan menggunakan *Polyline* pada Windows Phone untuk
 21 menggambarnya. *Polyline* yang digambar harus terlihat dengan jelas dan diberi warna yang
 22 kontras dengan tampilan peta. Warna polyline yang penulis akan pilih adalah merah dengan
 23 ketebalan 2.

1 3.2.5 Analisis Pemanfaatan Sumber Data

2 Aplikasi yang penulis buat memanfaatkan sumber data dari luar. Sumber data yang pe-
3 nulis dapatkan dalam format JSON (*Javascript object Notation*). Pengambilan sumber data
4 tersebut dilakukan dengan melakukan permintaan HTTP dari *Uniform Resource Identifier*
5 / URI. Pemanfaatan sumber data yang penulis gunakan adalah kelas *HttpClient*.

6 *method* yang penulis gunakan adalah *GetStringAsync()*. *method* ini akan mengirimkan
7 permintaan melalui URI dan mengembalikan hasilnya dalam tipe data *String* dan kemajuan
8 data. Karena *method* ini mengembalikan hasil dalam tipe data *String* maka mudah disesua-
9 ikan dengan kebutuhan tugas akhir ini. Selanjutnya penulis harus membuat pengurai untuk
10 keluaran untuk diolah menjadi informasi yang dibutuhkan.

11 3.2.6 Analisis Kiri API

12 Kiri API menyediakan 2 parameter untuk permintaan yaitu *POST* dan *GET*. Da-
13 lam tugas akhir ini penulis akan menggunakan parameter *GET*. Parameter **GET** penu-
14 lis pilih karena dalam tugas akhir ini penulis akan banyak mendapatkan data dan tidak
15 ada data sensitif yang dikirimkan. Untuk hal ini penulis akan mengirim ke URL <http://kiri.travel/handle.php>.

17 Untuk setiap permintaan terhadap Kiri API dibutuhkan *API key*. Kegunaan *API key*
18 adalah password untuk mengakses Kiri API. *API key* dapat didapatkan di <dev.kiri>.
19 <travel>. *API key* yang penulis gunakan pada tugas akhir ini adalah 97A7A1157A05ED6F.

20 Untuk tugas akhir ini penulis akan menggunakan 2 layanan yang ada pada Kiri API.
21 Layanan yang digunakan adalah pencarian lokasi dan penentuan rute. Pencarian lokasi
22 adalah layanan untuk menemukan tempat atau nama jalan yang terkait dengan masukan
23 pengguna. Penentuan rute adalah layanan untuk menemukan langkah yang harus ditempuh
24 pengguna untuk sampai ke lokasi tujuan dari lokasi asal.

25 Pemanfaatan layanan pencarian lokasi yaitu dengan parameter *GET* melalui protokol
26 HTTP. Berikut parameter yang harus dikirimkan beserta keterangannya.

- 27 • *version*: 2

28 Karena acuan penulis adalah Kiri API versi maka di parameter *version* penulis akan
29 menggunakan 2.

- 30 • *mode*: "searchplace"

31 Mode "searchplace" digunakan untuk mencari lokasi terkait.

- 32 • *region*: "cgk" untuk Jakarta, "bdo" untuk Bandung, dan "sub" untuk Surabaya

33 Karena Kiri API baru tersedia di 3 kota yaitu Jakarta, Bandung, dan Surabaya maka
34 region harus dimasukan untuk pencarian. Region harus dipilih antara "cgk"/"bdo"/"sub"
35 sebagai parameter. Pengguna dapat menentukan masukan region jika menuliskannya
36 pada lokasi asal atau lokasi tujuan. Tetapi, jika pengguna tidak menuliskannya maka
37 sistem yang akan menentukan. Cara penentuan region oleh sistem adalah sistem akan
38 menampung titik tengah dari ketiga region tersebut lalu membandingkannya dengan
39 lokasi pengguna berada. Jarak terdekat antara lokasi pengguna dan salah satu region
40 menandakan pengguna berada di region tersebut.

- 41 • *querystring*: merupakan kata kunci lokasi

1 • *apikey*: 16 digit heksadesimal

2 Format layanan yang dikirim melalui URL adalah [kiri.travel/handle.php?version=2&mode=searchplace®ion=cgk/bdo/sub&querystring="](http://kiri.travel/handle.php?version=2&mode=searchplace®ion=cgk/bdo/sub&querystring=) "string" &*apikey*=97A7A1157A05ED6F.

4

5 Penulis mencoba mencari lokasi bip dari kata kunci "bip" yang berada di bandung. La-

6 yan dan dikirimkan ke URL kiri.travel/handle.php. Berikut format layanan yang penulis

7 kirim:

8 <http://kiri.travel/handle.php?version=2&mode=searchplace®ion=bdo&querystring=bip&apikey=97A7A1157A05ED6F>

10

11 Berikut hasil kembalian dari Kiri API:

Listing 3.1: Kode kembalian dari pencarian rute

```

12 {
13     "status ":"ok",
14     "searchresult ":[
15         {
16             "placename ":"Hypermart - BIP Plaza",
17             "location ":"-6.90864,107.61108"
18         },
19         {
20             "placename ":"Stroberi - BIP",
21             "location ":"-6.90834,107.61115"
22         },
23         {
24             "placename ":"Kebab Kings (Hypermart BIP)",
25             "location ":"-6.91503,107.61017"
26         },
27         {
28             "placename ":"Pegadaian UPC Bip Mall",
29             "location ":"-6.90916,107.61052"
30         },
31         {
32             "placename ":"Rice Bowl BIP",
33             "location ":"-6.90873,107.61088"
34         },
35         {
36             "placename ":"Gee Eight - Bip",
37             "location ":"-6.90817,107.61080"
38         },
39         {
40             "placename ":"Jonas Photo - BIP",
41             "location ":"-6.91066,107.61016"
42         },
43         {
44             "placename ":"Bip Foodcourt",
45             "location ":"-6.91081,107.61015"
46         },
47         {
48             "placename ":"Mister Baso BIP",
49             "location ":"-6.90348,107.61709"
50         },
51         {
52             "placename ":"JH Moriska - Bip",
53             "location ":"-6.90868,107.61070"
54         }
55     ],
56     "attributions":null
57 }
```

58 Hasil dari kembalian berupa kumpulan *placename* dan *location*. Hasil tersebut akan

59 aplikasi tampung namun yang akan ditampilkan ke pengguna hanya *placename*. Menam-

60 pilkan *location* tidak efektif menurut penulis karena akan membingungkan pengguna. Dari

61 percobaan yang penulis lakukan, nilai dari *attributions* selalu bernilai "null". Karena hal

62 tersebut maka nilai *attributions* akan penulis abaikan.

63 Pemanfaatan layanan penentuan rute untuk mendapatkan langkah yang harus ditempuh

64 pengguna untuk mencapai lokasi tujuan dari lokasi asal. Pemanfaatan layanan ini yaitu

65 dengan parameter *GET* melalui protokol HTTP. Berikut parameter yang harus dikirim:

- 1 ● *version: 2*
- 2 Karena acuan penulis adalah Kiri API versi maka di parameter version penulis akan
- 3 menggunakan 2.
- 4 ● *mode: "findroute"*
- 5 Mode "findroute" digunakan untuk mendapatkan langkah yang harus ditempuh me-
- 6 nuju lokasi tujuan.
- 7 ● *locale: "en"* untuk bahasa Inggris dan "id" untuk bahasa Indonesia.
- 8 Karena aplikasi ini memungkinkan dipakai orang banyak maka penulis putuskan untuk
- 9 menggunakan bahasa Inggris.
- 10 ● *start:* koordinat lokasi awal dalam berupa latitude dan longitude.
- 11 Masukan untuk lokasi awal harus dalam bentuk koordinat. Jika masukan dari peng-
- 12 guna adalah alamat atau tempat maka perlu dicari kordinatnya dahulu.
- 13 ● *finish:* koordinat lokasi tujuan dalam berupa latitude dan longitude.
- 14 Masukan untuk lokasi tujuan harus dalam bentuk koordinat. Jika masukan dari peng-
- 15 guna adalah alamat atau tempat maka perlu dicari kordinatnya dahulu.
- 16 ● *presentation: "mobile"* untuk perangkat bergerak dan "desktop" untuk komputer.
- 17 Karena aplikasi ini dirancang untuk Windows Phone 8, presentasi yang penulis pi-
- 18 ilih adalah "desktop". Pemilihan ini didasarkan karena deskripsi yang ditampilkan
- 19 tidak ada tulisan "image from" dan "image to", selain itu presentasi "desktop" juga
- 20 memungkinkan rute alternatif.
- 21 ● *apikey:* 16 digit heksadesimal.

22 Format layanan yang dikirim melalui URL adalah `kiri.travel/handle.php?version=`
 23 `2&mode=findroute&locale=en/id&start=lat,lng&finish=lat,lng&presentation=mobile/`
 24 `desktop&apikey=97A7A1157A05ED6`

25 Penulis mencoba menuju jalan merdeka dari jalan ciumbuleuit. Layanan dikirimkan ke
 26 URL `kiri.travel/handle.php`. Berikut format layanan yang penulis kirim dengan *pre-*
sentation mobile `http://kiri.travel/handle.php?version=2&mode=findroute&locale=`
`id&start=-6.8747337,107.6048829&finish=-6.9114646,107.6104887&presentation=mobile&apikey=`
`97A7A1157A05ED6F`.

31 Berikut hasil kembalian dari Kiri API:

Listing 3.2: Kode kembalian pencarian rute dengan *presentation mobile*

```
32 {  

33   "status ":"ok ",  

34   "routingresults ":[  

35     {  

36       "steps ":[  

37         [  

38           "walk ",  

39           "walk ",  

40           ["-6.8747337,107.6048829","-6.87445,107.60465"],  

41           "Jalan dari lokasi mulai Anda \%fromicon ke Jalan Ciumbuleuit \%toicon sejauh  

42             kurang lebih 41 meter.",  

43             null ,  

44             null  

45           ],  

46           [  

47             "angkot ",
```

```

1   " ciumbuleuitsthallurus " ,
2   [ "-6.87445,107.60465", "-6.87541,107.60443", "-6.87637,107.60421", "-6.87734,107.60400",
3
4   " -6.87830,107.60378",
5   " -6.87926,107.60356", "-6.87926,107.60356", "-6.87963,107.60352",
6   " -6.87978,107.60352", "-6.88093,107.60392", "-6.88209,107.60433", "-6.88209,107.60433",
7
8   " -6.88328,107.60490", "-6.88328,107.60490", "-6.88347,107.60481", "-6.88452,107.60459",
9
10  " -6.88556,107.60436", "-6.88660,107.60413", "-6.88764,107.60390", "-6.88764,107.60391",
11
12  " -6.88782,107.60392", "-6.88887,107.60404", "-6.88991,107.60416", "-6.88991,107.60416",
13
14  " -6.89161,107.60428", "-6.89161,107.60428", "-6.89166,107.60421", "-6.89275,107.60424",
15
16  " -6.89275,107.60424", "-6.89405,107.60408", "-6.89405,107.60408", "-6.89496,107.60400"],
17
18  " Naik angkot Ciumbuleuit – St. Hall (lurus) di Jalan Ciumbuleuit \%fromicon ,
19  dan turun di Jalan Cihampelas \%toicon kurang lebih setelah 3,3 kilometer
20  .",
21  null ,
22  https:// angkot.web.id/go/route/640?ref=kir
23 ],
24 [
25  " walk ",
26  " walk ",
27  [ "-6.90424,107.60433", "-6.90429,107.60440"],
28  " Jalan dari Jalan Cihampelas \%fromicon ke Jalan Abdul Rivai \%toicon sejauh
29  kurang lebih 10 meter.",
30  null ,
31  null
32 ],
33 [
34  " angkot ",
35  " kalapaledeng ",
36  [ "-6.89501,107.60403", "-6.89562,107.60398", "-6.89623,107.60395", "-6.89732,107.60401",
37
38  " -6.89732,107.60401", "-6.89882,107.60414", "-6.89882,107.60414", "-6.89969,107.60418",
39
40  " -6.90071,107.60426", "-6.90173,107.60433", "-6.90173,107.60433", "-6.90297,107.60437",
41
42  " -6.90420,107.60440", "-6.90420,107.60440", "-6.90426,107.60456", "-6.90422,107.60481",
43
44  " -6.90399,107.60546", "-6.90406,107.60617", "-6.90454,107.60697", "-6.90454,107.60697",
45
46  " -6.90512,107.60745", "-6.90618,107.60778", "-6.90618,107.60778", "-6.90643,107.60787",
47
48  " -6.90651,107.60807", "-6.90675,107.60914", "-6.90675,107.60914", "-6.90694,107.60939",
49
50  " -6.90723,107.60939", "-6.90891,107.60943", "-6.90891,107.60943", "-6.90909,107.60934",
51
52  " -6.90914,107.60857", "-6.90933,107.60846", "-6.91021,107.60887", "-6.91021,107.60887",
53
54  " -6.91030,107.60897", "-6.91028,107.60927", "-6.90986,107.61040", "-6.90986,107.61040"],
55
56  " Naik angkot Kalapa – Ledeng di Jalan Abdul Rivai \%fromicon , dan turun di
57  Jalan Aceh \%toicon kurang lebih setelah 1,1 kilometer.",
58  https:// angkot.web.id/go/route/156?ref=kiri"
59 ],
60 [
61  " walk ",
62  " walk ",
63  [ "-6.90986,107.61040", "-6.9114646,107.6104887"],
64  " Walk about 178 meter from Jalan Aceh \%fromicon to your destination \%toicon
65  .",
66  null ,
67  null
68 ],
69 ],
70  " traveltime ":"30 minutes"
71 ]
72 ]
73 }

```

74 Berikut format layanan yang penulis kirim dengan *presentation mobile* <http://kiri.travel/handle.php?version=2&mode=findroute&locale=id&start=-6.8747337,107.6048829&finish=-6.9114646,107.6104887&presentation=desktop&apikey=97A7A1157A05ED6F>.

Listing 3.3: Kode kembalian pencarian rute dengan *presentation desktop*

```

1  {
2      "status ":"ok",
3      "routingresults ":[
4          {
5              "steps ":[
6                  [
7                      "walk",
8                      "walk",
9                      ["-6.8747337,107.6048829","-6.87445,107.60464"],
10                     "Jalan dari lokasi mulai Anda ke Jalan Ciumbuleuit sejauh kurang lebih 41
11                     meter.",
12                     null,
13                     null
14                 ],
15                 [
16                     "angkot",
17                     "ciumbuleuitsthallurus",
18                     ["-6.87445,107.60464","-6.87541,107.60443","-6.87541,107.60443","-6.87637,107.60422",
19                     "-6.87637,107.60422","-6.87734,107.60400","-6.87734,107.60400","-6.87830,107.60378",
20                     "-6.87830,107.60378","-6.87926,107.60356","-6.87926,107.60356","-6.87926,107.60356",
21                     "-6.87963,107.60352","-6.87978,107.60352","-6.88093,107.60393","-6.88093,107.60393",
22                     "-6.88209,107.60433","-6.88209,107.60433","-6.88209,107.60433","-6.88328,107.60490",
23                     "-6.88328,107.60490","-6.88328,107.60490","-6.88347,107.60481","-6.88452,107.60458",
24                     "-6.88452,107.60458","-6.88556,107.60435","-6.88556,107.60435","-6.88660,107.60413",
25                     "-6.88660,107.60413","-6.88764,107.60390","-6.88764,107.60390","-6.88764,107.60390",
26                     "-6.88782,107.60392","-6.88887,107.60404","-6.88887,107.60404","-6.88991,107.60416",
27                     "-6.88991,107.60416","-6.88991,107.60416","-6.89161,107.60428","-6.89161,107.60428",
28                     "-6.89161,107.60428","-6.89166,107.60420","-6.89275,107.60424","-6.89275,107.60424",
29                     "-6.89275,107.60424","-6.89405,107.60407","-6.89405,107.60407","-6.89405,107.60407",
30                     "-6.89496,107.60400","-6.89496,107.60400","-6.89586,107.60392","-6.89586,107.60392",
31                     "-6.89586,107.60392","-6.89759,107.60397","-6.89759,107.60397","-6.89759,107.60397",
32                     "-6.89895,107.60406","-6.89895,107.60406","-6.89895,107.60406","-6.89970,107.60413",
33                     "-6.89999,107.60416","-6.90114,107.60426","-6.90114,107.60426","-6.90114,107.60426",
34                     "-6.90218,107.60428","-6.90218,107.60428","-6.90321,107.60431","-6.90321,107.60431",
35                     "-6.90424,107.60433"],
36                     "Naik angkot Ciumbuleuit – St. Hall (lurus) di Jalan Ciumbuleuit , dan turun di
37                     Jalan Cihampelas kurang lebih setelah 3,3 kilometer.",
38                     null,
39                     "https:// angkot.web.id/go/route/640?ref=kiri"
40                 ],
41                 [
42                     "walk",
43                     "walk",
44                     ["-6.90424,107.60433","-6.90429,107.60440"],
45                     "Jalan dari Jalan Cihampelas ke Jalan Abdul Rivai sejauh kurang lebih 10 meter
46                     .",
47                     null,
48                     null
49                 ],
50                 [
51                     "angkot",
52                     "kalapaledeng",
53                     ["-6.90429,107.60440","-6.90434,107.60490","-6.90398,107.60558","-6.90417,107.60619",
54                     "-6.90465,107.60702","-6.90465,107.60702","-6.90465,107.60702","-6.90521,107.60748",
55                     "-6.90600,107.60771","-6.90646,107.60775","-6.90755,107.60760","-6.90755,107.60760",
56                     "-6.90755,107.60760","-6.90866,107.60789","-6.90866,107.60789","-6.90866,107.60789",
57                     "-6.90911,107.60826","-6.91034,107.60892","-6.91034,107.60892","-6.91034,107.60892",
58                     "-6.91007,107.60985"],
59                     "Naik angkot Kalapa – Ledeng di Jalan Abdul Rivai , dan turun di Jalan Aceh
60                     kurang lebih setelah 1,1 kilometer.",
61                     null,
62                     "https:// angkot.web.id/go/route/156?ref=kiri"
63             ]
64         ]
65     ]
66 ]
67 ]
68 ]
69 ]
70 ]
71 ]
72 ]
73 ]
74 ]
75 ]
76 ]
77 ]
78 ]
79 ]
80 ]
81 ]
82 ]
83 ]
84 ]

```

```

1   ],
2   [
3     "walk",
4     "walk",
5     ["-6.91007,107.60985", "-6.9114646,107.6104887"],
6     "Jalan dari Jalan Aceh ke tujuan akhir Anda sejauh kurang lebih 171 meter.",
7     null,
8     null
9   ],
10  "traveltime":"30 menit"}]}

```

Setiap langkah akan aplikasi tampung dalam elemen *array*. Untuk keterangan dan jenis angkutan umum akan aplikasi tampilkan dalam bentuk *pushpin* pada peta atau daftar. Sedangkan untuk titik-titik kordinat akan digambarkan pada peta. Dari analisa penulis setiap langkah menunjukkan perpindahan angkutan umum yang dipakai, berpindah dari angkutan umum atau jalan, dan dari jalan untuk menaiki angkutan umum. Keterangan yang penulis akan tambahkan harus berada antara setiap *steps* tersebut. Dari analisa penulis juga terdapat kata "%fromicon" dan "%toicon" yang tidak menunjukkan sesuatu. Karena itu kedua kata tersebut akan penulis hilangkan agar tidak mengganggu pengguna. Penulis juga akan mengambil gambar angkutan kota dan gambar jalan yang sudah disediakan dari Kiri dengan memanfaatkan URL yang disediakan.

3.2.7 Diagram *Use Case* dan Skenario

Diagram *use case* adalah diagram yang menjelaskan interaksi sistem dengan lingkungan (contoh: pengguna). Berdasarkan analisa di atas maka pengguna dapat:

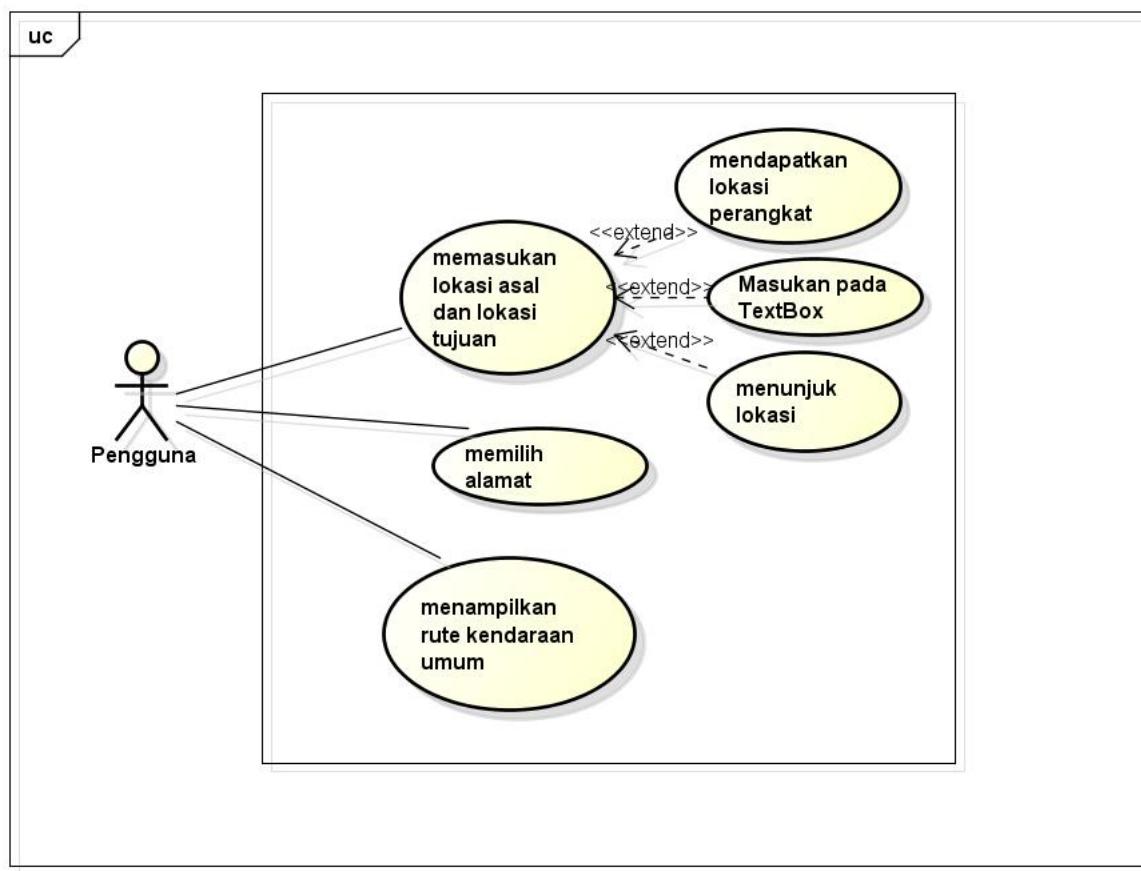
- Mendapatkan lokasi pengguna berada.
- Memasukan lokasi asal dan lokasi tujuan.
- Menunjuk langsung lokasi asal dan tujuan pada peta.
- Memilih alamat atau tempat dari pilihan yang disediakan.
- Menampilkan rute kendaraan umum dalam bentuk titik dan *pushpin* pada peta atau bentuk daftar dari tempat asal ke tempat tujuan.

