

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	21
2.1.8 Thread	23
2.2 Kiri API	24
2.2.1 <i>Web Service</i> Penentuan Rute	25
2.2.2 <i>Web Service</i> Pencarian Lokasi	26
2.2.3 <i>Web Service</i> Menemukan Transportasi Terdekat	27
3 ANALISIS	29
3.1 Analisis Aplikasi Sejenis	29
3.2 Analisis Aplikasi	31
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	35
3.2.6 Analisis Kiri API	36
3.2.7 Diagram <i>Use Case</i> dan Skenario	41
3.2.8 Kelas Diagram	44
4 PERANCANGAN	47
4.1 Diagram <i>Sequence</i>	47

4.2	Perancangan Kelas	50
4.2.1	Kelas <i>PhoneApplicationPage</i>	55
4.2.2	Kelas <i>MainPage</i>	55
4.2.3	Kelas <i>City</i>	57
4.2.4	Kelas <i>BackgroundWorker</i>	57
4.2.5	Kelas <i>Geocoordinate</i>	57
4.2.6	Kelas <i>Geolocator</i>	57
4.2.7	Kelas <i>Geoposition</i>	57
4.2.8	Kelas <i>LocationFinder</i>	58
4.2.9	Kelas <i>Map</i>	59
4.2.10	Kelas <i>HttpClient</i>	60
4.2.11	Kelas <i>JsonConvert</i>	60
4.2.12	Kelas <i>Protocol</i>	60
4.2.13	Kelas <i>RootObjectSearchPlace</i>	62
4.2.14	Kelas <i>SearchResult</i>	62
4.2.15	Kelas <i>RootObjectFindRoute</i>	62
4.2.16	Kelas <i>RoutingResult</i>	62
4.2.17	Kelas <i>Route</i>	63
4.3	Perancangan Antar Muka	65
4.3.1	Antarmuka Kelas <i>MainPage</i>	65
4.3.2	Antarmuka Kelas <i>Map</i>	67
4.3.3	Antarmuka Kelas <i>Route</i>	67
5	IMPLEMENTASI DAN PENGUJIAN APLIKASI	69
5.1	Implementasi	69
5.1.1	Perangkat Keras untuk Implementasi	69
5.1.2	Perangkat Lunak untuk Implementasi	69
5.1.3	Hasil Implementasi	70
5.2	Pengujian	74
5.2.1	Lingkungan Pengujian	74
5.2.2	Pengujian Fungsional	74
5.2.3	Pengujian Eksperimental	77
5.2.4	Perbandingan Aplikasi Pencari Rute di Android dengan Aplikasi Pencari Rute di Windows Phone	78
6	KESIMPULAN DAN SARAN	81
6.1	Kesimpulan	81
6.2	Saran	81
DAFTAR REFERENSI	83	
A	KODE PROGRAM KELAS MAINPAGE	85
B	KODE PROGRAM KELAS MAP	91
C	KODE PROGRAM KELAS ROUTE	93
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	47
4.2	Diagram <i>sequence</i> Mendapatkan Kordinat pada Peta	48
4.3	Diagram <i>sequence</i> Mendapatkan Rute	49
4.4	Diagram Kelas	50
4.5	Kelas Mainpage	51
4.6	Kelas City	51
4.7	Kelas JsonConvert	51
4.8	Kelas LocationFinder	52
4.9	Kelas Map	52
4.10	<i>Package</i> WindowsPhone	52
4.11	Kelas Protocol	53
4.12	Kelas Route	53
4.13	Kelas SearchPlace	54
4.14	Kelas FindRoute	54
4.15	Antarmuka <i>MainPage</i>	65
4.16	Antarmuka Daftar Tempat	66
4.17	Antarmuka <i>Map</i>	67
4.18	Antarmuka <i>Route</i>	67
4.19	Antarmuka Rute dalam Bentuk Daftar	68
5.1	Gambar antarmuka kelas MainPage	71
5.2	Gambar Antarmuka Splash di Kelas MainPage	71
5.3	Gambar Antarmuka <i>List</i> Asal dan <i>List</i> Tujuan di Kelas MainPage	72

5.4	Gambar Antarmuka Kelas Map	72
5.5	Gambar Antarmuka Menunggu di Kelas Route	73
5.6	Gambar Antarmuka Kelas Route	73
5.7	Gambar Antarmuka <i>List</i> di Kelas Route	74
5.8	Gambar Perbandingan Antarmuka Home di Android(kiri) dan Windows Phone(kanan)	78
5.9	Gambar Perbandingan Antarmuka Home di Android(kiri) dan Windows Phone(kanan)	78
5.10	Gambar Perbandingan Antarmuka Menunggu hasil pencarian rute di Android(kiri) dan Windows Phone(kanan)	79
5.11	Gambar Perbandingan Antarmuka Penentuan Rute di Android(kiri) dan Windows Phone(kanan)	79

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	25
2.6	Tabel Parameter Layanan Pencarian Lokasi	26
2.7	Tabel Parameter :ayanan Menemukan Transportasi Terdekat	27
3.1	Skenario Mendapatkan Lokasi untuk Masukan Lokasi Asal dan Lokasi Tujuan	42
3.2	Skenario Memasukan Lokasi Asal dan Lokasi Tujuan pada <i>TextBox</i>	43
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	44
5.1	Tabel Hasil yang Diharapkan Penulis	75
5.2	Tabel Hasil Pengujian Fungsionalitas Terhadap Pengguna 1	75
5.3	Tabel Hasil Pengujian Fungsionalitas Terhadap Pengguna 2	75
5.4	Tabel Hasil Pengujian Fungsionalitas Terhadap Pengguna 3	76
5.5	Tabel Hasil Pengujian Fungsionalitas Terhadap Pengguna 4	76
5.6	Tabel Hasil Pengujian Fungsionalitas Terhadap Pengguna 5	76
5.7	Tabel Perbedaan	79

¹

BAB 1

²

PENDAHULUAN

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

⁶ 1.1 Latar Belakang

⁷ Transportasi menjadi bagian yang penting bagi kehidupan manusia saat penelitian ini
⁸ dilakukan. Ada dua jenis transportasi yaitu kendaraan umum dan kendaraan pribadi. Pada
⁹ saat penelitian ini dilakukan banyak orang yang lebih memilih kendaraan pribadi diban-
¹⁰ ding kendaraan umum. Maraknya penggunaan kendaraan pribadi ditambah penambahan
¹¹ jalur kendaraan yang tidak sebanding banyaknya kendaraan di Indonesia akhirnya menim-
¹² bulkan kemacetan. Hal yang menimbulkan maraknya penggunaan kendaraan pribadi dik-
¹³ renakan kurang nyamannya kendaraan umum dan kesulitan dalam menentukan kendaraan
¹⁴ umum yang harus dinaiki. Kesulitan seseorang untuk memilih kendaraan umum disebabkan
¹⁵ banyaknya rute kendaraan umum membuat orang kebingungan dalam memilih kendaraan
¹⁶ umum untuk menuju lokasi yang diinginkan dan kurangnya publikasi mengenai rute ang-
¹⁷ kutan umum. Selain kebingungan seseorang cenderung malas untuk bertanya dan mencari
¹⁸ rute yang benar dan efisien. Karena hal tersebut, seseorang lebih memilih menggunakan
¹⁹ kendaraan pribadi ketimbang kendaraan umum.

²⁰ Ide pembuatan aplikasi yang memudahkan seseorang dalam menentukan rute kendaraan
²¹ umum sudah lebih dulu ada. Ide dalam penentuan rute angkutan umum tersebut dikenal
²² dengan nama Kiri. Kiri dibuat dengan latar belakang tiga masalah besar yaitu pemanasan
²³ global, kemacetan, dan harga bahan bakar minyak yang tinggi¹. Kiri pertama dikenal dalam
²⁴ bentuk situs web dengan alamat [kiri.travel](http://static.kiri.travel). Meskipun Kiri pertama dibuat di web tetapi
²⁵ layanan Kiri dalam menentukan rute kendaraan umum dapat dimanfaatkan untuk pencarian
²⁶ rute selain di web. Pemanfaatan layanan Kiri dalam mencari rute kendaraan umum tersebut
²⁷ yaitu dengan menggunakan Kiri API.

²⁸ Pesatnya perkembangan teknologi informasi dan komunikasi saat penelitian ini dilakukan
²⁹ memiliki pengaruh besar terhadap kehidupan manusia. Kebutuhan akan informasi mendo-
³⁰ rong perkembangan perangkat yang dapat mengakses informasi dimanapun dan kapanpun.
³¹ Teknologi yang dapat membantu manusia dalam hal tersebut adalah perangkat *mobile*. Ke-
³² mudahan dalam penggunaan yang dimiliki perangkat *mobile* kian digemari orang-orang
³³ terutama di Indonesia. Salah satu yang menarik perhatian adalah Windows Phone 8 yang

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

1 dibuat Microsoft. Antarmuka Windows Phone 8 yang disebut *Metro* cukup menarik dan
2 mudah digunakan.

3 Berdasarkan hal tersebut, penulis mencoba mengembangkan aplikasi Pencarian Rute
4 Kendaraan Umum di Windows Phone dalam tugas akhir ini. Aplikasi yang penulis kem-
5 bangkan akan memungkinkan pengguna menemukan rute kendaraan umum untuk sampai di
6 tujuan. Untuk memudahkan pengguna, penulis akan menampilkan dalam dua bentuk yaitu
7 peta dan daftar.

8 **1.2 Rumusan Masalah**

9 Sehubung dengan latar belakang diatas timbul permasalahan sebagai berikut:

- 10 • Bagaimana membuat aplikasi di Windows Phone?
- 11 • Bagaimana mengintegrasikan Kiri API dengan aplikasi pencari rute kendaraan umum
12 di Windows Phone?

13 **1.3 Tujuan**

14 Berdasarkan rumusan masalah pada sub-bab 1.2, maka tujuan dari pembuatan tugas akhir
15 ini adalah:

- 16 • Mempelajari cara pembuatan perangkat linak di Windows Phone lalu mengembangkan
17 aplikasi yang akan dibuat.
- 18 • Membuat aplikasi di di Windows Phone yang memanfaatkan Kiri API.

19 **1.4 Batasan Masalah**

20 Ruang lingkup pengembangan aplikasi Pencari Rute Kendaraan untuk Windows Phone ini
21 dibatasi hal berikut:

- 22 • Aplikasi ini akan berjalan di sistem operasi Windows Phone 8.
- 23 • Aplikasi ini membutuhkan koneksi internet.
- 24 • Aplikasi ini akan menampilkan rute jalur angkot, bus umum dan travel di tiga kota
25 besar yaitu Bandung, Jakarta, dan Surabaya.

26 **1.5 Metode Penelitian**

27 Metode Penelitian yang penulis gunakan dalam membuat tugas akhir ini adalah sebagai
28 berikut:

- 29 • Melakukan studi pustaka mengenai XAML, kontrol dan navigasi di Windows Phone,
30 Peta di Windows Phone, GPS di Windows Phone dan Kiri API.
- 31 • Melakukan analisis terhadap aplikasi lain yang menggunakan Kiri API.

- 1 ● Melakukan analisis terhadap dasar teori untuk pembangunan aplikasi Pencarian Rute Kendaraan Umum untuk Windows Phone.
- 2
- 3 ● Melakukan perancangan aplikasi Pencarian Rute Kendaraan Umum untuk Windows Phone.
- 4
- 5 ● Implementasi dari aplikasi Pencarian Rute Kendaraan Umum untuk Windows Phone.
- 6 ● Menguji aplikasi Pencarian Rute Kendaraan Umum untuk Windows Phone.
- 7 ● Membuat kesimpulan.

8 1.6 Sistematika Penulisan

9 Bab 1 membahas latar belakang masalah, rumusan masalah, tujuan penulisan tugas
10 akhir, batasan masalah, ruang lingkup masalah, metode penelitian, dan teknik pengumpulan
11 data tugas akhir ini.

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

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

23 Bab 4 membahas tentang perancangan aplikasi dari pembahasan di bab 2 dan bab 3. Bab
24 perancangan dimulai dengan perancangan diagram *sequence* yang dapat menggambarkan
25 interaksi antar objek, diagram kelas digunakan untuk menggambarkan hubungan antar suatu
26 kelas, dan antarmuka aplikasi.

27 Bab 5 membahas tentang implementasi dari perancangan di bab 4 lalu pengujian yang
28 penulis lakukan. Pengujian yang dilakukan ada dua yaitu pengujian fungsional dan eksperi-
29 mental.

30 Bab 6 membahas tentang kesimpulan dari penelitian ini dan saran dari penulis terhadap
31 penelitian ini.

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 *mobile* yang dikembangkan
⁹ oleh 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, and
¹⁹ Microsoft Azure^[1]. Microsoft .NET framework terdiri dari runtime bahasa umum dan *library*
²⁰ .NET Framework, yang meliputi kelas, *interface*, dan jenis nilai yang saling terintegrasi.
²¹ Microsoft .NET Framework menyediakan lingkungan yang mudah dikelola, pengembangan
²² yang disederhanakan, dan integrasi dengan berbagai bahasa pemrograman, termasuk Visual
²³ 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 Visu-
²⁶ al C#. Untuk masalah kehandalan keduanya menawarkan kehandalan yang baik. Kelebihan
²⁷ dari Visual Basic adalah bahasa pemrograman berorientasi objek yang kuat dan memiliki
²⁸ banyak pengembangan fitur di *inheritance*, *polymorphism*, *interfaces*, and *overloading*^[1].
²⁹ Kelebihan dari C# yang merupakan pengembangan dari C/C++ adalah sederhana, mode-
³⁰ ren, aman dan berorientasi objek^[1]. Satu hal yang dirasakan penulis adalah kenyamanan
³¹ ketika memilih bahasa .NET tersebut. Akan lebih mudah bagi developer yang menggunakan
³² Visual Basic 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 menggu-
10 nakan konvensi bahasa XAML dan ditulis pada *namespace* yang ditandai dengan *prefix* x
11 sebagai elemen paling atas. Setelah itu di baris ke dua dimulai dengan *xmlns* diikuti titik
12 dua, lalu nama dari *namespace*, diikuti tanda sama dengan dan *path* yang merepresentasikan
13 *namespace*[2]. *Prefix* x pada XAML mengandung beberapa struktur program yang sering
14 kita gunakan yaitu :

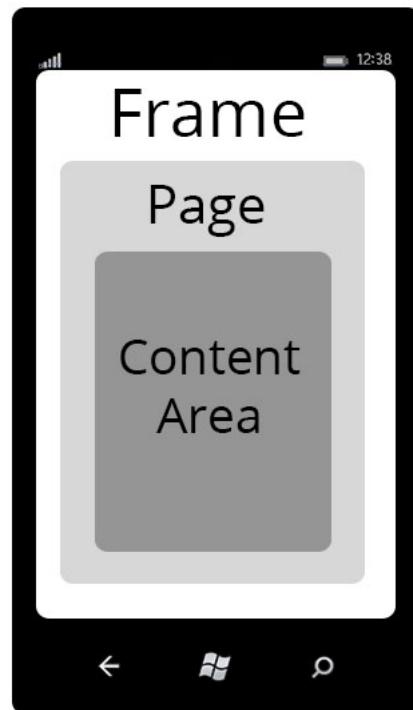
- 15 • x:Key : sebuah nama unik untuk menunjuk referensi ke suatu *resource*. Nilai ini dapat
16 dipanggil kembali untuk menggunakan *resource* tersebut.
- 17 • x:Class : menunjukkan nama kelas.
- 18 • x:Name : menunjukkan nama sebuah obyek dan untuk membedakan antar obyek yang
19 satu dengan obyek yang 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.

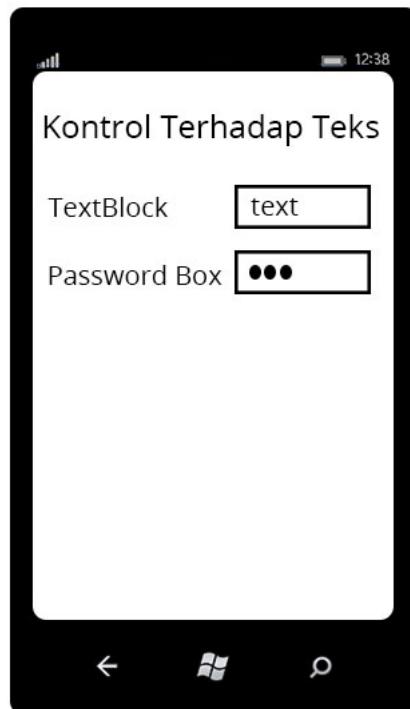
- 6
- *TextBlock* merupakan tempat menaruh potongan teks yang hanya bisa dilihat.

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**

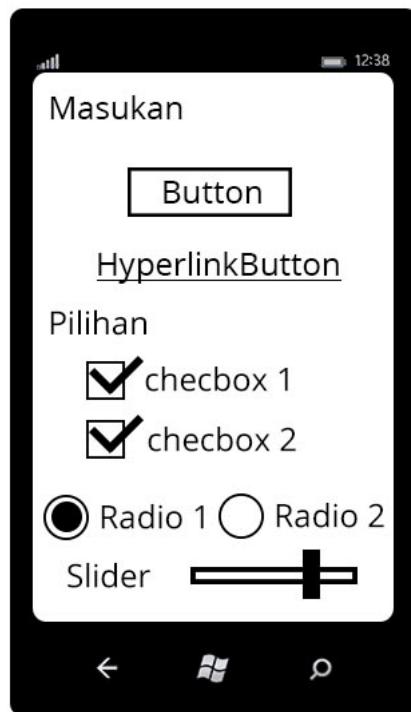
```
11 | <TextBlock x:Name="TextBlock1" Text="TextBlock"/>
12 | <TextBox x:Name="TextBox1" Text="TextBox"/>
```