Diagram *use case* saat pengguna mencari rute kendaraan umum dapat dilihat pada gambar (Gambar: 3.8):

Skenario pencarian rute kendaraan umum dapat dilihat pada tabel 3.1 sampai tabel 3.5.

Nama	Mendapatkan lokasi perangkat
Aktor	Pengguna
Deskripsi	Mendapatkan lokasi perangkat berada
Kondisi awal	TextBox masih kosong dan pengguna menekan tombol lokasi
Kondisi akhir	Lokasi ditemukan dan TextBox berisi "here"
Skenario utama	Pengguna menekan tombol lalu perangkat akan mencari lokasi perangkat dan TextBox berisi "here"
Eksespsi	lokasi tidak ditemukan jika GPS perangkat tidak aktif

Tabel 3.1: Skenario mandapatkan lokasi untuk masukan lokasi asal dan lokasi tujuan

Gambar 3.8: Diagram *use case*

Nama	Masukan pada <i>TextBox</i>
Aktor	Pengguna
Deskripsi	Memasukan lokasi asal pengguna dan tujuan pengguna(masukan dapat berupa alamat, kordinat, atau tempat)
Kondisi awal	TextBox masih dalam keadaan belum terisi
Kondisi akhir	Lokasi awal dan tujuan sudah dimasukan
Skenario utama	Pengguna mengetikan lokasi awal dan tujuan pada <i>TextBox</i> yang sudah disediakan
Eksespsi	tidak ada

Tabel 3.2: Skenario memasukan lokasi asal dan lokasi tujuan pada *TextBox*

Nama	Menunjuk lokasi
Aktor	Pengguna
Deskripsi	Memasukan lokasi asal pengguna dan tujuan pengguna dengan menunjuk pada peta
Kondisi awal	TextBox masih dalam keadaan belum terisi
Kondisi akhir	TextBox terisi dengan "lokasi dari peta"
Skenario utama	Pengguna menunjuk lokasi pada peta dan TextBox terisi dengan "lokasi dari peta"
Eksespsi	tidak ada

Tabel 3.3: Skenario menunjuk lokasi asal dan lokasi tujuan pada peta

Nama	Memilih alamat
Aktor	Pengguna
Deskripsi	Pengguna memilih alamat atau lokasi yang terkait masukan pengguna
Kondisi awal	Lokasi awal dan lokasi tujuan terisi dan pengguna menekan tombol "Find"
Kondisi akhir	Pengguna sudah memilih dan lokasi sudah dapat dipastikan
Skenario utama	Pengguna menekan tombol "Find". Sistem mengembalikan daftar yang berisi alamat atau tempat terkait masukan pengguna
Eksespsi	Lokasi masukan pengguna tidak ditemukan

Tabel 3.4: Skenario memilih alamat

Nama	Menampilkan rute kendaraan umum
Aktor	Pengguna
Deskripsi	Lokasi dari pengguna diolah menjadi rute kendaraan umum dari lokasi asal dan lokasi tujuan
Kondisi awal	Lokasi sudah dapat dipastikan
Kondisi akhir	Rute kendaraan umum dimunculkan pada peta dan dalam bentuk daftar
Skenario utama	Lokasi dapat dipastikan sistem. Sistem lalu akan memproses data masukan. Sistem akan mengembalikan hasil rute kendaraan umum pada peta dan dalam bentuk daftar
Eksespsi	Rute kendaraan umum tidak ditemukan

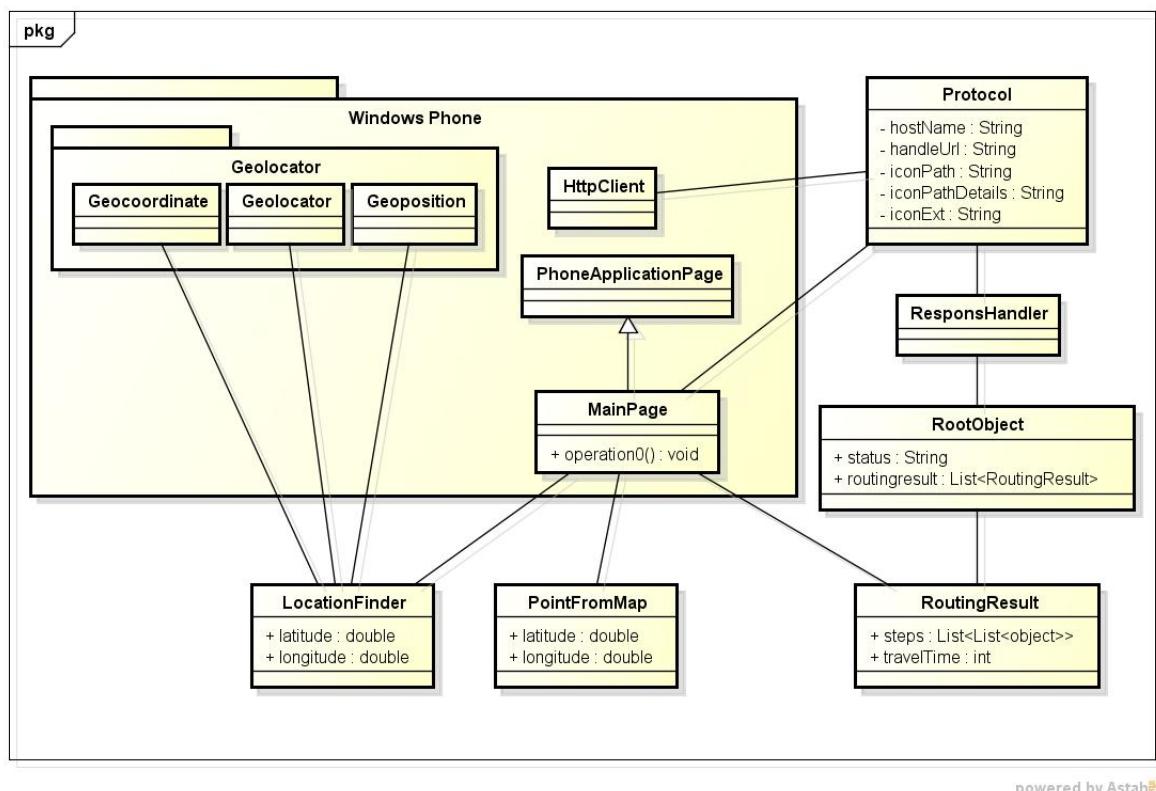
Tabel 3.5: Skenario menampilkan rute kendaraan umum

¹ 3.2.8 Kelas Diagram

² Pembuatan kelas diagram didasarkan pada skenario pada sub bab [3.2.7](#). Kelas diagram ³ dapat dilihat pada gambar [4.4](#).

⁴ Berikut deskripsi kelas pada gambar [4.4](#).

- ⁵ • Kelas *Protocol*
- ⁶ Merupakan kelas yang menampung semua alamat URL yang berhubungan dengan Kiri API. Semua pemanggilan akan ditangani oleh kelas ini.



powered by Astah

Gambar 3.9: Diagram Kelas

- 1 ● Kelas *ResponsHandler*
Merupakan kelas yang menangani masukan dari pemanggilan layanan.
- 2 ● Kelas *RootObject*
Merupakan kelas untuk menampung status dan daftar dari layanan *routing* Kiri API. Hasil kembalian akan dipisahkan di kelas ini untuk selanjutnya ditambahkan di kelas *RoutingResult*.
- 3 ● Kelas *RoutingResult*
Merupakan kelas untuk menampung setiap langkah dari rute sesuai masukan pengguna. Pada kelas ini juga rute akan digambarkan pada peta.
- 4 ● Kelas *PointFromMap*
Merupakan kelas yang dapat mengetahui lokasi yang ditunjuk pengguna pada peta. Kelas ini akan menyimpan lokasi yang ditunjuk pengguna dalam bentuk *latitude* dan *longitude*.
- 5 ● Kelas *LocationFinder*
Merupakan kelas yang digunakan untuk mencari lokasi. kelas ini akan memanfaatkan kelas *Geocoordinate* untuk mendapatkan lokasi. Setelah lokasi didapatkan dalam bentuk kelas *Geoposition* maka akan diubah ke *latitude* dan *longitude*.

1

BAB 4

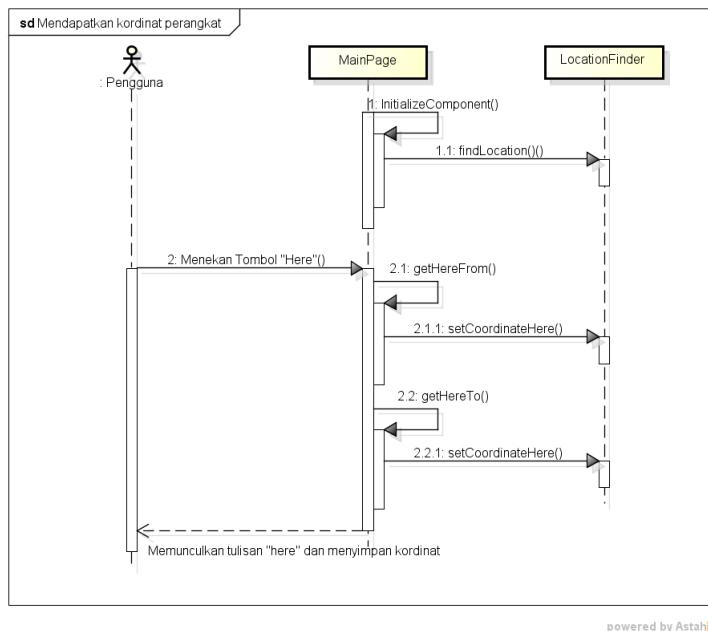
2

PERANCANGAN

- 3 Pada bab 4 akan dibahas mengenai perancangan seperti diagram *sequence*, diagram kelas
4 secara rinci, deskripsi atribut dan *method* dari setiap kelas, dan perancangan antarmuka.

5 4.1 Diagram *Sequence*

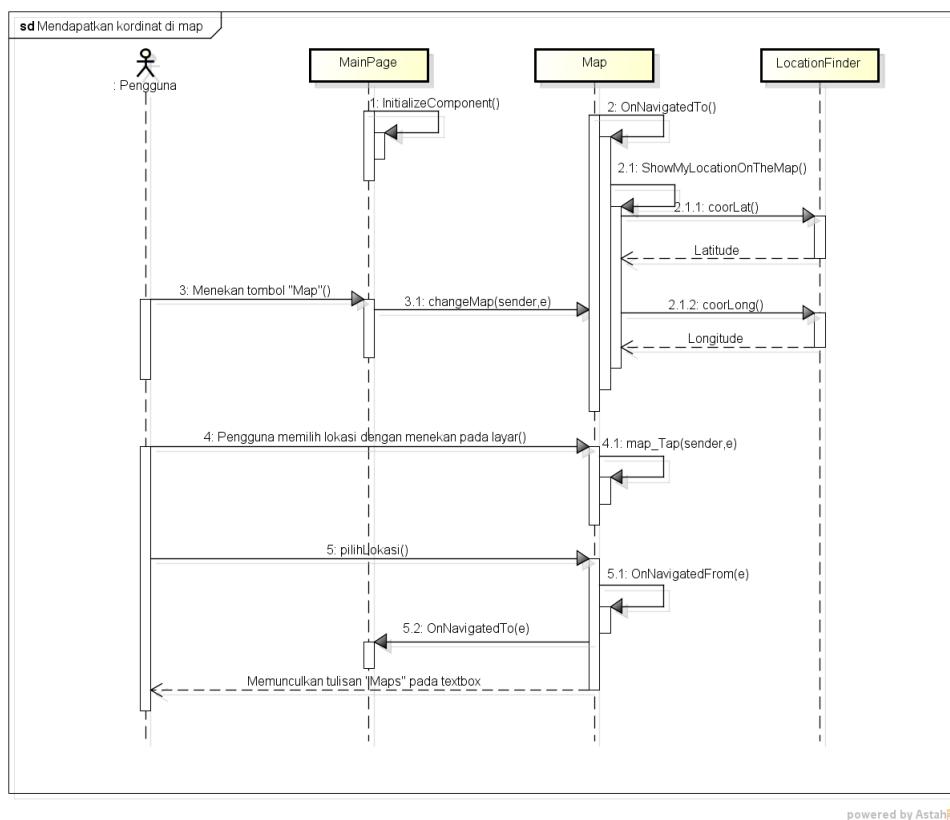
- 6 Diagram *sequence* merupakan diagram yang menggambarkan interaksi antar objek dalam
7 suatu skenario. Gambar diagram *sequence* dapat dilihat pada gambar reffig:sequence lokasi
8 perangkat sampai reffig:sequence route.



Gambar 4.1: Diagram *sequence* Mendapatkan Kordinat perangkat

9 Diagram 4.1 merupakan diagram *sequence* untuk memilih lokasi dengan lokasi per-
10 angkat berada. Diagram menunjukkan bahwa setelah aplikasi dibuka maka aplikasi akan
11 mencari dahulu lokasi perangkat dengan memanfaatkan kelas LocationFinder. Lalu setelah
12 aplikasi terbuka jika pengguna ingin memilih lokasi tersebut sebagai lokasi asal maka peng-
13 guna harus menekan tombol "here". Setelah tombol "here" ditekan maka kelas MainPage
14 akan mengambil nilai *Latitude* dan nilai *Longitude* dari kelas LocationFinder. Setelah lokasi
15 didapatkan maka akan muncul tulisan "here" pada masukan di kelas MainPage.

16 Diagram 4.2 merupakan diagram *sequence* untuk memilih lokasi. Diagram menun-
17 jukan bahwa setelah aplikasi dibuka maka pengguna dapat menekan tombol "map". Setelah

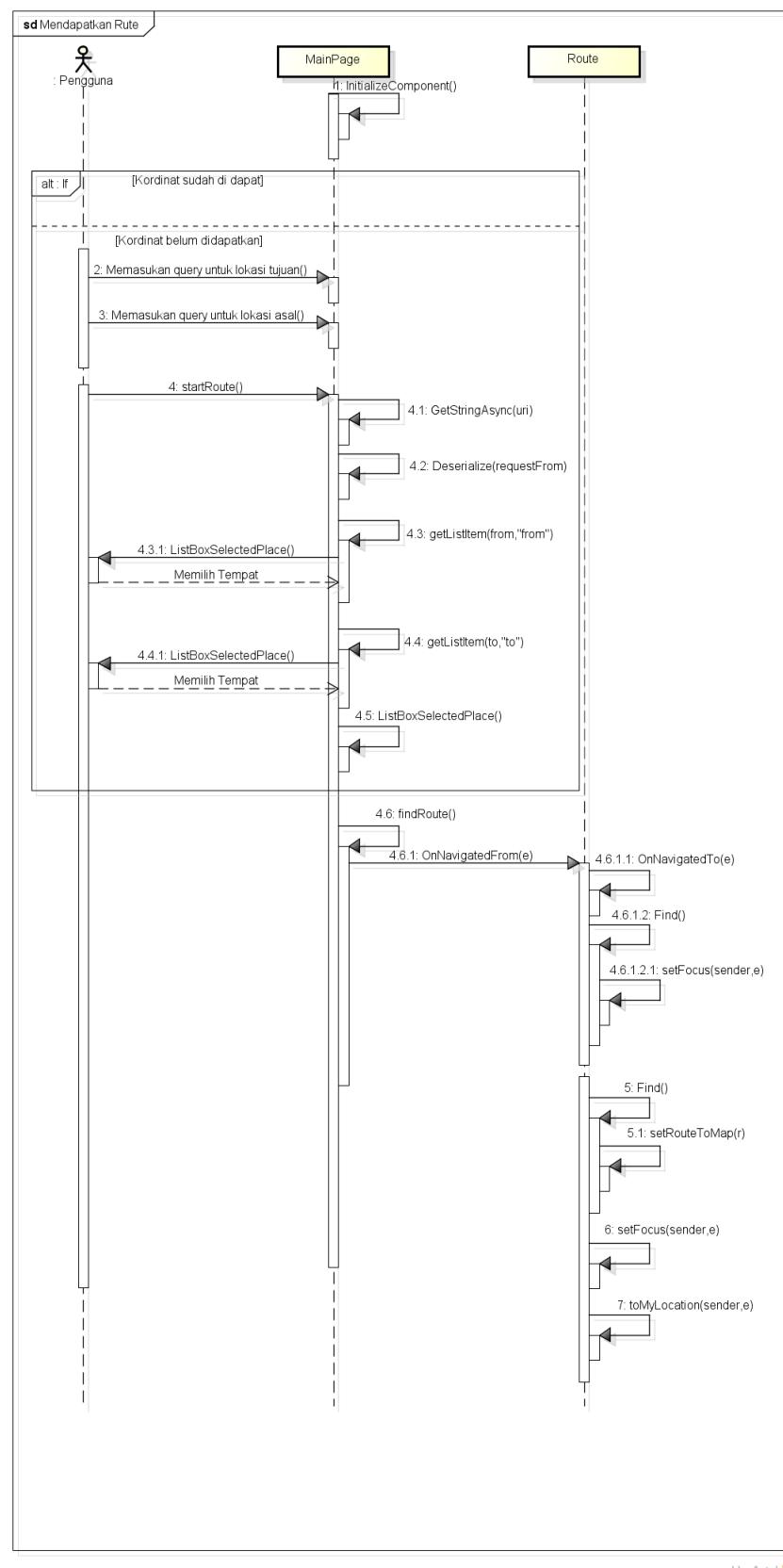


Gambar 4.2: Diagram *sequence* Mendapatkan Kordinat pada Peta

- 1 tombol "map" ditekan maka halaman akan dialihkan ke kelas Map. Saat kelas Map terbuka
- 2 maka lokasi yang ditunjukkan adalah lokasi dimana perangkat berada. Untuk mengetahui
- 3 lokasi kelas Map mengambil kordinat *latitude* dan *longitude* dari kelas LocationFinder. Di
- 4 kelas "Map" pengguna dapat memilih lokasi dengan memilih lokasi pada peta dan memang-
- 5 gil *method map_tap*, lalu setelah pengguna memilih tempat pengguna akan menekan tombol
- 6 Pilih Lokasi yang akan memanggil *method pilihLokasi*. Lokasi yang dipilih pengguna akan
- 7 disimpan di kelas MainPage dan pada masukan akan tertulis "map".

Diagram 4.3 merupakan diagram *sequence* untuk mencari rute. Diagram menunjukkan bahwa setelah aplikasi dibuka maka aplikasi akan melakukan inisialisasi. Untuk mencari rute dari lokasi asal ke lokasi tujuan dibutuhkan kordinat *latitude* lokasi asal, *longitude* lokasi asal, *latitude* lokasi tujuan, dan *longitude* lokasi tujuan. Jika pengguna mendapatkan lokasi dari peta atau sesuai lokasi maka yang didapatkan sudah pasti kordinat, namun jika pengguna memasukan kata kunci perlu didapat kordinat *latitude* dan *longitude* dari kata kunci tersebut. Jika masukan yang didapat berupa kata kunci maka akan dilakukan pemeriksaan apakah kordinat untuk kata kunci tersebut tersedia. Pemeriksaan dilakukan dengan melakukan pemanggilan Kiri API. Tahap pemanggilan meliputi pemanggilan *method GetStringAsync* lalu mengjadikan objek kembalinya dengan *method Deserialize*. Jika sudah didapat dan hasilnya lebih dari satu maka akan dipanggil *method getListItem* yang akan menampilkan daftar pilihan ke pengguna untuk dipilih. Pengguna dapat memilih tempat sesuai tempat asal maupun tujuan yang diinginkan. Setelah lokasi asal dan lokasi tujuan di dapat maka kelas MainPage akan mengarahkan ke kelas Route untuk menampilkan hasilnya. Kelas Route akan memanggil *method OnNavigatedTo* yang bertujuan untuk mendapatkan lokasi asal dan lokasi tujuan. Setelah itu akan memanggil *method Find* lalu mengembalikan

- 1 rute yang ditemukan kepada pengguna.

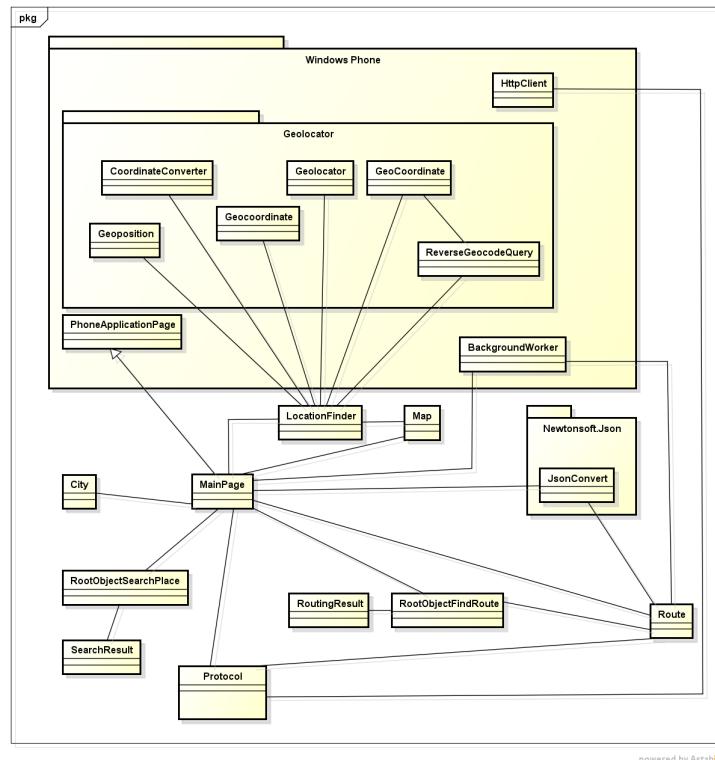


powered by Astah

Gambar 4.3: Diagram *sequence* Mendapatkan Rute

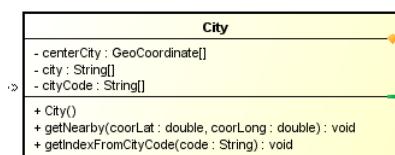
4.2 Perancangan Kelas

Pada sub bab ini akan dibahas mengenai deskripsi kelas secara rinci pada aplikasi Pencari Rute Kendaraan Umum untuk Windows Phone. Untuk lebih jelas mengenai kelas yang ada pada aplikasi ini, penulis menyajikan gambar diagram kelas yang dapat dilihat pada gambar 4.4.

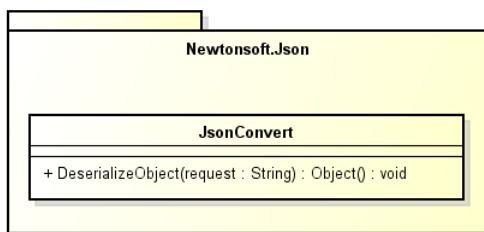


Gambar 4.4: Diagram Kelas

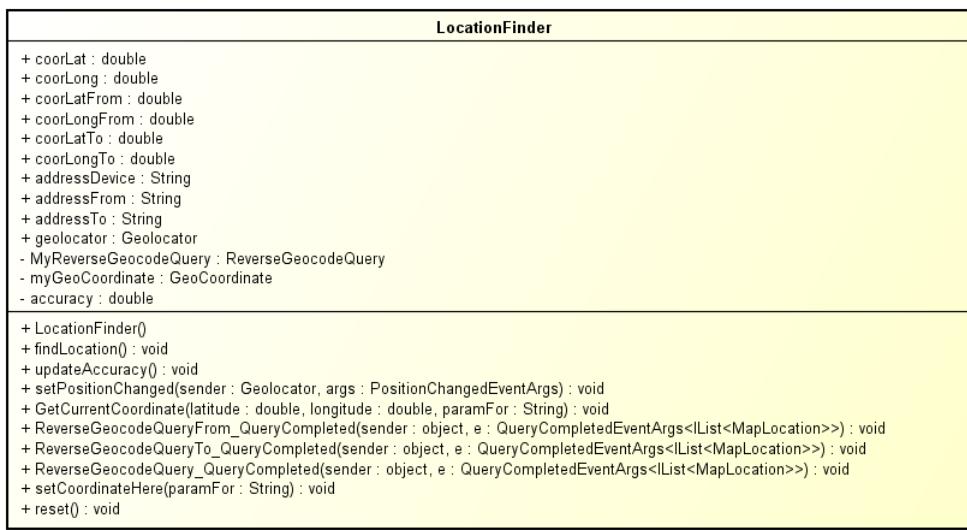
Berikut Penjelasan kelas diagram secara rinci dengan setiap kelas disertai atribut dan *method*.



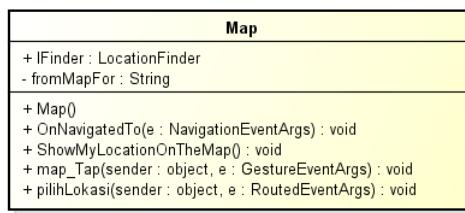
Gambar 4.5: Kelas City



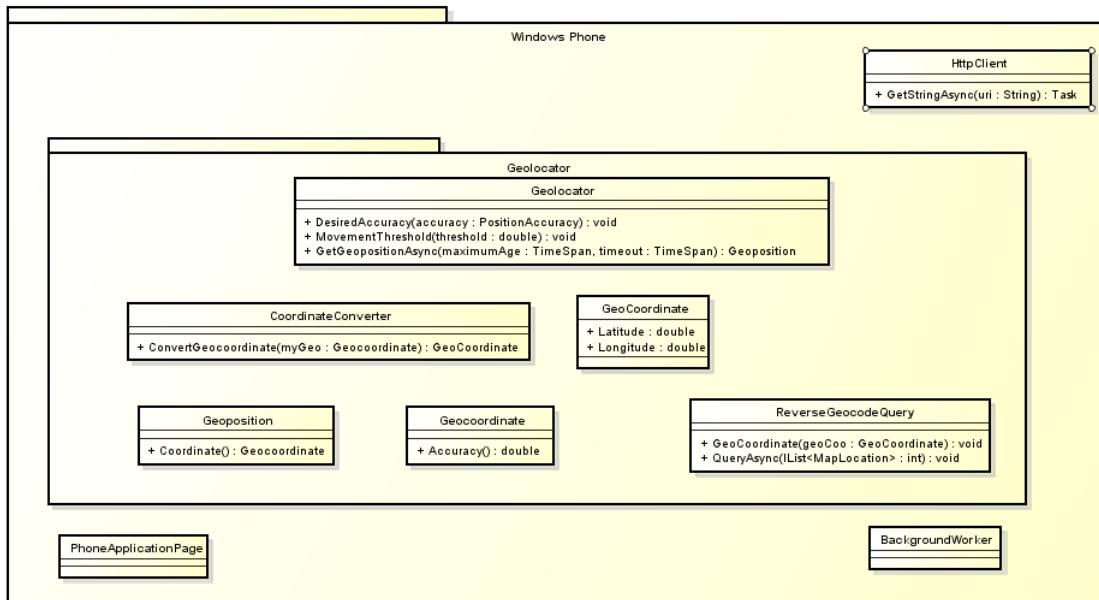
Gambar 4.6: Kelas JsonConvert



Gambar 4.7: Kelas LocationFinder



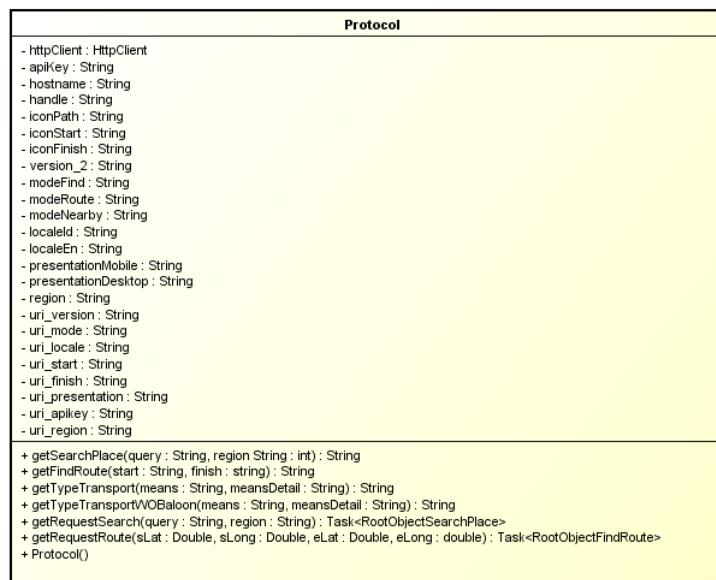
Gambar 4.8: Kelas Map



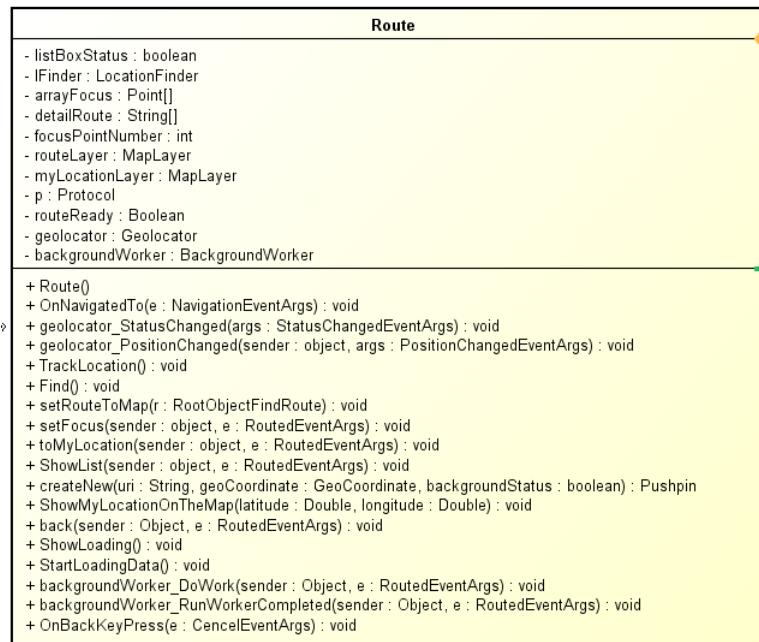
Gambar 4.9: Package WindowsPhone

1 4.2.1 Kelas *PhoneApplicationPage*

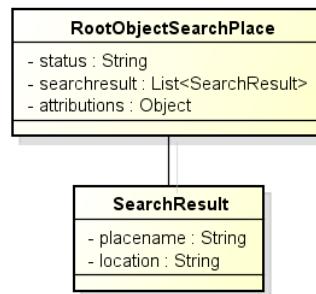
- 2 *PhoneApplicationPage* merupakan kelas bawaan Windows Phone yang menangani interaksi pengguna dengan aplikasi dan siklus hidup aplikasi.



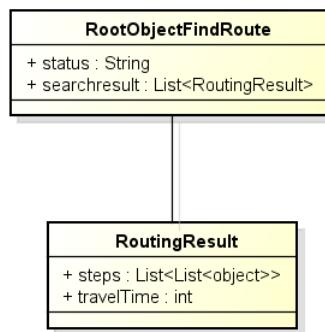
Gambar 4.10: Kelas Protocol



Gambar 4.11: Kelas Route



Gambar 4.12: Kelas SearchPlace



Gambar 4.13: Kelas FindRoute

4.2.2 Kelas MainPage

MainPage merupakan kelas turunan dari kelas *PhoneApplicationPage* yang menangani interaksi langsung antara halaman aplikasi dengan pengguna. Pada kelas ini akan ditaruh kontrol yang diperlukan. Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

1. protocol bertipe Protocol untuk mendapatkan URL yang digunakan dalam permintaan ke Kiri API.
2. IFinder bertipe LocationFinder digunakan untuk menampung semua informasi mengenai lokasi.
3. httpClient bertipe HttpClient merupakan objek yang akan mengurus permintaan dan kembalian dari Kiri API.
4. city bertipe kumpulan String untuk menampung kota yang didukung oleh layanan Kiri.
5. myCity bertipe String untuk menampung kode kota sesuai Kode Penerbangan IATA.
6. backgroundWorker bertipe BackgroundWorker untuk mengurus pencarian lokasi di belakang layar.

Berikut adalah penjelasan beberapa *method* yang dimiliki kelas ini:

1. Konstruktor MainPage digunakan untuk memuat komponen yang ada di halaman MainPage dan mendapatkan kota terdekat dari lokasi perangkat.
2. *method* OnNavigatedTo digunakan untuk mendapatkan lokasi dari peta dan mendapatkan objek LocationFinder. *Method* ini secara otomatis dipanggil jika pengguna kembali ke kelas MainPage. *method* ini memiliki parameter NavigationEventArgs.
3. *method* OnNavigatedFrom digunakan untuk mengirimkan *state* objek LocationFinder saat berpindah kelas. *Method* ini secara otomatis dipanggil jika pengguna berpindah dari kelas MainPage ke kelas lain. *method* ini memiliki parameter NavigationEventArgs.
4. *method* Application_Deactivated digunakan untuk menyimpan *state* objek saat meninggalkan aplikasi. *method* ini memiliki parameter object dan DeactivatedEventArgs.

- 1 5. *method Application_Activated* digunakan untuk mengambil *state* objek saat kembali ke aplikasi. *method* ini memiliki parameter *object* dan *ActivatedEventArgs*.
- 2 6. *method startRoute* digunakan untuk mendapatkan masukan pengguna. Jika masukan didapat dari peta atau lokasi perangkat berarti sudah dalam kordinat, namun jika masukan didapat dari *query* maka akan dicari lokasi yang terkait sesuai *query* tersebut. Lokasi terkait yang didapatkan akan dikembalikan ke pengguna untuk dipilih. Setelah pengguna lokasi dalam bentuk kordinat didapatkan maka kelas akan diarahkan ke kelas *Route*. *method* ini memiliki parameter objek tombol dan event.
- 3 7. *method changeMapFrom* digunakan untuk berpindah ke halaman *mapFrom*. *method* ini memiliki parameter objek tombol dan event.
- 4 8. *method changeMapTo* digunakan untuk berpindah ke halaman *mapTo*. *method* ini memiliki parameter objek tombol dan event.
- 5 9. *method getHereFrom* digunakan untuk mendapatkan kordinat perangkat lalu menyimpan nilainya di kordinat asal di kelas *LocationFinder* dan menulisakan "Here" pada *TextBox* lokasi asal. *method* ini memiliki parameter objek tombol dan event.
- 6 10. *method getHereTo* digunakan untuk mendapatkan kordinat perangkat lalu menyimpan nilainya di kordinat tujuan di kelas *LocationFinder* dan menulisakan "Here" pada *TextBox* lokasi tujuan. *method* ini memiliki parameter objek tombol dan event.
- 7 11. *method getListItem* digunakan untuk membuat *listBox* lalu menampilkan ke pengguna. *method* ini memiliki parameter *RootObjectSearchPlace* dan *string* yang menunjukkan *list* yang ditampilkan untuk lokasi asal dan lokasi tujuan.
- 8 12. *method ListBoxSelectedPlace* digunakan untuk mendapatkan tempat asal yang dipilih pengguna. *method* ini memiliki parameter objek dan *event SelectionChangedEventArgs*.
- 9 13. *method searchCoordinatePlace* digunakan untuk mencari kordinat dari tempat pilihan pengguna. *method* ini memiliki parameter *ListBox* dan tempat yang dipilih dalam bentuk *string*.
- 10 14. *method findRoute* digunakan untuk berpindah ke kelas *Route* jika lokasi asal dan lokasi tujuan sudah ditentukan.
- 11 15. *method changeCity* digunakan untuk mengubah kota tujuan dari pencarian. *method* ini memiliki parameter objek dan *event SelectionChangedEventArgs*.
- 12 16. *method showCity* digunakan untuk mencari kota yang paling dekat dengan lokasi perangkat.
- 13 17. *method ShowSplash* digunakan untuk menampilkan tampilan awal untuk proses inisialisasi aplikasi.
- 14 18. *method StartLoadingData* digunakan untuk memanggil *BackgroundWorker*. *BackgroundWorker* digunakan untuk melakukan aksi di belakang layar.

- 1 19. *method* backgroundWorker_DoWork digunakan untuk melakukan pemanggilan aksi
- 2 di belakang layar. *method* ini memiliki parameter objek dan DoWorkEventArgs.
- 3 20. *method* backgroundWorker_RunWorkerCompleted digunakan untuk melakukan pe-
- 4 manggilan saat BackgroundWorker selesai melakukna tugasnya. *method* ini memiliki
- 5 parameter objek dan RunWorkerCompletedEventArgs.

6 4.2.3 Kelas *City*

7 *City* merupakan kelas yang menyimpan kota-kota yang mendukung pencarian rute ken-
8 daraam umum dengan bantuan Kiri. Berikut adalah penjelasan atribut-atribut yang dimiliki
9 kelas ini:

- 10 1. centerCity bertipe *array of GeoCoordinate* untuk menyimpan kordinat pusat dari kota.
- 11 2. city bertipe *array of String* untuk menyimpan nama kota.
- 12 3. cityCode bertipe *array of String* untuk menyimpan kode kota dalam huruf kecil sesuai
13 aturan IATA Airport Code.

14 Berikut adalah penjelasan beberapa *method* yang dimiliki kelas ini:

- 15 1. Konstruktor City digunakan untuk untuk inisialisasi nilai atribut.
- 16 2. *method* getNearby digunakan untuk mencari kota terdekat dengan lokasi perangkat.
17 *method* ini mengembalikan interger yang merupakan index kota pada atribut city.
18 *method* ini memiliki 2 buah parameter yaitu *latitude* dan *longitude* yang bertipe double.
- 19 3. *method* getIndexFromCityCode digunakan untuk mencari index pada *array* sesuai kode
20 kota. *method* ini memiliki parameter bertipe String yang merupakan kode kota.

21 4.2.4 Kelas *BackgroundWorker*

22 *BackgroundWorker* merupakan kelas yang dipakai untuk mengeksekusi operasi pada
23 *thread* terpisah. Berikut adalah penjelasan *event* yang dimiliki kelas ini dan dipakai untuk
24 perancangan aplikasi:

- 25 1. *Event* DoWork
- 26 2. *Event* RunWorkerCompleted

27 Berikut adalah penjelasan beberapa *method* yang dimiliki kelas ini:

- 28 1. *method* RunWorkerAsync digunakan untuk memulai operasi di belakang layar.

29 4.2.5 Kelas *Geocoordinate*

30 *Geocoordinate* merupakan kelas bawaan dari Windows Phone yang akan dimanfaatkan
31 untuk membaca *latitude* dan *longitude*.

1 4.2.6 Kelas *Geolocator*

2 *Geolocator* merupakan kelas bawaan Windows Phone untuk mengkases lokasi. Dengan
3 bantuan kelas ini maka dapat mengetahui status lokasi dari perangkat dan menemukan lokasi
4 secara akurat.

5 4.2.7 Kelas *Geoposition*

6 *Geoposition* merupakan kelas yang menampung lokasi sesuak kembalian *Geolocator*.

7 4.2.8 Kelas *LocationFinder*

8 *LocationFinder* merupakan kelas yang akan menampung lokasi dan pencarian lokasi.
9 Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

- 10** 1. coorLat bertipe Double untuk menampung kordinat latitude pengguna.
- 11** 2. coorLong bertipe Double untuk menampung kordinat longitude pengguna.
- 12** 3. coorLatFrom bertipe Double untuk menampung kordinat latitude lokasi asal yang
13 diinginkan pengguna.
- 14** 4. coorLongFrom bertipe Double untuk menampung kordinat longitude lokasi asal yang
15 diinginkan pengguna.
- 16** 5. coorLatTo bertipe Double untuk menampung kordinat latitude lokasi tujuan yang
17 diinginkan pengguna.
- 18** 6. coorLongTo bertipe Double untuk menampung kordinat longitude lokasi tujuan yang
19 diinginkan pengguna.
- 20** 7. addressDevice bertipe String untuk menyimpan alamat perangkat berada.
- 21** 8. addressDeviceFrom bertipe String untuk menyimpan alamat berdasarkan lokasi lokasi
22 asal yang diinginkan pengguna.
- 23** 9. addressDeviceTo bertipe String untuk menyimpan alamat berdasarkan lokasi tujuan
24 yang diinginkan pengguna.
- 25** 10. geolocator bertipe Geolocator untuk menampung pengaturan mendapatkan lokasi.
- 26** 11. myReverseGeocodeQuery bertipe ReverseGeocodeQuery untuk konversi dari alamat
27 ke lokasi dan sebaliknya.
- 28** 12. myCoordinate bertipe GeoCoordinate untuk menampung kordinat geografis.
- 29** 13. accuracy bertipe double untuk menampung akurasi perangkat mendapatkan lokasi.
- 30** Berikut adalah penjelasan beberapa *method* yang dimiliki kelas ini:
 - 31** 1. Konstruktor LocationFinder berfungsi mengatur atribut geolocator dan mencari lokasi
32 perangkat.

- 1 2. *method* findLocation berfungsi inisialisasi GPS lalu mendapat kordinat dan menampungnya di atribut.
- 2 3. *method* updateAccuracy berfungsi untuk mengubah nilai akurasi dari perangkat.
- 4 4. *method* setPositionChanged berfungsi mengubah atribut coorLat, atribut coorLong, dan akurasi jika terdapat perubahan lokasi. *method* ini memiliki parameter Geolocator dan PositionChangedEventArgs yang akan menjalankan *method* jika terdapat perubahan yang diberitahukan melalui kelas Geolocator.
- 5 5. *method* GetCurrentCoordinate berfungsi mengubah posisi saat ini, posisi lokasi asal, dan lokasi tujuan. *method* ini memiliki tiga buah parameter *latitude* bertipe Double, *longitude* bertipe Double, dan paramFor bertipe String. Parameter *latitude* dan *longitude* merupakan lokasi sedangkan parameter paramFor digunakan sebagai tujuan perubahan lokasi.
- 6 6. *method* ReverseGeocodeQueryFrom_QueryCompleted berfungsi untuk mencari alamat lokasi asal. *method* ini memiliki parameter objek dan QueryCompletedEventArgs<IList<MapLocation>>.
- 7 7. *method* ReverseGeocodeQueryTo_QueryCompleted berfungsi untuk mencari alamat lokasi tujuan. *method* ini memiliki parameter objek dan QueryCompletedEventArgs<IList<MapLocation>>.
- 8 8. *method* ReverseGeocodeQuery_QueryCompleted berfungsi untuk mencari alamat lokasi perangkat. *method* ini memiliki parameter objek dan QueryCompletedEventArgs<IList<MapLocation>>.
- 9 9. *method* setCoordinateHere berfungsi untuk menyimpan kordinat dan alamat perangkat ke kordinat dan alamat lokasi asal dan lokasi tujuan. *method* ini memiliki parameter paramFor bertipe String yang akan digunakan sebagai masukan disimpannya lokasi perangkat.
- 10 10. *method* reset berfungsi untuk memasang kembali lokasi asal dan lokasi tujuan.