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

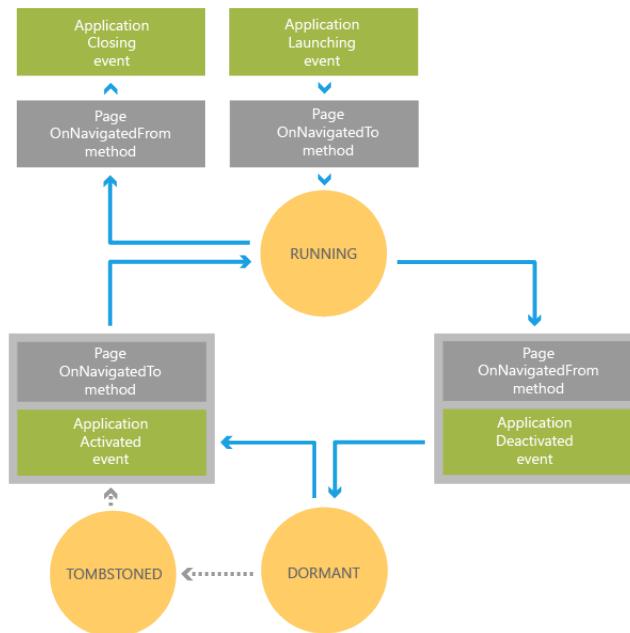
Gambar 2.4: Antarmuka *ListBox*Listing 2.4: Kode untuk menampilkan *listBox*

```

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 *Maps* menawarkan beberapa pilihan dalam tampilan peta mulai dari *cartographic*, penca-
7 hayaan dan pandangan. Selain tampilan pada sub-bab ini akan dibahas mengenai menda-
8 patkan 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
11 peta merupakan bagian dari *library* Windows Phone. Dengan begitu penggunaan peta di
12 Windows Phone perlu direferensikan. Untuk dapat menggunakannya juga harus ditambah
13 *capability ID_CAP_MAP*. Setelah hal tersebut dilakukan barulah peta dapat ditampilkan.
14 Berikut 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 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 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 rentetan 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*. Setelah penambahan Namespace perlu ditambahkan pula 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

- 1 Berikut *method* yang dapat digunakan pada kelas ini.
- 2 • *SetView(LocationRectangle)*
3 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai wilayah geogra-
4 fis. *Method* ini tidak mengembalikan nilai.
- 5 • *SetView(GeoCoordinate, Double)*
6 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah dan
7 tingkat zoom. *Method* ini tidak mengembalikan nilai.
- 8 • *SetView(LocationRectangle, MapAnimationKind)*
9 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai region geografis
10 dan animasi. *Method* ini tidak mengembalikan nilai.
- 11 • *SetView(LocationRectangle, Thickness)*
12 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai region geografis
13 dengan batas tertentu. *Method* ini tidak mengembalikan nilai.
- 14 • *SetView(GeoCoordinate, Double, MapAnimationKind)*
15 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah,
16 tingkat zoom, dan animasi. *Method* ini tidak mengembalikan nilai.
- 17 • *SetView(GeoCoordinate, Double, Double)*
18 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah,
19 tingkat zoom, dan *heading*. *Method* ini tidak mengembalikan nilai.
- 20 • *SetView(LocationRectangle, Thickness, MapAnimationKind)*
21 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai wilayah geografis
22 dengan batas tertentu, dan animasi. *Method* ini tidak mengembalikan nilai.
- 23 • *SetView(GeoCoordinate, Double, Double, MapAnimationKind)*
24 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah,
25 tingkat zoom, *heading*, dan animasi. *Method* ini tidak mengembalikan nilai.
- 26 • *SetView(GeoCoordinate, Double, Double, Double)*
27 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah,
28 tingkat zoom, *heading*, *pitch*. *Method* ini tidak mengembalikan nilai.
- 29 • *SetView(GeoCoordinate, Double, Double, Double, MapAnimationKind)*
30 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah,
31 tingkat zoom, *heading*, *pitch*, dan animasi. *Method* ini tidak mengembalikan nilai.
- 32 • *UpdateLayout*
33 *method* yang memastikan semua posisi objek turunan mengikuti tata letak.

34 **2.1.5.7 Kelas *Polyline***

35 Merupakan kelas yang dipakai untuk menggambarkan garis lurus yang saling terhubung.
36 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 Kelas *Pushpin*

12 Merupakan kelas yang dipakai untuk menggambarkan elemen terpisah diatas peta. Meskipun *pushpin* merupakan bawaan pada peta untuk menunjuk suatu lokasi tetapi *pushpin* dari peta tidak dapat diubah-ubah. Untuk dapat mengubah *Pushpin* pada Windows Phone 8 sesuai kebutuhan dibutuhkan penambahan *Windows Phone Toolkit*. *Windows Phone Toolkit* mempunyai komponen untuk menggambar pushpin diatas peta. Berikut Properti yang disediakan kelas *Pushpin*:

- 18 ● *background {set;}*
- 19 Untuk mengatur latar belakang.
- 20 ● *content {set;}*
- 21 Untuk mengatur konten pada *pushpin*.
- 22 ● *margin {set;}*
- 23 Untuk mengatur batas pada *pushpin*.
- 24 ● *GeoCoordinate {set;}*
- 25 Untuk mengatur kordinat peletakan *pushpin*.

26 2.1.6 Lokasi

27 Aplikasi di Windows Phone 8 dapat memanfaatkan lokasi di mana perangkat berada.

28 Aplikasi dapat melacak lokasi sesaat pengguna atau pelacakan selama periode tertentu.

1 Data lokasi perangkat berasal dari berbagai sumber termasuk *Global Positioning System*
2 (*GPS*), *Wireless Fidelity* (*Wi-Fi*), dan jaringan seluler. Ada 2 set API berbeda yang dapat
3 dimanfaatkan di Windows Phone yaitu *Runtime Location API* dan *.NET Location API*.
4 Windows Phone *Runtime Location* memiliki keunggulan fitur yang banyak sedangkan *.NET*
5 *Location* direkomendasikan jika aplikasi ditargetkan pada Windows Phone 7.1 dan Windows
6 Phone 8[1].

7 Hal yang perlu diperhatikan dalam menentukan layanan lokasi adalah penangkap *GPS*,
8 *Wi-Fi*, dan jaringan seluler. Perangkat tersebut berfungsi sebagai penyedia data lokasi de-
9 ngan berbagai tingkat akurasi dan konsumsi daya. Perangkat diatas juga berkomunikasi
10 langsung untuk memutuskan sumber mana yang digunakan untuk menentukan lokasi per-
11 angkat berdasarkan ketersediaan data lokasi dan prasyarat yang ditentukan aplikasi. Lapisan
12 diatas penyedia data lokasi tersebut adalah pengelola antarmuka. Aplikasi akan menggunakan
13 an antarmuka tersebut untuk memulai dan menghentikan layanan lokasi, mengatur tingkat
14 akurasi, dan menerima data lokasi.

15 Karena pengguna dapat berpindah tempat untuk menuju tempat yang lain, maka pela-
16 cakan lokasi harus dilakukan terus menerus. Pelacakan lokasi secara terus menerus ini dapat
17 dilakukan di depan maupun di belakang aplikasi Windows Phone 8. Pelacakan aplikasi di
18 depan akan memungkinkan aplikasi melacak lokasi pengguna sekaligus melakukan perbaruan
19 antarmuka. Jika pelacakan lokasi di belakang aplikasi maka tidak ada perubahan pada an-
20 tarmuka namun pelacakan dilakukan secara terus menerus. Pelacakan yang terus menerus di
21 belakang aplikasi akan membuat keadaan aplikasi cepat dipulihkan dari keadaan *Dormant*.

22 **2.1.6.1 Mendapatkan Posisi Pengguna**

23 Di Windows Phone 8 telah ada *GeoCoordinate class* yang dapat digunakan untuk menge-
24 tahui posisi pengguna. *Geolocator class* dari *Windows.Devices.Geolocation* akan mengemba-
25 likan posisi saat ini. Untuk menggunakan *Geolocator*, perlu menghidupkan *ID_CAP_LOCATION*
26 di [/properties/WMApManifest.xml](#). Dalam mendapatkan posisi perlu diperhatikan status
27 dari GPS karena membutuhkan waktu dari awal pengaktifan hingga mendapatkan lokasi
28 pengguna secara akurat. Untuk lebih jelas mengenai status posisi dapat dilihat pada nilai
29 status dibawah ini.

- 30 ● *Ready* : Jika lokasi tersedia.
- 31 ● *Initializing* : Jika status penangkap *GPS* belum memiliki cukup satelit untuk menda-
32 patkan posisi yang akurat.
- 33 ● *NoData* : Data lokasi belum tersedia. Status ini muncul jika aplikasi sedang mamanggil
34 method *GetGeopositionAsync()* atau method *register*.
- 35 ● *Disable* : Status mengindikasikan tidak diperbolehkannya pengaksesan lokasi.
- 36 ● *NotInitialized* : Data lokasi belum tersedia. Status ini muncul jika aplikasi belum
37 mamanggil method *GetGeopositionAsync()* atau method *register*.
- 38 ● *NotAvailable* : Jika sensor arah mata angin dan lokasi tidak tersedia.

2.1.6.2 Namespace Geolocator

Namespace merupakan nama yang dipakai untuk mengatur kelas-kelas. Windows Phone 8 sudah menyediakan namespace bawaan untuk mengakses lokasi. Namespace yang disediakan adalah *namespace geolocator*. Namespace ini akan mengakses lokasi geografis dari perangkat dan mendukung pelacakan lokasi dari waktu ke waktu. Agar dapat menggunakan kelas pada namespace tersebut perlu ditambahkan namespace dan *capabilities*. Namespace yang harus ditambahkan pada baris awal XAML adalah **Windows.Device.Geolocator**. Selanjutnya ada penambahan *capabilities ID_CAP_LOCATION*. Penambahan capabilities ditambahkan pada *WMAppManifest.xml*. Kelas yang diatur oleh namespace geolocator dapat di lihat 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

10

2.1.6.3 Kelas Geocoordinate

12 *Geocoordinate* adalah kelas yang menunjukkan lokasi sebagai kordinat geografis. Kelas ini hanya menyediakan properti yang hanya bisa dibaca. Kelas ini menyediakan properti 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

2.1.6.4 Kelas Geolocator

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

17 Berikut *method* yang disediakan *Geolocator*:

- *public IAsyncOperation<Geoposition> GetGeopositionAsync()*

19 Operator await diatas dimaksudkan untuk meminta posisi lokasi terus menerus sampai selesai dan menunda tugas yang lain.

21 *GetGeopositionAsync()* merupakan bawaan kelas *Geolocator* akan meminta data lokasi 22 dan menanganinya sampai selesai. Kembalian dari *GetGeopositionAsync()* adalah 23 objek *Geoposition*.

24 Berikut Properti yang disediakan kelas Geolocator:

- 1 ● *public PositionStatus LocationStatus { get; }*

2 Merupakan properti dari kelas *geolocator* untuk mendapatkan status posisi dengan
3 mengembalikan kelas *PositionStatus*. Status pada kelas *PositionStatus* adalah *Ready*,
4 *Initializing*, *NoData*, *Disable*, *NotInitialized*, dan *NotAvailable*.

- 5 ● *public PositionAccuracy DesiredAccuracy { get; set; }*

6 Properti yang digunakan untuk mengatur dan mendapatkan tingkat akurasi. Untuk
7 tingkat akurasi dapat dipilih tingkat *High* untuk tingkat akurasi tinggi dan dipilih
8 tingkat *Default* untuk menghemat daya. Keluaran dari properti ini adalah tipe data
9 *PositionAccuracy*.

- 10 ● *public Nullable<uint> DesiredAccuracyInMeters { get; set; }*

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

- 13 ● *public uint ReportInterval { get; set; }*

14 Merupakan properti untuk mendapatkan selang waktu pembaruan lokasi. Properti ini
15 mengejarkan tipe data unit.

16 2.1.6.5 Kelas *Geoposition*

17 *Geoposition* merupakan kelas yang memuat lokasi (*latitude* dan *longitude*). Berikut

18 Properti yang disediakan kelas *Geoposition*:

- 19 ● *public CivicAddress CivicAddress { get; }*

20 Data alamat sipil yang terkait dengan lokasi geografis.

- 21 ● *public Geocoordinate Coordinate { get; }*

22 Data latitude dan longitude yang terkait lokasi geografis.

23 2.1.6.6 Kelas *ReverseGeocodeQuery*

24 *ReverseGeocodeQuery* merupakan kelas yang digunakan untuk mambalikan objek *query*

25 *geocode*. Kelas *ReverseGeocodeQuery* dapat mendapatkan alamat dari kordinat *latitude* dan

26 *longitude*. Berikut Properti yang disediakan kelas *ReverseGeocodeQuery*:

- 27 ● *GeoCoordinate { get; set; }*

28 Untuk menampung kordinat yang akan digunakan.

29 Berikut *method* yang disediakan *ReverseGeocodeQuery*:

- 30 ● *QueryAsync()*

31 Memulai permintaan.

32 Berikut *event* yang disediakan *ReverseGeocodeQuery*:

- 33 ● *QueryCompleted*

34 Terjadi ketika permintaan selesai dijalankan.

1 2.1.7 Memanfaatkan Sumber Data

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

6 Sebelum data dapat dikirim antar perangkat perlu dilakukan *serialization*. *Serialization*
7 merupakan proses mentransformasikan objek ke format yang bisa dengan mudah dikirim
8 melewati jaringan atau disimpan di database[2]. Formatnya disini berupa string yang direpre-
9 sentasikan sebagai objek di XML atau JSON(Javascript Object Notation). Setelah data
10 yang dikirim didapatkan maka perlu dilakukan *deserialization*. *Deserialization* merupakan
11 proses mentransformasikan dari format yang bisa dengan mudah dikirim melewati jaringan
12 ke dalam bentuk objek.

13 Banyak *web service* yang mengembalikan data dalam format JSON. JSON memiliki
14 format *data-interchange* yang ringan [4]. Karena hal tersebut JSON mudah diurai dan di-
15 hasilkan oleh mesin. Kurung kurawal mengindikasikan objek, kurung siku berarti array,
16 dan properti berupa nama dan nilai pasangan yang dipisahkan oleh titik dua. JSON for-
17 mat memiliki ukuran data yang kecil dan baik untuk penggunaan perangkat *mobile*. Untuk
18 contoh format JSON dapat dilihat di bagian Kiri API pada Bab dua ini karena Kiri API
19 menggunakan format JSON. *Serializes* menggunakan *DataContractJsonSerializer* membuat
20 *serializes* mudah untuk menerjemahkan form String JSON ke objek yang dapat langsung
21 digunakan. *DataContractJsonSerializer* memakai *WriteObject()* untuk *serializes* and *Rea-*
22 *dObject()* untuk *deserializes*.

23 2.1.7.1 *HttpClient*

24 Merupakan Kelas yang dipakai untuk mengirim permintaan HTTP dan menerima kembal-
25 ian HTTP dari *Uniform Resource Identifier*(URI) yang dapat diidentifikasi. *Uniform Re-*
26 *source Identifier*(URI) merupakan urutan kompak karakter yang mengidentifikasi sumber
27 daya abstrak dan fisik [5]. Berikut *method* yang disediakan kelas *HttpClient*.

28 • *DeleteAsync(Uri)*

29 *Method* yang dipakai untuk mengirimkan permintaan DELETE ke URI yang spesifik
30 sebagai operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memung-
31 kinkan aplikasi untuk melanjutkan pekerjaan selagi *method* ini dipanggil². *Method* ini
32 membutuhkan parameter URI sebagai tujuan dari permintaan. Sedangkan kembali-
33 annya berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek
34 tersebut memiliki 2 parameter yaitu hasil berupa pesan kembalian dari http dan ke-
35 majuan dari data yang dikirim.

36 • *GetAsync(Uri)*

37 *Method* yang dipakai untuk mengirimkan permintaan GET ke URI yang spesifik seba-
38 gai operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan
39 aplikasi untuk melanjutkan pekerjaan selagi *method* ini dipanggil². *Method* ini mem-
40 butuhkan parameter URI sebagai tujuan dari permintaan. Sedangkan kembaliannya
41 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.

● *GetAsync(Uri,HttpCompletionOption)*

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 dan nilai tambahan yang dimasukan sebagai indikasi operasi dianggap selesai. Sedangkan kembaliannya 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.

● *GetBufferAsync(Uri)*

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 kembaliannya berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut memiliki 2 parameter yaitu hasil berupa pesan kembalian yang dikirimkan secara buffer(disimpan dalam memori) dan kemajuan dari data yang dikirim.

● *GetInputStreamAsync(Uri)*

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 kembaliannya berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut memiliki 2 parameter yaitu hasil berupa pesan kembalian yang dikirimkan secara stream(langsung sesuai waktu) dan kemajuan dari data yang dikirim.

● *GetStringAsync(Uri)*

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 kembaliannya 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.

● *PostAsync(Uri)*

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 kembaliannya berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut

1 memiliki 2 parameter yaitu hasil berupa pesan kembalian dari http dan kemajuan dari
2 data yang dikirim.

3 • *SendRequestAsync(HttpRequestMessage)*

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

9 • *SendRequestAsync(HttpRequestMessage, HttpCompletionOption)*

10 *Method* yang dipakai untuk mengirimkan permintaan HTTP sebagai operasi *asyn-
11 chronous*. Maksud dari operasi *asynchronous* adalah memungkinkan aplikasi untuk
12 melanjutkan pekerjaan selagi *method* ini dipanggil². *Method* ini membutuhkan pa-
13 rameter pesan dari permintaan dan nilai tambahan yang dimasukan sebagai indikasi
14 operasi dianggap selesai. Sedangkan kembalinya berupa objek yang mewakili ope-
15 rasi *asynchronous* disertai kemajuan. Objek tersebut memiliki 2 parameter yaitu hasil
16 berupa pesan kembalian dari http dan kemajuan dari data yang dikirim.

18 **2.1.7.2 Json.NET**

19 Json.NET merupakan *library* untuk melakukan *serialize* dan *deserialize* terhadap objek
20 dalam .NET [6]. *Library* ini memiliki 2 kelas yaitu kelas JsonConvert dan kelas JsonSerializer.
21 Berikut merupakan keterangan dari dua kelas tersebut.

22 • Kelas JsonConvert merupakan kelas yang dapat mengkonversi objek ke JSON dalam
23 bentuk String dan sebaliknya. Berikut 2 buah *method* yang disediakan kelas JsonConvert.
24

26 1. *Method SerializeObject()*

27 *Method* yang dipakai untuk mengkonversi objek ke JSON dalam bentuk String.

28 2. *Method DeserializeObject()*

29 *Method* yang dipakai untuk mengkonversi JSON dalam bentuk String ke objek.

30 • Kelas JsonSerializer merupakan kelas yang dapat melakukan *serializes* dan *deserializes*
31 objek ke dan dari JSON format.

32 **2.1.8 Thread**

33 Thread merupakan entitas dalam suatu proses yang dapat dijalankan untuk eksekusi
34 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[7]. 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          "traveltimes": any text string, null if and only if route is not found.
29        },
30        {
31          "steps": [ ... ],
32          "traveltimes": "..."
33        },
34        {
35          "steps": [ ... ],
36          "traveltimes": "..."
37        },
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 kendaran.
- 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 *mobile*.
 - %toicon = untuk menunjukkan ikon "to". Biasanya untuk mode presentasi di perangkat *mobile*.
- Indeks ke 4 atau bari 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 di akses 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 veris 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 Lembalian 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"	Mengintruksikan 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 :ayanan 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 kendaran.
- 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

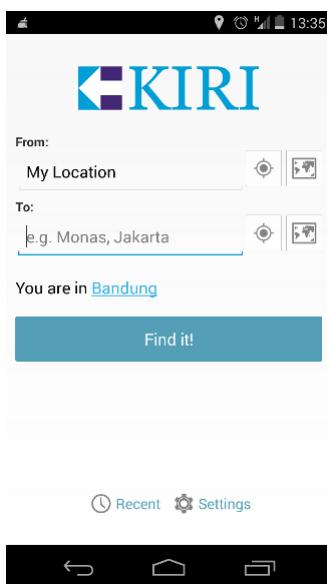
ANALISIS

- 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 Public Transport penggunaannya sederhana.
10 Di halaman awal pengguna dapat mengetik lokasi asal 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.

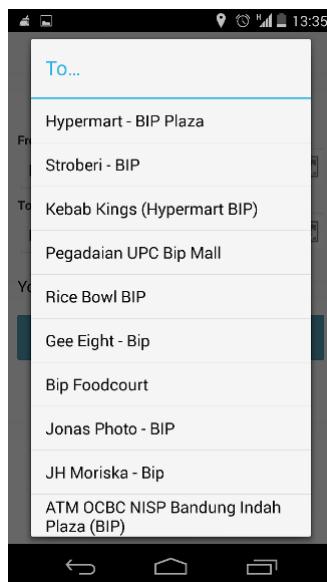
¹<https://play.google.com/store/apps/details?id=travel.kiri.smarttransportapp>

- 1 Bila pengguna ingin menunjuk lokasi pengguna berada dapat dilakukan dengan mengetuk
- 2 tombol kordinat. Tersedia juga pilihan 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 memilihnya.



Gambar 3.3: Memberikan Daftar Nama Tempat dan Nama Jalan Terkait

- 5 Pada gambar 3.3 pengguna dapat memilih nama tempat terkait. Pemilihan didasarkan
- 6 sesuai masukan pengguna untuk memastikan tempat asal maupun tempat tujuan. Jika
- 7 nama tempat sudah jelas maka tidak akan ada halaman ini.
- 8 Pada gambar 3.4 menampilkan daftar rute kendaraan umum yang harus dinaiki beserta
- 9 gambar untuk mempermudah pengguna. Selain itu disertakan juga jarak dan perkiraan
- 10 waktu sampai di lokasi tujuan.



Gambar 3.4: Tampilan Rute Kendaraan Umum dalam Bentuk Daftar



Gambar 3.5: Tampilan Rute Kendaraan Umum di Peta

- 1 Pada gambar 3.5 menampilkan rute kendaraan umum dan jalur yang harus dilalui pada peta. Dengan cara ini pengguna dapat mengetahui posisi dan jalur yang harus dilalui.

3.2 Analisis Aplikasi

- 4 Aplikasi akan dibuat menggunakan bahasa pemrograman C#. Aplikasi 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.

¹ 3.2.1 Kebutuhan Aplikasi

² Dari hasil observasi penulis dalam menentukan lokasi asal dan lokasi tujuan ada dua cara.
³ Kedua cara tersebut yaitu dengan menulis alamat atau tempat dan dengan menunjuk pada
⁴ peta. Cara menuliskan alamat atau tempat yaitu dengan menuliskan alamat atau tempat
⁵ pada tempat yang disediakan untuk asal dan tujuan. Cara menunjuk pada peta yaitu dengan
⁶ mengetuk layar di posisi yang diinginkan. Kedua hal tersebut pada dasarnya sama saja tetapi
⁷ ada faktor kemudahan pengguna dalam pemakaiannya. Jadi penulis menyediakan dua cara
⁸ tersebut pada aplikasi yang penulis buat agar pengguna dapat memilih salah satunya.

⁹ Pada saat menuliskan lokasi atau tempat atau menunjuk langsung pada peta mungkin
¹⁰ saja terjadi kesalahan. Kesalahan tersebut bisa saja disebabkan salah pengetikan atau nama
¹¹ tempat yang tidak ada. Maka dari itu dibutuhkan pemeriksaan terhadap masukan penggu-
¹² na. Pemeriksaan tersebut dilakukan setelah pengguna memulai pencarian dengan menekan
¹³ tombol "FIND".

¹⁴ Untuk hasil keluaran ada dua tipe seperti aplikasi peta lainnya. Kedua tipe tersebut
¹⁵ adalah bentuk daftar dan bentuk peta. Bentuk daftar memudahkan dalam melihat tiap
¹⁶ langkah rute. bentuk peta memudahkan pengguna dalam melihat arah dan posisi lingkungan
¹⁷ pada rute yang dipilih.

¹⁸ Berdasarkan kebutuhan aplikasi diatas maka aplikasi yang penulis bangun didasarkan
¹⁹ pada kebutuhan sebagai berikut.

- ²⁰ 1. Pengguna dapat memasukan lokasi asal dan lokasi tujuan pada *TextBox* yang disedi-
²¹ akan atau menunjuk langsung lokasi pada peta.
- ²² 2. Mendapatkan lokasi terkait menurut lokasi yang dimasukan pengguna.
- ²³ 3. Menampilkan hasil rute angkutan umum dari lokasi asal ke lokasi tujuan.

²⁴ 3.2.2 Analisis Kontrol yang Dipakai

²⁵ Dari kebutuhan yang telah disebutkan diatas penulis menyadari pentingnya kontrol yang
²⁶ harus dipakai. Kontrol yang dimaksud termasuk tata letak, teks, pilihan, dan daftar. Ke-
²⁷ butuhan akan kontrol penting bukan hanya untuk kebutuhan tapi memudahkan pengguna.

²⁸ Untuk kontrol tata letak penulis membayangkan pengaturan yang tertata rapih dan
²⁹ beberapa elemen dalam satu baris atau kolom. Tetapi juga penulis tidak mengharapkan
³⁰ penggunaan kontrol tata letak yang rumit. Dari hasil pengamatan penulis kontrol tata
³¹ letak yang cocok adalah Grid. Kontrol tata letak ini penulis pilih karena mudah diposisikan
³² sesuai baris dan kolom. Selain itu tampilan Grid akan menyesuaikan jika layar diputar dari
³³ posisi pemandangan ke posisi potret dan sebaliknya.

³⁴ Kontrol terhadap teks juga diperlukan untuk aplikasi. Kebutuhan yang diperlukan ada-
³⁵ lah mengeluarkan potongan teks yang dapat dibaca dan tempat pengguna memasukan teks.
³⁶ Untuk mengeluarkan teks yang dapat dilihat penulis akan menggunakan *TextBlock*. *Te-
37 xtBlock* digunakan untuk menampilkan tulisan "from" dan "to" pada halaman utama apli-
³⁸ kasi. Untuk masukan pengguna terhadap aplikasi penulis akan menyediakan *TextBox* sebagai
³⁹ tempat teks. *TextBox* digunakan sebagai masukan untuk lokasi asal dan lokasi tujuan.

⁴⁰ Suatu aplikasi tentunya tidak hanya mempunyai satu halaman. Sama hal dengan aplikasi
⁴¹ yang penulis buat memiliki beberapa halaman yang mempunyai tugas berbeda. Karena hal

1 tersebut dibutuhkan kontrol untuk berpindah dari satu halaman ke halaman lain. Kontrol
2 yang dibutuhkan yaitu kontrol tombol. Kontrol tombol akan mengeksekusi *event click* yang
3 memungkinkan pindah halaman dan melakukan perintah. Kontrol tombol akan penulis
4 gunakan untuk berpindah ke halaman peta, menemukan lokasi pengguna, dan pencarian
5 rute. Pada halaman utama terdapat 5 tombol yaitu tombol map pada bagian from, tombol
6 here pada bagian from, tombol map pada bagian to, tombol here pada bagian to, dan tombol
7 find. Berikut kegunaan dari tombol-tombol tersebut.

8 ● Tombol "map" pada bagian from

9 Tombol untuk berpindah dari halaman utama menuju halaman peta. Pada halaman
10 peta pengguna dapat menunjuk lokasi asal dan kembali lagi ke halaman utama. Saat
11 kembali ke halaman utama lokasi yang dipilih akan disimpan dan pada *TextBox* bagian
12 from akan tertulis "Maps".

13 ● Tombol "here" pada bagian from

14 Tombol untuk mencari lokasi pengguna. Setelah tombol ini di tekan maka lokasi
15 pengguna akan disimpan dan pada bagian *TextBox* bagian from akan tertulis "Here".

16 ● Tombol "map" pada bagian to

17 Tombol untuk berpindah dari halaman utama menuju halaman peta. Pada halaman
18 peta pengguna dapat menunjuk lokasi tujuan dan kembali lagi ke halaman utama.
19 Saat kembali ke halaman utama lokasi yang dipilih akan disimpan dan pada *TextBox*
20 bagian to akan tertulis "Maps".

21 ● Tombol "here" pada bagian to

22 Tombol untuk mencari lokasi pengguna. Setelah tombol ini di tekan maka lokasi
23 pengguna akan disimpan dan pada bagian *TextBox* bagian to akan tertulis "Here".

24 ● Tombol find Tombol ini akam mencari rute angkutan umum dan menampilkannya
25 pada halaman peta.

26 Pada aplikasi ini penulis akan menampilkan daftar tempat dan daftar rute angkutan
27 umum yang dipakai. Bentuk daftar digunakan penulis karena hasil tempat dan rute ang-
28 kutan umum akan banyak. Bentuk daftar yang dapat dipakai di Windows Phone adalah
29 menggunakan *ListBox*. *ListBox* akan menampilkan daftar tempat dan daftar rute satu per
30 satu menurun ke bawah.

31 **3.2.3 Analisis Terhadap Siklus Hidup Aplikasi**

32 Aplikasi pada Windows Phone memiliki siklus hidup yang dijelaskan pada bab 2.1.4.
33 Pengaturan aplikasi ini diatur sesuai konfigurasi awal sistem operasi Windows Phone. Tetapi
34 pengaturan ini dapat diatur sesuai kebutuhan aplikasi. Karena di aplikasi ini terdapat
35 keadaan yang berbeda dengan konfigurasi awal sistem operasi Windows Phone maka perlu
36 dilakukan pengaturan ulang siklus hidup.

37 Saat aplikasi dijalankan, pengguna memasukan lokasi asal dan lokasi tujuan. Setelah
38 memasukan lokasi pengguna akan mencari rute. Ketika rute berhasil ditemukan aplikasi akan
39 berada di keadaan Running. Tetapi ada kemungkinan pengguna berpindah aplikasi atau
40 mematikan layar untuk menghemat daya. Dalam kasus tersebut sistem operasi Windows

1 Phone akan menganggap aplikasi tidak aktif dan aplikasi akan masuk pada keadaan *dormant*.
2 Untuk menangani kasus tersebut maka penulis harus menyimpan keadaan dan informasi
3 sesaat sebelum aplikasi menjadi tidak aktif. Penanganan yang penulis akan lakukan adalah
4 menggunakan *method OnNavigatedFrom()*. Dengan *method* tersebut keadaan aplikasi akan
5 disimpan di memori.

6 Pada saat aplikasi masuk keadaan *Dormant* semua *thread* dan proses akan dihentikan.
7 Pada saat tersebut juga GPS Windows Phone akan terhenti dan tidak akan mengetahui
8 posisi pengguna. GPS akan kembali aktif mengetahui posisi pengguna jika pengguna masuk
9 ke aplikasi dan tentunya membutuhkan waktu untuk pelacakan lokasi. Tetapi Windows
10 Phone mendukung proses di belakang untuk pelacakan GPS selama keluar dari aplikasi
11 atau layar perangkat dimatikan. Maka dari itu aplikasi yang penulis buat akan mendukung
12 pengaksesan lokasi meskipun layar perangkat dimatikan atau berpindah aplikasi.

13 Ketika aplikasi sudah berada pada keadaan *Dormant* atau *Tombstoned*, pengguna masih
14 dapat memulihkan keadaan aplikasi saat aplikasi berada di keadaan *Running* sebelumnya.
15 Penanganan yang penulis akan lakukan untuk hal tersebut adalah menggunakan *method*
16 *OnNavigatedTo()*. Menggunakan *method* tersebut akan memulihkan informasi halaman pada
17 keadaan *Running* sebelumnya.

18 **3.2.4 Analisis Peta**

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

- 21 • Pemetaan terhadap peta atau *cartographic* dan mode warna
22 • Tingkat *zoom*
23 • Menampilkan gambar dan keterangan angkutan umum menggunakan *pushpin*
24 • Menggambar rute pada peta menggunakan *polyline*

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

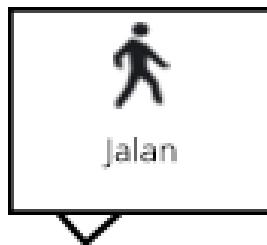
32 Tampilan awal peta di Windows Phone akan mengeluarkan peta dengan pandangan du-
33 nia. Karena aplikasi pencarian rute ini masih terbatas di Pulau Jawa, Indonesia terutama
34 Jawa Barat maka tingkat zoom harus diatur agar mengikuti lokasi pengguna dan di satu
35 daerah saja. Jika pengguna berada di daerah Bandung maka tingkat zoom pada peta di-
36 sesuaikan pada daerah tersebut. Tingkat zoom dapat dapat diatur dari kode dan XAML.
37 Tingkat zoom yang penulis akan gunakan adalah 14. Tingkat zoom 14 akan menampilkan
38 satu kota dengan jelas.

39 Di setiap kota ada satu angkutan umum yang banyak dipakai yaitu angkutan kota(angkot).
40 Namun bagi yang baru pertama mengunjungi suatu daerah dan mencari angkot mungkin

akan kesulitan membaca trayek/rute dari angkot tersebut. Namun ada satu cara yang mudah untuk membedakan angkot di setiap rute yaitu dari warna dan coraknya. Agar dapat memudahkan pengguna dan menghindari pengguna dari kesalahan naik angkot maka Kiri API sudah menyediakan gambar angkot yang sesuai dengan setiap rute. Gambar angkot tersebut akan ditempatkan di peta dengan suatu penampung beserta keterangannya. Salah satu teknik untuk menempatkan gambar tersebut adalah dengan membuat lapisan terpisah di atas tempat gambar tersebut. Untuk hal tersebut penulis akan memanfaatkan Pushpin sebagai lapisan terpisah untuk menaruh gambar dan keterangan angkutan umum. Berikut 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

Pencarian rute yang penulis gunakan untuk aplikasi yaitu dengan memakai Kiri API. Kiri API akan memberikan kembalian berupa titik-titik rute perjalanan dari lokasi asal ke lokasi tujuan. Karena hal itu penulis harus menggambar rute tersebut sesuai jalan pada peta. Untuk hal tersebut penulis akan menggunakan *Polyline* pada Windows Phone untuk menggambarnya. *Polyline* yang digambar harus terlihat dengan jelas dan diberi warna yang kontras dengan tampilan peta. Warna polyline yang penulis akan pilih adalah merah dengan ketebalan 4.

3.2.5 Analisis Pemanfaatan Sumber Data

Aplikasi yang penulis buat memanfaatkan sumber data dari luar. Sumber data yang penulis dapatkan dalam format JSON (*Javascript object Notation*). Pengambilan sumber data tersebut dilakukan dengan melakukan permintaan melalui protokol HTTP dengan parameter *Uniform Resource Identifier / URI*.

Untuk dapat memanfaatkan sumber data di Windows Phone penulis akan memanfaatkan kelas *HttpClient.Method* yang penulis gunakan adalah *GetStringAsync()*. *Method* ini akan

1 mengirimkan permintaan dengan parameter URI dan hasil yang dikembalikan memiliki tipe
2 data *String*. Karena *method* ini mengembalikan hasil dalam tipe data *String* maka mudah
3 disesuaikan dengan kebutuhan tugas akhir ini. Selanjutnya agar data dalam bentuk *String*
4 tersebut dapat dimanfaatkan maka diperlukan proses *Deserialization*. Proses *Deserialization*
5 dimaksudkan untuk mengubah bentuk *String* ke objek. Untuk melakukan *Deserialization*
6 penulis memanfaatkan *library* Json.NET. Penulis memanfaatkan *library* Json.NET karena
7 memiliki performa yang baik. Untuk hal tersebut penulis akan menggunakan *method* *De-
8 serializeObject()*.

9 **3.2.6 Analisis Kiri API**

10 Kiri API menyediakan 2 parameter untuk permintaan yaitu *POST* dan *GET*. Dalam
11 tugas akhir ini penulis akan menggunakan parameter *GET*. Parameter **GET** penulis pilih
12 karena dalam tugas akhir ini penulis akan banyak mendapatkan data dan tidak ada data
13 sensitif yang dikirimkan. Untuk hal ini penulis akan mengirim ke URI [http://kiri.travel/
14 handle.php](http://kiri.travel/handle.php).

15 Untuk setiap permintaan terhadap Kiri API dibutuhkan *API key*. Kegunaan *API key*
16 adalah password untuk mengakses Kiri API. *API key* dapat didapatkan di [dev.kiri.
travel](http://dev.kiri.
17 travel). *API key* yang penulis gunakan pada tugas akhir ini adalah 97A7A1157A05E***.

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

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

- 25 • *version: 2*

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

- 28 • *mode: "searchplace"*

29 Mode "searchplace" digunakan untuk mencari lokasi terkait.

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

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

- 39 • *querystring:* merupakan kata kunci lokasi

- 40 • *apikey:* 16 digit heksadesimal

- 1 Format layanan yang dikirim melalui URL adalah `kiri.travel/handle.php?version=2&mode=searchplace®ion=cgk/bdo/sub&querystring="string"&apikey=97A7A1157A05E***.`
- 2
- 3
- 4 Penulis mencoba mencari lokasi bip dari kata kunci "bip" yang berada di bandung. La-
- 5 yan dan dikirimkan ke URL `kiri.travel/handle.php`. Berikut format layanan yang penulis
- 6 kirim:
- 7 `http://kiri.travel/handle.php?version=2&mode=searchplace®ion=bdo&querystring=`
- 8 `bip&apikey=97A7A1157A05E***`
- 9

10 Berikut hasil kembalian dari Kiri API:

Listing 3.1: Kode Kembalian dari Pencarian Rute