27 4.2.9 Kelas *Map*

28 *Map* merupakan kelas yang akan mendapatkan titik yang ditunjuk pengguna pada peta
29 lalu menerjemahkannya dalam bentuk titik kordinat. Berikut adalah penjelasan atribut-
30 atribut yang dimiliki kelas ini:

- 31 1. IFinder bertipe LocationFinder digunakan untuk menampung semua informasi mengenai lokasi.
- 33 2. fromMapFor bertipe string digunakan sebagai indikator lokasi asal atau lokasi tujuan yang didapatkan dari map.

35 Berikut adalah penjelasan beberapa *method* yang dimiliki kelas ini:

- 36 1. Konstruktor Map untuk inisialisasi dan penambahan *event* mengetuk pada peta.

- 1 2. *method* OnNavigatedTo berfungsi untuk mendapatkan masukan lokasi asal atau lokasi
2 tujuan yang akan ditentukan dari map untuk kemudian ditampung di Objek Location-
3 Finder. *method* ini memiliki sebuah parameter NavigationEventArgs.
- 4 3. *method* OnNavigatedFrom digunakan untuk mengirimkan *state* objek LocationFinder
5 saat berpindah kelas. *method* ini memiliki parameter NavigationEventArgs.
- 6 4. *method* ShowMyLocationOnTheMap digunakan untuk memberitahu dan menandai
7 lokasi perangkat.
- 8 5. *method* map_Tap berfungsi untuk menandai lokasi yang ditunjuk pengguna lalu me-
9 nerjemahkan lokasi yang ditunjuk pengguna pada peta dan mengirimnya ke kelas Lo-
10 cationFinder.
- 11 6. *method* pilihLokasi berfungsi berpindah ke kelas MainPage dan memberitahu kelas
12 MainPage bahwa lokasi sudah dipilih. *method* ini memiliki parameter objek tombol
13 dan event.

14 **4.2.10 Kelas *HttpClient***

15 *HttpClient* merupakan kelas bawaan Windows Phone untuk mengatur pengiriman dan
16 kembalian menggunakan protokol HTTP. Berikut adalah penjelasan *method* kelas *HttpClient*
17 yang dipakai untuk perancangan aplikasi ini:

- 18 1. *method* GetStringAsync membutuhkan parameter alamat bertipe string dan mengem-
19 balikan kembalian dari Kiri dalam bentuk Task<string>.

20 **4.2.11 Kelas *JsonConvert***

21 *JsonConvert* merupakan kelas yang menyediakan *method* untuk mengkonversi berbagai
22 jenis komponen *common language runtime* dan *JSON*. Kelas ini merupakan bagian *namespa-*
23 ce *Newtonsoft*. Berikut adalah penjelasan *method* yang dipakai untuk perancangan aplikasi:

- 24 1. *method* DeserializeObject berfungsi untuk konversi dari bentuk string menjadi objek.
25 *method* ini memiliki satu parameter bertipe string lalu mengembalikan string tersebut
26 dalam bentuk objek.

27 **4.2.12 Kelas *Protocol***

28 *Protocol* merupakan kelas untuk menampung semua alamat dalam pengiriman menggu-
29 nakan protokol HTTP. Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

- 30 1. uri_version bertipe string digunakan untuk menyimpan nama dari parameter uri.
- 31 2. uri_mode bertipe string digunakan untuk menyimpan nama dari parameter mode.
- 32 3. uri_locale bertipe string digunakan untuk menyimpan nama dari parameter locale.
- 33 4. uri_start bertipe string digunakan untuk menyimpan nama dari parameter start.
- 34 5. uri_finish bertipe string digunakan untuk menyimpan nama dari parameter finish.

- 1 6. uri_presentation bertipe string digunakan untuk menyimpan nama dari parameter
2 presentation.
 - 3 7. uri_apikey bertipe string digunakan untuk menyimpan nama dari parameter apikey.
 - 4 8. uri_region bertipe string digunakan untuk menyimpan nama dari parameter region.
 - 5 9. uri_query bertipe string digunakan untuk menyimpan nama dari parameter query.
 - 6 10. apiKey bertipe string digunakan untuk menyimpan nilai kunci API untuk mengirim
7 permintaan ke Kiri.
 - 8 11. hostname bertipe string digunakan untuk digunakan untuk menyimpan alamat host
9 dari Kiri.
 - 10 12. handle bertipe string digunakan untuk menyimpan alamat host ditambah "handle.php".
 - 11 13. iconPath bertipe string digunakan untuk menyimpan lokasi gambar yang dibutuhkan.
 - 12 14. iconStart bertipe string digunakan untuk menyimpan lokasi gambar awal perjalanan
13 dari lokasi awal.
 - 14 15. iconFinish bertipe string digunakan untuk menyimpan lokasi gambar akhir perjalanan
15 ke lokasi tujuan.
 - 16 16. version_2 bertipe string digunakan untuk menyimpan nilai versi dari API yang di-
17 gunaikan (saat pembuatan penelitian ini versi Kiri API yang digunakan adalah versi
18 2).
 - 19 17. modeFind bertipe string yang digunakan untuk menyimpan nilai "searchplace" yang
20 merupakan mode mencari lokasi terkait pada Kiri API.
 - 21 18. modeRoute bertipe string yang digunakan untuk menyimpan nilai "findroute" yang
22 merupakan mode mencari rute pada Kiri API.
 - 23 19. modeNearby bertipe string yang digunakan untuk menyimpan nilai "nearbytransport"
24 " yang merupakan mode mencari lokasi terdekat pada Kiri API.
 - 25 20. localeId bertipe string yang digunakan untuk menyimpan nilai bahasa jika kembalian
26 yang diinginkan ingin berbahasa Indonesia.
 - 27 21. localeEn bertipe string yang digunakan untuk menyimpan nilai bahasa jika kembalian
28 yang diinginkan ingin berbahasa Inggris.
 - 29 22. presentationMobile bertipe string yang digunakan untuk menyimpan nilai penyajian
30 untuk perangkat *mobile*.
 - 31 23. presentationDesktop bertipe string yang digunakan untuk menyimpan nilai penyajian
32 untuk perangkat *desktop*.
- 33 Berikut adalah penjelasan beberapa *method* yang dimiliki kelas ini:

1. *method* getTypeTransport merupakan *method* yang akan mengembalikan alamat dari gambar transportasi dengan bingkai. *method* ini memiliki 2 parameter yaitu means sebagai tipe transportasi dan meansDetail sebagai nama kendaraan.
4. *method* getTypeTransportWOBaloon merupakan *method* yang akan mengembalikan alamat dari gambar transportasi tanpa bingkai tambahan. *method* ini memiliki 2 parameter yaitu means sebagai tipe transportasi dan meansDetail sebagai nama kendaraan.
7. getSearchPlace merupakan *method* yang akan mengembalikan URI pencarian lokasi sesuai parameter. Parameter yang dimaksud adalah kata kunci masukan pengguna.
9. *method* getFindRoute merupakan *method* yang akan mengembalikan URI pencarian rute sesuai parameter. Parameter yang dimaksud adalah kordinat lokasi asal dan kordinat lokasi tujuan yang bertipe string.
12. *method* getRequestSearch digunakan untuk mendapatkan lokasi terkait sesuai masukan pengguna. *method* ini akan mengembalikan Task<RootObjectSearchPlace> karena menggunakan operasi *asynchronous*. *method* ini memiliki parameter kata kunci masukan pengguna dan kota yang masing-masing parameter bertipe String.
16. *method* getRequestRoute digunakan untuk mendapatkan rute sesuai lokasi asal dan lokasi tujuan. *method* ini akan mengembalikan Task<RootObjectFindRoute> karena menggunakan operasi *asynchronous*. *method* ini memiliki parameter *latitude* lokasi asal, *longitude* lokasi asal, *latitude* lokasi tujuan, dan *longitude* lokasi tujuan yang masing-masing bertipe double.

21 4.2.13 Kelas *RootObjectSearchPlace*

22 *RootObjectSearchPlace* merupakan kelas untuk menampung objek hasil pencarian lokasi. Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

24. 1. status bertipe *string* digunakan untuk menampung hasil kembalian status dari Kiri.
25. 2. searchresult bertipe *list* dan menampung banyak objek *SearchResult*.
26. 3. attributions bertipe objek untuk menampung attributions.

27 4.2.14 Kelas *SearchResult*

28 *SearchResult* merupakan kelas untuk menampung nama tempat dan kordinat dari nama tempat tersebut. Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

30. 1. placename bertipe *string* digunakan untuk menampung nama tempat.
31. 2. location bertipe *string* digunakan untuk menampung nama tempat.

32 4.2.15 Kelas *RootObjectFindRoute*

33 *RootObjectFindRoute* merupakan kelas untuk menampung hasil pencarian rute. Berikut 34 adalah penjelasan atribut-atribut yang dimiliki kelas ini:

35. 1. status

1 2. routingresults

2 **4.2.16 Kelas *RoutingResult***

3 *RoutingResult* merupakan kelas untuk menampung langkah menuju tempat tujuan dan
4 waktu yang dibutuhkan. Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

5 1. steps

6 2. traveltimes

7 **4.2.17 Kelas *Route***

8 *Route* merupakan kelas untuk pencarian rute dan menampilkannya kepada pengguna.

9 Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

10 1. listBoxStatus bertipe boolean digunakan untuk menentukan nilai status rute dalam
11 bentuk daftar sedang tertutup atau terbuka.

12 2. IFinder bertipe LocationFinder digunakan untuk menampung semua informasi meng-
13 enai lokasi.

14 3. arrayFocus bertipe *array of Point* digunakan untuk menampung titik fokus perubahan
15 jenis transportasi yang digunakan pengguna.

16 4. detailRoute bertipe *array of String* digunakan untuk menampung keterangan yang
17 dibutuhkan pengguna dari Kiri API.

18 5. focusPointNumber bertipe *integer* digunakan untuk menentukan *index of* dari atribut
19 arrayFocus dan detailRoute.

20 6. routeLayer bertipe MapLayer digunakan untuk menampung *Polyline* dan *Pushpin*

21 7. myLocationLayer bertipe MapLayer digunakan untuk menampung titik dari lokasi
22 perangakt berada.

23 8. p bertipe Protocol digunakan untuk menampung permintaan data dengan Kiri API.

24 9. geolocator bertipe Geolocator untuk menangani perubahan lokasi yang terjadi.

25 10. newTimer bertipe DispatcherTimer digunakan untuk membuat perngatur waktu.

26 11. timeOut bertipe boolean digunakan untuk menentukan nilai status apakah waktu un-
27 tuk satu proses sudah mencapai batas.

28 12. backgroundWorker bertipe BackgroundWorker digunakan untuk menangain proses di
29 belakang layar.

30 Berikut adalah penjelasan beberapa *method* yang dimiliki kelas ini:

31 1. Konstruktor Route digunakan untuk memuat komponen yang ada di halaman Route,
32 inisialisasi atribu, dan pemanggilan backgroundWorker.

- 1 2. *method* OnNavigatedTo digunakan untuk mendapatkan objek LocationFinder dan me-
2 manggil *method* TrackLocation. *method* ini memiliki parameter NavigationEventArgs.
- 3 3. *method* OnNavigatedFrom digunakan untuk mengirimkan *state* objek LocationFinder
4 saat berpindah kelas. *method* ini memiliki parameter NavigationEventArgs.
- 5 4. *method* Application_Deactivated digunakan untuk menyimpan *state* objek saat me-
6 ninggalkan aplikasi. *method* ini memiliki parameter object dan DeactivatedEventArgs.
- 7 5. *method* TrackLocation digunakan untuk inisialisasi GeoLocator dan memulai pe-
8 lacakan lokasi terus menerus.
- 9 6. *method* geolocator_StatusChanged digunakan untuk mengetahui status GeoLocator.
- 10 7. *method* geolocator_PositionChanged digunakan untuk mengetahui perubahan lokasi
11 yang terjadi dan menyimpannya di kelas LocationFinder.
- 12 8. *method* Find digunakan untuk mencari rute yang dicari pengguna.
- 13 9. *method* setRouteToMap digunakan untuk menggambar rute dan lokasi pada *layer* pe-
14 ta. *method* ini memiliki parameter RootObjectFindRoute yang merupakan objek un-
15 tuk pencarian rute.
- 16 10. *method* setFocus digunakan untuk mengarahkan pusat pandangan ke titik lokasi per-
17 gantian jenis transportasi. *method* ini memiliki parameter objek dan RoutedEventArgs.
- 19 11. *method* toMyLocation digunakan untuk mengarahkan pusat pandangan ke titik lokasi
20 perangakat berada. *method* ini memiliki parameter objek dan RoutedEventArgs.
- 21 12. *method* ShowList digunakan untuk membuka dan menutup rute dalam bentuk daftar.
22 *method* ini memiliki parameter objek dan RoutedEventArgs.
- 23 13. *method* createNew digunakan untuk membuat objek Pushpins. *method* ini memiliki
24 parameter uri bertipe String, transport bertipe String yang menandakan jenis trans-
25 portasi, dan geoCoordinate bertipe GeoCoordinate sebagai lokasi dari Pushpins.
- 26 14. *method* drawMyLocationOnTheMap digunakan untuk membuat penanda lokasi di la-
27 pisan myLocationLayer pada peta. *method* ini memiliki parameter latitude bertipe
28 double dan longitude bertipe double.
- 29 15. *method* back digunakan untuk konfirmasi ke pengguna jika pengguna ingin mening-
30 galkan aplikasi. *method* ini memiliki dua buah parameter sender bertipe objek dan e
31 bertipe RoutedEventArgs.
- 32 16. *method* ShowLoading digunakan untuk memunculkan *popup* menunggu.
- 33 17. *method* StartLoadingData digunakan untuk pemanggilan BackgroundWorker.
- 34 18. *method* backgroundWorker_DoWork digunakan untuk eksekusi *method* dengan Bac-
35 kgroundWorker. *method* ini memiliki dua buah parameter sender bertipe objek dan e
36 bertipe DoWorkEventArgs.

19. *method* backgroundWorker_RunWorkerCompleted digunakan untuk menutup *popup* menunggu jika semua *method* sudah selesai dijalankan. *method* ini memiliki dua buah parameter sender bertipe objek dan e bertipe RunWorkerCompletedEventArgs.
20. *method* OnBackKeyPress digunakan untuk konfirmasi ke pengguna jika pengguna ingin meninggalkan aplikasi dengan menekan tombol "back". *method* ini memiliki parameter e bertipe CancelEventArgs.

7 4.3 Perancangan Antar Muka

8 Pada sub bab ini akan dibahas mengenai antarmuka pada aplikasi Pencari Rute Kendaraan Umum untuk Windows Phone. Antarmuka berfungsi sebagai jembatan yang menghubungkan antara aplikasi dengan pengguna. Berikut ini akan dijelaskan mengenai rancangan antarmuka aplikasi Pencari Rute Kendaraan Umum untuk Windows Phone.

12 4.3.1 Antarmuka Kelas *MainPage*



Gambar 4.14: Antarmuka *MainPage*

13 Antarmuka Kelas Map pada gambar 4.14 merupakan tampilan awal saat aplikasi
14 dijalankan. Antarmuka Kelas Map memiliki dua buah masukan, lima buah tombol, dan satu
15 menu daftar. Berikut adalah detailnya.

16 Dua buah masukan yaitu.

- 17 • Masukan lokasi asal

18 Merupakan masukan lokasi asal mula pengguna ingin melakukan perjalanan.

- 19 • Masukan lokasi tujuan

20 Merupakan masukan lokasi tujuan berhentinya perjalanan.

21 Lima buah tombol yaitu.

- 22 • Tombol map untuk lokasi asal

23 Jika tombol ditekan maka akan berpindah ke kelas map untuk memilih lokasi asal
24 di peta. Jika di kelas Map pengguna memilih lokasi maka pada masukan lokasi asal
25 terdapat tulisan "Maps".

- 1 • Tombol here untuk lokasi asal
- 2 Jika tombol ditekan maka lokasi asal adalah lokasi perangkat saat tombol ditekan dan masukan lokasi asal menjadi "here".
- 4 • Tombol map untuk lokasi tujuan
- 5 Jika tombol ditekan maka akan berpindah ke kelas map untuk memilih lokasi tujuan di peta. Jika di kelas Map pengguna memilih lokasi maka pada masukan lokasi tujuan terdapat tulisan "Maps".
- 8 • Tombol here untuk lokasi tujuan
- 9 Jika tombol ditekan maka lokasi tujuan adalah lokasi perangkat saat tombol ditekan dan masukan lokasi tujuan menjadi "here".
- 11 • Tombol find
- 12 Jika tombol ditekan maka akan menampilkan daftar tempat asal dan tempat tujuan lalu mengarahkan ke Kelas Route.
- 14 Satu buah daftar yaitu.
- 15 • Daftar kota yang tersedia
- 16 Merupakan daftar kota yang tersedia (kota yang rute angkutan umumnya dapat dimunculkan dengan aplikasi ini). Disaat aplikasi dijalankan maka daftar akan menunjuk ke kota terdekat tempat perangkat berada.

Pilih Tempat

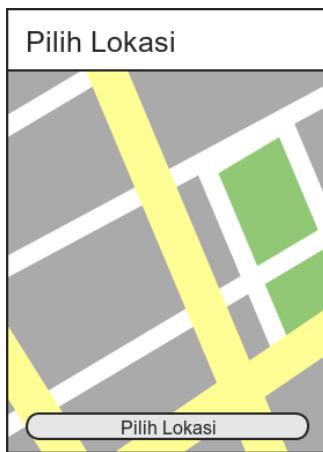
Kota 1
 Kota 2
 Kota 3
 Kota 4
 Kota 5

Gambar 4.15: Antarmuka Daftar Tempat

19 Antarmuka Daftar Tempat pada gambar 4.15 merupakan daftar yang akan dimunculkan jika pengguna memasukan kata kunci pada pencarian tempat asal dan tempat tujuan.
 20 Daftar akan muncul jika didapat kembalian hasil pencarian lebih dari satu.

22 **4.3.2 Antarmuka Kelas Map**

23 Antarmuka Kelas Map pada gambar 4.16 merupakan antarmuka untuk menunjuk lokasi pada peta. Terdapat satu buah tombol yang akan dimunculkan jika pengguna sudah memilih lokasi. Jika tombol ditekan maka kordinat lokasi akan disimpan dan dikirim pada kelas MainPage dan halaman akan diarahkan ke kelas MainPage.



Gambar 4.16: Antarmuka *Map*

¹ 4.3.3 Antarmuka Kelas *Route*



Gambar 4.17: Antarmuka *Route*

² Antarmuka Kelas Route pada gambar [4.17](#) merupakan antarmuka untuk melihat rute
³ dari lokasi asal ke lokasi tujuan dalam bentuk daftar maupun peta. Terdapat empat buah
⁴ tombol pada antarmuka Kelas Route. Berikut tombol yang terdapat pada Kelas Route.

- ⁵ • Tombol prev
⁶ Jika tombol ditekan maka akan menunjuk titik sebelumnya pada rute peta.
- ⁷ • Tombol next
⁸ Jika tombol ditekan maka akan menunjuk titik setelahnya pada rute peta.
- ⁹ • Tombol here
¹⁰ Jika tombol ditekan maka akan menunjuk lokasi perangkat berada pada peta.
- ¹¹ • Tombol Show List
¹² Jika tombol ditekan maka akan menunjuk atau menyembunyikan daftar rute.

¹³ Antarmuka rute dalam bentuk daftar pada gambar [4.18](#) merupakan antarmuka untuk
¹⁴ melihat rute secara lebih jelas dengan keterangan tahap demi tahap disertai jarak dan waktu
¹⁵ perjalanan. Antarmuka daftar dapat dilihat atau disembunyikan sesuai keinginan pengguna
¹⁶ namun saat kelas Rute dibuka antarmuka daftar rute akan disembunyikan.



Gambar 4.18: Antarmuka Rute dalam bentuk Daftar

BAB 5

IMPLEMENTASI DAN PENGUJIAN APLIKASI

³ Pada bab 5 akan dibahas implementasi dan pengujian aplikasi pencari rute kendaraan umum
⁴ untuk Windows Phone.

5.1 Implementasi

⁶ Pada sub bab ini akan dijelaskan mengenai lingkungan yang digunakan untuk membangun
⁷ aplikasi Pencari Rute Kendaraan Umum untuk Windows Phone. Pada lingkungan yang akan
⁸ dibahas juga penulis membangun aplikasi sesuai rancangan yang telah dibahas pada bab 4
⁹ dan mengujinya.

5.1.1 Perangkat Keras untuk Implementasi

¹⁰ Dalam membangun aplikasi ini perangkat keras yang digunakan adalah sebagai berikut:

¹¹ 1. Komputer

- ¹² (a) Processor: intel Core i7-2620M CPU 2,7 GHz
- ¹³ (b) RAM: 4 GB
- ¹⁴ (c) Hardisk: 640 GB
- ¹⁵ (d) VGA: Intel HD 3000

¹⁶ 2. Perangkat Bergerak

- ¹⁷ (a) Processor: 1,2 GHz
- ¹⁸ (b) RAM: 1 GB
- ¹⁹ (c) ROM: 8 GB
- ²⁰ (d) Layar: 720 x 1280 pixel, 4,7 inch
- ²¹ (e) GPS
- ²² (f) Sensor: kompas, *accelerometer*

5.1.2 Perangkat Lunak untuk Implementasi

²³ Dalam membangun aplikasi ini perangkat lunak yang digunakan adalah sebagai berikut:

²⁴ 1. Komputer

- ²⁵ (a) Sistem Operasi Windows 8.1

- 1 (b) IDE Visual Studio Express 2012
 - 2 (c) Bahasa Pemrograman C#
 - 3 (d) Library .Net Framework 4.5
- 4 2. Perangkat Bergerak
- 5 (a) Sistem Operasi Windows Phone 8.1

6 5.1.3 Hasil Implementasi

7 Hasil implementasi dari perangkat lunak ini terbagi dalam tiga bagian, yaitu:

- 8 1. Kode Program
9 Kode Program pada perangkat lunak ditulis dengan menggunakan bahasa c#. Bahasa
10 C# dipilih berdasarkan analisa pada bab 3 dan kemampuan penulis.
- 11 2. Hasil kompilasi program
12 Hasil dari kompilasi program berupa file Kiri_Debug_AnyCPU.xap. File ini dapat
13 dipasang pada perangkat dengan sistem operasi Windows Phone versi 8 atau lebih
14 tinggi.
- 15 3. Antarmuka Aplikasi
16 Berikut merupakan hasil implementasi antarmuka aplikasi Pencari Rute Kendaraan
17 Umum untuk Windows phone.