```

11 {
12     "status ":"ok",
13     "searchresult ":[
14         {
15             "placename ":"Hypermart - BIP Plaza",
16             "location ":"-6.90864,107.61108"
17         },
18         {
19             "placename ":"Stroberi - BIP",
20             "location ":"-6.90834,107.61115"
21         },
22         {
23             "placename ":"Kebab Kings (Hypermart BIP)",
24             "location ":"-6.91503,107.61017"
25         },
26         {
27             "placename ":"Pegadaian UPC Bip Mall",
28             "location ":"-6.90916,107.61052"
29         },
30         {
31             "placename ":"Rice Bowl BIP",
32             "location ":"-6.90873,107.61088"
33         },
34         {
35             "placename ":"Gee Eight - Bip",
36             "location ":"-6.90817,107.61080"
37         },
38         {
39             "placename ":"Jonas Photo - BIP",
40             "location ":"-6.91066,107.61016"
41         },
42         {
43             "placename ":"Bip Foodcourt",
44             "location ":"-6.91081,107.61015"
45         },
46         {
47             "placename ":"Mister Baso BIP",
48             "location ":"-6.90348,107.61709"
49         },
50         {
51             "placename ":"JH Moriska - Bip",
52             "location ":"-6.90868,107.61070"
53         }
54     ],
55     "attribution":null
56 }
```

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

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

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

60 percobaan yang penulis lakukan, nilai dari *attribution* selalu bernilai "null". Karena hal

61 tersebut maka nilai *attribution* akan penulis abaikan.

62 Layanan lainnya yang penulis akan manfaatkan adalah layanan penentuan rute. Pe-

63 manfaatan layanan penentuan rute digunakan mendapatkan langkah-langkah yang harus

64 ditempuh pengguna untuk menuju lokasi tujuan dari lokasi asal. Pemanfaatan layanan

65 ini yaitu dengan parameter *GET* melalui protokol HTTP. Berikut parameter yang harus

1 dikirim:

2 ● *version*: 2

3 Karena acuan penulis adalah Kiri API versi 2 maka di parameter penulis akan meng-
4 gunakan versi 2.

5 ● *mode*: "findroute"

6 Mode "findroute" digunakan untuk mendapatkan langkah yang harus ditempuh me-
7 nuju lokasi tujuan.

8 ● *locale*: "en" untuk bahasa Inggris dan "id" untuk bahasa Indonesia.

9 Karena aplikasi ini bertujuan untuk memudahkan pemakaian transportasi di Indonesia
10 maka penulis putuskan untuk menggunakan bahasa Indonesia.

11 ● *start*: koordinat lokasi asal dalam bentuk *string* bernilai *latitude* dan *longitude* yang
12 dipisahkan koma.

13 Masukan untuk lokasi awal harus dalam bentuk kordinat. Jika masukan dari pengguna
14 adalah alamat atau tempat maka perlu dicari kordinatnya dahulu.

15 ● *finish*: koordinat lokasi tujuan dalam bentuk *string* bernilai *latitude* dan *longitude*
16 yang dipisahkan koma.

17 Masukan untuk lokasi tujuan harus dalam bentuk kordinat. Jika masukan dari peng-
18 guna adalah alamat atau tempat maka perlu dicari kordinatnya dahulu.

19 ● *presentation*: "mobile" untuk perangkat *mobile* dan "desktop" untuk komputer.

20 Karena aplikasi ini dirancang untuk Windows Phone 8, presentasi yang penulis pi-
21 ilih adalah "desktop". Pemilihan ini didasarkan karena deskripsi yang ditampilkan
22 tidak ada tulisan "image from" dan "image to", selain itu presentasi "desktop" juga
23 memungkinkan rute alternatif.

24 ● *apikey*: 16 digit heksadesimal.

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

26 Penulis mencoba menuju jalan merdeka dari jalan ciumbuleuit. Layanan dikirimkan ke
27 URL kiri.travel/handle.php?version=2&mode=findroute&locale=id&start=-6.8747337,107.6048829&finish=-6.9114646,107.6104887&presentation=mobile&apikey=97A7A1157A05ED6.

34 Berikut hasil kembalian dari Kiri API:

Listing 3.2: Kode Kembalian Pencarian Rute dengan *Presentation Mobile*

```
35 {  
36     "status": "ok",  
37     "routingresults": [  
38         {  
39             "steps": [  
40                 [  
41                     "walk",  
42                     "walk",  
43                     ["-6.8747337,107.6048829", "-6.87445,107.60465"],
```

```

1      " Jalan dari lokasi mulai Anda \%fromicon ke Jalan Ciumbuleuit \%toicon sejauh
2          kurang lebih 41 meter.", 
3          null ,
4          null
5      ],
6      [
7          "angkot",
8          "ciumbuleuitsthallurus",
9          [" -6.87445,107.60465"," -6.87541,107.60443"," -6.87637,107.60421"," -6.87734,107.60400",
10
11         "-6.87830,107.60378",
12         "-6.87926,107.60356"," -6.87926,107.60356"," -6.87963,107.60352",
13         "-6.87978,107.60352"," -6.88093,107.60392"," -6.88209,107.60433"," -6.88209,107.60433",
14
15         "-6.88328,107.60490"," -6.88328,107.60490"," -6.88347,107.60481"," -6.88452,107.60459",
16
17         "-6.88556,107.60436"," -6.88660,107.60413"," -6.88764,107.60390"," -6.88764,107.60391",
18
19         "-6.88782,107.60392"," -6.88887,107.60404"," -6.88991,107.60416"," -6.88991,107.60416",
20
21         "-6.89161,107.60428"," -6.89161,107.60428"," -6.89166,107.60421"," -6.89275,107.60424",
22
23         "-6.89275,107.60424"," -6.89405,107.60408"," -6.89405,107.60408"," -6.89496,107.60400"],
24
25         "Naik angkot Ciumbuleuit – St. Hall (lurus) di Jalan Ciumbuleuit \%fromicon,
26             dan turun di Jalan Cihampelas \%toicon kurang lebih setelah 3,3 kilometer
27             .",
28         null ,
29         "https://angkot.web.id/go/route/640?ref=kiri"
30     ],
31     [
32         "walk",
33         "walk",
34         [" -6.90424,107.60433"," -6.90429,107.60440"],
35         "Jalan dari Jalan Cihampelas \%fromicon ke Jalan Abdul Rivai \%toicon sejauh
36             kurang lebih 10 meter.",
37         null ,
38         null
39     ],
40     [
41         "angkot",
42         "kalapaledeng",
43         [" -6.89501,107.60403"," -6.89562,107.60398"," -6.89623,107.60395"," -6.89732,107.60401",
44
45         "-6.89732,107.60401"," -6.89882,107.60414"," -6.89882,107.60414"," -6.89969,107.60418",
46
47         "-6.90071,107.60426"," -6.90173,107.60433"," -6.90173,107.60433"," -6.90297,107.60437",
48
49         "-6.90420,107.60440"," -6.90420,107.60440"," -6.90426,107.60456"," -6.90422,107.60481",
50
51         "-6.90399,107.60546"," -6.90406,107.60617"," -6.90454,107.60697"," -6.90454,107.60697",
52
53         "-6.90512,107.60745"," -6.90618,107.60778"," -6.90618,107.60778"," -6.90643,107.60787",
54
55         "-6.90651,107.60807"," -6.90675,107.60914"," -6.90675,107.60914"," -6.90694,107.60939",
56
57         "-6.90723,107.60939"," -6.90891,107.60943"," -6.90891,107.60943"," -6.90909,107.60934",
58
59         "-6.90914,107.60857"," -6.90933,107.60846"," -6.91021,107.60887"," -6.91021,107.60887",
60
61         "-6.91030,107.60897"," -6.91028,107.60927"," -6.90986,107.61040"," -6.90986,107.61040"],
62
63         "Naik angkot Kalapa – Ledeng di Jalan Abdul Rivai \%fromicon, dan turun di
64             Jalan Aceh \%toicon kurang lebih setelah 1,1 kilometer.",
65         "https://angkot.web.id/go/route/156?ref=kiri"
66     ],
67     [
68         "walk",
69         "walk",
70         [" -6.90986,107.61040"," -6.9114646,107.6104887"],
71         "Walk about 178 meter from Jalan Aceh \%fromicon to your destination \%toicon
72             .",
73         null ,
74         null
75     ],
76     [
77         "traveltime ":"30 minutes"
78     }
79 ]
80 }

```

81 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=>

1 -6.9114646,107.6104887&presentation=desktop&apikey=97A7A1157A05ED6F.

2

3 Berikut hasil kembalian dari Kiri API:

Listing 3.3: Kode Kembalian Pencarian Rute dengan *Presentation Desktop*