Gambar 5.1: Gambar antarmuka kelas MainPage



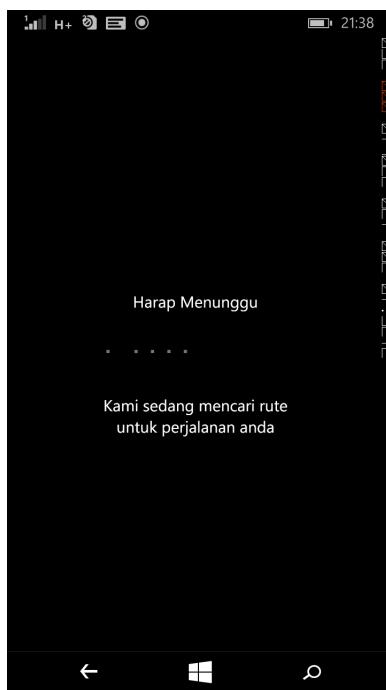
Gambar 5.2: Gambar antarmuka Splash di kelas MainPage



Gambar 5.3: Gambar antarmuka *list* asal dan *list* tujuan di kelas MainPage



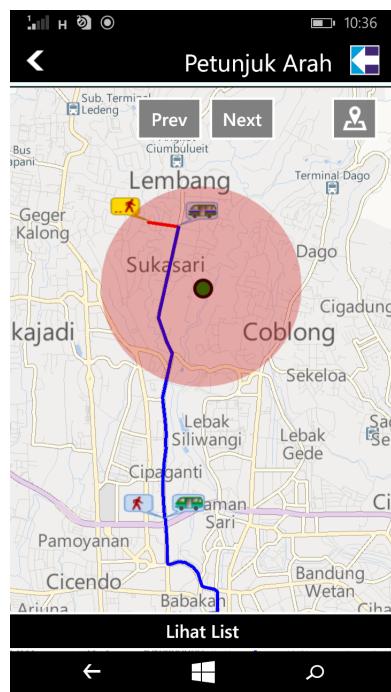
Gambar 5.4: Gambar antarmuka kelas Map



Gambar 5.5: Gambar antarmuka menunggu di kelas Route

¹ 5.2 Pengujian

² Pada bagian ini akan dibahas mengenai hasil pengujian yang telah dilakukan terhadap
³ aplikasi yang telah dibangun penulis. Pengujian yang dilakukan terdiri dari dua bagian
⁴ yaitu pengujian fungsional dan pengujian eksperimental. Pengujian fungsional bertujuan
⁵ untuk memastikan semua fungsi aplikasi berjalan sesuai harapan. Sementara pengujian
⁶ eksperimental bertujuan untuk mengetahui keberhasilan proses kerja dari aplikasi.



Gambar 5.6: Gambar antarmuka kelas Route



Gambar 5.7: Gambar antarmuka *list* di kelas Route

¹ 5.2.1 Lingkungan Pengujian

- ² Dalam proses pengujian perangkat lunak penulis menggunakan sistem operasi Windows Phone 8.1 dengan spesifikasi perangkat keras sebagai berikut.
- ⁴ 1. *Processor* : 1.2 Ghz Quad Core
- ⁵ 2. RAM : 1 GB
- ⁶ 3. Layar : 1280 x 720 pixels, 4,7 inch

- ¹ 4. GPS : A-GPS, GLONASS, Beidou

² 5.2.2 Pengujian Fungsional

³ Pengujian fungsional dilakukan untuk menguji kesesuaian reaksi yang terjadi dengan
⁴ reaksi yang diharapkan. Untuk menguji kesesuaian reaksi yang terjadi dengan reaksi yang
⁵ diharapkan penulis menguji 5 orang pengguna dengan perintah sebagai berikut.

- ⁶ 1. Carilah rute dari sini ke BIP!
 - ⁷ 2. Carilah rute dari BIP ke Ciwalk!
 - ⁸ 3. Carilah rute dari sini ke ITB pintu jalan Ganesa!
- ⁹ Dari tiga perintah berikut didapat hasil sebagai berikut.

Pengguna	Perintah	Aksi Pengguna	Reaksi yang Diharapkan
1	1	Menekan tombol "Here" dan menulis "BIP" pada TextBox	sesuai
	2	Menulis "BIP" pada TextBox dan menulis "Ciwalk" pada TextBox	sesuai
	3	Menekan tombol "Here" dan menulis "jl Ganesa" pada TextBox	tidak sesuai
2	1	Menekan tombol "Here" dan menulis "BIP" pada TextBox	sesuai
	2	Menulis "BIP" pada TextBox dan menulis "Ciwalk" pada TextBox	sesuai
	3	Menekan tombol "Here" dan menunjuk pada peta	sesuai
3	1	Menekan tombol "Here" dan menulis "BIP" pada TextBox	sesuai
	2	Menulis "BIP" pada TextBox dan menulis "Ciwalk" pada TextBox	sesuai
	3	Menekan tombol "Here" dan menulis "ITB Ganeca" pada TextBox lalu menyadari tombol map dan menggunakannya	sesuai
4	1	Menekan tombol "Here" dan menulis "BIP" pada TextBox	sesuai
	2	Menulis "BIP" pada TextBox dan menulis "Ciwalk" pada TextBox	sesuai
	3	Menekan tombol "Here" dan menulis "Institut Teknologi Bandung" pada TextBox	tidak sesuai

Tabel 5.1: Tabel Hasil pengujian fungsionalitas terhadap pengguna

¹⁰ Penulis juga mencoba melakukan pengujian fungsionalitas untuk menguji aksi dan reaksi
¹¹ dari aplikasi. Hasil pengujian tersebut ditunjukkan pada tabel 5.2.

1 Pada pengujian keadaan aplikasi dari keadaan *running* ke *dormant* tidak terjadi.
2 Pada saat aplikasi dalam keadaan *running* dan pengguna menekan tombol "windows" seha-
3 rusnya keadaan aplikasi berpindah dari keadaan *running* ke *dormant* tetapi keadaan aplikasi
4 malah *terminate*. Hal tersebut dikarenakan aplikasi yang penulis kembangkan masih dalam
5 mode debug.

6 **5.2.3 Pengujian Eksperimental**

7 Pengujian eksperimental dilakukan untuk mengetahui keberhasilan aplikasi yang dibu-
8 gun penulis. Berikut pengujian eksperimental yang dilakukan penulis.

9 Pengujian 1

- 10 • Tempat : Rumah(Jalan Kejaksaan menuju Unpar)
11 • Waktu : Pukul 9 pada tanggal 7 April 2014
12 • Pengalaman :

13 Pada pengujian 1 penulis memasukan alamat di dalam rumah untuk menuju Unpar.
14 Saat penulis memasukan kata Unpar muncul daftar tempat sesuai kata kunci "Un-
15 par", lalu penulis memilih "Universitas Katolik Parahyangan". Saat rute ditampilkan
16 ada ketidaktepatan lokasi yang ditunjukan untuk lokasi asal(rumah) sedangkan un-
17 tuk lokasi tujuan(Unpar) ditunjukan dengan tepat. Ketidaktepatan tersebut kira-kira
18 seratus meter dan lokasi ditandai lingkaran merah dengan ukuran besar. Hal tersebut
19 dikarenakan tempat tertutup yang membuat GPS tidak menunjukan lokasi yang
20 akurat. Tapi saat pengujian penulis berusaha mengikuti titik naik angkutan umum.
21 Penulis mulai dengan mengikuti petunjuk berjalan kaki menuju Jalan Tamblong. Saat
22 keluar rumah lingkaran merah disekitar lokasi perangkat mulai mengecil yang menan-
23 dakan akurasi GPS semakin baik. Angkutan kota yang penulis naiki pertama kali
24 adalah angkutan kota kalapa. Sambil berada di angkot penulis mengamati rute yang
25 ditunjukan aplikasi sudah sesuai dengan rute angkutan kota dan penunjuk lokasi me-
26 nunjukan pergerakan dengan akurat. Angkutan kota yang penulis naiki berhenti di
27 stasiun lalu penulis turun dan menuju Jalan Kebon Jukut. Di jalan kebon jukut pe-
28 nulis naik angkutan kota Ciumbuleuit - St. Hall. Selama perjalanan petunjuk rute
29 dan *polyline* sudah tepat sesuai rute angkutan kota. Angkutan kota ke dua langsung
30 mengantarkan penulis menuju Jalan Ciumbuleuit untuk selanjutnya penulis berjalan
31 menuju Unpar. Ketika sampai di Unpar, kordinat lokasi pada peta juga menunjukan
32 titik di Unpar dan memakan waktu 65 menit. Untuk perjalanan pulang penulis me-
33 mulai dengan mencari rute dari Unpar menuju ke rumah di jalan Kejaksaan dengan
34 mencari di peta. Perjalanan penulis mulai dengan naik angkot Ciumbuleuit - St. Hall
35 (lurus). Angkot Ciumbuleuit - St. Hall (lurus) penulis naiki dari jalan Ciumbuleuit
36 sampai jalan Cihampelas. Penulis turun di jalan Cihampelas dan naik angkot Kalapa
37 - Ledeng di jalan Cihampelas. Namun di jalan Wastukencana rute pada peta menun-
38 jukan rute menyusuri jalan Wastukencana lalu ke jalan Aceh, namun angkot malah
39 berbelok ke jalan Riau lalu ke jalan Purnawarman lalu ke jalan Aceh. Rute yang
40 benar yaitu rute angkot dan bukan pada peta. Dari jalan Aceh angkot menuju jalan
41 Sumatera lalu ke jalan Tamblong. Perjalannan penulispun berakhir setelah sampai di
1 rumah.

**1 5.2.4 Perbandingan Aplikasi Pencari Rute di Android dengan Aplikasi
2 Pencari Rute di Windows Phone**

3 Aplikasi Pencari rute yang penulis buat dengan yang ada di Android memiliki beberapa
4 perbedaan. Berikut perbedaan antara aplikasi pencari rute di Android dengan Windows
5 Phone.

No	Aksi Pengguna	Reaksi yang Diharapkan	Reaksi Perangkat lunak
1	Menjalankan aplikasi	Aplikasi menampilkan SplashScreen selama beberapa saat dan tampilan awal halaman MainPage ditampilkan	sesuai
2	Memasukan lokasi asal dan lokasi tujuan dari <i>textbox</i>	<i>Textbox</i> terisi sesuai masukan	sesuai
3	Memasukan lokasi asal atau lokasi tujuan berdasarkan lokasi perangkat dengan menekan tombol "here"	<i>Textbox</i> lokasi asal atau lokasi tujuan terisi "here"	sesuai
4	Menekan tombol "maps" untuk memilih lokasi pada peta	Pindah halaman ke kelas Map	sesuai
5	Menekan lokasi pada peta lalu menekan tombol "pilih lokasi"	Lokasi ditandai dan pengguna diarahkan kembali ke kelas Main Page	sesuai
6	Memilih kota	Kota berubah sesuai yang dipilih pengguna	sesuai
7	Menekan tombol "Find"	Bila lokasi diambil dari peta dan lokasi perangkat maka tidak akan tampil <i>ListBox</i> dan antarmuka dialihkan ke kelas Route	sesuai
8		Bila input masukan berupa kata kunci tempat maka akan tampil <i>ListBox</i> untuk dipilih, lalu setelah dipilih antarmuka dialihkan ke kelas Route	sesuai
9	Bila <i>ListBox</i> ditampilkan dan pengguna memilih	<i>ListBox</i> akan tertutup	sesuai
10	Pada kelas Route pengguna menekan tombol "here"	Pusat peta akan diarahkan ke lokasi pengguna berada	sesuai
11	Pada kelas Route pengguna menekan tombol "Tunjukan Daftar"	Jika daftar tertutup maka daftar akan terbuka	sesuai
12		Jika daftar terbuka maka daftar akan tertutup	sesuai
13	Pada kelas Route pengguna menekan tombol "Next"	Pusat peta akan diarahkan ke pertemuan transportasi berikutnya se-sua kembalian Kiri disertai keterangan	sesuai
14	Pada kelas Route pengguna menekan tombol "Prev"	Pusat peta akan diarahkan ke pertemuan transportasi sebelumnya se-sua kembalian Kiri disertai keterangan	sesuai
15	Pada kelas Route pengguna menekan tombol "windows"	Keadaan aplikasi menjadi <i>dormant</i>	tidak sesuai

Tabel 5.2: Tabel Hasil pengujian fungsionalitas

Perbedaan	Android	Windows Phone
Mendapatkan Lokasi	Memanfaatkan pemanggilan lokasi secara berulang-ulang dan akan mengunah lokasi dari yang kurang tepat menjadi lebih tepat.	Memanfaatkan pemanggilan sampai mendapat akurasi yang cukup tepat.
Map	Google Map	Windows Phone Map
Map	Google Map	Windows Phone Map

Tabel 5.3: Tabel perbedaan

6

BAB 6

7

KESIMPULAN DAN SARAN

8 Bab ini berisi kesimpulan dari pembangunan aplikasi beserta saran untuk pengembangan
9 selanjutnya.

10 6.1 Kesimpulan

11 Dari hasil penelitian penulis terhadap perancangan aplikasi pencari rute kendaraan
12 umum didapat kesimpulan sebagai berikut.

- 13 1. Pengguna dapat memasukan lokasi berdasarkan peta, lokasi perangkat, dan dengan
14 kata kunci.
- 15 2. Penggunaan Kiri API sudah dapat digunakan untuk mencari rute angkutan umum.
- 16 3. Akses internet mempengaruhi performansi mendapatkan rute.

17 6.2 Saran

18 Berdasarkan hasil kesimpulan yang telah dipaparkan, penulis memberi saran sebagai
19 berikut.

- 20 1. Memanfaatkan tombol "enter" untuk memanggil *method startRoute*. Masukan dari
21 pengguna karena setelah pengguna menentukan tujuan tombol "Find" terhalang *key-*
22 *board* maka pengguna harus menekan tombol "back" terlebih dahulu untuk menekan
23 tombol "Find".
- 24 2. Memanfaatkan pencarian lokasi pada peta agar pengguna dapat memilih lokasi dengan
1 lebih mudah.
- 2 3. Menambahkan *gesture* dan animasi yang akan meningkatkan interaksi pengguna ter-
3 hadap aplikasi.
- 4 4. Mengubah beberapa tampilan yang menjadi ciri khas Windows Phone seperti meng-
5 gunakan *application bar*.

DAFTAR REFERENSI

- 7 [1] "Windows phone silverlight development." <http://msdn.microsoft.com/library/windows/apps/ff402535.aspx>, 2014. Accessed: 2014-8-17.
- 8
- 9 [2] P. Manning, *Pro Windows Phone App Development*. Apress, 2013.
- 10 [3] A. Whitechapel and S. McKenna, *Windows Phone 8 Development Internals*. Microsoft,
- 11 2013.
- 12 [4] T. Bray, "The JavaScript Object Notation (JSON) Data Interchange Format." RFC 7159
- 13 (Proposed Standard), Mar. 2014.
- 1 [5] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifier (URI): Gene-
- 2 ric Syntax." RFC 3986 (INTERNET STANDARD), Jan. 2005. Updated by RFCs 6874,
- 3 7320.
- 4 [6] "Kiri api v2 documentation." https://bitbucket.org/projectkiri/kiri_api/wiki/KIRI%20API%20v2%20Documentation, 2014. Accessed: 2014-8-17.
- 5

LAMPIRAN A

KODE PROGRAM KELAS MAINPAGE

Listing A.1: MainPage.xaml.cs

```
8  using System;
9  using System.Collections.Generic;
10 using System.Linq;
11 using System.Net;
12 using System.Windows;
13 using System.Windows.Controls;
14 using System.Windows.Navigation;
15 using Microsoft.Phone.Controls;
16 using Microsoft.Phone.Shell;
17 using Kiri.Resources;
18 using System.Net.Http;
19 using System.Threading.Tasks;
20 using System.Windows.Input;
21 using Windows.Devices.Geolocation;
22 using System.IO.IsolatedStorage;
23 using System.Windows.Media;
24 using System.IO;
25 using System.Runtime.Serialization.Json;
26 using System.Text;
27 using Newtonsoft.Json;
28
29 using System.Windows.Controls.Primitives;
30 using System.ComponentModel;
31 using System.Threading;
32
33 using System.Device.Location; //ilangin kalo buat kelas city
34
35 namespace Kiri
36 {
37     public partial class MainPage : PhoneApplicationPage
38     {
39         // Constructor
40         private Protocol protocol;
41         private LocationFinder lFinder;
42         private HttpClient httpClient = new HttpClient();
43         private City c;
44         private String myCity;
45
46         private BackgroundWorker backgroundWorker;
47         public String test = "aaa";
48
49         public MainPage()
50     {
51         InitializeComponent();
52         this.lFinder = new LocationFinder();
53         this.c = new City();
54         this.cmbCurrFrom.ItemsSource = c.city;
55         this.protocol = new Protocol();
56         ShowSplash();
57     }
58
59     protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
60     {
61         base.OnNavigatedTo(e);
62         if (PhoneApplicationService.Current.State.ContainsKey("location"))
63     {
64         this.lFinder = (LocationFinder)PhoneApplicationService.Current.State["location"];
65         //this.lFinder = (LocationFinder)PhoneApplicationService.Current.State["location"];
66         if (lFinder.coorLongFrom != 0.0)
67         {
68             fromBox.Text = lFinder.coorLongFrom + "aaa";
69         }
70         if (lFinder.coorLongTo != 0.0)
71         {
72             fromBox.Text = lFinder.coorLongTo + "";
73         }
74
75         //check form
76         if (lFinder.coorLatFrom == 0.0 && lFinder.coorLongFrom == 0.0 && !lFinder.
77             addressFrom.Equals("Maps") && !lFinder.addressFrom.Equals("Here"))
78         {
79             fromBox.Text = lFinder.addressFrom;
80         }
81         else if (lFinder.coorLatFrom != 0.0 && lFinder.coorLongFrom != 0.0)
82         {
83             fromBox.Text = lFinder.addressFrom;
84         }
85     }
86 }
```

```

6         if (lFinder.addressFrom.Equals("Here"))
7         {
8             fromBox.Text = "Here";
9         }
10        else
11        {
12            fromBox.Text = "Maps";
13        }
14    }
15    //System.Diagnostics.Debug.WriteLine("stuff");
16    if (lFinder.coorLatTo == 0.0 && lFinder.coorLongTo == 0.0 && !lFinder.addressTo.
17        Equals("Maps") && lFinder.addressTo.Equals("Here"))
18    {
19        toBox.Text = lFinder.addressTo;
20    }
21    else if (lFinder.coorLatTo != 0.0 && lFinder.coorLongTo != 0.0)
22    {
23        if (lFinder.addressTo.Equals("Here"))
24        {
25            toBox.Text = "Here";
26        }
27        else
28        {
29            toBox.Text = "Maps";
30        }
31    }
32    string forMaps = "";
33    if (NavigationContext.QueryString.TryGetValue("for", out forMaps)) ;
34    if (forMaps!=null)
35    {
36        if (forMaps.Equals("from"))
37        {
38            fromBox.Text = "Maps";
39            //lFinder.addressFrom = "Maps";
40        }
41        else
42        {
43            toBox.Text = "Maps";
44            //lFinder.addressTo = "Maps";
45        }
46    }
47 }
48 }
49
50 protected override void OnNavigatedFrom(NavigationEventArgs e)
51 {
52     base.OnNavigatedFrom(e);
53     PhoneApplicationService.Current.State["location"] = lFinder;
54     State["lFinder"] = lFinder;
55 }
56
57 private void Application_Deactivated(object sender, DeactivatedEventArgs e){
58     PhoneApplicationService.Current.State["location"] = lFinder;
59 }
60
61 private void Application_Activated(object sender, ActivatedEventArgs e) {
62     //Console.WriteLine("Test");
63     if (PhoneApplicationService.Current.State.ContainsKey("location"))
64     {
65         this.lFinder = (LocationFinder)PhoneApplicationService.Current.State["location"];
66         this.findRoute();
67         /*
68         if (!lFinder.coorLongFrom != 0.0)
69         {
70             fromBox.Text = lFinder.coorLongFrom + "aaa";
71         }
72         if (!lFinder.coorLongTo != 0.0)
73         {
74             fromBox.Text = lFinder.coorLongTo + "";
75         }
76         */
77     }
78 }
79
80 private async void startRoute(object sender, RoutedEventArgs e)
81 {
82
83     String queryFrom = fromBox.Text;
84     String queryTo = toBox.Text;
85     RootObjectSearchPlace from = null; //Untuk Asal
86     RootObjectSearchPlace to = null; //Untuk Tujuan
87     //Penting!
88     //Dokumentasi zhttps://bitbucket.org/projectkiri/kiri_api/wiki/KIRI%20API%20v2%20
89     //Documentation
90     //Contoh searchplace zhttp://kiri.travel/handle.php?version=2&mode=searchplace&
91     //region=bdo&querystring=bip&apikey=97A7A1157A05ED6F
92     // zhttp://kiri.travel/handle.php?version=2&mode=searchplace&
93     //region=bdo&querystring=pvj&apikey=97A7A1157A05ED6F
94     //Contoh findroute zhttp://kiri.travel/handle.php?version=2&mode=findroute&locale
95     // =id&start=-6.90864,107.61108&finish=-6.88929,107.59574&presentation=mobile&apikey
96     // =97A7A1157A05ED6F
97     //Contoh findroute zhttp://kiri.travel/handle.php?version=2&mode=findroute&locale
98     // =id&start=-6.87474,107.60491&finish=-6.88909,107.59614&presentation=mobile&apikey
99     // =97A7A1157A05ED6F
100
101    //Check form
102    if (!queryFrom.Equals("") && !queryTo.Equals(""))
103    {
104        //Validate From
105        Boolean routeStatus = true;
106        progressFindPlace.IsIndeterminate = true;
107        //Get place from query

```

```

6     if ((!queryFrom.Equals("Here") || !queryFrom.Equals("Maps")) && (!Finder.
7         coorLatFrom == 0.0 && !Finder.coorLongFrom == 0.0)) //Check get location from
8         GPS
9     {
10        //Reference zhttps://msdn.microsoft.com/en-us/library/hh191443.aspx
11        //Task<string> requestFromTask = httpClient.GetStringAsync(new Uri(protocol.
12            getSearchPlace(queryFrom, myCity)));
13        //requestFrom = await requestFromTask;
14        //from = new RootObjectSearchPlace();
15        //from = JsonConvert.DeserializeObject<RootObjectSearchPlace>(requestFrom); //Mengubah String menjadi objek
16        from = await protocol.getRequestSearch(queryFrom, myCity); //Mengubah String
17        menjadi objek
18        if (from.searchresult.Count() == 0)
19        {
20            MessageBox.Show("Pencarian_untuk_kata_" + fromBox.Text + "_tidak_ditemukan");
21            routeStatus = false;
22        }
23    }
24    if ((!queryTo.Equals("Here") || !queryTo.Equals("Maps")) && (!Finder.coorLatTo ==
25        0.0 && !Finder.coorLongTo == 0.0)) //Check get location from GPS
26    {
27        //Task<string> requestToTask = httpClient.GetStringAsync(new Uri(protocol.
28            getSearchPlace(queryTo, myCity)));
29        //requestTo = await requestToTask;
30        //to = new RootObjectSearchPlace();
31        //to = JsonConvert.DeserializeObject<RootObjectSearchPlace>(requestTo); //Mengubah String
32        menjadi objek
33        to = await protocol.getRequestSearch(queryTo, myCity); //Mengubah String
34        menjadi objek
35        if (to.searchresult.Count() == 0)
36        {
37            MessageBox.Show("Pencarian_untuk_kata_" + toBox.Text + "_tidak_ditemukan");
38            routeStatus = false;
39        }
40    }
41    //Check Query
42    if (routeStatus == true)
43    {
44        if ((!Finder.coorLatFrom == 0.0 && !Finder.coorLongFrom == 0.0))
45        {
46            getListItem(from, "from"); //Show Listbox for location From
47        }
48        if ((!Finder.coorLatTo == 0.0 && !Finder.coorLongTo == 0.0))
49        {
50            getListItem(to, "to"); //Show Listbox for location To
51        }
52        this.findRoute();
53    }
54    progressFindPlace.Indeterminate = false;
55}
56}
57}
58}
59}
60}
61}
62}
63}
64}
65}
66}
67}
68}
69}
70}
71}
72}
73}
74}
75}
76}
77}
78}
79}
80}
81}
82}
83}
84}
85}
86}
87}
88}
89}
90}
91}
92}
93}
94}
95}
96}
97}
98}
99}
100}
101}
102}
103}
1}
2}
3}
4}
5}

```

```

6             else
7             {
8                 String locTo = request.searchresult[0].location.ToString();
9                 string[] coordinate = locTo.Split(',');
10                lFinder.coorLatTo = Double.Parse(coordinate[0]);
11                lFinder.coorLongTo = Double.Parse(coordinate[1]);
12            }
13            this.findRoute();
14        }
15    }
16    {
17        //LayoutRoot.Children.Add(getListItem(requestFrom));
18        if (forRequest.Equals("from"))
19        {
20            for (int c = 0; c < request.searchresult.Count; c++)
21            {
22                listPlaceFrom.Items.Add(request.searchresult[c].placename);
23            }
24            panelFrom.Visibility = Visibility.Visible;
25            listPlaceFrom.DataContext = request.searchresult;
26            listPlaceFrom.SelectionChanged += ListBoxSelectedPlace;
27        }
28        else
29        {
30            panelTo.Visibility = Visibility.Visible;
31            for (int c = 0; c < request.searchresult.Count; c++)
32            {
33                listPlaceTo.Items.Add(request.searchresult[c].placename);
34            }
35            listPlaceTo.DataContext = request.searchresult;
36            listPlaceTo.SelectionChanged += ListBoxSelectedPlace;
37        }
38    }
39}
40    else
41    {
42        MessageBox.Show("Error!");
43    }
44 });
45}
46
47 private void ListBoxSelectedPlace(object sender, SelectionChangedEventArgs e)
48 {
49    if (panelFrom.Visibility.Equals(Visibility.Visible))
50    { //Check visibility
51        if (null != (sender as ListBox).SelectedItem)
52        {
53            if ((sender as ListBox).SelectedIndex >= 0)
54            {
55                String locFrom = searchCoordinatePlace((List<Searchresult>)listPlaceFrom.
56                DataContext, listPlaceFrom.SelectedItem.ToString()); //((sender as
57                ListBox).SelectedItem as Searchresult).placename;
58                string[] coordinate = locFrom.Split(',');
59                lFinder.coorLatFrom = Double.Parse(coordinate[0]);
60                lFinder.coorLongFrom = Double.Parse(coordinate[1]);
61                lFinder.addressFrom = listPlaceFrom.SelectedItem.ToString();
62            }
63        }
64        panelFrom.Children.Clear();
65        panelFrom.Visibility = Visibility.Collapsed;
66        //panelTo.Visibility = Visibility.Visible;
67    }
68    else {
69        if (null != (sender as ListBox).SelectedItem)
70        {
71            if ((sender as ListBox).SelectedIndex >= 0)
72            {
73                String locTo = searchCoordinatePlace((List<Searchresult>)listPlaceTo.
74                DataContext, listPlaceTo.SelectedItem.ToString()); //((sender as
75                ListBox).SelectedItem as Searchresult).placename;
76                string[] coordinate = locTo.Split(',');
77                lFinder.coorLatTo = Double.Parse(coordinate[0]);
78                lFinder.coorLongTo = Double.Parse(coordinate[1]);
79                lFinder.addressTo = listPlaceTo.SelectedItem.ToString();
80            }
81        }
82        panelTo.Children.Clear();
83        panelTo.Visibility = Visibility.Collapsed;
84    }
85    this.findRoute();
86}
87
88 public string searchCoordinatePlace(List<Searchresult> listResult, string place) { //Pasti
89     ketemu
90     String coordinate = "0.0";
91     for (int c = 0; c < listResult.Count; c++)
92     {
93         if (place.Equals(listResult[c].placename))
94         {
95             coordinate = listResult[c].location;
96             c = listResult.Count;
97         }
98     }
99     return coordinate;
100}
101
102 public void findRoute()
103 {
1        if ((lFinder.coorLatFrom != 0.0 && lFinder.coorLongFrom != 0.0) && (lFinder.coorLatTo
1        != 0.0 && lFinder.coorLongTo != 0.0))
2        {
3            NavigationService.Navigate(new Uri("/Route.xaml", UriKind.Relative)); //start=" +
4            locationFrom + "&nameFrom=" + fromBox.Text + "&finish=" + locationTo + "&
5

```

```

6         nameTo=" + toBox.Text
7     }
8 }
9
10    private void changeCity(object sender, SelectionChangedEventArgs e)
11    {
12        if (!cmbCurrFrom.SelectedItem.Equals(null))
13        {
14            switch (cmbCurrFrom.SelectedItem.ToString())
15            {
16                case "Bandung":
17                    this.myCity="bdo";
18                    break;
19                case "Jakarta":
20                    this.myCity="cgk";
21                    break;
22                case "Malang":
23                    this.myCity="mlg";
24                    break;
25                case "Surabaya":
26                    this.myCity="sub";
27                    break;
28                default:
29                    this.myCity="bdo"; //Finder Auto
30                    break;
31            }
32        }
33    }
34
35    public void showCity()
36    {
37        int indexCity = -1;
38        while (indexCity == -1)
39        {
40            indexCity = c.getNearby(lFinder.coorLat, lFinder.coorLong);
41        }
42        this.myCity = c.cityCode[indexCity];
43    }
44
45    private void ShowSplash()
46    {
47        this.popup.IsOpen = true;
48        StartLoadingData();
49    }
50
51    private void StartLoadingData()
52    {
53        backgroundWorker = new BackgroundWorker();
54        backgroundWorker.DoWork += new DoWorkEventHandler(backgroundWorker_DoWork);
55        backgroundWorker.RunWorkerCompleted += new RunWorkerCompletedEventHandler(
56            backgroundWorker_RunWorkerCompleted);
57        backgroundWorker.RunWorkerAsync();
58    }
59
60    private void backgroundWorker_DoWork(object sender, DoWorkEventArgs e)
61    {
62        showCity();
63    }
64
65    private void backgroundWorker_RunWorkerCompleted(object sender,
66        RunWorkerCompletedEventArgs e)
67    {
68        this.Dispatcher.BeginInvoke(() =>
69        {
70            this.cmbCurrFrom.SelectedIndex = c.getIndexFromCityCode(myCity);
71            this.popup.IsOpen = false;
72        });
73    }
74 }

```

Listing A.2: MainPage.xaml

```

75 <!--Logo kiri from kiri.travel
76 Icon from modernuiicons.com
77 -->
78 <phone:PhoneApplicationPage
79 x:Class="Kiri.MainPage"
80 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
81 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
82 xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
83 xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
84 xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
85 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
86 xmlns:toolkit="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.
87 Toolkit"
88 mc:Ignorable="d"
89 FontFamily="{StaticResource PhoneFontFamilyNormal}"
90 FontSize="{StaticResource PhoneFontSizeNormal}"
91 Foreground="{StaticResource PhoneForegroundBrush}"
92 SupportedOrientations="Portrait" Orientation="Portrait"
93 shell:SystemTray.Visible="True">
94
95 <!--LayoutRoot is the root grid where all page content is placed-->
96 <Grid x:Name="LayoutRoot" Background="Transparent" Margin="0,0,0,0">
97     <Grid.RowDefinitions>
98         <RowDefinition Height="158"/>
99         <RowDefinition Height="3"/>
100        <RowDefinition Height="*"/>
101    </Grid.RowDefinitions>
102
103    <!-- LOCALIZATION NOTE:
104        To localize the displayed strings copy their values to appropriately named
105        resources in your application's resources file.
106    -->

```

```

6      keys in the app's neutral language resource file (AppResources.resx) then
7      replace the hard-coded text value between the attributes' quotation marks
8      with the binding clause whose path points to that string name.
9
10     For example:
11
12         Text="{Binding Path=LocalizedResources.ApplicationTitle, Source={StaticResource
13             LocalizedStrings}}"
14
15         This binding points to the template's string resource named "ApplicationTitle".
16
17         Adding supported languages in the Project Properties tab will create a
18         new resx file per language that can carry the translated values of your
19         UI strings. The binding in these examples will cause the value of the
20         attributes to be drawn from the .resx file that matches the
21         CurrentUICulture of the app at run time.
22         -->
23
24         <!--TitlePanel contains the name of the application and page title -->
25         <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="6,10,6,0" RenderTransformOrigin
26             ="0.5,0.5" Height="108" VerticalAlignment="Top">
27             <StackPanel.RenderTransform>
28                 <CompositeTransform SkewX="0.781" TranslateX="0.736" />
29             </StackPanel.RenderTransform>
30             <TextBlock Text="Public Transport" Style="{StaticResource PhoneTextNormalStyle}">
31                 Margin="12,0" />
32             <TextBlock Text="Kiri_WP_8" Margin="9,-7,0,0" Style="{StaticResource
33                 PhoneTextTitleStyle}" Height="111" />
34         </StackPanel>
35         <Popup x:Name="popup" Margin="6,0,-6,10" Grid.RowSpan="3">
36             <Grid Background="Black" Height="756" Width="484" />
37             <Image Source="/icons/kiri_logo.png" Margin="144,175,159,358" /
38                 RenderTransformOrigin="1.509,0.516" />
39             <ProgressBar Height="34" Width="481" IsIndeterminate="True" Margin
40                 ="-7,489,10,233" />
41         </Grid>
42     </Popup>
43     <!--ContentPanel-->
44     <Grid x:Name="ContentPanel" Margin="0,0,0,0" Grid.Row="1" Grid.RowSpan="2">
45         <TextBlock Text="From" Style="{StaticResource PhoneTextNormalStyle}" Margin
46             ="26,0,278,484" FontSize="50" />
47         <TextBlock Text="To" Style="{StaticResource PhoneTextNormalStyle}" Margin
48             ="26,137,361,274" FontSize="50" />
49         <TextBox x:Name="fromBox" HorizontalAlignment="Left" Margin="12,70,0,0" TextWrapping="Wrap"
50             VerticalAlignment="Top" FontSize="24" Width="325" Height="76" />
51         <TextBox x:Name="toBox" HorizontalAlignment="Left" Margin="12,209,0,0" TextWrapping="Wrap"
52             VerticalAlignment="Top" FontSize="24" Width="325" Height="76" />
53         <Button Content="FIND" HorizontalAlignment="Left" Margin="10,376,0,0" /
54             VerticalAlignment="Top" Click="startRoute" Height="81" Width="460" />
55         <Button HorizontalAlignment="Left" Margin="330,209,0,0" VerticalAlignment="Top" Click
56             ="changeMapTo" Height="77" Width="83" />
57         <Image Source="icons/map.png" Margin="-14,-12,0,0" Width="60" Height="60" />
58     </Button>
59     <Button HorizontalAlignment="Left" Margin="329,69,0,0" VerticalAlignment="Top" Click
60             ="changeMapFrom" Height="77" Width="83" />
61     <Image Source="icons/map.png" Margin="-14,-12,0,0" Width="60" Height="60" />
62     </Button>
63     <Button HorizontalAlignment="Left" Margin="393,210,0,0" VerticalAlignment="Top" Click
64             ="getHereTo" Height="77" Width="83" />
65     <Image Source="icons/marker.png" Margin="-14,-12,0,0" Width="60" Height="60" />
66     <Image />
67     </Button>
68     <Button HorizontalAlignment="Left" Margin="393,70,0,0" VerticalAlignment="Top" Click
69             ="getHereFrom" Height="77" Width="83" />
70     <Image Source="icons/marker.png" Margin="-14,-12,0,0" Width="60" Height="60" />
71     <Image />
72     </Button>
73     <TextBlock HorizontalAlignment="Left" Margin="20,315,0,0" TextWrapping="Wrap" Text="Anda berada di kota" VerticalAlignment="Top" />
74     <toolkit:ListPicker BorderThickness="0" ScrollViewer.VerticalScrollBarVisibility="Auto"
75             Margin="217,298,74,-45" Name="cmbCurrFrom" SelectionChanged="changeCity" Visibility="Visible"
76             >
77     </toolkit:ListPicker>
78     <ProgressBar x:Name="progressFindPlace" Background="Black" HorizontalAlignment="Left"
79             Height="19" Margin="0,-24,0,0" VerticalAlignment="Top" Width="456" RenderTransformOrigin
80             ="0.493,1.056" />
81
82     <StackPanel x:Name="panelFrom" HorizontalAlignment="Stretch" Canvas.ZIndex="10" /
83         Margin="0,-150,0,5" VerticalAlignment="Stretch" Visibility="Collapsed" Background="Black" />
84     <TextBlock x:Name="textFrom" TextWrapping="Wrap" Text="Pilih lokasi Asal" FontSize
85             ="40" />
86     <ListBox x:Name="listPlaceFrom" Padding="10" Height="700" FontSize="30" />
87     </StackPanel>
88
89     <StackPanel x:Name="panelTo" HorizontalAlignment="Stretch" Canvas.ZIndex="5" Margin
90             ="0,-150,0,5" VerticalAlignment="Stretch" Visibility="Collapsed" Background="Black" Grid.
91             ColumnSpan="2" />
92     <TextBlock x:Name="textTo" TextWrapping="Wrap" Text="Pilih lokasi Tujuan" FontSize
93             ="40" />
94     <ListBox x:Name="listPlaceTo" Padding="10" Height="700" FontSize="30" />
95     </StackPanel>
96
97
98
99
100    <!--Uncomment to see an alignment grid to help ensure your controls are
101        aligned on common boundaries. The image has a top margin of -32px to
102        account for the System Tray. Set this to 0 (or remove the margin altogether)
103        if the System Tray is hidden.
1
2        Before shipping remove this XAML and the image itself.-->
1    <!--<Image Source="/Assets/AlignmentGrid.png" VerticalAlignment="Top" Height="800" Width
2        ="480" Margin="0,-32,0,0" Grid.Row="0" Grid.RowSpan="2" IsHitTestVisible="False" />-->
3    </Grid>
```

4 |
5 </phone:PhoneApplicationPage>

LAMPIRAN B

KODE PROGRAM KELAS MAP

Listing B.1: Map.xaml.cs

```

8  using System;
9  using System.Collections.Generic;
10 using System.Linq;
11 using System.Net;
12 using System.Windows;
13 using System.Windows.Controls;
14 using System.Windows.Navigation;
15 using Microsoft.Phone.Controls;
16 using Microsoft.Phone.Shell;
17 using System.Device.Location;
18
19 using Windows.Devices.Geolocation; //Provides the Geocoordinate class.
20 using System.Windows.Media;
21 using System.Windows.Shapes;
22 using Microsoft.Phone.Maps.Controls;
23 using Microsoft.Phone.Maps.Toolkit;
24
25 namespace Kiri
26 {
27     public partial class Map : PhoneApplicationPage
28     {
29         private LocationFinder lFinder;
30         public string fromMapFor;
31
32         public Map()
33         {
34             this.lFinder = null;
35             InitializeComponent();
36             MyMap.Tap += new EventHandler<System.Windows.Input.GestureEventArgs>(map_Tap);
37         }
38
39         protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
40         {
41             base.OnNavigatedTo(e);
42             if (NavigationContext.QueryString.TryGetValue("fromMapFor", out fromMapFor)) ;
43             this.fromMapFor = fromMapFor + "";
44             if (PhoneApplicationService.Current.State.ContainsKey("location"))
45             {
46                 this.lFinder = (LocationFinder)PhoneApplicationService.Current.State["location"];
47             }
48             ShowMyLocationOnTheMap();
49         }
50
51         protected override void OnNavigatedFrom(NavigationEventArgs e)
52         {
53             base.OnNavigatedFrom(e);
54             PhoneApplicationService.Current.State["location"] = lFinder;
55         }
56
57         private void ShowMyLocationOnTheMap()
58         {
59
60             // Get my current location.
61             loadingFrom.Indeterminate = true;
62             GeoCoordinate myGeoCoordinate = new GeoCoordinate(lFinder.coorLat, lFinder.coorLong);
63             this.MyMap.Center = myGeoCoordinate;
64             this.MyMap.ZoomLevel = 13;
65             loadingFrom.Indeterminate = false;
66         }
67
68         private void map_Tap(object sender, System.Windows.Input.GestureEventArgs e)
69         {
70             ButtonPilih.Visibility = Visibility.Visible;
71             Point p = e.GetPosition(MyMap);
72             GeoCoordinate s = MyMap.ConvertViewportPointToGeoCoordinate(p);
73             MyMap.Layers.Clear();
74
75             Ellipse myCircle = new Ellipse();
76             myCircle.Stroke = new SolidColorBrush(Colors.Black);
77             myCircle.StrokeThickness = 4;
78             myCircle.Fill = new SolidColorBrush(Colors.Green);
79             myCircle.Height = 25;
80             myCircle.Width = 25;
81             myCircle.Opacity = 50;
82
83             MapOverlay myLocationOverlay = new MapOverlay();
84             myLocationOverlay.Content = myCircle;

```

```

6     myLocationOverlay.PositionOrigin = new Point(0, 0);
7     myLocationOverlay.GeoCoordinate = s;
8
9     if (fromMapFor.Equals("from"))
10    {
11        this.lFinder.GetCurrentCoordinate(s.Latitude, s.Longitude, "from");
12    }
13    else {
14        this.lFinder.GetCurrentCoordinate(s.Latitude, s.Longitude, "to");
15    }
16    MapLayer myLocationLayer = new MapLayer();
17    myLocationLayer.Add(myLocationOverlay);
18
19    MyMap.Layers.Add(myLocationLayer);
20}
21
22 private void pilihLokasi(object sender, RoutedEventArgs e)
23 {
24     if (fromMapFor.Equals("from"))
25     {
26         NavigationService.Navigate(new Uri("/ MainPage.xaml?for=from", UriKind.Relative));
27     }
28     else
29     {
30         NavigationService.Navigate(new Uri("/ MainPage.xaml?for=to", UriKind.Relative));
31     }
32 }
33 }
34 }
```

Listing B.2: Map.xaml

```

35 <phone:PhoneApplicationPage
36   x:Class="Kiri.Map"
37   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
38   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
39   xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
40   xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
41   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
42   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
43   xmlns:Controls="clr-namespace:Microsoft.Phone.Maps.Controls;assembly=Microsoft.Phone.Maps"
44   xmlns:toolkit="clr-namespace:Microsoft.Phone.Maps.Toolkit;assembly=Microsoft.Phone.Controls.
45   Toolkit"
46   FontFamily="{StaticResource PhoneFontFamilyNormal}"
47   FontSize="{StaticResource PhoneFontSizeNormal}"
48   Foreground="{StaticResource PhoneForegroundBrush}"
49   SupportedOrientations="Portrait" Orientation="Portrait"
50   mc:Ignorable="d"
51   shell:SystemTray.Visibile="true">
52
53 <!--LayoutRoot is the root grid where all page content is placed-->
54 <Grid x:Name="LayoutRoot" Background="Transparent">
55   <Grid.RowDefinitions>
56     <RowDefinition Height="Auto"/>
57     <RowDefinition Height="*"/>
58   </Grid.RowDefinitions>
59
60   <!--TitlePanel contains the name of the application and page title-->
61   <StackPanel Grid.Row="0" Margin="12,17,0,28">
62     <TextBlock Text="Kiri" Style="{StaticResource PhoneTextNormalStyle}"/>
63     <TextBlock Text="Pilih lokasi" FontSize="40" Margin="9,-7,0,0" Style="{StaticResource .
64     PhoneTextTitle1Style}"/>
65   </StackPanel>
66
67   <!--ContentPanel - place additional content here-->
68   <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
69
70     <Controls:Map x:Name="MyMap" ZoomLevel="14" Margin="0,10,-16,0">
71       </Controls:Map>
72       <Button x:Name="ButtonPilih" Content="Pilih Lokasi" HorizontalAlignment="Left" Margin=
73           "-12,586,-16,-9" VerticalAlignment="Top" Width="484" Click="pilihLokasi"
74           Visibility="Collapsed" Background="Black"/>
75     </Grid>
76     <ProgressBar x:Name="loadingFrom" Background="Black" HorizontalAlignment="Left" Height="16
77           " Margin="0,96,0,0" VerticalAlignment="Top" Width="480"/>
78   </Grid>
79
80 </phone:PhoneApplicationPage>
```