```

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

```

```

1      " -6.90911,107.60826", "-6.91034,107.60892", "-6.91034,107.60892", "-6.91034,107.60892",
2      "-6.91007,107.60985"],
3      "Naik angkot Kalapa – Ledeng di Jalan Abdul Rivai, dan turun di Jalan Aceh
4      kurang lebih setelah 1,1 kilometer.", null,
5      "https://angkot.web.id/go/route/156?ref=kiri"
6      ],
7      [
8          "walk",
9          "walk",
10         ["-6.91007,107.60985", "-6.9114646,107.6104887"],
11         "Jalan dari Jalan Aceh ke tujuan akhir Anda sejauh kurang lebih 171 meter.", null,
12         null
13     ],
14     "traveltime": "30 menit"}]}
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37

```

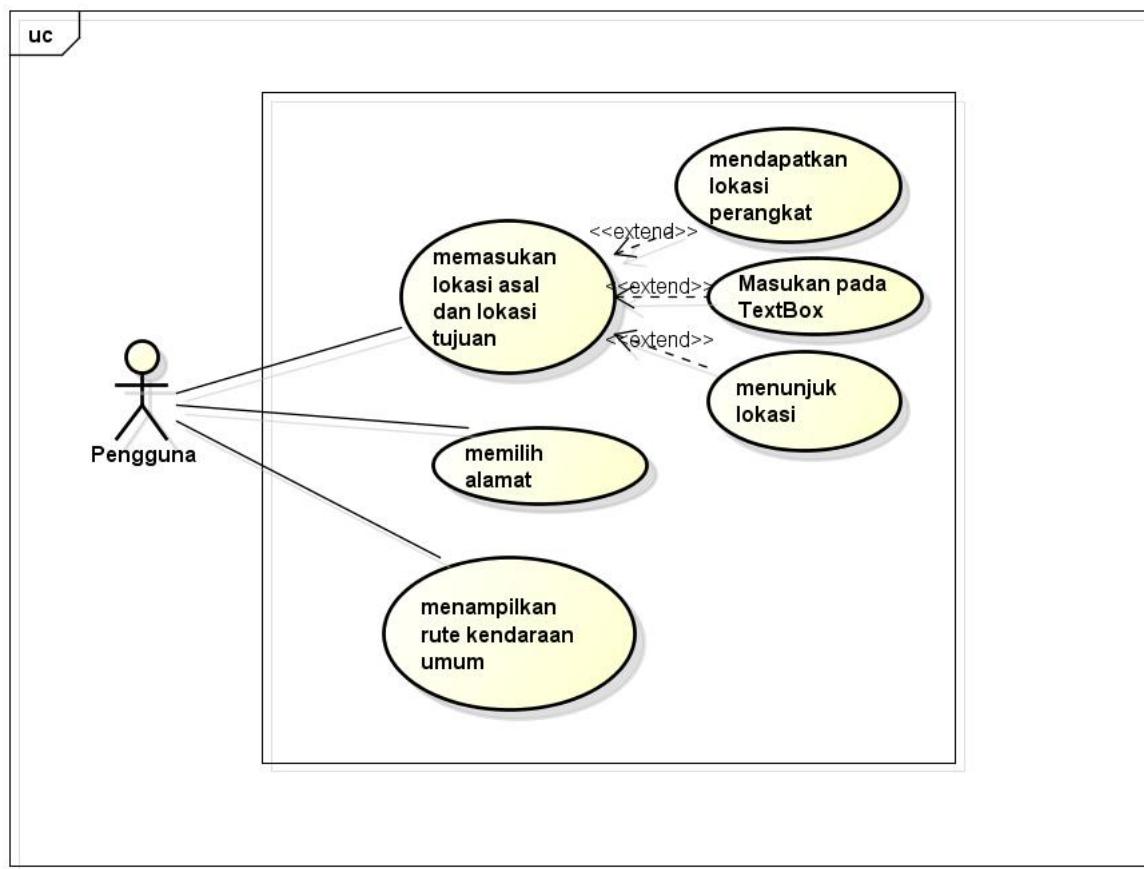
Setiap langkah akan aplikasi tampung dalam elemen *array*. Untuk keterangan dan jenis angkutan umum akan aplikasi tampilkan dalam bentuk *pushpin* pada peta atau daftar. Sedangkan untuk titik-titik kordinat akan digambarkan pada peta. Dari analisa penulis setiap langkah menunjukkan perpindahan angkutan 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 untuk keluaran jika melakukan permintaan dengan *presentation mobile*. Karena hal tersebut mode *presentation* yang akan digunakan adalah *desktop*. 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.

- ¹ Diagram *use case* saat pengguna mencari rute kendaraan umum dapat dilihat pada gambar (Gambar: 3.8):



powered by Astah

Gambar 3.8: Diagram *Use Case*

²

- ³ 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	<i>TextBox</i> masih kosong dan pengguna menekan tombol lokasi
Kondisi akhir	Lokasi ditemukan dan <i>TextBox</i> berisi "here"
Skenario utama	Pengguna menekan tombol lalu perangkat akan mencari lokasi perangkat dan <i>TextBox</i> berisi "here"
Eksespsi	Lokasi tidak ditemukan jika GPS perangkat tidak aktif

Tabel 3.1: Skenario Mendapatkan Lokasi untuk Masukan Lokasi Asal dan Lokasi Tujuan

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	<i>TextBox</i> 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 pada peta
Aktor	Pengguna
Deskripsi	Memasukan lokasi asal pengguna dan tujuan pengguna dengan menunjuk pada peta
Kondisi awal	<i>TextBox</i> masih dalam keadaan belum terisi
Kondisi akhir	<i>TextBox</i> terisi dengan "Maps"
Skenario utama	Pengguna menunjuk lokasi pada peta dan <i>TextBox</i> terisi dengan "Maps"
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

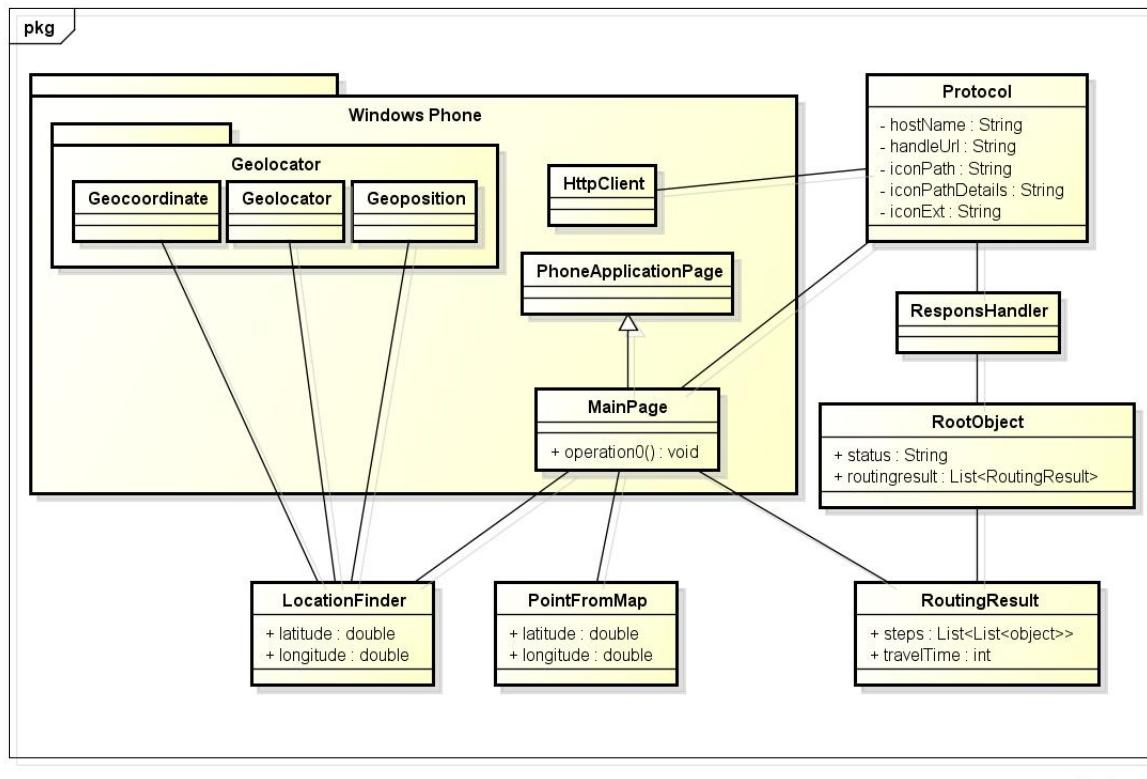
Tabel 3.4: Skenario Memilih Alamat

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

3.2.8 Kelas Diagram

Pembuatan kelas diagram didasarkan pada skenario pada sub-bab 3.2.7. Kelas diagram dapat dilihat pada gambar 4.4.



Gambar 3.9: Diagram Kelas

Berikut deskripsi kelas pada gambar 4.4.

- Kelas *Protocol*

Merupakan kelas yang menampung semua alamat URL yang berhubungan dengan Kiri API. Semua pemanggilan akan ditangani oleh kelas ini.

- Kelas *ResponsHandler*

Merupakan kelas yang menangani masukan dari pemanggilan layanan.

1 ● Kelas *RootObject*

2 Merupakan kelas untuk menampung status dan daftar dari layanan *routing* Kiri API.

3 Hasil kembalian akan dipisahkan di kelas ini untuk selanjutnya ditambahkan di kelas

4 *RoutingResult*.

5 ● Kelas *RoutingResult*

6 Merupakan kelas untuk menampung setiap langkah dari rute sesuai masukan pengguna.
7 Pada kelas ini juga rute akan digambarkan pada peta.

8 ● Kelas *PointFromMap*

9 Merupakan kelas yang dapat mengetahui lokasi yang ditunjuk pengguna pada peta.

10 Kelas ini akan menyimpan lokasi yang ditunjuk pengguna dalam bentuk *latitude* dan

11 *longitude*.

12 ● Kelas *LocationFinder*

13 Merupakan kelas yang digunakan untuk mencari lokasi. kelas ini akan memanfaatkan

14 kelas *Geocoordinate* untuk mendapatkan lokasi. Setelah lokasi didapatkan dalam

15 bentuk kelas *Geoposition* maka akan diubah ke *latitude* dan *longitude*.

1

BAB 4

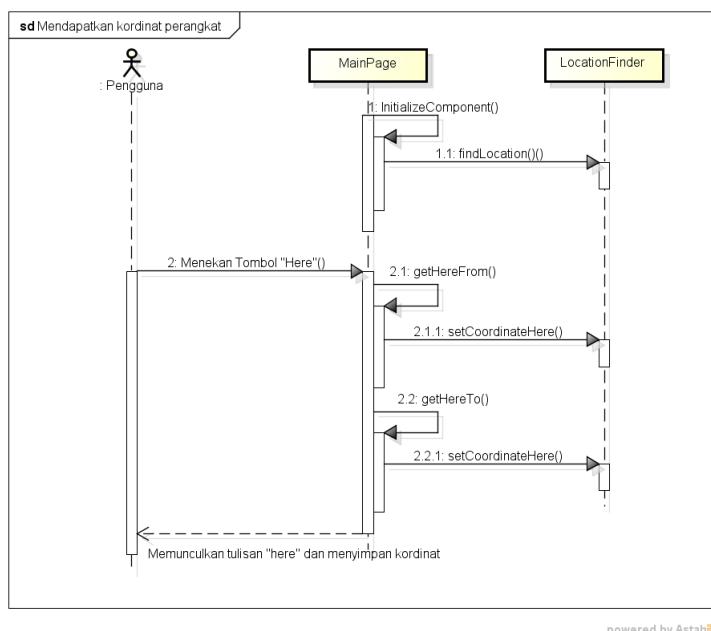
2

PERANCANGAN

- 3 Pada bab 4 akan dibahas mengenai perancangan aplikasi mulai dari diagram *sequence*, di-
4 agram kelas secara rinci, deskripsi artibut dan *method* dari setiap kelas, dan perancangan
5 antarmuka.

6 **4.1 Diagram *Sequence***

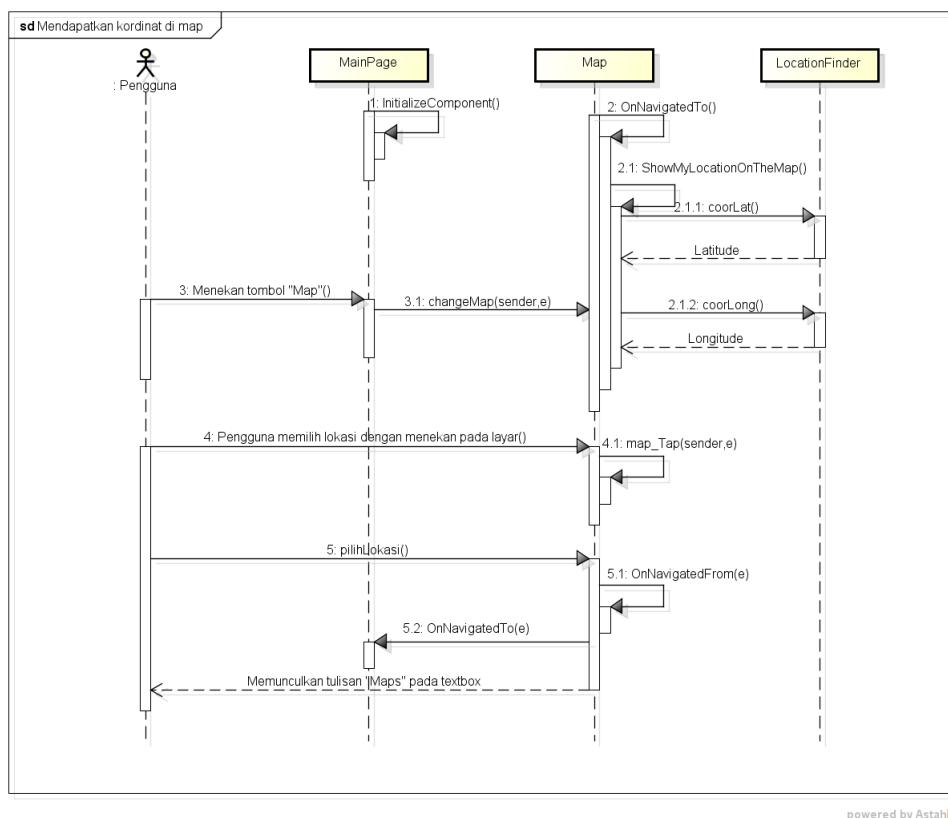
7 Diagram *sequence* merupakan diagram yang menggambarkan interaksi antar objek dalam
8 suatu skenario. Gambar diagram *sequence* dapat dilihat pada gambar 4.1 sampai 4.3.



Gambar 4.1: Diagram *sequence* Mendapatkan Kordinat Perangkat

9 Diagram 4.1 merupakan diagram *sequence* untuk penentuan lokasi asal/tujuan de-
10 ngan masukan lokasi perangkat. Diagram menunjukkan bahwa setelah aplikasi dibuka maka
11 aplikasi akan mencari dahulu lokasi perangkat dengan memanfaatkan kelas LocationFinder.
12 Lalu setelah aplikasi terbuka jika pengguna ingin memilih lokasi tersebut sebagai lokasi asal
13 maka pengguna harus menekan tombol "here". Setelah tombol "here" ditekan maka kelas
14 MainPage akan mengambil nilai *Latitude* dan nilai *Longitude* dari kelas LocatonFinder. Se-
15 telah lokasi didapatkan maka akan muncul tulisan "here" pada *TextBox* 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

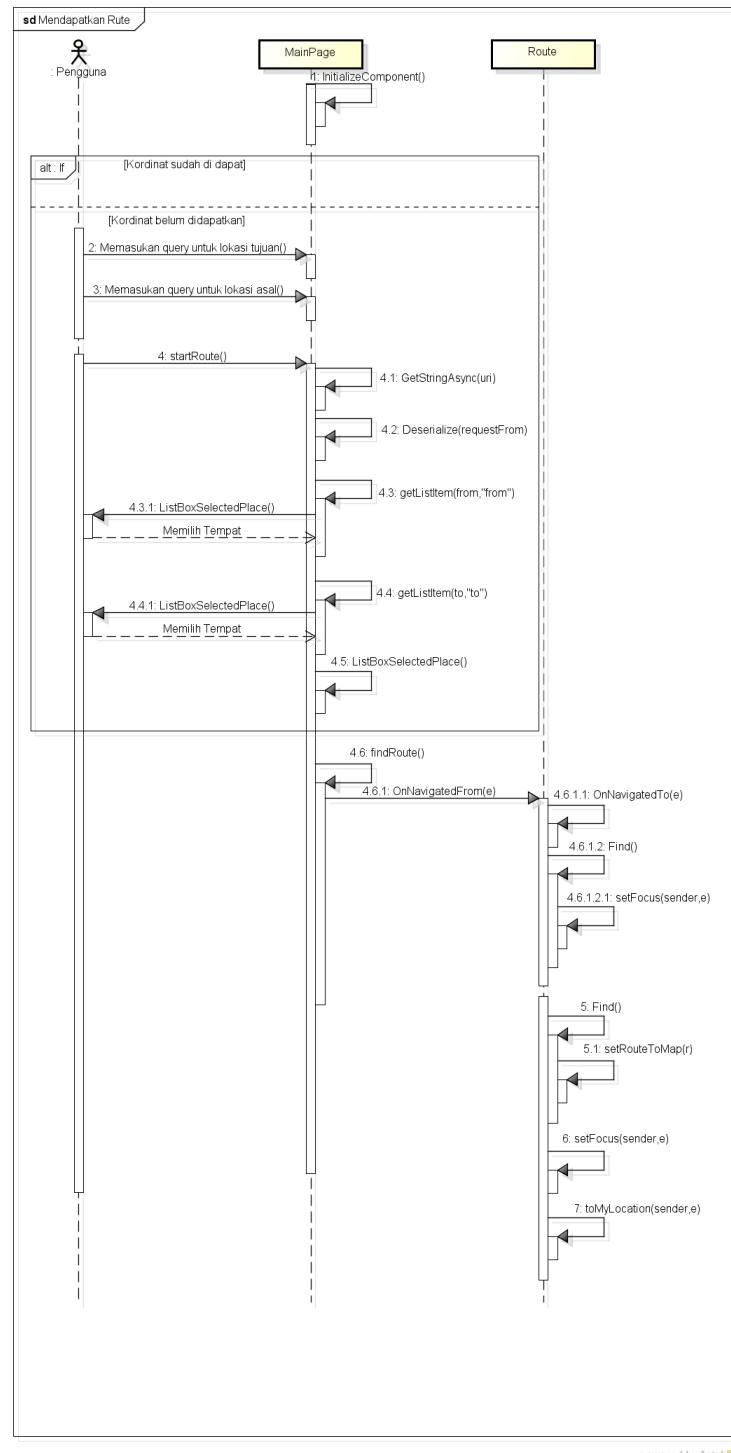


Gambar 4.2: Diagram *sequence* Mendapatkan Kordinat pada Peta

- 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 memanggil
- 5 *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 "Maps".

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 didapat 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

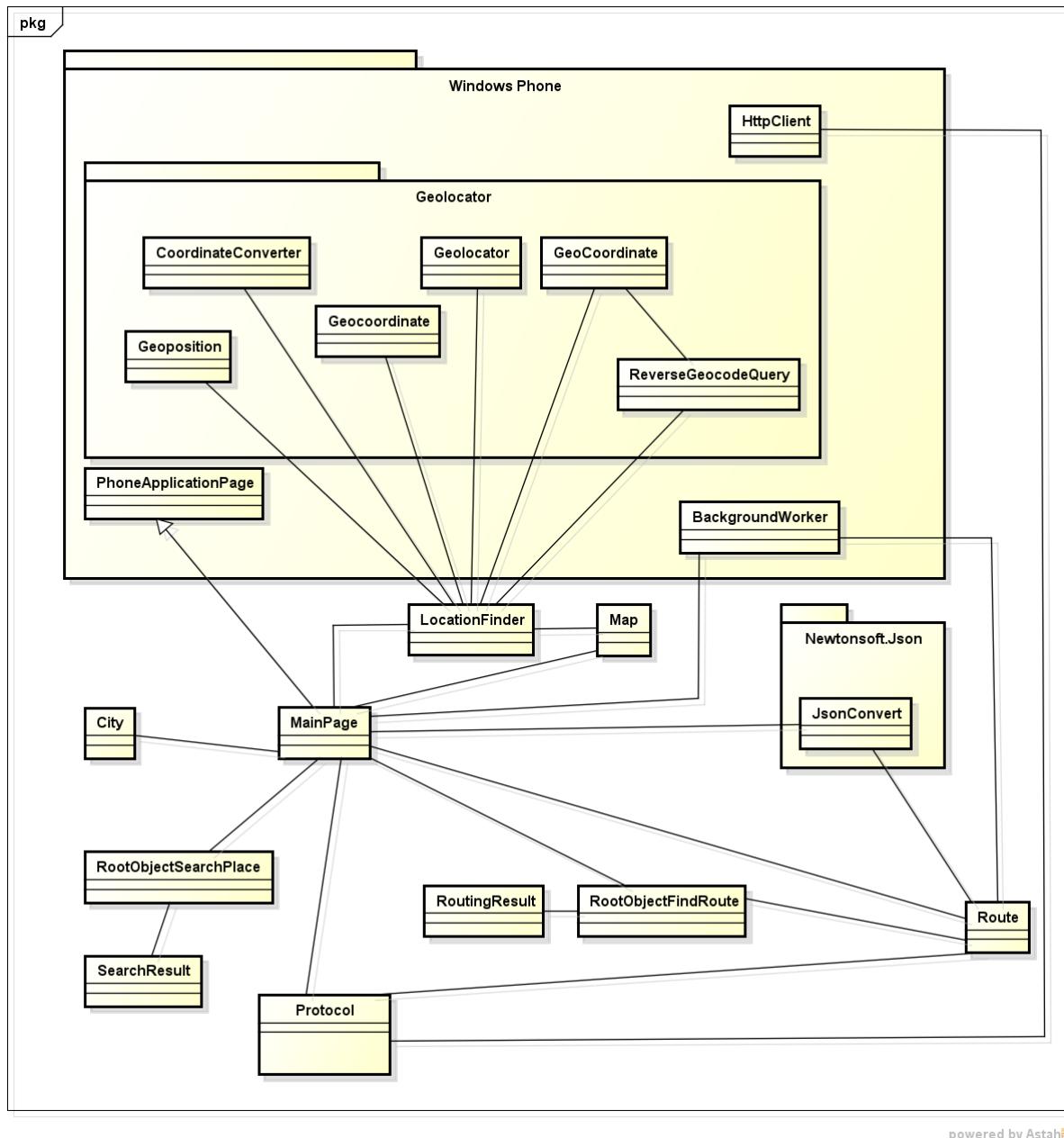
- 1 mengembalikan rute yang ditemukan kepada pengguna.



Gambar 4.3: Diagram *sequence* Mendapatkan Rute

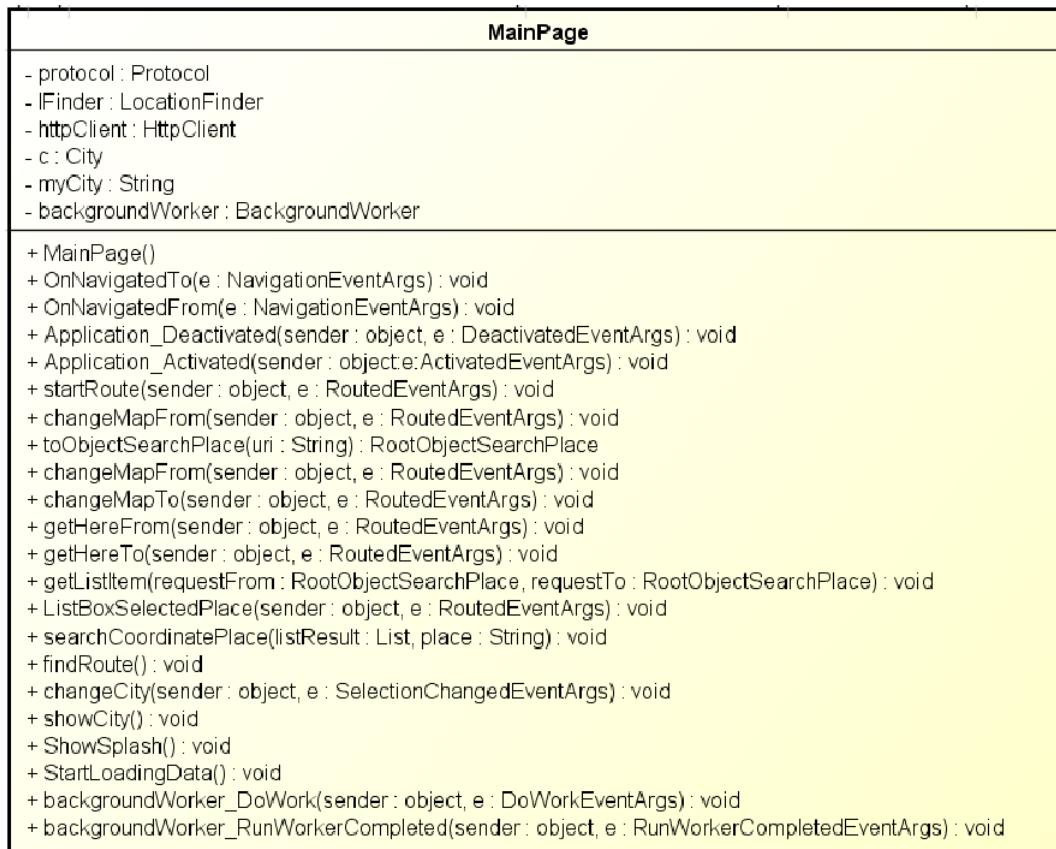
4.2 Perancangan Kelas

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.

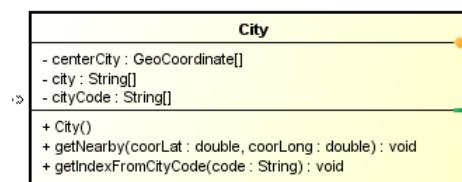


Gambar 4.4: Diagram Kelas

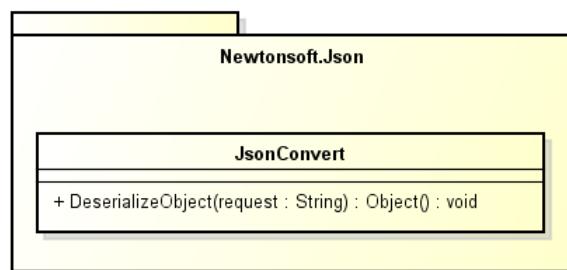
Berikut Penjelasan kelas diagram secara rinci dengan setiap kelas disertai atribut dan *method*.



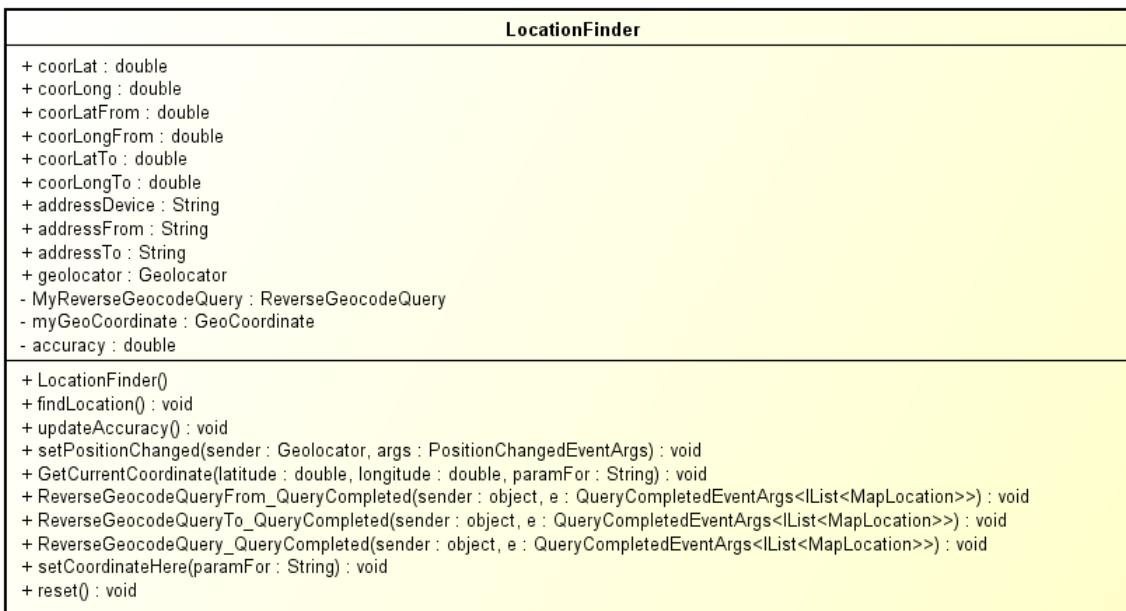
Gambar 4.5: Kelas Mainpage



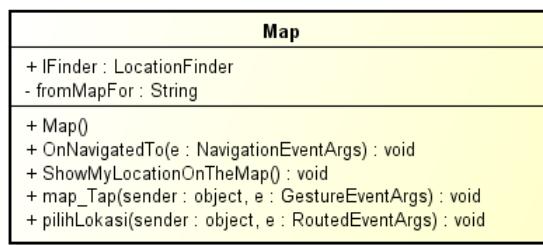
Gambar 4.6: Kelas City



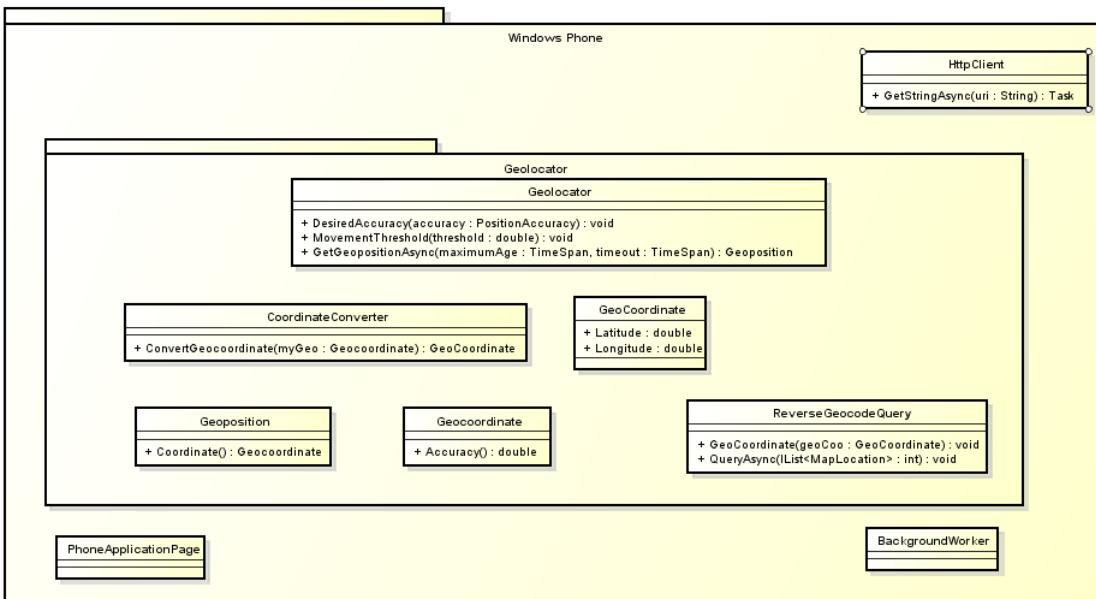
Gambar 4.7: Kelas JsonConvert



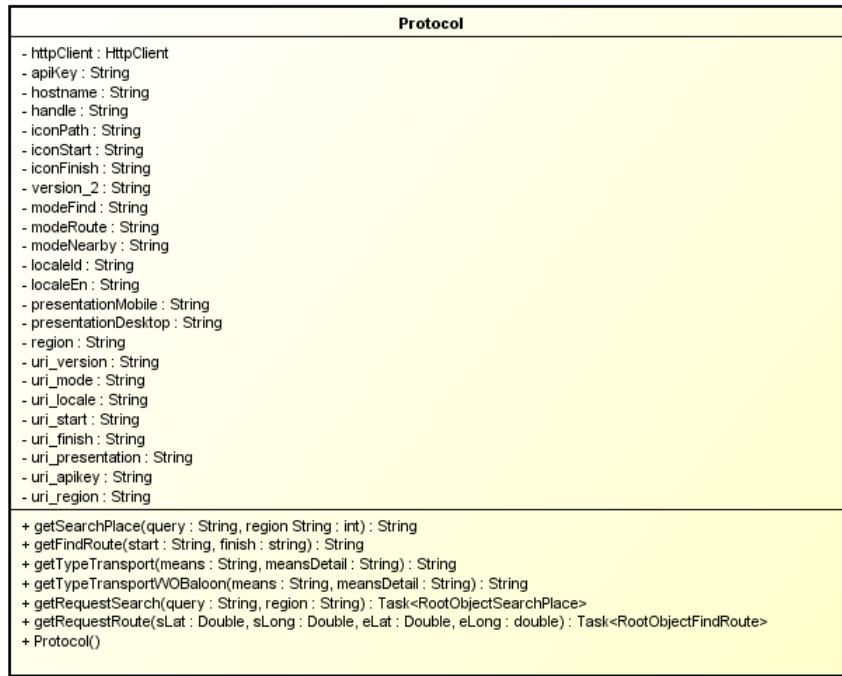
Gambar 4.8: Kelas LocationFinder



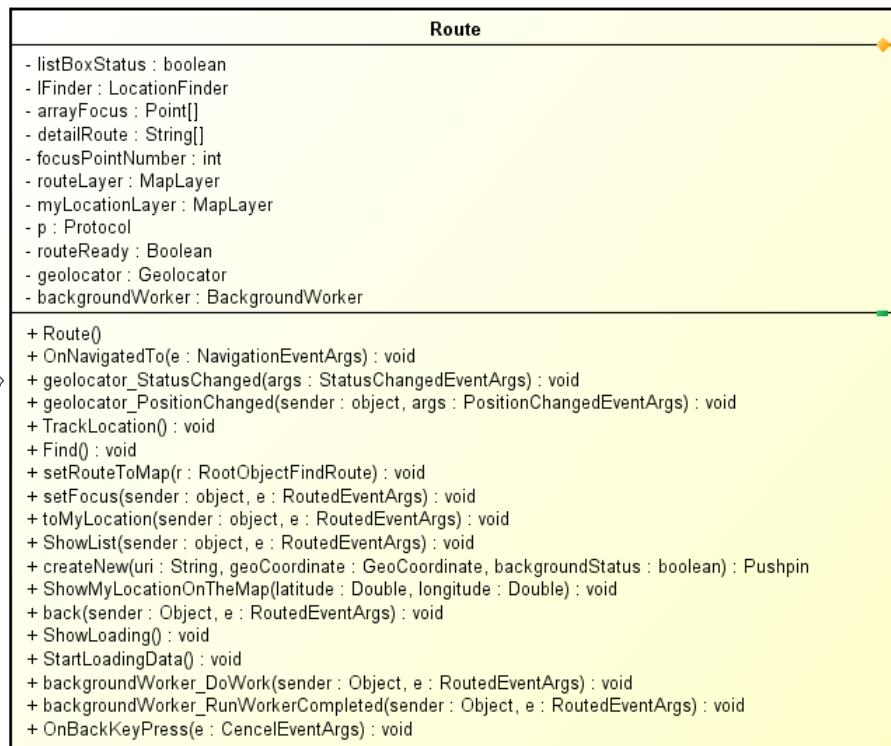
Gambar 4.9: Kelas Map



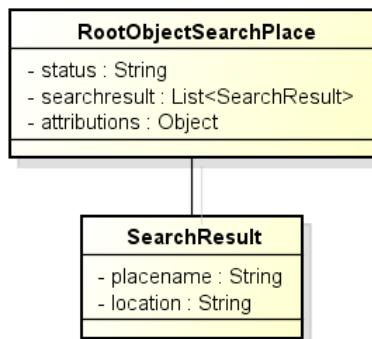
Gambar 4.10: Package WindowsPhone



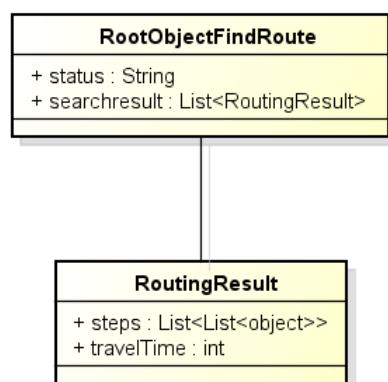
Gambar 4.11: Kelas Protocol



Gambar 4.12: Kelas Route



Gambar 4.13: Kelas SearchPlace



Gambar 4.14: Kelas FindRoute

1 **4.2.1 Kelas *PhoneApplicationPage***

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

4 **4.2.2 Kelas *MainPage***

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

- 8 1. protocol bertipe Protocol untuk mendapatkan URL yang digunakan dalam permintaan
9 ke Kiri API.
- 10 2. lFinder bertipe LocationFinder digunakan untuk menampung semua informasi meng-
11 enai lokasi.
- 12 3. httpClient bertipe HttpClient merupakan objek yang akan mengurus permintaan dan
13 kembalian dari Kiri API.
- 14 4. city bertipe kumpulan String untuk menampung kota yang didukung oleh layanan
15 Kiri.
- 16 5. myCity bertipe String untuk menampung kode kota sesuai Kode Penerbangan IATA.
- 17 6. backgroundWorker bertipe BackgroundWorker untuk mengurus pencarian lokasi di
18 belakang layar.

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

- 20 1. Konstruktor MainPage digunakan untuk memuat komponen yang ada di halaman
21 MainPage dan mendapatkan kota terdekat dari lokasi perangkat.
- 22 2. *method* OnNavigatedTo digunakan untuk mendapatkan lokasi dari peta dan menda-
23 patkan objek LocationFinder. *Method* ini secara otomatis dipanggil jika pengguna
24 kembali ke kelas MainPage. *Method* ini memiliki parameter NavigationEventArgs.
- 25 3. *method* OnNavigatedFrom digunakan untuk mengirimkan *state* objek LocationFinder
26 saat berpindah kelas. *Method* ini secara otomatis dipanggil jika pengguna berpindah
27 dari kelas MainPage ke kelas lain. *Method* ini memiliki parameter NavigationEven-
28 tArgs.
- 29 4. *method* Application_Deactivated digunakan untuk menyimpan *state* objek saat me-
30 ninggalkan aplikasi. *method* ini memiliki parameter object dan DeactivatedEventArgs.
- 31 5. *method* Application_Activated digunakan untuk mengambil *state* objek saat kembali
32 ke aplikasi. *method* ini memiliki parameter object dan ActivatedEventArgs.
- 33 6. *method* startRoute digunakan untuk mendapatkan masukan pengguna. Jika masukan
34 didapat dari peta atau lokasi perangkat berarti sudah dalam kordinat, namun jika
35 masukan didapat dari *query* maka akan dicari lokasi yang terkait sesuai *query* tersebut.
36 Lokasi terkait yang didapatkan akan dikembalikan ke pengguna untuk dipilih. Setelah

- 1 pengguna lokasi dalam bentuk kordinat didapatkan maka kelas akan diarahkan ke kelas
2 Route. *method* ini memiliki parameter objek tombol dan event.
- 3 7. *method* changeMapFrom digunakan untuk berpindah ke halaman mapFrom. *method*
4 ini memiliki parameter objek tombol dan event.
- 5 8. *method* changeMapTo digunakan untuk berpindah ke halaman mapTo. *method* ini
6 memiliki parameter objek tombol dan event.
- 7 9. *method* getHereFrom digunakan untuk mendapatkan kordinat perangkat lalu menyimpan
8 nilainya di kordinat asal di kelas LocationFinder dan menulisakan "Here" pada
9 *TextBox* lokasi asal. *method* ini memiliki parameter objek tombol dan event.
- 10 10. *method* getHereTo digunakan untuk mendapatkan kordinat perangkat lalu menyimpan
11 nilainya di kordinat tujuan di kelas LocationFinder dan menulisakan "Here" pada
12 *TextBox* lokasi tujuan. *method* ini memiliki parameter objek tombol dan event.
- 13 11. *method* getListItem digunakan untuk membuat *listBox* lalu menampilkan ke pengguna.
14 *method* ini memiliki parameter RootObjectSearchPlace dan string yang menunjukan
15 *list* yang ditampilkan untuk lokasi asal dan lokasi tujuan.
- 16 12. *method* ListBoxSelectedPlace digunakan untuk mendapatkan tempat asal yang dipilih
17 pengguna. *method* ini memiliki parameter objek dan *event SelectionChangedEventArgs*.
18 *gs*.
- 19 13. *method* searchCoordinatePlace digunakan untuk mencari kordinat dari tempat pilihan
20 pengguna. *method* ini memiliki parameter *ListBox* dan tempat yang dipilih dalam
21 bentuk *string*.
- 22 14. *method* findRoute digunakan untuk berpindah ke kelas Route jika lokasi asal dan lokasi
23 tujuan sudah ditentukan.
- 24 15. *method* changeCity digunakan untuk mengubah kota tujuan dari pencarian. *method*
25 ini memiliki parameter objek dan *event SelectionChangedEventArgs*.
- 26 16. *method* showCity digunakan untuk mencari kota yang paling dekat dengan lokasi per-
27 angkat.
- 28 17. *method* ShowSplash digunakan untuk menampilkan tampilan awal untuk proses inisi-
29 alisasi aplikasi.
- 30 18. *method* StartLoadingData digunakan untuk memanggil BackgroundWorker. Backgro-
31 undWorker digunakan untuk melakukan aksi di belakang layar.
- 32 19. *method* backgroundWorker_DoWork digunakan untuk melakukan pemanggilan aksi
33 di belakang layar. *method* ini memiliki parameter objek dan *DoWorkEventArgs*.
- 34 20. *method* backgroundWorker_RunWorkerCompleted digunakan untuk melakukan pe-
35 manggilan saat BackgroundWorker selesai melakukan tugasnya. *method* ini memiliki
36 parameter objek dan *RunWorkerCompletedEventArgs*.

¹ **4.2.3 Kelas *City***

² *City* merupakan kelas yang menyimpan kota-kota yang mendukung pencarian rute ken-
³ daraam umum dengan bantuan Kiri. Berikut adalah penjelasan atribut-atribut yang dimiliki
⁴ kelas ini:

- ⁵ 1. centerCity bertipe *array of GeoCoordinate* untuk menyimpan kordinat pusat dari kota.
- ⁶ 2. city bertipe *array of String* untuk menyimpan nama kota.
- ⁷ 3. cityCode bertipe *array of String* untuk menyimpan kode kota dalam huruf kecil sesuai
⁸ aturan IATA Airport Code.

⁹ Berikut adalah penjelasan beberapa *method* yang dimiliki kelas ini:

- ¹⁰ 1. Konstruktor City digunakan untuk untuk inisialisasi nilai atribut.
- ¹¹ 2. *method* getNearby digunakan untuk mencari kota terdekat dengan lokasi perangkat.
¹² *method* ini mengembalikan interger yang merupakan index kota pada atribut city.
¹³ *method* ini memiliki 2 buah parameter yaitu *latitude* dan *longitude* yang bertipe *double*.
- ¹⁴ 3. *method* getIndexFromCityCode digunakan untuk mencari indeks pada *array* sesuai
¹⁵ kode kota. *method* ini memiliki parameter bertipe *string* yang merupakan kode kota.

¹⁶ **4.2.4 Kelas *BackgroundWorker***

¹⁷ *BackgroundWorker* merupakan kelas yang dipakai untuk mengeksekusi operasi pada
¹⁸ *thread* terpisah. Berikut adalah penjelasan *event* yang dimiliki kelas ini dan dipakai untuk
¹⁹ perancangan aplikasi:

- ²⁰ 1. *Event* DoWork
- ²¹ 2. *Event* RunWorkerCompleted

²² Berikut adalah penjelasan beberapa *method* yang dimiliki kelas ini:

- ²³ 1. *method* RunWorkerAsync() digunakan untuk memulai operasi di belakang layar.

²⁴ **4.2.5 Kelas *Geocoordinate***

²⁵ *Geocoordinate* merupakan kelas bawaan dari Windows Phone yang akan dimanfaatkan
²⁶ untuk membaca *latitude* dan *longitude*.

²⁷ **4.2.6 Kelas *Geolocator***

²⁸ *Geolocator* merupakan kelas bawaan Windows Phone untuk mengkases lokasi. Dengan
²⁹ bantuan kelas ini maka dapat mengetahui status lokasi dari perangkat dan menemukan lokasi
³⁰ secara akurat.

³¹ **4.2.7 Kelas *Geoposition***

³² *Geoposition* merupakan kelas yang menampung lokasi sesuak kembalian *Geolocator*.

1 4.2.8 Kelas *LocationFinder*

2 *LocationFinder* merupakan kelas yang akan menampung lokasi dan pencarian lokasi.

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

- 4** 1. coorLat bertipe Double untuk menampung kordinat latitude pengguna.
- 5** 2. coorLong bertipe Double untuk menampung kordinat longitude pengguna.
- 6** 3. coorLatFrom bertipe Double untuk menampung kordinat latitude lokasi asal yang
7 diinginkan pengguna.
- 8** 4. coorLongFrom bertipe Double untuk menampung kordinat longitude lokasi asal yang
9 diinginkan pengguna.
- 10** 5. coorLatTo bertipe Double untuk menampung kordinat latitude lokasi tujuan yang
11 diinginkan pengguna.
- 12** 6. coorLongTo bertipe Double untuk menampung kordinat longitude lokasi tujuan yang
13 diinginkan pengguna.
- 14** 7. addressDevice bertipe String untuk menyimpan alamat perangkat berada.
- 15** 8. addressDeviceFrom bertipe String untuk menyimpan alamat berdasarkan lokasi lokasi
16 asal yang diinginkan pengguna.
- 17** 9. addressDeviceTo bertipe String untuk menyimpan alamat berdasarkan lokasi tujuan
18 yang diinginkan pengguna.
- 19** 10. geolocator bertipe Geolocator untuk menampung pengaturan mendapatkan lokasi.
- 20** 11. myReverseGeocodeQuery bertipe ReverseGeocodeQuery untuk konversi dari alamat
21 ke lokasi dan sebaliknya.
- 22** 12. myCoordinate bertipe GeoCoordinate untuk menampung kordinat geografis.
- 23** 13. accuracy bertipe double untuk menampung akurasi perangkat mendapatkan lokasi.

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

- 25** 1. Konstruktor *LocationFinder* berfungsi mengatur atribut geolocator dan mencari lokasi
26 perangkat.
- 27** 2. *method* *findLocation* berfungsi inisialisasi GPS lalu mendapat kordinat dan menam-
28 pungnya di atribut.
- 29** 3. *method* *updateAccuracy* berfungsi untuk mengubah nilai akurasi dari perangkat.
- 30** 4. *method* *setPositionChanged* berfungsi mengubah atribut coorLat, atribut coorLong,
31 dan akurasi jika terdapat perubahan lokasi. *method* ini memiliki parameter Geolo-
32 cator dan PositionChangedEventArgs yang akan menjalankan *method* jika terdapat
33 perubahan yang diberitahukan melalui kelas Geolocator.

- 1 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>>.
- 9 7. *method* ReverseGeocodeQueryTo_QueryCompleted berfungsi untuk mencari alamat lokasi tujuan. *method* ini memiliki parameter objek dan QueryCompletedEventArgs<IList<MapLocation>>.
- 12 8. *method* ReverseGeocodeQuery_QueryCompleted berfungsi untuk mencari alamat lokasi perangkat. *method* ini memiliki parameter objek dan QueryCompletedEventArgs<IList<MapLocation>>.
- 15 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.
- 19 10. *method* reset berfungsi untuk memasang kembali lokasi asal dan lokasi tujuan.

20 **4.2.9 Kelas Map**

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

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

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

- 29 1. Konstruktor Map untuk inisialisasi dan penambahan *event* mengetuk pada peta.
- 30 2. *method* OnNavigatedTo berfungsi untuk mendapatkan masukan lokasi asal atau lokasi tujuan yang akan ditentukan dari map untuk kemudian ditampung di objek LocationFinder. *method* ini memiliki sebuah parameter NavigationEventArgs.
- 33 3. *method* OnNavigatedFrom digunakan untuk mengirimkan *state* objek LocationFinder saat berpindah kelas. *method* ini memiliki parameter NavigationEventArgs.
- 35 4. *method* ShowMyLocationOnTheMap digunakan untuk memberitahu dan menandai lokasi perangkat.

5. *method* map_Tap berfungsi untuk menandai lokasi yang ditunjuk pengguna lalu merjemahkan lokasi yang ditunjuk pengguna pada peta dan mengirimnya ke kelas LocationFinder.
6. *method* pilihLokasi berfungsi berpindah ke kelas MainPage dan memberitahu kelas MainPage bahwa lokasi sudah dipilih. *method* ini memiliki parameter objek tombol dan *event*.

7 4.2.10 Kelas *HttpClient*

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

- 11 1. *method* GetStringAsync membutuhkan parameter alamat bertipe *string* dan mengembalikan kembalian dari Kiri dalam bentuk *Task<string>*.

13 4.2.11 Kelas *JsonConvert*

14 *JsonConvert* merupakan kelas yang menyediakan *method* untuk mengkonversi berbagai
15 jenis komponen *common language runtime* dan *JSON*. Kelas ini merupakan bagian *namespace*
16 *Newtonsoft*. Berikut adalah penjelasan *method* yang dipakai untuk perancangan aplikasi:

- 17 1. *method* DeserializeObject berfungsi untuk konversi dari bentuk *string* menjadi objek.
18 *Method* ini memiliki satu parameter bertipe *string* lalu mengembalikan *string* tersebut
19 dalam bentuk objek.

20 4.2.12 Kelas *Protocol*

21 *Protocol* merupakan kelas untuk menampung semua alamat dalam pengiriman menggunakan
22 protokol HTTP. Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

- 23 1. *uri_version* bertipe *string* digunakan untuk menyimpan nama dari parameter uri.
- 24 2. *uri_mode* bertipe *string* digunakan untuk menyimpan nama dari parameter mode.
- 25 3. *uri_locale* bertipe *string* digunakan untuk menyimpan nama dari parameter locale.
- 26 4. *uri_start* bertipe *string* digunakan untuk menyimpan nama dari parameter start.
- 27 5. *uri_finish* bertipe *string* digunakan untuk menyimpan nama dari parameter finish.
- 28 6. *uri_presentation* bertipe *string* digunakan untuk menyimpan nama dari parameter presentation.
- 29 7. *uri_apikey* bertipe *string* digunakan untuk menyimpan nama dari parameter apikey.
- 30 8. *uri_region* bertipe *string* digunakan untuk menyimpan nama dari parameter region.
- 31 9. *uri_query* bertipe *string* digunakan untuk menyimpan nama dari parameter query.
- 32 10. *apiKey* bertipe *string* digunakan untuk menyimpan nilai kunci API untuk mengirim
33 permintaan ke Kiri.

- 1 11. hostname bertipe *string* digunakan untuk digunakan untuk menyimpan alamat host
2 dari Kiri.
- 3 12. handle bertipe *string* digunakan untuk menyimpan alamat host ditambah "handle.php".
- 4 13. iconPath bertipe *string* digunakan untuk menyimpan lokasi gambar yang dibutuhkan.
- 5 14. iconStart bertipe *string* digunakan untuk menyimpan lokasi gambar awal perjalanan
6 dari lokasi awal.
- 7 15. iconFinish bertipe *string* digunakan untuk menyimpan lokasi gambar akhir perjalanan
8 ke lokasi tujuan.
- 9 16. version_2 bertipe *string* digunakan untuk menyimpan nilai versi dari API yang di-
10 gunakan (saat pembuatan penelitian ini versi Kiri API yang digunakan adalah versi
11 2).
- 12 17. modeFind bertipe *string* yang digunakan untuk menyimpan nilai "searchplace" yang
13 merupakan mode mencari lokasi terkait pada Kiri API.
- 14 18. modeRoute bertipe *string* yang digunakan untuk menyimpan nilai "findroute" yang
15 merupakan mode mencari rute pada Kiri API.
- 16 19. modeNearby bertipe *string* yang digunakan untuk menyimpan nilai "nearbytransport"
17 " yang merupakan mode mencari lokasi terdekat pada Kiri API.
- 18 20. localeId bertipe *string* yang digunakan untuk menyimpan nilai bahasa jika kembalian
19 yang diinginkan ingin berbahasa Indonesia.
- 20 21. localeEn bertipe *string* yang digunakan untuk menyimpan nilai bahasa jika kembalian
21 yang diinginkan ingin berbahasa Inggris.
- 22 22. presentationMobile bertipe *string* yang digunakan untuk menyimpan nilai penyajian
23 untuk perangkat *mobile*.
- 24 23. presentationDesktop bertipe *string* yang digunakan untuk menyimpan nilai penyajian
25 untuk perangkat *desktop*.

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

- 27 1. *method* getTypeTransport merupakan *method* yang akan mengembalikan alamat dari
28 gambar transportasi dengan bingkai. *method* ini memiliki 2 parameter yaitu means
29 sebagai tipe transportasi dan meansDetail sebagai nama kendaraan.
- 30 2. *method* getTypeTransportWOBaloon merupakan *method* yang akan mengembalikan
31 alamat dari gambar transportasi tanpa bingkai tambahan. *method* ini memiliki 2 par-
32 meter yaitu means sebagai tipe transportasi dan meansDetail sebagai nama kendaraan.
- 33 3. getSearchPlace merupakan *method* yang akan mengembalikan URI pencarian lokasi
34 sesuai paramater. Parameter yang dimaksud adalah kata kunci masukan pengguna.

- 1 4. *method* getFindRoute merupakan *method* yang akan mengembalikan URI pencarian
2 route sesuai parameter. Parameter yang dimaksud adalah kordinat lokasi asal dan
3 kordinat lokasi tujuan yang bertipe string.
- 4 5. *method* getRequestSearch digunakan untuk mendapatkan lokasi terkait sesuai masuk-
5 an pengguna. *method* ini akan mengembalikan Task<RootObjectSearchPlace> karena
6 menggunakan operasi *asynchronous*. *method* ini memiliki parameter kata kunci ma-
7 sukan pengguna dan kota yang masing-masing parameter bertipe string.
- 8 6. *method* getRequestRoute digunakan untuk mendapatkan rute sesuai lokasi asal dan
9 lokasi tujuan. *method* ini akan mengembalikan Task<RootObjectFindRoute> karena
10 menggunakan operasi *asynchronous*. *method* ini memiliki parameter *latitude* lokasi
11 asal, *longitude* lokasi asal, *latitude* lokasi tujuan, dan *longitude* lokasi tujuan yang
12 masing-masing bertipe *double*.

13 **4.2.13 Kelas RootObjectSearchPlace**

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

- 16 1. status bertipe string digunakan untuk menampung hasil kembalian status dari Kiri.
- 17 2. searchresult bertipe *list* dan menampung banyak objek SearchResult.
- 18 3. attributions bertipe objek untuk menampung *attributions*.

19 **4.2.14 Kelas SearchResult**

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

- 22 1. placename bertipe *string* digunakan untuk menampung nama tempat.
- 23 2. location bertipe *string* digunakan untuk menampung nama tempat.

24 **4.2.15 Kelas RootObjectFindRoute**

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

- 27 1. status bertipe *string* digunakan untuk menampung hasil kembalian status dari Kiri.
- 28 2. routingresults bertipe *list* dan menampung banyak objek RoutingResult.

29 **4.2.16 Kelas RoutingResult**

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

- 32 1. steps bertipe *list* digunakan untuk menampung tiap langkah dari lokasi asal ke lokasi
33 tujuan. Tiap langkah yang dimaksud adalah kendaraan yang digunakan, jalur yang
34 dilewati, dan informasi yang dibutuhkan pengguna.

- 1 2. traveltimes bertipe *string* digunakan untuk menampung waktu yang dibutuhkan dari lokasi asal ke lokasi tujuan.
- 2

3 **4.2.17 Kelas *Route***

4 *Route* merupakan kelas untuk pencarian rute dan menampilkannya kepada pengguna.
5 Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

- 6 1. listBoxStatus bertipe *boolean* digunakan untuk menentukan nilai status rute dalam bentuk daftar sedang tertutup(bernilai *false*) atau terbuka(bernilai *true*).
- 7
- 8 2. lFinder bertipe LocationFinder digunakan untuk menampung semua informasi mengenai lokasi.
- 9
- 10 3. arrayFocus bertipe *array of Point* digunakan untuk menampung titik fokus perubahan jenis transportasi yang digunakan pengguna.
- 11
- 12 4. detailRoute bertipe *array of String* digunakan untuk menampung keterangan yang dibutuhkan pengguna dari Kiri API.
- 13
- 14 5. focusPointNumber bertipe *integer* digunakan untuk menentukan *index of* dari atribut arrayFocus dan detailRoute.
- 15
- 16 6. routeLayer bertipe MapLayer digunakan untuk menampung *Polyline* dan *Pushpin*.
- 17
- 18 7. myLocationLayer bertipe MapLayer digunakan untuk menampung titik dari lokasi perangakt berada.
- 19
- 20 8. p bertipe Protocol digunakan untuk menampung permintaan data dengan Kiri API.
- 21
- 22 9. geolocator bertipe Geolocator untuk menangani perubahan lokasi yang terjadi.
- 23
- 24 10. newTimer bertipe DispatcherTimer digunakan untuk membuat perngatur waktu.
- 25
- 26 11. timeOut bertipe boolean digunakan untuk menentukan nilai status apakah waktu untuk satu proses sudah mencapai batas.
- 27
- 28 12. backgroundWorker bertipe BackgroundWorker digunakan untuk menangain proses di belakang layar.
- 29

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

- 31 1. Konstruktor Route digunakan untuk memuat komponen yang ada di halaman Route, inisialisasi atribu, dan pemanggilan backgroundWorker.
- 32
- 33 2. *method* OnNavigatedTo digunakan untuk mendapatkan objek LocationFinder dan memanggil *method* TrackLocation. *method* ini memiliki parameter NavigationEventArgs.
- 34
- 35 3. *method* OnNavigatedFrom digunakan untuk mengirimkan *state* objek LocationFinder saat berpindah kelas. *method* ini memiliki parameter NavigationEventArgs.
- 36
- 37 4. *method* Application_Deactivated digunakan untuk menyimpan *state* objek saat meninggalkan aplikasi. *method* ini memiliki parameter object dan DeactivatedEventArgs.
- 38

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

¹ 4.3 Perancangan Antar Muka

² 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.

⁶ 4.3.1 Antarmuka Kelas *MainPage*



Gambar 4.15: Antarmuka *MainPage*

⁷ Antarmuka Kelas Map pada gambar [4.15](#) merupakan tampilan awal saat aplikasi ⁸ dijalankan. Antarmuka Kelas Map memiliki dua buah masukan, lima buah tombol, dan satu ⁹ menu daftar. Berikut adalah detailnya.

¹⁰ Dua buah masukan yaitu.

- ¹¹ • Masukan lokasi asal

¹² Merupakan masukan lokasi asal mula pengguna ingin melakukan perjalanan.

- ¹³ • Masukan lokasi tujuan

¹⁴ Merupakan masukan lokasi tujuan berhentinya perjalanan.

¹⁵ Lima buah tombol yaitu.

- ¹⁶ • Tombol map untuk lokasi asal

¹⁷ Jika tombol ditekan maka akan berpindah ke kelas map untuk memilih lokasi asal ¹⁸ di peta. Jika di kelas Map pengguna memilih lokasi maka pada masukan lokasi asal ¹⁹ terdapat tulisan "Maps".

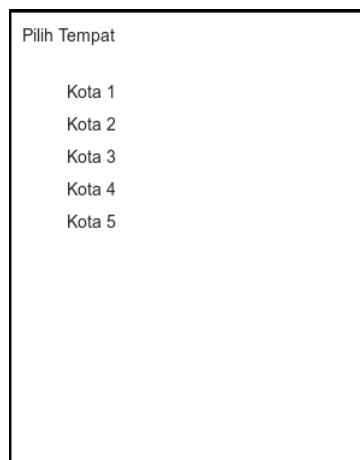
- ²⁰ • Tombol here untuk lokasi asal

²¹ Jika tombol ditekan maka lokasi asal adalah lokasi perangkat saat tombol ditekan dan ²² masukan lokasi asal menjadi "here".

- ²³ • Tombol map untuk lokasi tujuan

²⁴ Jika tombol ditekan maka akan berpindah ke kelas map untuk memilih lokasi tujuan ²⁵ di peta. Jika di kelas Map pengguna memilih lokasi maka pada masukan lokasi tujuan ²⁶ terdapat tulisan "Maps".

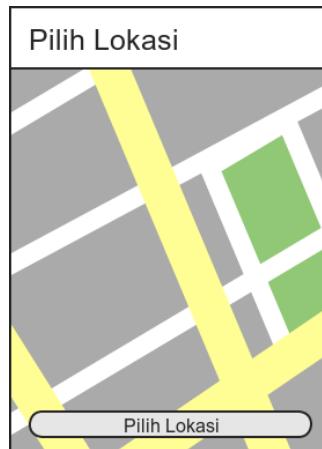
- 1 ● Tombol here untuk lokasi tujuan
- 2 Jika tombol ditekan maka lokasi tujuan adalah lokasi perangkat saat tombol ditekan dan masukan lokasi tujuan menjadi "here".
- 3
- 4 ● Tombol find
- 5 Jika tombol ditekan maka akan menampilkan daftar tempat asal dan tempat tujuan
- 6 lalu mengarahkan ke Kelas Route.
- 7 Satu buah daftar yaitu.
- 8 ● Daftar kota yang tersedia
- 9 Merupakan daftar kota yang tersedia (kota yang rute angkutan umumnya dapat ditemukan dengan aplikasi ini). Disaat aplikasi dijalankan maka daftar akan menunjuk ke kota terdekat tempat perangkat berada.
- 10
- 11



Gambar 4.16: Antarmuka Daftar Tempat