LAMPIRAN C

KODE PROGRAM KELAS ROUTE

Listing C.1: Route.xaml.cs

```

8  using System;
9  using System.Collections.Generic;
10 using System.Linq;
11 using System.Net;
12 using System.Windows;
13 using System.Windows.Controls;
14 using System.Windows.Navigation;
15 using Microsoft.Phone.Controls;
16 using Microsoft.Phone.Shell;
17
18 using System.Device.Location; // Provides the GeoCoordinate class.
19 using Windows.Devices.Geolocation; //Provides the Geocoordinate class.
20 using System.Windows.Media;
21 using System.Windows.Shapes;
22 using Microsoft.Phone.Maps.Controls;
23 using Microsoft.Phone.Maps.Toolkit;
24
25 //optional
26 using System.Threading.Tasks;
27 using System.Net.Http;
28 using System.IO;
29 using System.Runtime.Serialization.Json;
30 using System.Text;
31 using Newtonsoft.Json;
32 using System.Windows.Media.Imaging;
33 using System.ComponentModel;
34 using System.Windows.Controls.Primitives;
35 using System.Threading;
36 using System.IO.IsolatedStorage;
37 using System.Windows.Threading;
38 using System.Diagnostics;
39
40 namespace Kiri
41 {
42     public partial class Route : PhoneApplicationPage
43     {
44         //get location
45         //zhttps://msdn.microsoft.com/en-us/library/windows/apps/jj247548%28v=vs.105%29.aspx
46         private Boolean listBoxStatus; //true == show, false == hide
47         private LocationFinder lFinder;
48         private Point[] arrayFocus;
49         private String[] detailRoute;
50         private int focusPointNumber;
51         private MapLayer routeLayer;
52         private MapLayer myLocationLayer;
53         private Protocol p;
54         private Boolean routeReady;
55
56         private Geolocator geolocator = null;
57         private BackgroundWorker backgroundWorker;
58         private DispatcherTimer newTimer;
59         private Boolean timeOut = false;
60         //private Boolean routeReady;
61         //private AutoResetEvent _workerCompleted = new AutoResetEvent(false);
62         public string AppStatus = String.Empty;
63         public string SavedData = String.Empty;
64
65         public Route()
66         {
67             InitializeComponent();
68             this.lFinder = null;
69             this.arrayFocus = null;
70             this.detailRoute = null;
71             this.focusPointNumber = 0;
72             this.routeLayer = new MapLayer();
73             this.myLocationLayer = new MapLayer();
74             this.p = new Protocol();
75             this.routeReady = false;
76             this.newTimer = new DispatcherTimer();
77             newTimer.Interval = new TimeSpan(0, 0, 60);
78             newTimer.Tick += OnTimerTick;
79             newTimer.Start();
80
81             ShowLoading();
82             //PhoneApplicationService.Current.UserIdleDetectionMode = IdleDetectionMode.Disabled;
83             //this.StartLoadingData();
84             //startLocation();
85         }

```

```

6     }
7
8 //old zhttp://www.geekchamp.com/articles/understanding-the-windows-phone-location-service-
9 //how-to-get-current-gps-coordinates
10 //old zhttps://msdn.microsoft.com/en-us/library/windows/apps/ff431782(v=vs.105).aspx
11
12 //source zhttps://msdn.microsoft.com/en-us/library/windows/apps/ff626521%28v=vs.105%29.asp
13 protected override void OnNavigatedTo(System.Windows.NavigationEventArgs e)
14 {
15     //System.Diagnostics.Debug.WriteLine("route");
16     base.OnNavigatedTo(e);
17     //if (PhoneApplicationService.Current.State.ContainsKey("route"))
18     //{
19     //    this.lFinder = (LocationFinder)PhoneApplicationService.Current.State["route"];
20     //}
21     if (PhoneApplicationService.Current.State.ContainsKey("location"))
22     {
23         this.lFinder = (LocationFinder)PhoneApplicationService.Current.State["location"];
24     }
25     this.TrackLocation();
26     //this.route.Center = new GeoCoordinate(lFinder.coorLatFrom, lFinder.coorLongFrom);
27     //GeoCoordinate[] recLocation = new GeoCoordinate[]{new GeoCoordinate(lFinder.
28     //    coorLatFrom, lFinder.coorLongFrom),new GeoCoordinate(lFinder.coorLatTo, lFinder.
29     //    coorLongTo)};
30     //LocationRectangle lr = LocationRectangle.CreateBoundingRectangle(recLocation);
31     //this.route.SetView(new GeoCoordinate(lFinder.coorLatFrom, lFinder.coorLongFrom), 1,
32     //    MapAnimationKind.Linear);
33
34     //detect location
35     if (IsolatedStorageSettings.ApplicationSettings.Contains("LocationConsent"))
36     {
37         // User has opted in or out of Location
38         return;
39     }
40
41     //Find(); // After get start coordinate and finish coordinate then call Find Method
42 }
43
44 protected override void OnNavigatedFrom(NavigationEventArgs e)
45 {
46     base.OnNavigatedFrom(e);
47     this.lFinder.reset();
48     PhoneApplicationService.Current.State["location"] = lFinder;
49 }
50
51 private void Application_Deactivated(object sender, DeactivatedEventArgs e)
52 {
53     PhoneApplicationService.Current.State["location"] = lFinder;
54 }
55
56 private void TrackLocation()
57 {
58     geolocator = lFinder.geolocator;
59
60     geolocator.StatusChanged += geolocator_StatusChanged;
61     geolocator.PositionChanged += geolocator_PositionChanged;
62 }
63
64 void geolocator_StatusChanged(Geolocator sender, StatusChangedEventArgs args)
65 {
66     string status = "";
67     switch (args.Status)
68     {
69         case PositionStatus.Disabled:
70             // the application does not have the right capability or the location master
71             // switch is off
72             status = "location_is_disabled_in_phone_settings";
73             break;
74         case PositionStatus.Initializing:
75             // the geolocator started the tracking operation
76             status = "initializing";
77             break;
78         case PositionStatus.NoData:
79             // the location service was not able to acquire the location
80             status = "no_data";
81             break;
82         case PositionStatus.Ready:
83             // the location service is generating geopositions as specified by the
84             // tracking parameters
85             status = "ready";
86             break;
87         case PositionStatus.NotAvailable:
88             status = "not_available";
89             // not used in WindowsPhone, Windows desktop uses this value to signal that
90             // there is no hardware capable to acquire location information
91             break;
92         case PositionStatus.NotInitialized:
93             // the initial state of the geolocator, once the tracking operation is stopped
94             // by the user the geolocator moves back to this state
95             break;
96     }
97     System.Console.WriteLine(status);
98 }
99
100 void geolocator_PositionChanged(Geolocator sender, PositionChangedEventArgs args)
101 {
102     Dispatcher.BeginInvoke(() =>
103     {
104         //MessageBox.Show(args.Position.Coordinate.Latitude.ToString() + ", " + args.
105         //Position.Coordinate.Longitude.ToString());
106         drawMyLocationOnTheMap(args.Position.Coordinate.Latitude, args.Position.Coordinate
107         .Longitude);
108         lFinder.setPositionChanged(sender, args);
109     });
110 }

```

```

6     });
7 }
8
9     public async void Find()
10 {
11     //this.TrackLocation_Click();
12     Boolean status = true;
13     HttpClient httpClient = new HttpClient();
14
15     //p.getRequestRoute(lFinder.coorLatFrom, lFinder.coorLongFrom, lFinder.coorLatTo,
16     //lFinder.coorLongTo);
17     //String uri = p.getFindRoute(lFinder.coorLatFrom + "," + lFinder.coorLongFrom,
18     //lFinder.coorLatTo + "," + lFinder.coorLongTo);
19     //Task<String> requestRouteTask = httpClient.GetStringAsync(new Uri(uri));
20     //String requestRoute = await requestRouteTask;
21     //RootObjectFindRoute r = JsonConvert.DeserializeObject<RootObjectFindRoute>(
22     //requestRoute); //Mengubah String menjadi objek
23
24     RootObjectFindRoute r = await p.getRequestRoute(lFinder.coorLatFrom, lFinder.
25         coorLongFrom, lFinder.coorLatTo, lFinder.coorLongTo);
26     if (r.status.Equals("ok"))
27     {
28         //source zhttps://msdn.microsoft.com/en-us/library/windows/apps/xaml/dn792121.aspx
29         if (!r.routingresults[0].steps[0][3].Equals("Maaf, kami tidak dapat menemukan rute
30             transportasi publik untuk perjalanan Anda."))
31         {
32             this.setRouteToMap(r);
33         }
34         else
35         {
36             MessageBox.Show("Maaf, kami tidak dapat menemukan rute transportasi publik
37                 untuk perjalanan Anda.");
38             status = false;
39         }
40     }
41     else
42     {
43         MessageBox.Show("Error");
44         status = false;
45     }
46     if (status == false)
47     {
48         this.lFinder.reset();
49         NavigationService.Navigate(new Uri("/ MainPage.xaml", UriKind.Relative));
50     }
51 }
52
53 private void setRouteToMap(RootObjectFindRoute r)
54 {
55     MapPolyline routeRoad = new MapPolyline();
56     for (int i = 0; i < 1; i++) // Ambil Routing result yg pertama
57     {
58         this.arrayFocus = new Point[r.routingresults[i].steps.Count + 1];
59         this.detailRoute = new String[r.routingresults[i].steps.Count + 1];
60         for (int j = 0; j < r.routingresults[i].steps.Count; j++)
61         {
62             routeRoad = new MapPolyline();
63             if (r.routingresults[i].steps[j][0].ToString().Equals("walk"))
64             {
65                 //MessageBox.Show(r.routingresults[i].steps[j][0].ToString() + " walk");
66                 routeRoad.StrokeColor = Color.FromArgb(255, 255, 0, 0);
67             }
68             else
69             {
70                 //MessageBox.Show(r.routingresults[i].steps[j][0].ToString() + " angkot");
71                 routeRoad.StrokeColor = Color.FromArgb(255, 0, 0, 255);
72             }
73             routeRoad.StrokeThickness = 4;
74             GeoCoordinate geoCoo = new GeoCoordinate();
75             //Trim character
76             String temp = r.routingresults[i].steps[j][2].ToString();
77             char[] charsToTrim = { '[', ']', ',', ' ' };
78             String result = temp.Trim(charsToTrim);
79             result = result.Replace("\n", "");
80             //Split string coordinate to array
81             string[] coordinate = result.Split(',');
82             for (int c = 0; c < coordinate.Length; c = c + 2)
83             {
84                 geoCoo = new GeoCoordinate();
85                 geoCoo.Latitude = double.Parse(coordinate[c]);
86                 geoCoo.Longitude = double.Parse(coordinate[c + 1]);
87                 routeRoad.Path.Add(geoCoo);
88                 MapOverlay overlay1 = new MapOverlay();
89                 //Process add icon/pushpin
90                 if (j == 0 && c == 0)
91                 {
92                     overlay1.Content = createNew(p.iconStart, geoCoo, "start");
93                     this.route.Center = geoCoo;
94                     this.route.ZoomLevel = 14;
95                 }
96                 else if (j == r.routingresults[i].steps.Count - 1 && c == coordinate.
97                     Length - 2)
98                 {
99                     //MessageBox.Show("finish");
100                    overlay1.Content = createNew(p.iconFinish, geoCoo, "finish");
101                }
102                else if (c == 0)
103                {
104                    String iconLoc = p.getTypeTransport(r.routingresults[i].steps[j][0].
105                        ToString(), r.routingresults[i].steps[j][1].ToString());
106                    if (r.routingresults[i].steps[j][0].ToString().Equals("walk"))
107                    {
108                        overlay1.Content = createNew(iconLoc, geoCoo, "walk");
109                    }
110                }
111            }
112        }
113    }
114 }

```

```

6             else
7             {
8                 overlay1.Content = createNew(iconLoc, geoCoo, "umum");
9             }
10            overlay1.GeoCoordinate = geoCoo;
11            routeLayer.Add(overlay1);
12        }
13    }
14
15    //reference add image zhttp://stackoverflow.com/questions/676869/add-image-to-
16    //listbox
17    this.arrayFocus[j] = new Point(double.Parse(coordinate[0]), double.Parse(
18        coordinate[1]));
19    this.detailRoute[j] = r.routingresults[i].steps[j][3].ToString();
20    if (j == r.routingresults[i].steps.Count - 1) // Untuk yg terakhir/sampai di
21        tujuan
22    {
23        this.arrayFocus[j + 1] = new Point(double.Parse(coordinate[coordinate.
24            Length - 2]), double.Parse(coordinate[coordinate.Length - 1]));
25        this.detailRoute[j + 1] = "Sampai_di_tujuan!";
26    }
27    //Add image
28    Uri imgUri = new Uri(p.getTypeTransportWOBaloon(r.routingresults[i].steps[j].
29        [0].ToString(), r.routingresults[i].steps[j][1].ToString()), UriKind.
30        RelativeOrAbsolute);
31    BitmapImage imgSourceR = new BitmapImage(imgUri);
32    Image image = new Image();
33    image.Source = imgSourceR;
34    image.Height = 30;
35    image.Width = 50;
36    //add description
37    listRoute.Items.Add(image);
38    listRoute.Items.Add(r.routingresults[i].steps[j][3].ToString()); // Add text
39    to listBox
40    routeRoad.Path.Add(geoCoo);
41    route.MapElements.Add(routeRoad);
42}
43 addFindRoute.Text = lFinder.addressFrom + " ke " + lFinder.addressTo + "( " + r.
44 routingresults[i].traveltime + " )";
45
46 // Add the list box to a parent container in the visual tree.
47 route.Layers.Add(routeLayer);
48 this.routeReady = true;
49}
50
51 public void setFocus(object sender, RoutedEventArgs e)
52{
53    routePanel.Visibility = Visibility.Collapsed;
54    detailPanel.Visibility = Visibility.Visible;
55    this.route.ZoomLevel = 18;
56    if ((sender as Button).Name.Equals("prev"))
57    {
58        this.focusPointNumber--;
59        if (this.focusPointNumber < 0)
60        {
61            this.focusPointNumber = arrayFocus.Length - 1;
62        }
63        this.route.Center = new GeoCoordinate(arrayFocus[focusPointNumber].X, arrayFocus[
64            focusPointNumber].Y);
65    }
66    else
67    {
68        this.focusPointNumber++;
69        if (this.focusPointNumber > arrayFocus.Length - 1)
70        {
71            this.focusPointNumber = 0;
72        }
73        this.route.Center = new GeoCoordinate(arrayFocus[focusPointNumber].X, arrayFocus[
74            focusPointNumber].Y);
75    }
76    detailShows.Text = detailRoute[focusPointNumber];
77}
78
79 public void toMyLocation(object sender, RoutedEventArgs e)
80{
81    this.route.Center = new GeoCoordinate(lFinder.coorLat, lFinder.coorLong);
82    this.route.ZoomLevel = 15;
83}
84
85 private void ShowList(object sender, RoutedEventArgs e)
86{
87    detailPanel.Visibility = Visibility.Collapsed;
88    if (listBoxStatus == false)
89    {
90        routePanel.Visibility = Visibility.Visible;
91        buttonList.Content = "Tutup_List";
92        //LayoutRoot.Children.Add(listRoute);
93        this.listBoxStatus = true;
94    }
95    else
96    {
97        routePanel.Visibility = Visibility.Collapsed;
98        buttonList.Content = "Lihat_List";
99        //LayoutRoot.Children.Remove(listRoute);
100       this.listBoxStatus = false;
101    }
102}
103
104 public Pushpin createNew(string uri, GeoCoordinate geoCoordinate, String transport) {
105     Pushpin p = new Pushpin();
106     Uri imgUri = new Uri(uri, UriKind.RelativeOrAbsolute);
107     BitmapImage imgSourceR = new BitmapImage(imgUri);
108     ImageBrush imgBrush = new ImageBrush() { ImageSource = imgSourceR };

```

```

6     p.Background = new SolidColorBrush(Color.FromArgb(0, 0, 0, 0));
7     p.Content = new Rectangle()
8     {
9         Fill = imgBrush,
10        Height = 30,
11        Width = 50,
12    };
13    if (transport.Equals("start"))
14    {
15        p.Margin = new Thickness(-51, -35, 0, 0);
16    }
17    else if(transport.Equals("finish"))
18    {
19        p.Margin = new Thickness(-6, -34, 0, 0);
20    }else if(transport.Equals("walk")){
21        p.Margin = new Thickness(-55, -35, 0, 0);
22    }else{
23        p.Margin = new Thickness(-5, -35, 0, 0);
24    }
25    p.GeoCoordinate = geoCoordinate;
26    return p;
27}
28
29 private void drawMyLocationOnTheMap(Double latitude, Double longitude)
30 {
31     this.route.Layers.Remove(myLocationLayer);
32     GeoCoordinate myGeoCoordinate = new GeoCoordinate(latitude, longitude);
33     double myAccuracy = lFinder.accuracy;
34     //MessageBox.Show(lFinder.accuracy + "");
35     double metersPerPixel = (Math.Cos(myGeoCoordinate.Latitude * Math.PI / 180) * 2 *
36     Math.PI * 6378137) / (256 * Math.Pow(2, this.route.ZoomLevel));
37     double radius = myAccuracy / metersPerPixel;
38
39     Ellipse ellipse = new Ellipse();
40     ellipse.Width = radius * 2;
41     ellipse.Height = radius * 2;
42     ellipse.Margin = new Thickness(-radius+10, -radius+10, 0, 0);
43     ellipse.Fill = new SolidColorBrush(Color.FromArgb(75, 200, 0, 0));
44
45     Ellipse myCircle = new Ellipse();
46     myCircle.Stroke = new SolidColorBrush(Colors.Black);
47     myCircle.StrokeThickness = 4;
48     myCircle.Fill = new SolidColorBrush(Colors.Green);
49     myCircle.Height = 25;
50     myCircle.Width = 25;
51     myCircle.Opacity = 50;
52
53     MapOverlay myLocationOverlayPosition = new MapOverlay();
54     myLocationOverlayPosition.Content = myCircle;
55     myLocationOverlayPosition.PositionOrigin = new Point(0, 0);
56     myLocationOverlayPosition.GeoCoordinate = myGeoCoordinate;
57
58     MapOverlay myLocationOverlayAccuracy = new MapOverlay();
59     myLocationOverlayAccuracy.Content = ellipse;
60     myLocationOverlayAccuracy.PositionOrigin = new Point(0, 0);
61     myLocationOverlayAccuracy.GeoCoordinate = myGeoCoordinate;
62
63     myLocationLayer = new MapLayer();
64     myLocationLayer.Add(myLocationOverlayPosition);
65     myLocationLayer.Add(myLocationOverlayAccuracy);
66     this.route.Layers.Add(myLocationLayer);
67 }
68
69 private void back(object sender, RoutedEventArgs e)
70 {
71     MessageBoxResult result = MessageBox.Show("Anda yakin mengakhiri Navigasi?", "Kembali ke Menu Utama", MessageBoxButton.OKCancel);
72
73     if (result == MessageBoxResult.OK)
74     {
75         this.lFinder.reset();
76         NavigationService.Navigate(new Uri("/ MainPage.xaml", UriKind.Relative));
77     }
78 }
79
80 private void ShowLoading()
81 {
82     this.popup.IsOpen = true;
83     StartLoadingData();
84 }
85
86 private void StartLoadingData()
87 {
88     backgroundWorker = new BackgroundWorker();
89     backgroundWorker.RunWorkerAsync();
90     backgroundWorker.DoWork += new DoWorkEventHandler(backgroundWorker_DoWork);
91     backgroundWorker.RunWorkerCompleted += new RunWorkerCompletedEventHandler(
92         backgroundWorker_RunWorkerCompleted);
93     // _workerCompleted.WaitOne();
94 }
95
96 private void backgroundWorker_DoWork(object sender, DoWorkEventArgs e)
97 {
98     this.Dispatcher.BeginInvoke(() =>
99     {
100        this.Find();
101    });
102
103    while (this.routeReady == false)
104    {
105        if (this.timeOut == true) {
106            //this.timeOutReached();
107            this.routeReady = true;
108        }
109    }
110 }

```

```

6         }
7     }
8
9     void OnTimerTick(Object sender, EventArgs args)
10    {
11        newTimer.Stop();
12        this.timeOut = true;
13    }
14
15    private void backgroundWorker_RunWorkerCompleted(object sender,
16        RunWorkerCompletedEventArgs e)
17    {
18        this.Dispatcher.BeginInvoke(() =>
19        {
20            this.popup.IsOpen = false;
21            if (this.timeOut == true)
22            {
23                MessageBox.Show("Maaf, saat ini kami tidak dapat memproses rute anda!");
24                this.lFinder.reset();
25                NavigationService.Navigate(new Uri("/ MainPage.xaml", UriKind.Relative));
26            }
27        });
28    }
29
30    protected override void OnBackKeyPress(CancelEventArgs e)
31    {
32        if (MessageBox.Show("Anda yakin mengakhiri Navigasi?", "Kembali ke Menu Utama?", MessageBoxButton.OKCancel) == MessageBoxResult.OK)
33        {
34            this.lFinder.reset();
35            NavigationService.Navigate(new Uri("/ MainPage.xaml", UriKind.Relative));
36        }
37        else
38        {
39            e.Cancel = true;
40        }
41    }
42
43
44    //JSON data to c# using JSON.NET
45    //package from zhttp://www.nuget.org/packages/newtonsoft.json
46    //dok zhttp://www.newtonsoft.com/json/help/html/SerializingJSON.htm
47
48 }
}