- 12 Antarmuka Daftar Tempat pada gambar 4.16 merupakan daftar yang akan dimunculkan jika pengguna memasukan kata kunci pada pencarian tempat asal dan tempat tujuan.
- 13
- 14 Daftar akan muncul jika didapat kembalian hasil pencarian lebih dari satu.

¹ **4.3.2 Antarmuka Kelas *Map***



Gambar 4.17: Antarmuka *Map*

² Antarmuka Kelas Map pada gambar [4.17](#) merupakan antarmuka untuk menunjuk
³ lokasi pada peta. Terdapat satu buah tombol yang akan dimunculkan jika pengguna sudah
⁴ memilih lokasi. Jika tombol ditekan maka kordinat lokasi akan di simpan dan dikirim pada
⁵ kelas MainPage dan halaman akan diarahkan ke kelas MainPage.

⁶ **4.3.3 Antarmuka Kelas *Route***



Gambar 4.18: Antarmuka *Route*

⁷ Antarmuka Kelas Route pada gambar [4.18](#) merupakan antarmuka untuk melihat rute
⁸ dari lokasi asal ke lokasi tujuan dalam bentuk daftar maupun peta. Terdapat empat buah
⁹ tombol pada antarmuka Kelas Route. Berikut tombol yang terdapat pada Kelas Route.

- ¹⁰ • Tombol prev
¹¹ Jika tombol ditekan maka akan menunjuk titik sebelumnya pada rute peta.
- ¹² • Tombol next
¹³ Jika tombol ditekan maka akan menunjuk titik setelahnya pada rute peta.
- ¹⁴ • Tombol here
¹⁵ Jika tombol ditekan maka akan menunjuk lokasi perangkat berada pada peta.

- 1 ● Tombol Show List
- 2 Jika tombol ditekan maka akan menunjuk atau menyembunyikan daftar rute.



Gambar 4.19: Antarmuka Rute dalam Bentuk Daftar

- 3 Antarmuka rute dalam bentuk daftar pada gambar 4.19 merupakan antarmuka untuk
- 4 melihat rute secara lebih jelas dengan keterangan tahap demi tahap disertai jarak dan waktu
- 5 perjalanan. Antarmuka daftar dapat dilihat atau disembunyikan sesuai keinginan pengguna
- 6 namun saat kelas Rute dibuka antarmuka daftar rute akan disembunyikan.

BAB 5

IMPLEMENTASI DAN PENGUJIAN APLIKASI

³ Pada bab 5 akan dibahas implementasi dan pengujian aplikasi Pencari Rute Kendaraan
⁴ Umum untuk Windows Phone.

5.1 Implementasi

⁶ Pada sub-bab ini akan dijelaskan mengenai lingkungan yang digunakan untuk memba-
⁷ ngun aplikasi Pencari Rute Kendaraan Umum untuk Windows Phone dan hasil dari im-
⁸ plementasi. Lingkungan yang akan dibahas penulis meliputi perangkat lunak untuk im-
⁹ plementasi dan perangkat keras untuk implementasi. Hasil dari implementasi juga penulis
¹⁰ sampaikan dalam bentuk *screenshoot* ketika aplikasi dijalankan.

5.1.1 Perangkat Keras untuk Implementasi

¹¹ Dalam membangun aplikasi ini perangkat keras yang digunakan adalah sebagai berikut:

¹² 1. Komputer

¹³ (a) Processor: intel Core i7-2620M CPU 2,7 GHz

¹⁴ (b) RAM: 4 GB

¹⁵ (c) Hardisk: 640 GB

¹⁶ (d) VGA: Intel HD 3000

¹⁷ 2. Perangkat *Mobile*

¹⁸ (a) Processor: 1,2 GHz

¹⁹ (b) RAM: 1 GB

²⁰ (c) ROM: 8 GB

²¹ (d) Layar: 720 x 1280 pixel, 4,7 inch

²² (e) GPS

²³ (f) Sensor: kompas, *accelerometer*

5.1.2 Perangkat Lunak untuk Implementasi

²⁴ Dalam membangun aplikasi ini perangkat lunak yang digunakan adalah sebagai berikut:

²⁵ 1. Komputer

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

5 2. Perangkat *mobile*

- 6 (a) Sistem Operasi Windows Phone 8.1

7 **5.1.3 Hasil Implementasi**

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

9 1. Kode Program

10 Kode Program pada perangkat lunak ditulis dengan menggunakan bahasa C#. Bahasa
11 C# dipilih berdasarkan analisa pada bab 3 dan kemampuan penulis.

12 2. Hasil kompilasi program

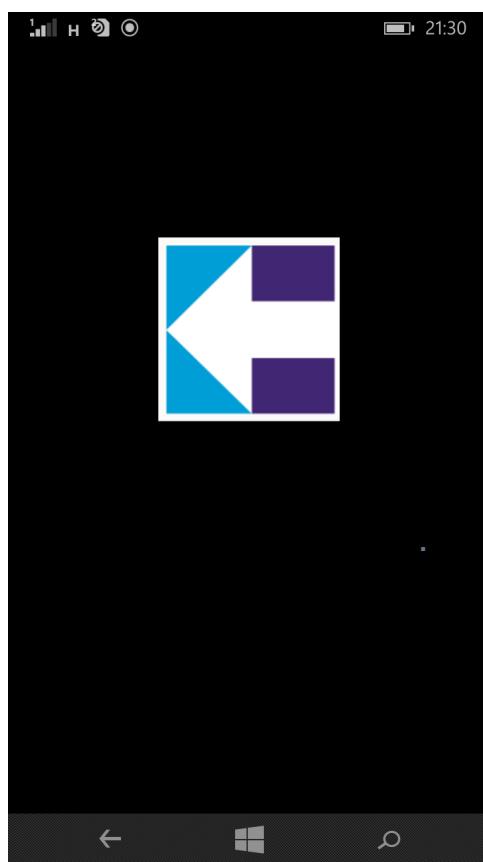
13 Hasil dari kompilasi program berupa *file* Kiri_Debug_AnyCPU.xap. *File* ini dapat
14 dipasang pada perangkat dengan sistem operasi Windows Phone versi 8 atau lebih
15 tinggi.

16 3. Antarmuka Aplikasi

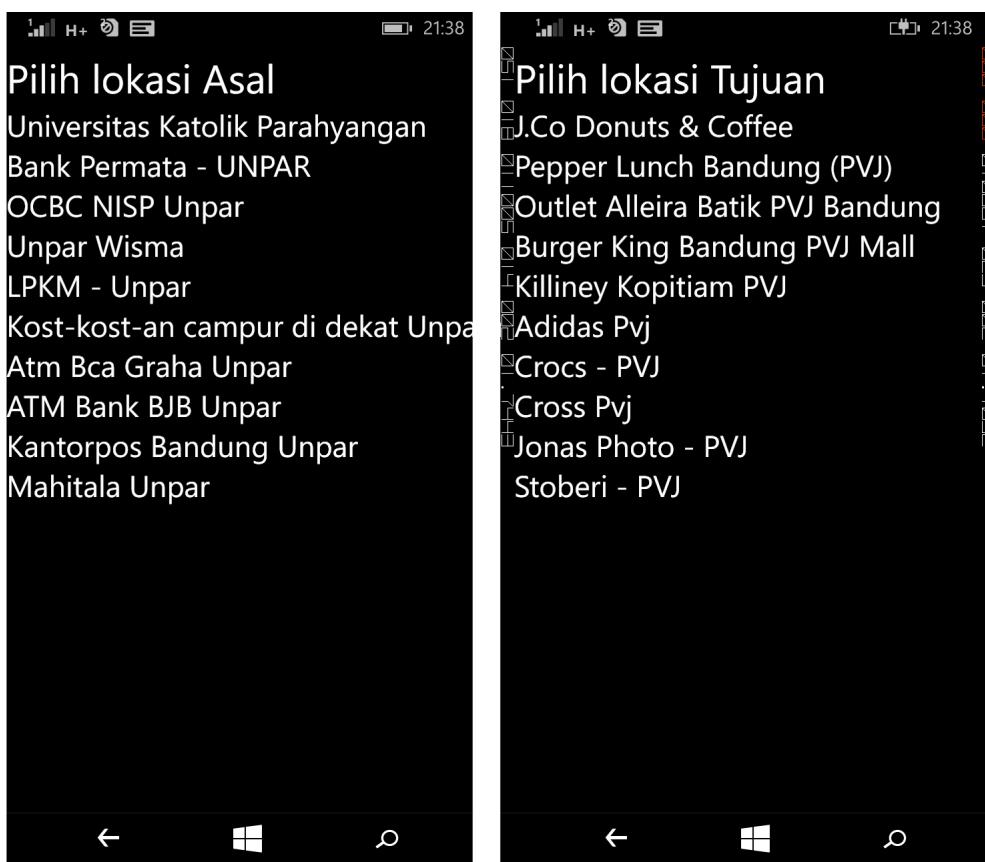
17 Berikut merupakan hasil implementasi antarmuka aplikasi Pencari Rute Kendaraan
18 Umum untuk Windows phone.



Gambar 5.1: Gambar antarmuka kelas MainPage



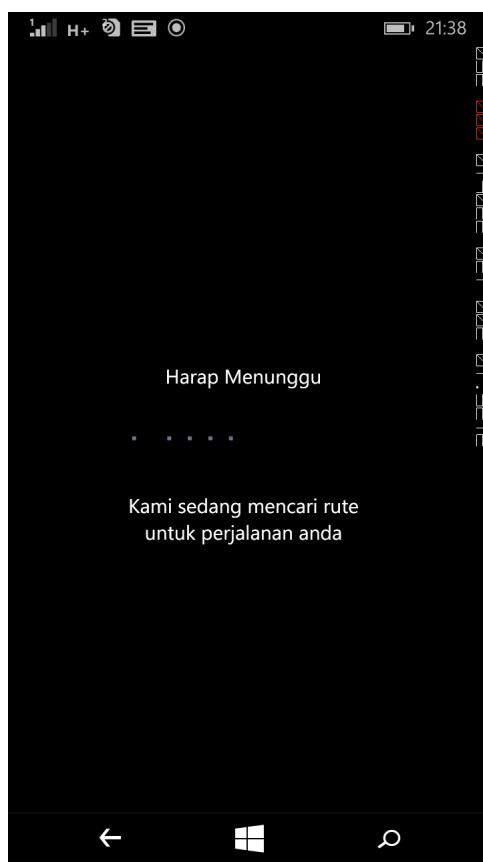
Gambar 5.2: Gambar Antarmuka Splash di Kelas MainPage



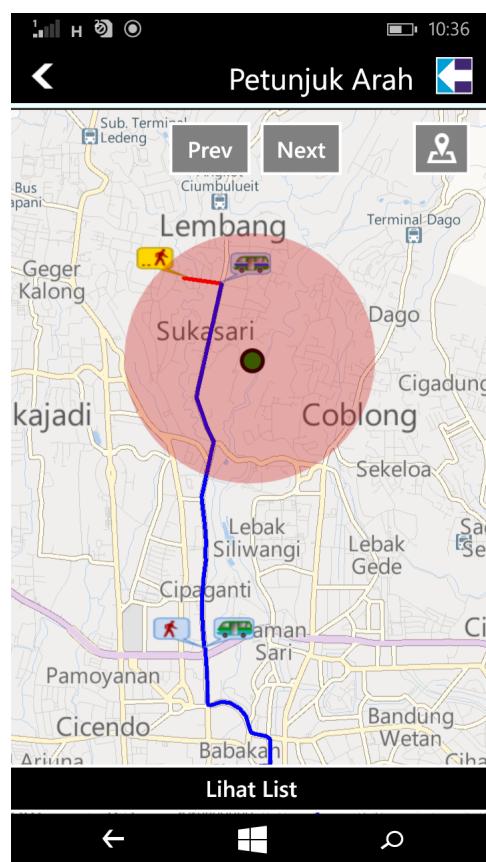
Gambar 5.3: Gambar Antarmuka *List* Asal dan *List* Tujuan di Kelas MainPage



Gambar 5.4: Gambar Antarmuka Kelas Map



Gambar 5.5: Gambar Antarmuka Menunggu di Kelas Route



Gambar 5.6: Gambar Antarmuka Kelas Route



Gambar 5.7: Gambar Antarmuka *List* di Kelas Route

5.2 Pengujian

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

5.2.1 Lingkungan Pengujian

Dalam proses pengujian perangkat lunak penulis menggunakan sistem operasi Windows Phone 8.1 dengan spesifikasi perangkat keras sebagai berikut.

1. Processor : 1.2 Ghz Quad Core
2. RAM : 1 GB
3. Layar : 1280 x 720 pixels, 4,7 inch
4. GPS : A-GPS, GLONASS, Beidou

5.2.2 Pengujian Fungsional

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

1. Carilah rute dari sini ke BIP!
2. Carilah rute dari BIP ke Ciwalk!
3. Carilah rute dari sini ke ITB pintu jalan Ganesa!

Dari tiga perintah berikut penulis mengharapkan hasil sebagai berikut.

Perintah	Aksi Pengguna
1	Menekan tombol "Here" dan menulis kata "bip" pada <i>TextBox</i>
2	Menulis kata "bip" pada <i>TextBox</i> dan menulis kata "Ciwalk" pada <i>TextBox</i>
3	Menekan tombol "Here" dan menunjuk pintu masuk ITB yang di jalan Ganesa

Tabel 5.1: Tabel Hasil yang Diharapkan Penulis

4

Dari tiga perintah berikut didapat hasil sebagai berikut.

Perintah	Aksi Pengguna	Kesesuaian
1	Menekan tombol "Here" dan menulis "BIP" pada <i>TextBox</i>	sesuai
2	Menulis "BIP" pada <i>TextBox</i> dan menulis "CIwalk" pada <i>TextBox</i>	sesuai
3	Menekan tombol "Here" dan menulis "jl Ganesa" pada <i>TextBox</i>	tidak sesuai

Tabel 5.2: Tabel Hasil Pengujian Fungsionalitas Terhadap Pengguna 1

5

Perintah	Aksi Pengguna	Kesesuaian
1	Menekan tombol "Here" dan menulis "BIP" pada <i>TextBox</i>	sesuai
2	Menulis "BIP" pada <i>TextBox</i> dan menulis "CIwalk" pada <i>TextBox</i>	sesuai
3	Menekan tombol "Here" dan menunjuk pada peta	sesuai

Tabel 5.3: Tabel Hasil Pengujian Fungsionalitas Terhadap Pengguna 2

Perintah	Aksi Pengguna	Kesesuaian
1	Menekan tombol "Here" dan menulis "BIP" pada <i>TextBox</i>	sesuai
2	Menulis "BIP" pada <i>TextBox</i> dan menulis "Ciwalk" pada <i>TextBox</i>	sesuai
3	Menekan tombol "Here" dan menulis "ITB Ganeca" pada <i>TextBox</i> lalu menyadari tombol map dan menggunakan	sesuai

Tabel 5.4: Tabel Hasil Pengujian Fungsionalitas Terhadap Pengguna 3

Perintah	Aksi Pengguna	Kesesuaian
1	Menekan tombol "Here" dan menulis "BIP" pada <i>TextBox</i>	sesuai
2	Menulis "BIP" pada <i>TextBox</i> dan menulis "Ciwalk" pada <i>TextBox</i>	sesuai
3	Menekan tombol "Here" dan menulis "Institut Teknologi Bandung" pada <i>TextBox</i>	tidak sesuai

Tabel 5.5: Tabel Hasil Pengujian Fungsionalitas Terhadap Pengguna 4

Perintah	Aksi Pengguna	Kesesuaian
1	Menekan tombol "Here" dan menulis "BIP" pada <i>TextBox</i>	sesuai
2	Menulis "BIP" pada <i>TextBox</i> dan menulis "Ciwalk" pada <i>TextBox</i>	sesuai
3	Menekan tombol "Here" dan menulis "jalan ganesa" pada <i>TextBox</i>	tidak sesuai

Tabel 5.6: Tabel Hasil Pengujian Fungsionalitas Terhadap Pengguna 5

Hasil pengujian menunjukan untuk perintah 1 dan 2 semua pengguna dapat menyelesaikan perintah dengan aksi yang sesuai. Sedangkan untuk perintah ke 3 sebagian pengguna masih salah dalam melakukan aksi yang sesuai. Dari hasil pengujian terhadap pengguna dapat dilihat pengguna lebih memilih untuk mengetikan kata kunci karena hal tersebut adalah yang paling mudah untuk dilakukan. Pengguna cenderung memikirkan suatu lokasi, memikirkan kata kunci lalu menuliskannya pada textbox di aplikasi. Menuliskan pada textbox memang lebih mudah dilakukan tapi ada beberapa pencarian spesifik yang tidak dapat dilakukan dengan mengetikan kata kunci.

Pada pengujian keadaan aplikasi dari keadaan *running* ke *dormant* tidak terjadi. Pada saat aplikasi dalam keadaan *running* dan pengguna menekan tombol "windows" seharusnya keadaan aplikasi berpindah dari keadaan *running* ke *dormant* tetapi keadaan aplikasi malah *terminate*. Hal tersebut dikarenakan aplikasi yang penulis kembangkan masih dalam mode debug.

5.2.3 Pengujian Eksperimental

Pengujian eksperimental dilakukan untuk mengetahui keberhasilan aplikasi yang dibangun penulis. Berikut pengujian eksperimental yang dilakukan penulis.

• Tempat : Jalan Kejaksaan menuju Unpar dan kembali ke jalan Kejaksaan

• Waktu : Pukul 9 pada tanggal 7 April 2014

• Pengalaman :

Pada pengujian penulis memasukan alamat di dalam rumah untuk menuju Unpar. Saat penulis memasukan kata Unpar muncul daftar tempat sesuai kata kunci "Unpar", lalu penulis memilih "Universitas Katolik Parahyangan". Saat rute ditampilkan ada ketidaktepatan lokasi yang ditunjukan untuk lokasi asal(rumah) sedangkan untuk lokasi tujuan(Unpar) ditunjukan dengan tepat. Ketidaktepatan tersebut kira-kira se ratus meter dan lokasi ditandai lingkaran merah dengan ukuran besar. Hal tersebut dikarenakan tempat tertutup yang membuat GPS tidak menunjukan lokasi yang akurat. Tapi saat pengujian penulis berusaha mengikuti titik naik angkutan umum.

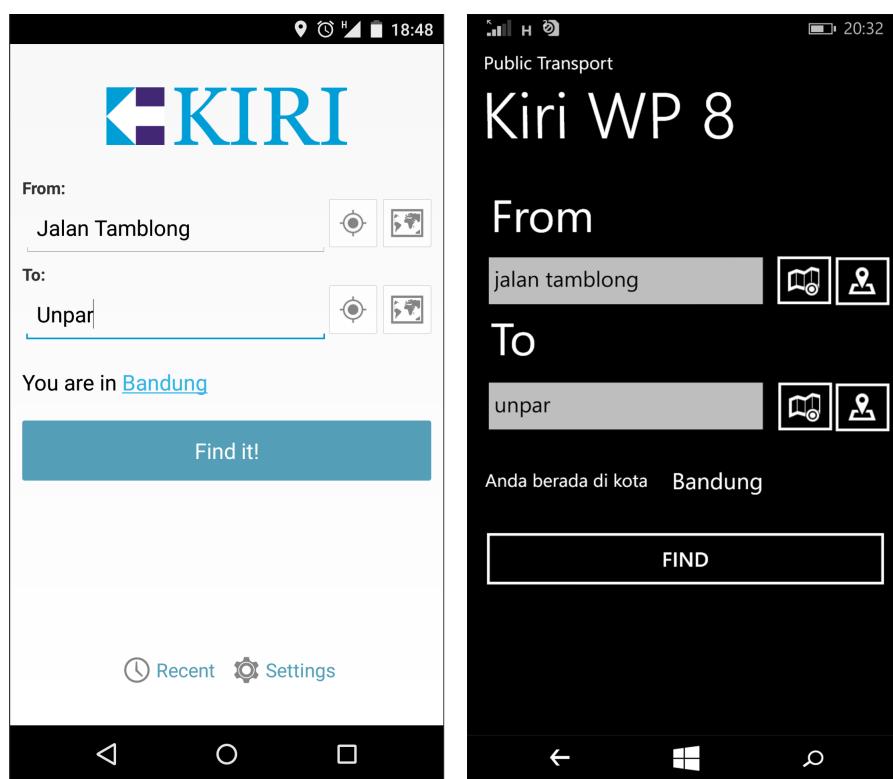
15

Penulis mulai dengan mengikuti petunjuk berjalan kaki menuju Jalan Tamblong. Saat keluar rumah lingkaran merah disekitar lokasi perangkat mulai mengecil yang menandakan akurasi GPS semakin baik. Angkutan kota yang penulis naiki pertama kali adalah angkutan kota kalapa. Sambil berada di angkot penulis mengamati rute yang ditunjukan aplikasi sudah sesuai dengan rute angkutan kota dan penunjuk lokasi menunjukan pergerakan dengan akurat. Angkutan kota yang penulis naiki berhenti di stasiun lalu penulis turun dan menuju Jalan Kebon Jukut. Di jalan kebon jukut penulis naik angkutan kota Ciumbuleuit - St. Hall. Selama perjalanan petunjuk rute dan *polyline* sudah tepat sesuai rute angkutan kota. Angkutan kota ke dua langsung mengantarkan penulis menuju Jalan Ciumbuleuit untuk selanjutnya penulis berjalan menuju Unpar. Ketika sampai di Unpar, kordinat lokasi pada peta juga menunjukan titik di Unpar dan memakan waktu 65 menit.

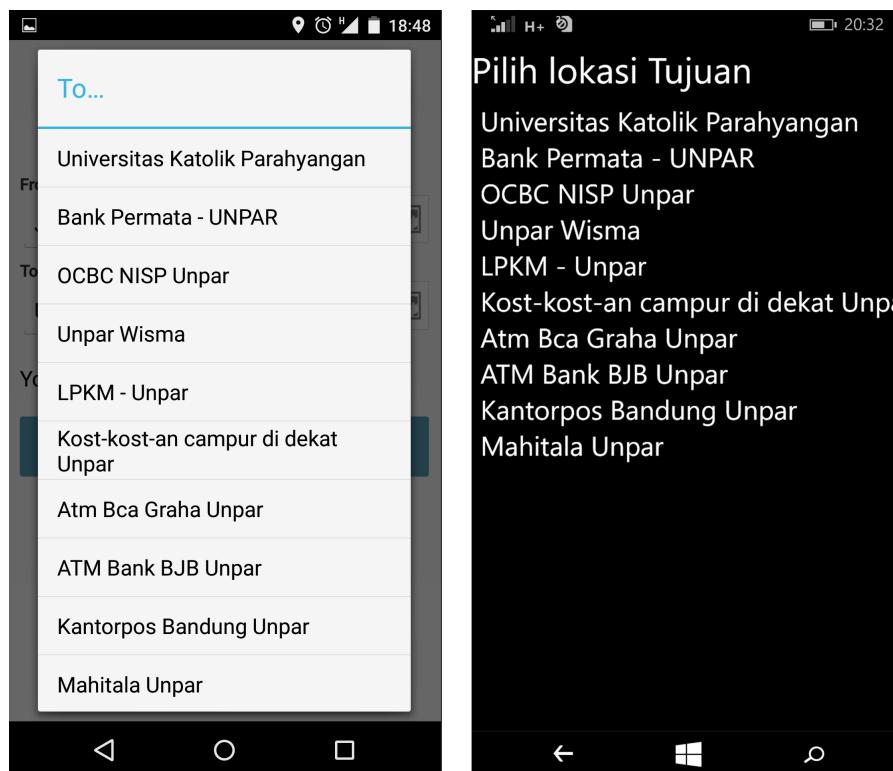
28

Untuk perjalanan pulang penulis memulai dengan mencari rute dari Unpar menuju ke rumah di jalan Kejaksaan dengan mencari di peta. Perjalanan penulis mulai dengan naik angkot Ciumbuleuit - St. Hall (lurus). Angkot Ciumbuleuit - St. Hall (lurus) penulis naiki dari jalan Ciumbuleuit sampai jalan Cihampelas. Penulis turun di jalan Cihampelas dan naik angkot Kalapa - Ledeng di jalan Cihampelas. Namun di jalan Wastukencana rute pada peta menunjukan rute menyusuri jalan Wastukencana lalu ke jalan Aceh, namun angkot malah berbelok ke jalan Riau lalu ke jalan Purnawarman lalu ke jalan Aceh. Rute yang benar yaitu rute angkot dan bukan pada peta. Dari jalan Aceh angkot menuju jalan Sumatera lalu ke jalan Tamblong. Perjalanan penulis pun berakhir setelah sampai di rumah.

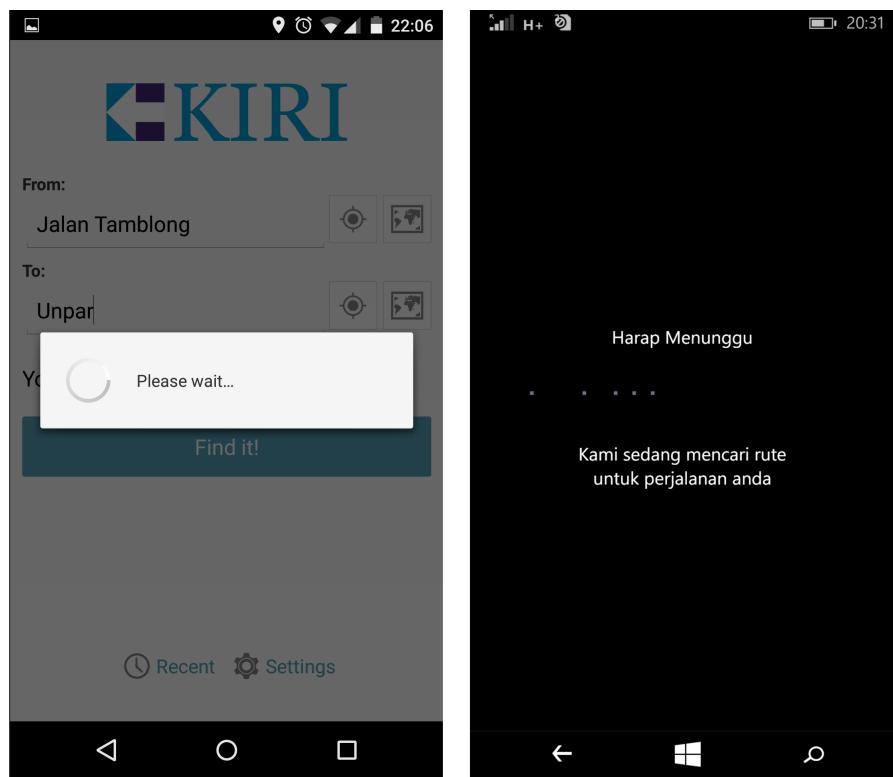
Selain pengujian langsung penulis juga coba mambandingkan hasilnya pencarian di android dan di Windows Phone. Hasil perbandingan dapat dilihat pada gambar dibawah ini.



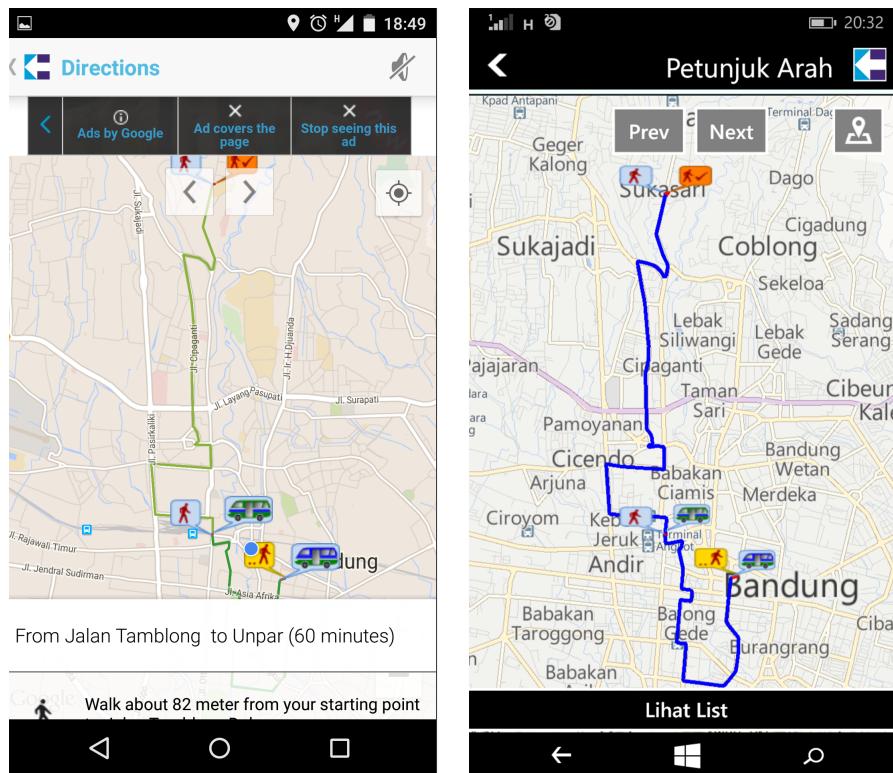
Gambar 5.8: Gambar Perbandingan Antarmuka Home di Android(kiri) dan Windows Phone(kanan)



Gambar 5.9: Gambar Perbandingan Antarmuka Home di Android(kiri) dan Windows Phone(kanan)



Gambar 5.10: Gambar Perbandingan Antarmuka Menunggu hasil pencarian rute di Android(kiri) dan Windows Phone(kanan)



Gambar 5.11: Gambar Perbandingan Antarmuka Penentuan Rute di Android(kiri) dan Windows Phone(kanan)

1 5.2.4 Perbandingan Aplikasi Pencari Rute di Android dengan Aplikasi 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.

Perbedaan	Android	Windows Phone
Bahasa	Java dan antarmuka menggunakan XML	C# dan antarmuka menggunakan XAML
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
Navigasi antar halaman	Pengiriman parameter berupa nilai "key" dan "value"	Pengiriman state objek
Pemanfaatan sumber data	JSON in Java	Json.NET

Tabel 5.7: Tabel Perbedaan

¹

BAB 6

²

KESIMPULAN DAN SARAN

³ Bab ini berisi kesimpulan dari pembangunan aplikasi beserta saran untuk pengembangan
⁴ selanjutnya.

⁵ 6.1 Kesimpulan

⁶ Dari hasil penelitian penulis terhadap perancangan aplikasi pencari rute kendaraan
⁷ umum didapat kesimpulan sebagai berikut.

- ⁸ 1. Pengguna dapat memasukan lokasi berdasarkan peta, lokasi perangkat, dan dengan
⁹ kata kunci.
- ¹⁰ 2. Penggunaan Kiri API sudah dapat digunakan untuk mencari rute angkutan umum.
- ¹¹ 3. Akses internet mempengaruhi performansi mendapatkan rute.

¹² 6.2 Saran

¹³ Berdasarkan hasil kesimpulan yang telah dipaparkan, penulis memberi saran sebagai
¹⁴ berikut.

- ¹⁵ 1. Memanfaatkan tombol "enter" untuk memanggil *method startRoute*. Masukan dari
¹⁶ pengguna karena setelah pengguna menentukan tujuan tombol "Find" terhalang *ke-*
¹⁷ *yboard* maka pengguna harus menekan tombol "back" terlebih dahulu untuk menekan
¹⁸ tombol "Find".
- ¹⁹ 2. Memanfaatkan pencarian lokasi pada peta agar pengguna dapat memilih lokasi dengan
²⁰ lebih mudah.
- ²¹ 3. Menambahkan *gesture* dan animasi yang akan meningkatkan interaksi pengguna ter-
²² hadap aplikasi.
- ²³ 4. Mengubah beberapa tampilan yang menjadi ciri khas Windows Phone seperti meng-
²⁴ gunakan *application bar*.

1

DAFTAR REFERENSI

- 2 [1] "Windows phone silverlight development." <http://msdn.microsoft.com/library/windows/apps/ff402535.aspx>, 2014. Accessed: 2014-8-17.
- 3
- 4 [2] P. Manning, *Pro Windows Phone App Development*. Apress, 2013.
- 5 [3] A. Whitechapel and S. McKenna, *Windows Phone 8 Development Internals*. Microsoft,
- 6 2013.
- 7 [4] T. Bray, "The JavaScript Object Notation (JSON) Data Interchange Format." RFC 7159
- 8 (Proposed Standard), Mar. 2014.
- 9 [5] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifier (URI): Gene-
- 10 ric Syntax." RFC 3986 (INTERNET STANDARD), Jan. 2005. Updated by RFCs 6874,
- 11 7320.
- 12 [6] "Newtonsoft json." <http://www.newtonsoft.com/json/help/html/SerializingJSON.htm>, 2015. Accessed: 2015-2-15.
- 13
- 14 [7] "Kiri api v2 documentation." https://bitbucket.org/projectkiri/kiri_api/wiki/KIRI%20API%20v2%20Documentation, 2014. Accessed: 2014-8-17.
- 15

1

LAMPIRAN A

2

KODE PROGRAM KELAS MAINPAGE

Listing A.1: MainPage.xaml.cs

```

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

```

```

1      {
2          toBox.Text = lFinder.addressTo;
3      }
4      else if (lFinder.coorLatTo != 0.0 && lFinder.coorLongTo != 0.0)
5      {
6          if (lFinder.addressTo.Equals("Here"))
7          {
8              toBox.Text = "Here";
9          }
10         else
11         {
12             toBox.Text = "Maps";
13         }
14     }
15     string forMaps = "";
16     if (NavigationContext.QueryString.TryGetValue("for", out forMaps)) ;
17     if (forMaps!=null)
18     {
19         if (forMaps.Equals("from"))
20         {
21             fromBox.Text = "Maps";
22             //lFinder.addressFrom = "Maps";
23         }
24         else
25         {
26             toBox.Text = "Maps";
27             //lFinder.addressTo = "Maps";
28         }
29     }
30 }
31 }
32
33 protected override void OnNavigatedFrom(NavigationEventArgs e)
34 {
35     base.OnNavigatedFrom(e);
36     PhoneApplicationService.Current.State["location"] = lFinder;
37 }
38
39 private void Application_Deactivated(object sender, DeactivatedEventArgs e){
40     PhoneApplicationService.Current.State["location"] = lFinder;
41 }
42
43 private void Application_Activated(object sender, ActivatedEventArgs e) {
44     //Console.WriteLine("Test");
45     if (PhoneApplicationService.Current.State.ContainsKey("location"))
46     {
47         this.lFinder = (LocationFinder)PhoneApplicationService.Current.State["location"];
48         this.findRoute();
49     }
50 }
51
52 private async void startRoute(object sender, RoutedEventArgs e)
53 {
54
55     String queryFrom = fromBox.Text;
56     String queryTo = toBox.Text;
57     RootObjectSearchPlace from = null; //Untuk Asal
58     RootObjectSearchPlace to = null; //Untuk Tujuan
59
60     //Check form
61     if (!queryFrom.Equals("") && !queryTo.Equals(""))
62     {
63         //Validate From
64         Boolean routeStatus = true;
65         progressFindPlace.IsIndeterminate = true;
66         //Get place from query
67         if ((!queryFrom.Equals("Here") || !queryFrom.Equals("Maps")) && (lFinder.
68             coorLatFrom == 0.0 && lFinder.coorLongFrom == 0.0)) //Check get location from
69             GPS
70         {
71             from = await protocol.getRequestSearch(queryFrom, myCity); //Mengubah String
72             menjadi objek
73             if (from.searchresult.Count() == 0)
74             {
75                 MessageBox.Show("Pencarian untuk kata " + fromBox.Text + " tidak ditemukan");
76             }
77             routeStatus = false;
78         }
79         if ((!queryTo.Equals("Here") || !queryTo.Equals("Maps")) && (lFinder.coorLatTo ==
80             0.0 && lFinder.coorLongTo == 0.0)) //Check get location from GPS
81         {
82             to = await protocol.getRequestSearch(queryTo, myCity); //Mengubah String
83             menjadi objek
84             if (to.searchresult.Count() == 0)
85             {
86                 MessageBox.Show("Pencarian untuk kata " + toBox.Text + " tidak ditemukan");
87             }
88             routeStatus = false;
89         }
90     }
91     //Check Query
92     if (routeStatus == true)
93     {
94         if ((lFinder.coorLatFrom == 0.0 && lFinder.coorLongFrom == 0.0))
95         {
96             getListItem(from, "from"); //Show Listbox for location From
97         }
98         if ((lFinder.coorLatTo == 0.0 && lFinder.coorLongTo == 0.0))
99         {
100            getListItem(to, "to"); //Show Listbox for location To
101        }
102    this.findRoute();
103 }
```

```

1         }
2         progressFindPlace.IsIndeterminate = false;
3     }
4     else {
5         MessageBox.Show("Harap melengkapi tempat asal dan tempat tujuan");
6     }
7 }
8
9 private void changeMapFrom(object sender, RoutedEventArgs e)
10 {
11     NavigationService.Navigate(new Uri("/Map.xaml?fromMapFor=from", UriKind.Relative));
12 }
13 private void changeMapTo(object sender, RoutedEventArgs e)
14 {
15     NavigationService.Navigate(new Uri("/Map.xaml?fromMapFor=to", UriKind.Relative));
16 }
17
18 private void getHereFrom(object sender, RoutedEventArgs e)
19 {
20     this.lFinder.setCoordinateHere("from");
21     fromBox.Text = "Here";
22     lFinder.addressFrom = "Here";
23 }
24
25 private void getHereTo(object sender, RoutedEventArgs e)
26 {
27     this.lFinder.setCoordinateHere("to");
28     toBox.Text = "Here";
29     lFinder.addressTo = "Here";
30 }
31
32 private void getListItem(RootObjectSearchPlace request, String forRequest)
33 {
34     Dispatcher.BeginInvoke(new Action(delegate
35     {
36         if (request.status.ToString().Equals("ok")) //check status
37         {
38             if (request.searchresult.Count() == 1)
39             {
40                 if (forRequest.Equals("from"))
41                 {
42                     String locFrom = request.searchresult[0].location.ToString();
43                     string[] coordinate = locFrom.Split(',');
44                     lFinder.coorLatFrom = Double.Parse(coordinate[0]);
45                     lFinder.coorLongFrom = Double.Parse(coordinate[1]);
46                 }
47                 else
48                 {
49                     String locTo = request.searchresult[0].location.ToString();
50                     string[] coordinate = locTo.Split(',');
51                     lFinder.coorLatTo = Double.Parse(coordinate[0]);
52                     lFinder.coorLongTo = Double.Parse(coordinate[1]);
53                 }
54                 this.findRoute();
55             }
56         }
57     });
58     if (forRequest.Equals("from"))
59     {
60         for (int c = 0; c < request.searchresult.Count; c++)
61         {
62             listPlaceFrom.Items.Add(request.searchresult[c].placename);
63         }
64         panelFrom.Visibility = Visibility.Visible;
65         listPlaceFrom.DataContext = request.searchresult;
66         listPlaceFrom.SelectionChanged += ListBoxSelectedPlace;
67     }
68     else
69     {
70         panelTo.Visibility = Visibility.Visible;
71         for (int c = 0; c < request.searchresult.Count; c++)
72         {
73             listPlaceTo.Items.Add(request.searchresult[c].placename);
74         }
75         listPlaceTo.DataContext = request.searchresult;
76         listPlaceTo.SelectionChanged += ListBoxSelectedPlace;
77     }
78 }
79 else
80 {
81     MessageBox.Show("Error!");
82 }
83 }));
84 }
85
86 private void ListBoxSelectedPlace(object sender, SelectionChangedEventArgs e)
87 {
88     if (panelFrom.Visibility.Equals(Visibility.Visible))
89     { //Check visibility
90         if (null != (sender as ListBox).SelectedItem)
91         {
92             if ((sender as ListBox).SelectedIndex >= 0)
93             {
94                 String locFrom = searchCoordinatePlace((List<Searchresult>)listPlaceFrom.
95                     DataContext, listPlaceFrom.SelectedItem.ToString()); //((sender as
96                     //ListBox).SelectedItem as Searchresult).placename;
97                 string[] coordinate = locFrom.Split(',');
98                 lFinder.coorLatFrom = Double.Parse(coordinate[0]);
99                 lFinder.coorLongFrom = Double.Parse(coordinate[1]);
100                lFinder.addressFrom = listPlaceFrom.SelectedItem.ToString();
101            }
102        }
103    }
104 }

```

```

1     panelFrom.Children.Clear();
2     panelFrom.Visibility = Visibility.Collapsed;
3 }
4 else {
5     if (null != (sender as ListBox).SelectedItem)
6     {
7         if ((sender as ListBox).SelectedIndex >= 0)
8         {
9             String locTo = searchCoordinatePlace((List<Searchresult>)listPlaceTo.
10                DataContext, listPlaceTo.SelectedItem.ToString()); //((sender as
11                ListBox).SelectedItem as Searchresult).placename;
12                string[] coordinate = locTo.Split(',');
13                lFinder.coorLatTo = Double.Parse(coordinate[0]);
14                lFinder.coorLongTo = Double.Parse(coordinate[1]);
15                lFinder.addressTo = listPlaceTo.SelectedItem.ToString();
16            }
17        }
18        panelTo.Children.Clear();
19        panelTo.Visibility = Visibility.Collapsed;
20    }
21    this.findRoute();
22}
23
24 public string searchCoordinatePlace(List<Searchresult> listResult, string place) { //Pasti
25     ketemu
26     String coordinate = "0.0";
27     for (int c = 0; c < listResult.Count; c++)
28     {
29         if (place.Equals(listResult[c].placename))
30         {
31             coordinate = listResult[c].location;
32             c = listResult.Count;
33         }
34     }
35     return coordinate;
36 }
37
38 public void findRoute()
39 {
40     if ((lFinder.coorLatFrom != 0.0 && lFinder.coorLongFrom != 0.0) && (lFinder.coorLatTo
41         != 0.0 && lFinder.coorLongTo != 0.0))
42     {
43         NavigationService.Navigate(new Uri("/Route.xaml", UriKind.Relative));
44     }
45 }
46
47 private void changeCity(object sender, SelectionChangedEventArgs e)
48 {
49     if (!cmbCurrFrom.SelectedItem.Equals(null))
50     {
51         switch (cmbCurrFrom.SelectedItem.ToString())
52     {
53         case "Bandung":
54             this.myCity="bdo";
55             break;
56         case "Jakarta":
57             this.myCity="cgk";
58             break;
59         case "Malang":
60             this.myCity="mlg";
61             break;
62         case "Surabaya":
63             this.myCity="sub";
64             break;
65         default:
66             this.myCity="bdo"; //Finder Auto
67             break;
68     }
69 }
70 }
71
72 public void showCity() {
73     int indexCity = -1;
74     while (indexCity == -1)
75     {
76         indexCity = c.getNearby(lFinder.coorLat, lFinder.coorLong);
77     }
78     this.myCity = c.cityCode[indexCity];
79 }
80
81 private void ShowSplash()
82 {
83     this.popup.IsOpen = true;
84     StartLoadingData();
85 }
86
87 private void StartLoadingData()
88 {
89     backgroundWorker = new BackgroundWorker();
90     backgroundWorker.DoWork += new DoWorkEventHandler(backgroundWorker_DoWork);
91     backgroundWorker.RunWorkerCompleted += new RunWorkerCompletedEventHandler(
92         backgroundWorker_RunWorkerCompleted);
93     backgroundWorker.RunWorkerAsync();
94 }
95
96 private void backgroundWorker_DoWork(object sender, DoWorkEventArgs e)
97 {
98     showCity();
99 }
100
101 private void backgroundWorker_RunWorkerCompleted(object sender,
102     RunWorkerCompletedEventArgs e)
103 {

```

```

1   this.Dispatcher.BeginInvoke(() =>
2   {
3       this.cmbCurrFrom.SelectedIndex = c.getIndexFromCityCode(myCity);
4       this.popup.IsOpen = false;
5   });
6 }
7 }
8 }

```

Listing A.2: MainPage.xaml

```

9  <!--Logo kiri from kiri.travel
10 Icon from modernuiicons.com
11 -->
12 <phone:PhoneApplicationPage
13 x:Class="Kiri.MainPage"
14 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
15 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
16 xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
17 xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
18 xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
19 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
20 xmlns:toolkit="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.
21 Toolkit"
22 mc:Ignorable="d"
23 FontFamily="{StaticResource PhoneFontFamilyNormal}"
24 FontSize="{StaticResource PhoneFontSizeNormal}"
25 Foreground="{StaticResource PhoneForegroundBrush}"
26 SupportedOrientations="Portrait" Orientation="Portrait"
27 shell:SystemTray.Visibile="True">
28
29 <!--LayoutRoot is the root grid where all page content is placed-->
30 <Grid x:Name="LayoutRoot" Background="Transparent" Margin="0,0,0,0">
31     <Grid.RowDefinitions>
32         <RowDefinition Height="158"/>
33         <RowDefinition Height="3"/>
34         <RowDefinition Height="*"/>
35     </Grid.RowDefinitions>
36
37     <!-- LOCALIZATION NOTE:
38         To localize the displayed strings copy their values to appropriately named
39         keys in the app's_neutral_language_resource_file_(AppResources.resx)_then
40         replace_the_hard-coded_text_value_between_the_attributes'_quotation_marks
41         with the binding clause whose path points to that string name.
42
43         For example:
44
45             Text="{Binding Path=LocalizedResources.ApplicationTitle, Source={StaticResource
46             LocalizedStrings}}"
47
48             This binding points to the template's_string_resource_named_"ApplicationTitle".
49
50             Adding_supported_languages_in_the_Project_Properties_tab_will_create_a
51             new_resx_file_per_language_that_can_carry_the_translated_values_of_your
52             UI_strings_.The_binding_in_these_examples_will_cause_the_value_of_the
53             attributes_to_be_drawn_from_the_.resx_file_that_matches_the
54             CurrentUICulture_of_the_app_at_run_time.
55
56
57             <!-- TitlePanel contains the name of the application and page title -->
58             <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="6,10,6,0" RenderTransformOrigin
59             ="0.5,0.5" Height="108" VerticalAlignment="Top">
60                 <StackPanel.RenderTransform>
61                     <CompositeTransform SkewX="0.781" TranslateX="0.736" />
62                 </StackPanel.RenderTransform>
63                 <TextBlock Text="Public_Transport" Style="{StaticResource PhoneTextNormalStyle}">
64                     Margin="12,0"/>
65                     <TextBlock Text="Kiri_WP_8" Margin="9,-7,0,0" Style="{StaticResource
66             PhoneTextTitle1Style}" Height="111"/>
67                 </StackPanel>
68                 <Popup x:Name="popup" Margin="6,0,-6,10" Grid.RowSpan="3">
69                     <Grid Background="Black" Height="756" Width="484">
70                         <Image Source="/icons/kiri_logo.png" Margin="144,175,159,358" /
71                             RenderTransformOrigin="1.509,0.516" />
72                         <ProgressBar Height="34" Width="481" IsIndeterminate="True" Margin
73                             ="-7,489,10,233" />
74                     </Grid>
75                 </Popup>
76             <!--ContentPanel place additional content here-->
77             <Grid x:Name="ContentPanel" Margin="0,0,0,0" Grid.Row="1" Grid.RowSpan="2">
78                 <TextBlock Text="From" Style="{StaticResource PhoneTextNormalStyle}" Margin
79                     ="26,0,278,484" FontSize="50" />
80                 <TextBlock Text="To" Style="{StaticResource PhoneTextNormalStyle}" Margin
81                     ="26,137,361,274" FontSize="50" />
82                 <TextBox x:Name="fromBox" HorizontalAlignment="Left" Margin="12,70,0,0" TextWrapping="Wrap"
83                     VerticalAlignment="Top" FontSize="24" Width="325" Height="76" />
84                 <TextBlock x:Name="toBox" HorizontalAlignment="Left" Margin="12,209,0,0" TextWrapping="Wrap"
85                     VerticalAlignment="Top" FontSize="24" Width="325" Height="76" />
86                 <Button Content="FIND" HorizontalAlignment="Left" Margin="10,376,0,0" VerticalAlignment="Top"
87                     Click="startRoute" Height="81" Width="460" />
88                 <Button HorizontalAlignment="Left" Margin="330,209,0,0" VerticalAlignment="Top" Click
89                     ="changeMapTo" Height="77" Width="83" />
90                 <Image Source="icons/map.png" Margin="-14,-12,0,0" Width="60" Height="60" /></Image>
91                 <Button HorizontalAlignment="Left" Margin="329,69,0,0" VerticalAlignment="Top" Click
92                     ="changeMapFrom" Height="77" Width="83" />
93                 <Image Source="icons/map.png" Margin="-14,-12,0,0" Width="60" Height="60" /></Image>
94                 <Button HorizontalAlignment="Left" Margin="393,210,0,0" VerticalAlignment="Top" Click
95                     ="getHereTo" Height="77" Width="83" />
96                 <Image Source="icons/marker.png" Margin="-14,-12,0,0" Width="60" Height="60" /></Image>

```

```

1  <!--</Button>
2  <--<Button HorizontalAlignment="Left" Margin="393,70,0,0" VerticalAlignment="Top" Click
3   = "getHereFrom" Height="77" Width="83">
4  <--<Image Source="icons/marker.png" Margin="-14,-12,0,0" Width="60" Height="60"></
5   Image>
6  </Button>
7  <--<TextBlock HorizontalAlignment="Left" Margin="20,315,0,0" TextWrapping="Wrap" Text=
8   "Anda berada di kota" VerticalAlignment="Top"/>
9  <--<toolkit : ListPicker BorderThickness="0" ScrollViewer.VerticalScrollBarVisibility="Auto
10  " Margin="217,298,74,-45" Name="cmbCurrFrom" SelectionChanged="changeCity" Visibility="Visible
11  ">
12 </toolkit : ListPicker>
13 <--<ProgressBar x:Name="progressFindPlace" Background="Black" HorizontalAlignment="Left"
14  " Height="19" Margin="0,-24,0,0" VerticalAlignment="Top" Width="456" RenderTransformOrigin
15  ="0.493,1.056" />
16
17 <--<StackPanel x:Name="panelFrom" HorizontalAlignment="Stretch" Canvas.ZIndex="10"
18  " Margin="0,-150,0,5" VerticalAlignment="Stretch" Visibility="Collapsed" Background="Black">
19 <--<TextBlock x:Name="textFrom" TextWrapping="Wrap" Text="Pilih lokasi Asal" FontSize
20  ="40"/>
21 <--<ListBox x:Name="listPlaceFrom" Padding="10" Height="700" FontSize="30"></ListBox>
22 </StackPanel>
23 <--<StackPanel x:Name="panelTo" HorizontalAlignment="Stretch" Canvas.ZIndex="5"
24  " Margin="0,-150,0,5" VerticalAlignment="Stretch" Visibility="Collapsed" Background="Black" Grid.
25  ColumnSpan="2">
26 <--<TextBlock x:Name="textTo" TextWrapping="Wrap" Text="Pilih lokasi Tujuan" FontSize
27  ="40"/>
28 <--<ListBox x:Name="listPlaceTo" Padding="10" Height="700" FontSize="30"></ListBox>
29 </StackPanel>
30 </Grid>
31
32
33
34 <!--Uncomment to see an alignment grid to help ensure your controls are
35  aligned on common boundaries. The image has a top margin of -32px to
36  account for the System Tray. Set this to 0 (or remove the margin altogether)
37  if the System Tray is hidden.
38
39 <!--Before shipping remove this XAML and the image itself.-->
40 <--<Image Source="/Assets/AlignmentGrid.png" VerticalAlignment="Top" Height="800" Width
41  ="480" Margin="0,-32,0,0" Grid.Row="0" Grid.RowSpan="2" IsHitTestVisible="False" />-->
42 </Grid>
43
44 </phone:PhoneApplicationPage>

```

LAMPIRAN B

KODE PROGRAM KELAS MAP

Listing B.1: Map.xaml.cs

```

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

```

```

1     myLocationOverlay.PositionOrigin = new Point(0, 0);
2     myLocationOverlay.GeoCoordinate = s;
3
4     if (fromMapFor.Equals("from"))
5     {
6         this.lFinder.GetCurrentCoordinate(s.Latitude, s.Longitude, "from");
7     }
8     else {
9         this.lFinder.GetCurrentCoordinate(s.Latitude, s.Longitude, "to");
10    }
11    MapLayer myLocationLayer = new MapLayer();
12    myLocationLayer.Add(myLocationOverlay);
13
14    MyMap.Layers.Add(myLocationLayer);
15}
16
17 private void pilihLokasi(object sender, RoutedEventArgs e)
18 {
19     if (fromMapFor.Equals("from"))
20     {
21         NavigationService.Navigate(new Uri("/ MainPage.xaml?for=from", UriKind.Relative));
22     }
23     else
24     {
25         NavigationService.Navigate(new Uri("/ MainPage.xaml?for=to", UriKind.Relative));
26     }
27 }
28 }
29 }
```

Listing B.2: Map.xaml

```

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

LAMPIRAN C

KODE PROGRAM KELAS ROUTE

Listing C.1: Route.xaml.cs

```

3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Net;
7  using System.Windows;
8  using System.Windows.Controls;
9  using System.Windows.Navigation;
10 using Microsoft.Phone.Controls;
11 using Microsoft.Phone.Shell;
12
13 using System.Device.Location; // Provides the GeoCoordinate class.
14 using Windows.Devices.Geolocation; //Provides the Geocoordinate class.
15 using System.Windows.Media;
16 using System.Windows.Shapes;
17 using Microsoft.Phone.Maps.Controls;
18 using Microsoft.Phone.Maps.Toolkit;
19
20 using System.Threading.Tasks;
21 using System.Net.Http;
22 using System.IO;
23 using System.Runtime.Serialization.Json;
24 using System.Text;
25 using Newtonsoft.Json;
26 using System.Windows.Media.Imaging;
27 using System.ComponentModel;
28 using System.Windows.Controls.Primitives;
29 using System.Threading;
30 using System.IO.IsolatedStorage;
31 using System.Windows.Threading;
32 using System.Diagnostics;
33
34 namespace Kiri
35 {
36     public partial class Route : PhoneApplicationPage
37     {
38         private Boolean listBoxStatus; //true == show, false == hide
39         private LocationFinder lFinder;
40         private Point[] arrayFocus;
41         private String[] detailRoute;
42         private int focusPointNumber;
43         private MapLayer routeLayer;
44         private MapLayer myLocationLayer;
45         private Protocol p;
46         private Boolean routeReady;
47
48         private Geolocator geolocator = null;
49         private BackgroundWorker backgroundWorker;
50         private DispatcherTimer newTimer;
51         private Boolean timeOut = false;
52
53         public Route()
54         {
55             InitializeComponent();
56             this.lFinder = null;
57             this.arrayFocus = null;
58             this.detailRoute = null;
59             this.focusPointNumber = 0;
60             this.routeLayer = new MapLayer();
61             this.myLocationLayer = new MapLayer();
62             this.p = new Protocol();
63             this.routeReady = false;
64             this.newTimer = new DispatcherTimer();
65             newTimer.Interval = new TimeSpan(0, 0, 60);
66             newTimer.Tick += OnTimerTick;
67             newTimer.Start();
68
69             ShowLoading();
70         }
71
72         protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
73         {
74             base.OnNavigatedTo(e);
75             if (PhoneApplicationService.Current.State.ContainsKey("location"))
76             {
77                 this.lFinder = (LocationFinder)PhoneApplicationService.Current.State["location"];
78             }
79             this.TrackLocation();
}