```

Listing C.2: MainPage.xaml

```

49 <phone:PhoneApplicationPage
50     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
51     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
52     xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
53     xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
54     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
55     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
56
57     xmlns:Controls="clr-namespace:Microsoft.Phone.Maps.Controls;assembly=Microsoft.Phone.Maps"
58     xmlns:toolkit="clr-namespace:Microsoft.Phone.Maps.Toolkit;assembly=Microsoft.Phone.Controls.
59     Toolkit"
60     xmlns:Maps="clr-namespace:Microsoft.Phone.Controls.Maps;assembly=Microsoft.Phone.Controls.Maps
61     "
62     x:Class="Kiri.Route"
63     FontFamily="{StaticResource PhoneFontFamilyNormal}"
64     FontSize="{StaticResource PhoneFontSizeNormal}"
65     Foreground="{StaticResource PhoneForegroundBrush}"
66     SupportedOrientations="Portrait" Orientation="Portrait"
67     mc:Ignorable="d"
68     shell:SystemTray.Visibile="True">
69
70     <!--xmlns:maps="clr-namespace:Microsoft.Phone.Maps.Controls;assembly=Microsoft.Phone.Maps"-->
71     <!--LayoutRoot is the root grid where all page content is placed-->
72     <Grid x:Name="LayoutRoot" Background="Transparent">
73         <Grid.RowDefinitions>
74             <RowDefinition Height="Auto"/>
75             <RowDefinition Height="*"/>
76         </Grid.RowDefinitions>
77         <!--TitlePanel contains the name of the application and page title-->
78         <Grid HorizontalAlignment="Left" Height="57" Margin="10,10,0,0" Grid.RowSpan="2"
79             VerticalAlignment="Top" Width="460">
80             <Border BorderBrush="Azure" BorderThickness="0,0,0,5" HorizontalAlignment="Left"
81                 Height="57" VerticalAlignment="Top" Width="511" Margin="-21,-2,-30,0"/>
82             <TextBlock HorizontalAlignment="Left" Margin="207,4,0,0" FontSize="30" TextWrapping="Wrap"
83                 Text="Petunjuk Arah" VerticalAlignment="Top" Width="198"/>
84             <Button BorderThickness="0" HorizontalAlignment="Left" FontSize="30" Margin=
85                 "-39,-12,0,-2" VerticalAlignment="Top" Height="71" Width="137" Click="back">
86                 <Image Source="icons/back.png" Height="60" Width="60" Margin="-14,-12,0,0"/></Image>
87             >
88             </Button>
89             <Image Source="icons/kiri_logo.png" Margin="405,4,0,16" RenderTransformOrigin="
90                 1.509,0.516"/>
91         </Grid>
92         <Controls:Map x:Name="route" ZoomLevel="14" Margin="-9,67,-8,0" Grid.RowSpan="2"/>
93         <Button Name="prev" Content="Prev" Background="Gray" HorizontalAlignment="Left" Margin=
94             "146,67,0,0" VerticalAlignment="Top" Height="77" Width="105" Click="setFocus" Grid.
95             RowSpan="2"/>
96         <Button Name="next" Content="Next" Background="Gray" HorizontalAlignment="Left" Margin=
97             "237,67,0,0" VerticalAlignment="Top" Height="77" Width="105" Click="setFocus" Grid.
98             RowSpan="2"/>
99         <Button HorizontalAlignment="Left" Background="Gray" Margin="389,67,0,0"
100             VerticalAlignment="Top" Height="77" Width="81" Click="toMyLocation" Grid.RowSpan="2"/>
101             <Image Source="icons/marker.png" Height="60" Width="60" Margin="-14,-12,0,0"/></Image>
102         </Button>
103         <StackPanel x:Name="detailPanel" HorizontalAlignment="Left" Background="Black" Height="111
104             " Margin="0,610,0,0" Grid.Row="1" VerticalAlignment="Top" Width="480" Visibility="
105             Collapsed">

```

```

6   <TextBlock x:Name="detailShows" HorizontalAlignment="Left" Margin="0,0,0,0" Grid.Row="
7     1" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Height="100" Width="478"
8     Padding="4"/>
9   </StackPanel>
10  <Button x:Name="buttonList" Content="Lihat List" Background="Black" HorizontalAlignment="
11    Left" Margin="-22,709,-21,-13" VerticalAlignment="Top" Height="72" Width="523" Click="
12    ShowList" Grid.Row="1"/>
13  <StackPanel x:Name="routePanel" Background="White" Height="auto" Margin="0,250,0,47" Grid.
14    Row="1" Visibility="Collapsed">
15    <TextBlock x:Name="addFindRoute" Text="Rute Perjalanan" Foreground="Black" FontSize="
16      25" TextWrapping="Wrap" Margin="3"/>
17    <ListBox x:Name="listRoute" Foreground="White" Height="426" Background="White"
18      FontSize="20" Padding="5" Margin="3">
19      <ListBox.ItemTemplate>
20        <DataTemplate>
21          <Grid>
22            <Image Grid.Column="0" Margin="3" Width="60"/>
23            <Border BorderBrush="Black" BorderThickness="0,0,0,2">
24              <TextBlock Grid.Column="1" Margin="3" Padding="5" Foreground="
25                Black" TextWrapping="Wrap" Text="{Binding}"/>
26            </Border>
27          </Grid>
28        </DataTemplate>
29      </ListBox.ItemTemplate>
30    </ListBox>
31  </StackPanel>
32  <Popup x:Name="popup" Margin="-9,0,-8,10" Grid.RowSpan="2">
33    <Grid Background="Black" Height="774" Width="498">
34      <ProgressBar Height="34" Width="481" IsIndeterminate="True" Margin=" -7,370,10,352 "
35        />
36      <TextBlock HorizontalAlignment="Left" Margin="166,320,0,0" TextWrapping="Wrap"
37        Text="Harap Menunggu" VerticalAlignment="Top"/>
38      <TextBlock HorizontalAlignment="Left" Margin="83,449,0,0" TextWrapping="Wrap"
39        TextAlignment="Center" Text="Kami sedang mencari rute
40        untuk perjalanan anda" VerticalAlignment="Top" Width="322"/>
1      </Grid>
2    </Popup>
3  </Grid>
4
5 | </phone:PhoneApplicationPage>

```


LAMPIRAN D

KODE PROGRAM KELAS CITY

Listing D.1: city.cs

```

8  using System;
9  using System.Collections.Generic;
10 using System.Device.Location;
11 using System.Linq;
12 using System.Text;
13 using System.Threading.Tasks;
14
15 namespace Kiri
16 {
17     class City
18     {
19         private GeoCoordinate[] centerCity;
20         //String[,] city = { {"Auto", "Bandung", "Jakarta", "Malang", "Surabaya"}, {"Auto", "bdo", "cgk", "mlg", "sub"} };
21         public String[] city;
22         public String[] cityCode;
23
24         public City()
25         {
26             this.city = new String[] {"Bandung", "Jakarta", "Malang", "Surabaya"};
27             this.cityCode = new String[] {"bdo", "cgk", "mlg", "sub"};
28             this.centerCity = new GeoCoordinate[] { new GeoCoordinate(-6.91474, 107.60981), new
29                 GeoCoordinate(-6.21154, 106.84517), new GeoCoordinate(-7.9812985, 112.6319264),
30                 new GeoCoordinate(-7.27421, 112.71908) };
31         }
32
33         public int getNearby(Double coorLat, Double coorLong)
34         {
35             int s = -1;
36             GeoCoordinate deviceLocation = new GeoCoordinate(coorLat, coorLong);
37             double distance = Double.MaxValue;
38             if (!coorLat.Equals(0.0) && !coorLong.Equals(0.0))
39             {
40                 for (int c = 0; c < centerCity.Length; c++)
41                 {
42                     if (deviceLocation.GetDistanceTo(centerCity[c]) < distance)
43                     {
44                         distance = deviceLocation.GetDistanceTo(centerCity[c]);
45                         s = c;
46                     }
47                 }
48             }
49             return s;
50         }
51
52         public int getIndexFromCityCode(String code)
53         {
54             int i = -1;
55             for (int c = 0; c < cityCode.Length; c++)
56             {
57                 if (cityCode[c].Equals(code))
58                 {
59                     i = c;
60                 }
61             }
62             return i;
63         }
64     }
65 }
```


LAMPIRAN E

KODE PROGRAM KELAS LOCATIONFINDER

Listing E.1: LocationFinder.cs

```

8  using Microsoft.Phone.Maps.Services;
9  using System;
10 using System.Collections.Generic;
11 using System.Device.Location;
12 using System.IO.IsolatedStorage;
13 using System.Linq;
14 using System.Text;
15 using System.Threading.Tasks;
16 using System.Windows;
17 using Windows.Devices.Geolocation;
18
19 namespace Kiri
20 {
21     class LocationFinder
22     {
23         public Double coorLat = 0.0; //device coordinate
24         public Double coorLong = 0.0;
25         public Double coorLatFrom = 0.0; //from coordinate
26         public Double coorLongFrom = 0.0;
27         public Double coorLatTo = 0.0; //to coordinate
28         public Double coorLongTo = 0.0;
29
30         public string addressDevice = "";
31         public string addressFrom = "";
32         public string addressTo = "";
33
34         public Geolocator geolocator;
35         private ReverseGeocodeQuery MyReverseGeocodeQuery = null;
36         private GeoCoordinate MyCoordinate = null;
37         public double accuracy;
38
39         public LocationFinder()
40         {
41             this.geolocator = new Geolocator();
42             this.geolocator.DesiredAccuracy = PositionAccuracy.High;
43             this.geolocator.MovementThreshold = 20; // The units are meters.
44             this.accuracy = 0.0;
45             findLocation();
46         }
47
48         public async void findLocation()
49         {
50             // Get my current location.
51             try
52             {
53                 Geoposition myGeoposition = await geolocator.GetGeopositionAsync(
54                     TimeSpan.FromMinutes(1),
55                     TimeSpan.FromSeconds(10)
56                 );
57                 this.accuracy = myGeoposition.Coordinate.Accuracy;
58                 Deployment.Current.Dispatcher.BeginInvoke(() =>
59                 {
60                     Geocoordinate myGeocoordinate = myGeoposition.Coordinate;
61                     GeoCoordinate myGeoCoordinate = CoordinateConverter.ConvertGeocoordinate(
62                         myGeocoordinate);
63                     GetCurrentCoordinate((Double)myGeoCoordinate.Latitude, (Double)myGeoCoordinate
64                         .Longitude, "device");
65                 });
66             }
67             catch (Exception ex)
68             {
69                 // Couldn't get current location - location might be disabled in settings
70                 MessageBox.Show("Tidak_dapat_menemukan_lokasi");
71             }
72         }
73
74         public async void updateAccuracy()
75         {
76             Geoposition myGeoposition = await geolocator.GetGeopositionAsync(
77                 TimeSpan.FromMinutes(1),
78                 TimeSpan.FromSeconds(10)
79             );
80             this.accuracy = myGeoposition.Coordinate.Accuracy;
81         }
82
83
84         public void setPositionChanged(Geolocator sender, PositionChangedEventArgs args)
85         {

```

```

6     Deployment.Current.Dispatcher.BeginInvoke(() =>
7     {
8         coorLat = args.Position.Coordinate.Latitude;
9         coorLong = args.Position.Coordinate.Longitude;
10        updateAccuracy();
11    });
12}
13
14 //Get address zhttps://msdn.microsoft.com/en-us/library/windows/apps/xaml/dn631249.aspx
15 //old zhttp://stackoverflow.com/questions/16685088/windows-phone-reversegeocoding-to-get-
16 address-from-lat-and-long
17
18 //from zhttp://blogs.msdn.com/b/jrspinella/archive/2012/11/02/using-reversegeocodequery-
19 //for-windows-phone-8.aspx
20 public void GetCurrentCoordinate(Double latitude, Double longitude, String paramFor)
21 {
22     try
23     {
24         MyCoordinate = new GeoCoordinate(latitude, longitude);
25
26         if (MyReverseGeocodeQuery == null || !MyReverseGeocodeQuery.IsBusy)
27         {
28             MyReverseGeocodeQuery = new ReverseGeocodeQuery();
29             MyReverseGeocodeQuery.GeoCoordinate = new GeoCoordinate(MyCoordinate.Latitude
30                 , MyCoordinate.Longitude);
31             if (paramFor.Equals("from"))
32             {
33                 MyReverseGeocodeQuery.QueryCompleted +=
34                     ReverseGeocodeQueryFrom_QueryCompleted;
35                 this.coorLatFrom = latitude;
36                 this.coorLongFrom = longitude;
37             }
38             else if (paramFor.Equals("to"))
39             {
40                 MyReverseGeocodeQuery.QueryCompleted +=
41                     ReverseGeocodeQueryTo_QueryCompleted;
42                 this.coorLatTo = latitude;
43                 this.coorLongTo = longitude;
44             }
45             else
46             {
47                 MyReverseGeocodeQuery.QueryCompleted +=
48                     ReverseGeocodeQuery_QueryCompleted;
49                 this.coorLat = latitude;
50                 this.coorLong = longitude;
51             }
52             MyReverseGeocodeQuery.QueryAsync();
53         }
54     }
55     catch (Exception ex)
56     {
57     }
58 }
59
60
61 private void ReverseGeocodeQueryFrom_QueryCompleted(object sender, QueryCompletedEventArgs<
62     <IList<MapLocation>> e)
63 {
64     if (e.Error == null)
65     {
66         if (e.Result.Count > 0)
67         {
68             MapAddress add = e.Result[0].Information.Address;
69             addressFrom = add.Street;
70         }
71     }
72 }
73
74 private void ReverseGeocodeQueryTo_QueryCompleted(object sender, QueryCompletedEventArgs<
75     <IList<MapLocation>> e)
76 {
77     if (e.Error == null)
78     {
79         if (e.Result.Count > 0)
80         {
81             MapAddress add = e.Result[0].Information.Address;
82             addressTo = add.Street;
83         }
84     }
85 }
86
87 private void ReverseGeocodeQuery_QueryCompleted(object sender, QueryCompletedEventArgs<
88     <IList<MapLocation>> e)
89 {
90     if (e.Error == null)
91     {
92         if (e.Result.Count > 0)
93         {
94             MapAddress add = e.Result[0].Information.Address;
95             addressDevice = add.Street;
96         }
97     }
98 }
99
100 public void setCoordinateHere(string paramFor){
101     if (paramFor.Equals("from"))
102     {
103         this.coorLatFrom = this.coorLat;
104         this.coorLongFrom = this.coorLong;
105         this.addressFrom = this.addressDevice;
106     }
107     else {
108         this.coorLatTo = this.coorLat;
109     }
110 }

```

```
6         this.coorLongTo = this.coorLong;
7         this.addressTo = this.addressDevice;
8     }
9 }
10
11    public void reset() {
12        this.coorLatFrom = 0.0; //from coordinate
13        this.coorLongFrom = 0.0;
14        this.coorLatTo = 0.0; ////to coordinate
15        this.coorLongTo = 0.0;
16
17        this.addressDevice = "";
18        this.addressFrom = "";
19    }
20
21 }
```


LAMPIRAN F

KODE PROGRAM KELAS PROTOCOL

Listing F.1: Protocol.cs

```

8  using Newtonsoft.Json;
9  using System;
10 using System.Collections.Generic;
11 using System.Linq;
12 using System.Net.Http;
13 using System.Text;
14 using System.Threading.Tasks;
15 using System.Windows.Controls;
16 using System.Windows.Media;
17
18 namespace Kiri
19 {
20     class Protocol
21     {
22         HttpClient httpClient = new HttpClient();
23         public String uri_version
24         {
25             get
26             {
27                 return "version=";
28             }
29         }
30         public String uri_mode
31         {
32             get
33             {
34                 return "&mode=";
35             }
36         }
37         public String uri_locale
38         {
39             get
40             {
41                 return "&locale=";
42             }
43         }
44         public String uri_start
45         {
46             get
47             {
48                 return "&start=";
49             }
50         }
51         public String uri_finish
52         {
53             get
54             {
55                 return "&finish=";
56             }
57         }
58         public String uri_presentation
59         {
60             get
61             {
62                 return "&presentation=";
63             }
64         }
65         public String uri_apikey
66         {
67             get
68             {
69                 return "&apikey=";
70             }
71         }
72         public String uri_region
73         {
74             get
75             {
76                 return "&region=";
77             }
78         }
79         public String uri_query
80         {
81             get
82             {
83                 return "&querystring=";
84             }
85         }

```

```
6     }
7
8     private static String apiKey
9     {
10        get
11        {
12            return "97A7A1157A05ED6F";
13        }
14    }
15    private static String hostname
16    {
17        get
18        {
19            return "http://kiri.travel/";
20        }
21    }
22    private static String handle
23    {
24        get
25        {
26            return hostname+"handle.php?";
27        }
28    }
29    private static String iconPath
30    {
31        get
32        {
33            return hostname + "images/means/";
34        }
35    }
36    public String iconStart
37    {
38        get
39        {
40            return hostname + "images/stepicon-walkstart.png";
41        }
42    }
43    public String iconFinish
44    {
45        get
46        {
47            return hostname + "images/stepicon-finish.png";
48        }
49    }
50
51    private static String version_
52    {
53        get
54        {
55            return "2";
56        }
57    }
58
59    private static String modeFind
60    {
61        get
62        {
63            return "searchplace";
64        }
65    }
66
67    private static String modeRoute
68    {
69        get
70        {
71            return "findroute";
72        }
73    }
74    private static String modeNearby
75    {
76        get
77        {
78            return "nearbytransport";
79        }
80    }
81    private static String localeId
82    {
83        get
84        {
85            return "id";
86        }
87    }
88    private static String localeEn
89    {
90        get
91        {
92            return "en";
93        }
94    }
95    private static String presentationMobile
96    {
97        get
98        {
99            return "mobile";
100       }
101      }
102      private static String presentationDesktop
103      {
104        get
105        {
106            return "desktop";
107        }
108      }
```

```

6     }
7
8     public string getTypeTransport(string means, string meansDetail)
9     {
10
11         String uri = iconPath + means + "/baloon/" + meansDetail + ".png";
12         return uri;
13     }
14
15     public string getTypeTransportWOBaloon(string means, string meansDetail)
16     {
17
18         String uri = iconPath + means + "/" + meansDetail + ".png";
19         return uri;
20     }
21
22     public string getSearchPlace(string query, string region)
23     {
24
25         String uri = handle + uri_version + version_2 + uri_mode + modeFind + uri_region +
26             region + uri_query + query + uri_apikey + apiKey;
27         return uri;
28     }
29
30     public string getFindRoute(string start, string finish)
31     {
32
33         String uri = handle + uri_version + version_2 + uri_mode + modeRoute + uri_locale +
34             localeId + uri_start + start + uri_finish + finish + uri_presentation +
35             presentationDesktop + uri_apikey + apiKey;
36         return uri;
37     }
38
39     public async Task<RootObjectSearchPlace> getRequestSearch(string query, string region)
40     {
41
42         String uri = getSearchPlace(query, region);
43         Task<String> requestRouteTask = httpClient.GetStringAsync(new Uri(uri));
44         String request = await requestRouteTask;
45         RootObjectSearchPlace objectRootSearch = JsonConvert.DeserializeObject<
46             RootObjectSearchPlace>(request);
47         return objectRootSearch;
48     }
49
50
51     public async Task<RootObjectFindRoute> getRequestRoute(Double startLat, Double startLong,
52
53         Double finishLat, Double finishLong)
54     {
55
56         String uri = getFindRoute(startLat + "," + startLong, finishLat + "," + finishLong);
57         Task<String> requestRouteTask = httpClient.GetStringAsync(new Uri(uri));
58         String request = await requestRouteTask;
59         RootObjectFindRoute objectRootRoute = JsonConvert.DeserializeObject<
60             RootObjectFindRoute>(request);
61         return objectRootRoute;
62     }
63
64     }
65 }
```


LAMPIRAN G

KODE PROGRAM OBJEK DARI JSON

Listing G.1: RootObjectFindRoute.cs

```

8  using System;
9  using System.Collections.Generic;
10 using System.Linq;
11 using System.Text;
12 using System.Threading.Tasks;
13
14 namespace Kiri
15 {
16     class RootObjectFindRoute
17     {
18         public string status { get; set; }
19         public List<Routingresult> routingresults { get; set; }
20     }
21 }
```

Listing G.2: RootObjectSearchPlace.cs

```

22 using System;
23 using System.Collections.Generic;
24 using System.Linq;
25 using System.Text;
26 using System.Threading.Tasks;
27
28 namespace Kiri
29 {
30     public class RootObjectSearchPlace
31     {
32         public string status { get; set; }
33         public List<Searchresult> searchresult { get; set; }
34         public object attributions { get; set; }
35     }
36 }
37 }
```

Listing G.3: Routingresult.cs

```

38 using System;
39 using System.Collections.Generic;
40 using System.Linq;
41 using System.Runtime.Serialization;
42 using System.Text;
43 using System.Threading.Tasks;
44
45 namespace Kiri
46 {
47     class Routingresult
48     {
49         public List<List<object>> steps { get; set; }
50         public string travertime { get; set; }
51     }
52 }
```

Listing G.4: Searchresult.cs

```

53 using System;
54 using System.Collections.Generic;
55 using System.Linq;
56 using System.Text;
57 using System.Threading.Tasks;
58
59 namespace Kiri
60 {
61     public class Searchresult
3863     {
3864         public string placename { get; set; }
3865         public string location { get; set; }
3866     }
3867 }
```