```

```

1     //detect location
2     if (IsolatedStorageSettings.ApplicationSettings.Contains("LocationConsent"))
3     {
4         return;
5     }
6 }
7
8 protected override void OnNavigatedFrom(NavigationEventArgs e)
9 {
10    base.OnNavigatedFrom(e);
11    this.lFinder.reset();
12    PhoneApplicationService.Current.State["location"] = lFinder;
13 }
14
15 private void Application_Deactivated(object sender, DeactivatedEventArgs e)
16 {
17    PhoneApplicationService.Current.State["location"] = lFinder;
18 }
19
20 private void TrackLocation()
21 {
22    geolocator = lFinder.geolocator;
23
24    geolocator.StatusChanged += geolocator_StatusChanged;
25    geolocator.PositionChanged += geolocator_PositionChanged;
26 }
27
28 void geolocator_PositionChanged(Geolocator sender, PositionChangedEventArgs args)
29 {
30    Dispatcher.BeginInvoke(() =>
31    {
32        drawMyLocationOnTheMap(args.Position.Coordinate.Latitude, args.Position.Coordinate
33            .Longitude);
34        lFinder.setPositionChanged(sender, args);
35    });
36 }
37
38 public async void Find()
39 {
40    Boolean status = true;
41    HttpClient httpClient = new HttpClient();
42
43    RootObjectFindRoute r = await p.getRequestRoute(lFinder.coorLatFrom, lFinder.
44        coorLongFrom, lFinder.coorLatTo, lFinder.coorLongTo);
45    if (r.status.Equals("ok"))
46    {
47        if (!r.routingresults[0].steps[0][3].Equals("Maaf, kami tidak dapat menemukan rute
48            _transportasi_publik_untuk_perjalanan Anda."))
49        {
50            this.setRouteToMap(r);
51        }
52        else
53        {
54            MessageBox.Show("Maaf, kami tidak dapat menemukan rute transportasi publik_
55                untuk perjalanan Anda.");
56            status = false;
57        }
58    }
59    else
60    {
61        MessageBox.Show("Error");
62        status = false;
63    }
64
65    if (status == false)
66    {
67        this.lFinder.reset();
68        NavigationService.Navigate(new Uri("/ MainPage.xaml", UriKind.Relative));
69    }
70
71 private void setRouteToMap(RootObjectFindRoute r) {
72    MapPolyline routeRoad = new MapPolyline();
73    for (int i = 0; i < 1; i++) // Ambil Routing result yg pertama
74    {
75        this.arrayFocus = new Point[r.routingresults[i].steps.Count + 1];
76        this.detailRoute = new String[r.routingresults[i].steps.Count + 1];
77        for (int j = 0; j < r.routingresults[i].steps.Count; j++)
78        {
79            routeRoad = new MapPolyline();
80            if (r.routingresults[i].steps[j][0].ToString().Equals("walk"))
81            {
82                //MessageBox.Show(r.routingresults[i].steps[j][0].ToString() + " + walk");
83                routeRoad.StrokeColor = Color.FromArgb(255, 255, 0, 0);
84            }
85            else
86            {
87                //MessageBox.Show(r.routingresults[i].steps[j][0].ToString() + " " +
88                angkot");
89                routeRoad.StrokeColor = Color.FromArgb(255, 0, 0, 255);
90            }
91            routeRoad.StrokeThickness = 4;
92            GeoCoordinate geoCoo = new GeoCoordinate();
93            //Trim character
94            String temp = r.routingresults[i].steps[j][2].ToString();
95            char[] charsToTrim = { '[', ']', ' ', ',' };
96            String result = temp.Trim(charsToTrim);
97            result = result.Replace("\\", "");
98            //Split string coordinate to array
99            string[] coordinate = result.Split(',');
100           for (int c = 0; c < coordinate.Length; c = c + 2)
101           {
102               geoCoo = new GeoCoordinate();
103               geoCoo.Latitude = double.Parse(coordinate[c]);
104           }
105       }
106   }
107 }
```

```

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

```

```

1     detailPanel.Visibility = Visibility.Collapsed;
2     if (listBoxStatus == false)
3     {
4         routePanel.Visibility = Visibility.Visible;
5         buttonList.Content = "Tutup_List";
6         this.listBoxStatus = true;
7     }
8     else
9     {
10        routePanel.Visibility = Visibility.Collapsed;
11        buttonList.Content = "Lihat_List";
12        this.listBoxStatus = false;
13    }
14}
15
16 public Pushpin createNew(string uri, GeoCoordinate geoCoordinate, String transport) {
17     Pushpin p = new Pushpin();
18     Uri imgUri = new Uri(uri, UriKind.RelativeOrAbsolute);
19     BitmapImage imgSourceR = new BitmapImage(imgUri);
20     ImageBrush imgBrush = new ImageBrush() { ImageSource = imgSourceR };
21     p.Background = new SolidColorBrush(Color.FromArgb(0, 0, 0, 0));
22     p.Content = new Rectangle()
23     {
24         Fill = imgBrush,
25         Height = 30,
26         Width = 50,
27     };
28     if (transport.Equals("start"))
29     {
30         p.Margin = new Thickness(-51, -35, 0, 0);
31     }
32     else if(transport.Equals("finish")) {
33         p.Margin = new Thickness(-6, -34, 0, 0);
34     } else if(transport.Equals("walk")){
35         p.Margin = new Thickness(-55, -35, 0, 0);
36     } else{
37         p.Margin = new Thickness(-5, -35, 0, 0);
38     }
39     p.GeoCoordinate = geoCoordinate;
40     return p;
41 }
42
43 private void drawMyLocationOnTheMap(Double latitude, Double longitude)
44 {
45     this.route.Layers.Remove(myLocationLayer);
46     GeoCoordinate myGeoCoordinate = new GeoCoordinate(latitude, longitude);
47     double myAccuracy = lFinder.accuracy;
48     double metersPerPixels = (Math.Cos(myGeoCoordinate.Latitude * Math.PI / 180) * 2 *
49     Math.PI * 6378137) / (256 * Math.Pow(2, this.route.ZoomLevel));
50     double radius = myAccuracy / metersPerPixels;
51
52     Ellipse ellipse = new Ellipse();
53     ellipse.Width = radius * 2;
54     ellipse.Height = radius * 2;
55     ellipse.Margin = new Thickness(-radius+10, -radius+10, 0, 0);
56     ellipse.Fill = new SolidColorBrush(Color.FromArgb(75, 200, 0, 0));
57
58     Ellipse myCircle = new Ellipse();
59     myCircle.Stroke = new SolidColorBrush(Colors.Black);
60     myCircle.StrokeThickness = 4;
61     myCircle.Fill = new SolidColorBrush(Colors.Green);
62     myCircle.Height = 25;
63     myCircle.Width = 25;
64     myCircle.Opacity = 50;
65
66     MapOverlay myLocationOverlayPosition = new MapOverlay();
67     myLocationOverlayPosition.Content = myCircle;
68     myLocationOverlayPosition.PositionOrigin = new Point(0, 0);
69     myLocationOverlayPosition.GeoCoordinate = myGeoCoordinate;
70
71     MapOverlay myLocationOverlayAccuracy = new MapOverlay();
72     myLocationOverlayAccuracy.Content = ellipse;
73     myLocationOverlayAccuracy.PositionOrigin = new Point(0, 0);
74     myLocationOverlayAccuracy.GeoCoordinate = myGeoCoordinate;
75
76     myLocationLayer = new MapLayer();
77     myLocationLayer.Add(myLocationOverlayPosition);
78     myLocationLayer.Add(myLocationOverlayAccuracy);
79     this.route.Layers.Add(myLocationLayer);
80 }
81
82 private void back(object sender, RoutedEventArgs e)
83 {
84     MessageBoxResult result = MessageBox.Show("Anda yakin mengakhiri Navigasi?", "Kembali ke Menu Utama", MessageBoxButton.OKCancel);
85
86     if (result == MessageBoxResult.OK)
87     {
88         this.lFinder.reset();
89         NavigationService.Navigate(new Uri("/ MainPage.xaml", UriKind.Relative));
90     }
91 }
92
93
94 private void ShowLoading()
95 {
96     this.popup.IsOpen = true;
97     StartLoadingData();
98 }
99
100 private void StartLoadingData()
101 {
102     backgroundWorker = new BackgroundWorker();
103     backgroundWorker.RunWorkerAsync();

```

```

1     backgroundWorker.DoWork += new DoWorkEventHandler(backgroundWorker_DoWork);
2     backgroundWorker.RunWorkerCompleted += new RunWorkerCompletedEventHandler(
3         backgroundWorker_RunWorkerCompleted);
4 }
5
6     private void backgroundWorker_DoWork(object sender, DoWorkEventArgs e)
7     {
8         this.Dispatcher.BeginInvoke(() =>
9         {
10            this.Find();
11        });
12
13        while (this.routeReady == false)
14        {
15            if (this.timeOut == true) {
16                //this.timeOutReached();
17                this.routeReady = true;
18            }
19        }
20    }
21
22    void OnTimerTick(Object sender, EventArgs args)
23    {
24        newTimer.Stop();
25        this.timeOut = true;
26    }
27
28    private void backgroundWorker_RunWorkerCompleted(object sender,
29             RunWorkerCompletedEventArgs e)
30    {
31        this.Dispatcher.BeginInvoke(() =>
32        {
33            this.popup.IsOpen = false;
34            if (this.timeOut == true)
35            {
36                MessageBox.Show("Maaf, saat ini kami tidak dapat memproses rute anda!");
37                this.lFinder.reset();
38                NavigationService.Navigate(new Uri("/ MainPage.xaml", UriKind.Relative));
39            }
40        });
41    }
42
43    protected override void OnBackKeyPress(CancelEventArgs e)
44    {
45        if (MessageBox.Show("Anda yakin mengakhiri Navigasi?", "Kembali ke Menu Utama?", MessageBoxButton.OKCancel) == MessageBoxResult.OK)
46        {
47            this.lFinder.reset();
48            NavigationService.Navigate(new Uri("/ MainPage.xaml", UriKind.Relative));
49        }
50        else
51        {
52            e.Cancel = true;
53        }
54    }
55
56
57    //JSON data to c# using JSON.NET
58    //package from zhttp://www.nuget.org/packages/newtonsoft.json
59    //dok http://www.newtonsoft.com/json/help/html/SerializingJSON.htm
60
61 }
}

```

Listing C.2: MainPage.xaml

```

62 <phone:PhoneApplicationPage
63     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
64     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
65     xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
66     xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
67     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
68     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
69
70     xmlns:Controls="clr-namespace:Microsoft.Phone.Maps.Controls;assembly=Microsoft.Phone.Maps"
71     xmlns:toolkit="clr-namespace:Microsoft.Phone.Maps.Toolkit;assembly=Microsoft.Phone.Controls.Toolkit"
72     xmlns:Maps="clr-namespace:Microsoft.Phone.Controls.Maps;assembly=Microsoft.Phone.Controls.Maps"
73
74     x:Class="Kiri.Route"
75     FontFamily="{StaticResource PhoneFontFamilyNormal}"
76     FontSize="{StaticResource PhoneFontSizeNormal}"
77     Foreground="{StaticResource PhoneForegroundBrush}"
78     SupportedOrientations="Portrait" Orientation="Portrait"
79     mc:Ignorable="d"
80     shell:SystemTray.Visibile="True">
81
82     <!--xmlns:maps="clr-namespace:Microsoft.Phone.Maps.Controls;assembly=Microsoft.Phone.Maps"-->
83     <!--LayoutRoot is the root grid where all page content is placed-->
84     <Grid x:Name="LayoutRoot" Background="Transparent">
85         <Grid.RowDefinitions>
86             <RowDefinition Height="Auto"/>
87             <RowDefinition Height="*"/>
88         </Grid.RowDefinitions>
89         <!--TitlePanel contains the name of the application and page title-->
90         <Grid HorizontalAlignment="Left" Height="57" Margin="10,10,0,0" Grid.RowSpan="2"
91             VerticalAlignment="Top" Width="460">
92             <Border BorderBrush="Azure" BorderThickness="0,0,0,5" HorizontalAlignment="Left"
93                 Height="57" VerticalAlignment="Top" Width="511" Margin="-21,-2,-30,0"/>
94             <TextBlock HorizontalAlignment="Left" Margin="207,4,0,0" FontSize="30" TextWrapping="Wrap"
95                 Text="Petunjuk Arah" VerticalAlignment="Top" Width="198"/>
96             <Button BorderThickness="0" HorizontalAlignment="Left" FontSize="30" Margin=
97                 "-39,-12,0,-2" VerticalAlignment="Top" Height="71" Width="137" Click="back">
98                 <Image Source="icons/back.png" Height="60" Width="60" Margin="-14,-12,0,0"/></Image>
99
100

```

```

1      </Button>
2      <Image Source="/icons/kiri_logo.png" Margin="405,4,0,16" RenderTransformOrigin="
3          1.509,0.516"/>
4  </Grid>
5  <Controls:Map x:Name="route" ZoomLevel="14" Margin="-9,67,-8,0" Grid.RowSpan="2"/>
6  <Button Name="prev" Content="Prev" Background="Gray" HorizontalAlignment="Left" Margin=
7      146,67,0,0 VerticalAlignment="Top" Height="77" Width="105" Click="setFocus" Grid.
8      RowSpan="2"/>
9  <Button Name="next" Content="Next" Background="Gray" HorizontalAlignment="Left" Margin=
10     237,67,0,0 VerticalAlignment="Top" Height="77" Width="105" Click="setFocus" Grid.
11    RowSpan="2"/>
12 <Button HorizontalAlignment="Left" Background="Gray" Margin="389,67,0,0"
13     VerticalAlignment="Top" Height="77" Width="81" Click="toMyLocation" Grid.RowSpan="2">
14     <Image Source="icons/marker.png" Height="60" Width="60" Margin="-14,-12,0,0"></Image>
15 </Button>
16 <StackPanel x:Name="detailPanel" HorizontalAlignment="Left" Background="Black" Height="111
17     " Margin="0,610,0,0" Grid.Row="1" VerticalAlignment="Top" Width="480" Visibility="
18     Collapsed">
19     <TextBlock x:Name="detailShows" HorizontalAlignment="Left" Margin="0,0,0,0" Grid.Row="
20         1" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Height="100" Width="478"
21         Padding="4"/>
22 </StackPanel>
23 <Button x:Name="buttonList" Content="Lihat List" Background="Black" HorizontalAlignment="
24     Left" Margin="-22,709,-21,-13" VerticalAlignment="Top" Height="72" Width="523" Click="
25     ShowList" Grid.Row="1"/>
26 <StackPanel x:Name="routePanel" Background="White" Height="auto" Margin="0,250,0,47" Grid.
27     Row="1" Visibility="Collapsed">
28     <TextBlock x:Name="addFindRoute" Text="Rute Perjalanan" Foreground="Black" FontSize="
29         25" TextWrapping="Wrap" Margin="3"/>
30     <ListBox x:Name="listRoute" Foreground="White" Height="426" Background="White"
31         FontSize="20" Padding="5" Margin="3">
32         <ListBox.ItemTemplate>
33             <DataTemplate>
34                 <Grid>
35                     <Image Grid.Column="0" Margin="3" Width="60"/>
36                     <Border BorderBrush="Black" BorderThickness="0,0,0,2">
37                         <TextBlock Grid.Column="1" Margin="3" Padding="5" Foreground="
38                             Black" TextWrapping="Wrap" Text="{Binding}"/>
39                     </Border>
40                 </Grid>
41             </DataTemplate>
42         </ListBox.ItemTemplate>
43     </ListBox>
44 </StackPanel>
45 <Popup x:Name="popup" Margin="-9,0,-8,10" Grid.RowSpan="2">
46     <Grid Background="Black" Height="774" Width="498">
47         <ProgressBar Height="34" Width="481" IsIndeterminate="True" Margin=" -7,370,10,352 "
48             />
49         <TextBlock HorizontalAlignment="Left" Margin="166,320,0,0" TextWrapping="Wrap"
50             Text="Harap Menunggu" VerticalAlignment="Top"/>
51         <TextBlock HorizontalAlignment="Left" Margin="83,449,0,0" TextWrapping="Wrap"
52             TextAlignment="Center" Text="Kami sedang mencari rute
53             -----untuk perjalanan anda" VerticalAlignment="Top" Width="322"/>
54     </Grid>
55 </Popup>
56 </Grid>
57
58 </phone:PhoneApplicationPage>
```

1

LAMPIRAN D

2

KODE PROGRAM KELAS CITY

Listing D.1: city.cs

```

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

```


1

LAMPIRAN E

2

KODE PROGRAM KELAS LOCATIONFINDER

Listing E.1: LocationFinder.cs

```

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

```

```

1     Deployment.Current.Dispatcher.BeginInvoke(() =>
2     {
3         coorLat = args.Position.Coordinate.Latitude;
4         coorLong = args.Position.Coordinate.Longitude;
5         updateAccuracy();
6     });
7 }
8
9     public void GetCurrentCoordinate(Double latitude, Double longitude, String paramFor)
10    {
11        try
12        {
13            MyCoordinate = new GeoCoordinate(latitude, longitude);
14
15            if (MyReverseGeocodeQuery == null || !MyReverseGeocodeQuery.IsBusy)
16            {
17                MyReverseGeocodeQuery = new ReverseGeocodeQuery();
18                MyReverseGeocodeQuery.GeoCoordinate = new GeoCoordinate(MyCoordinate.Latitude
19                                , MyCoordinate.Longitude);
20                if (paramFor.Equals("from"))
21                {
22                    MyReverseGeocodeQuery.QueryCompleted +=
23                        ReverseGeocodeQueryFrom_QueryCompleted;
24                    this.coorLatFrom = latitude;
25                    this.coorLongFrom = longitude;
26                }
27                else_if (paramFor.Equals("to"))
28                {
29                    MyReverseGeocodeQuery.QueryCompleted +=
30                        ReverseGeocodeQueryTo_QueryCompleted;
31                    this.coorLatTo = latitude;
32                    this.coorLongTo = longitude;
33                }
34                else
35                {
36                    MyReverseGeocodeQuery.QueryCompleted +=
37                        ReverseGeocodeQuery_QueryCompleted;
38                    this.coorLat = latitude;
39                    this.coorLong = longitude;
40                }
41                MyReverseGeocodeQuery.QueryAsync();
42            }
43        catch (Exception ex)
44        {
45        }
46    }
47 }
48
49
50     private void ReverseGeocodeQueryFrom_QueryCompleted(object sender, QueryCompletedEventArgs<
51             <IList<MapLocation>> e)
52    {
53        if (e.Error == null)
54        {
55            if (e.Result.Count > 0)
56            {
57                MapAddress add = e.Result[0].Information.Address;
58                addressFrom = add.Street;
59            }
60        }
61    }
62
63     private void ReverseGeocodeQueryTo_QueryCompleted(object sender, QueryCompletedEventArgs<
64             <IList<MapLocation>> e)
65    {
66        if (e.Error == null)
67        {
68            if (e.Result.Count > 0)
69            {
70                MapAddress add = e.Result[0].Information.Address;
71                addressTo = add.Street;
72            }
73        }
74    }
75
76     private void ReverseGeocodeQuery_QueryCompleted(object sender, QueryCompletedEventArgs<
77             <IList<MapLocation>> e)
78    {
79        if (e.Error == null)
80        {
81            if (e.Result.Count > 0)
82            {
83                MapAddress add = e.Result[0].Information.Address;
84                addressDevice = add.Street;
85            }
86        }
87    }
88
89     public void setCoordinateHere(string paramFor){
90         if (paramFor.Equals("from"))
91         {
92             this.coorLatFrom = this.coorLat;
93             this.coorLongFrom = this.coorLong;
94             this.addressFrom = this.addressDevice;
95         }
96         else {
97             this.coorLatTo = this.coorLat;
98             this.coorLongTo = this.coorLong;
99             this.addressTo = this.addressDevice;
100        }
101    }
102
103     public void reset() {

```

```
1 |     this.coorLatFrom = 0.0; //from coordinate
2 |     this.coorLongFrom = 0.0;
3 |     this.coorLatTo = 0.0; //to coordinate
4 |     this.coorLongTo = 0.0;
5 |
6 |     this.addressDevice = "";
7 |     this.addressFrom = "";
8 |
9 |
10| }
11| }
```


LAMPIRAN F

KODE PROGRAM KELAS PROTOCOL

Listing F.1: Protocol.cs

```

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

```

```
1      }
2
3      private static String apiKey
4      {
5          get
6          {
7              return "97A7A1157A05ED6F";
8          }
9      }
10     private static String hostname
11    {
12        get
13        {
14            return "http://kiri.travel/";
15        }
16    }
17    private static String handle
18    {
19        get
20        {
21            return hostname+"handle.php?";
22        }
23    }
24    private static String iconPath
25    {
26        get
27        {
28            return hostname + "images/means/";
29        }
30    }
31    public String iconStart
32    {
33        get
34        {
35            return hostname + "images/stepicon-walkstart.png";
36        }
37    }
38    public String iconFinish
39    {
40        get
41        {
42            return hostname + "images/stepicon-finish.png";
43        }
44    }
45
46    private static String version_
47    {
48        get
49        {
50            return "2";
51        }
52    }
53
54    private static String modeFind
55    {
56        get
57        {
58            return "searchplace";
59        }
60    }
61
62    private static String modeRoute
63    {
64        get
65        {
66            return "findroute";
67        }
68    }
69    private static String modeNearby
70    {
71        get
72        {
73            return "nearbytransport";
74        }
75    }
76    private static String localeId
77    {
78        get
79        {
80            return "id";
81        }
82    }
83    private static String localeEn
84    {
85        get
86        {
87            return "en";
88        }
89    }
90    private static String presentationMobile
91    {
92        get
93        {
94            return "mobile";
95        }
96    }
97    private static String presentationDesktop
98    {
99        get
100       {
101           return "desktop";
102       }
103   }
```

```

1      }
2
3      public string getTypeTransport(string means, string meansDetail)
4      {
5
6          String uri = iconPath + means + "/baloon/" + meansDetail + ".png";
7          return uri;
8      }
9
10     public string getTypeTransportWOBaloon(string means, string meansDetail)
11     {
12
13         String uri = iconPath + means + "/" + meansDetail + ".png";
14         return uri;
15     }
16
17     public string getSearchPlace(string query, string region)
18     {
19
20         String uri = handle + uri_version + version_2 + uri_mode + modeFind + uri_region +
21             region + uri_query + query + uri_apikey + apiKey;
22         return uri;
23     }
24
25     public string getFindRoute(string start, string finish)
26     {
27
28         String uri = handle + uri_version + version_2 + uri_mode + modeRoute + uri_locale +
29             localeId + uri_start + start + uri_finish + finish + uri_presentation +
30             presentationDesktop + uri_apikey + apiKey;
31         return uri;
32     }
33
34     public async Task<RootObjectSearchPlace> getRequestSearch(string query, string region)
35     {
36
37         String uri = getSearchPlace(query, region);
38         Task<String> requestRouteTask = httpClient.GetStringAsync(new Uri(uri));
39         String request = await requestRouteTask;
40         RootObjectSearchPlace objectRootSearch = JsonConvert.DeserializeObject<
41             RootObjectSearchPlace>(request);
42         return objectRootSearch;
43     }
44
45     public async Task<RootObjectFindRoute> getRequestRoute(Double startLat, Double startLong,
46             Double finishLat, Double finishLong)
47     {
48
49         String uri = getFindRoute(startLat + "," + startLong, finishLat + "," + finishLong);
50         Task<String> requestRouteTask = httpClient.GetStringAsync(new Uri(uri));
51         String request = await requestRouteTask;
52         RootObjectFindRoute objectRootRoute = JsonConvert.DeserializeObject<
53             RootObjectFindRoute>(request);
54         return objectRootRoute;
55     }
56 }

```


1

LAMPIRAN G

2

KODE PROGRAM OBJEK DARI JSON

Listing G.1: RootObjectFindRoute.cs

```
3 | using System;
4 | using System.Collections.Generic;
5 | using System.Linq;
6 | using System.Text;
7 | using System.Threading.Tasks;
8 |
9 | namespace Kiri
10| {
11|     class RootObjectFindRoute
12|     {
13|         public string status { get; set; }
14|         public List<Routingresult> routingresults { get; set; }
15|     }
16| }
```

Listing G.2: RootObjectSearchPlace.cs

```
17 | using System;
18 | using System.Collections.Generic;
19 | using System.Linq;
20 | using System.Text;
21 | using System.Threading.Tasks;
22 |
23 | namespace Kiri
24 | {
25|     public class RootObjectSearchPlace
26|     {
27|         public string status { get; set; }
28|         public List<Searchresult> searchresult { get; set; }
29|         public object attributions { get; set; }
30|
31|     }
32| }
```

Listing G.3: Routingresult.cs

```
33 | using System;
34 | using System.Collections.Generic;
35 | using System.Linq;
36 | using System.Runtime.Serialization;
37 | using System.Text;
38 | using System.Threading.Tasks;
39 |
40 | namespace Kiri
41 | {
42|     class Routingresult
43|     {
44|         public List<List<object>> steps { get; set; }
45|         public string travertime { get; set; }
46|     }
47| }
```

Listing G.4: Searchresult.cs

```
48 | using System;
49 | using System.Collections.Generic;
50 | using System.Linq;
51 | using System.Text;
52 | using System.Threading.Tasks;
53 |
54 | namespace Kiri
55 | {
56|     public class Searchresult
57|     {
58|         public string placename { get; set; }
59|         public string location { get; set; }
60|     }
61| }
```