

SKRIPSI

**PENGEMBANGAN APLIKASI NAVIGASI
KENDARAAN UMUM UNTUK WINDOWS PHONE
BERBASIS KIRI API**



YOHAN

NPM: 2011730048

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2015**

UNDERGRADUATE THESIS

**DEVELOPMENT OF PUBLIC TRANSPORT
NAVIGATION APPLICATION FOR WINDOWS PHONE
BASED ON KIRI API**



YOHAN

NPM: 2011730048

**DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2015**

LEMBAR PENGESAHAN

PENGEMBANGAN APLIKASI NAVIGASI KENDARAAN UMUM UNTUK WINDOWS PHONE BERBASIS KIRI API

YOHAN

NPM: 2011730048

Bandung, 28 Mei 2015

Menyetujui,

Pembimbing Tunggal

Thomas Anung Basuki, Ph.D.

Ketua Tim Penguji

Anggota Tim Penguji

Joanna Helga, M.Sc.

Lionov, M.Sc.

Mengetahui,

Ketua Program Studi

Thomas Anung Basuki, Ph.D.

PERNYATAAN

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

PENGEMBANGAN APLIKASI NAVIGASI KENDARAAN UMUM UNTUK WINDOWS PHONE BERBASIS KIRI API

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

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

Dinyatakan di Bandung,
Tanggal 28 Mei 2015

Meterai

Yohan
NPM: 2011730048

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.

Aplikasi 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. Aplikasi yang dibuat memanfaatkan protokol HTTP untuk permintaan data lalu mendapatkan kembalian berupa JSON. Antarmuka aplikasi menampilkan hasil pencarian rute dalam 2 bentuk yaitu bentuk daftar dan bentuk peta.

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 Application 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. The application is made utilizing the HTTP protocol for request data and get a return in the form of JSON. Application interface displays search results in two forms, namely the form of lists and maps.

Keywords: Route, Public Transport, Windows Phone

Dipersembahkan untuk diri sendiri dan orang tua

KATA PENGANTAR

Puji Syukur penulis kepada Tuhan yang telah memberikan rahmatnya sehingga penulis dapat menyelesaikan skripsi yang berjudul "**PENGEMBANGAN APLIKASI NAVIGASI KENDARAAN UMUM UNTUK WINDOWS PHONE BERBASIS KIRI API**". Skripsi ini disusun dengan maksud memenuhi salah satu prasyarat menyelesaikan pendidikan di Jurusan Teknik Informatika, Fakultas Teknologi Informasi dan Sains, Universitas Katolik Parahyangan. Penulis menyadari bahwa dalam penulisan skripsi ini tidak terlepas dari bantuan dan dukungan berbagai pihak. Oleh karena itu, penulis ingin mengucapkan terima kasih kepada:

- Pak Thomas Anung Basuki selaku pembimbing yang telah memberikan banyak masukan untuk skripsi ini sehingga skripsi ini dapat diselesaikan dengan baik.
- Pak Pascal Alfadian atas masukan dan tambahan wawasan untuk skripsi ini sehingga skripsi ini dapat diselesaikan dengan baik.
- Orang tua penulis yang selalu memberi dukungan baik moril maupun materil dan doa.
- Adik penulis yaitu Yovin yang selalu memberi dukungan dan doa.
- Ibu Joanna Helga dan Pak Lionov selaku penguiyang sudah memberikan banyak masukan dalam penyusunan skripsi ini.
- Segenap teman penulis Alexius, Tommy, Winel, Jovan, Kevin Ferdi, Rico, David, Sam, Radit, Neta, Mario, Oswin, Edbert, Kiki, Jeane, Vinny, Reanta, Putri, Vania, Melissa, Willianto
- Segenap dosen Jurusan Teknik Informatika Universitas Katolik Parahyangan yang telah memberikan ilmu untuk penulis.
- Seluruh teman jurusan Teknik Informatika, Fakultas Teknologi Informasi dan Sains Universitas Katolik Parahyangan.

Semoga segala bantuan dan dukungan berbagai pihak tersebut mendapat berkat dari Tuhan. Semoga skripsi ini berguna bagi semua orang dan dapat dijadikan bahan pembelajaran. Akhir kata, penulis mohon maaf apabila kesalahan dan kekurangan dalam penulisan skripsi ini.

Bandung, Mei 2015

Penulis

DAFTAR ISI

KATA PENGANTAR	xv
DAFTAR ISI	xvii
DAFTAR GAMBAR	xx
DAFTAR TABEL	xxii
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 [1]	6
2.1.3 Navigasi [1]	6
2.1.4 Kontrol Tata Letak [1]	7
2.1.5 Kontrol Terhadap Teks [1]	8
2.1.6 Tombol dan Kontrol Pilihan [1]	9
2.1.7 Kontrol Daftar [1]	10
2.1.8 Siklus Hidup Aplikasi [1]	11
2.1.9 Peta di Windows Phone [1][2]	12
2.1.10 Pushpin pada Peta[1]	15
2.1.11 Polyline pada Peta[1]	16
2.1.12 Namespace Control Map	18
2.1.13 Lokasi	20
2.1.14 Memanfaatkan Sumber Data	23
2.1.15 Thread	26
2.2 Kiri API	26
2.2.1 Web Service Penentuan Rute	27
2.2.2 Web Service Pencarian Lokasi	28
2.2.3 Web Service Menemukan Transportasi Terdekat	29
3 ANALISIS	31
3.1 Analisis Aplikasi Sejenis	31
3.2 Analisis Aplikasi	33
3.2.1 Kebutuhan Aplikasi	33
3.2.2 Analisis Kontrol yang Dipakai	34

3.2.3	Analisis Terhadap Siklus Hidup Aplikasi	35
3.2.4	Analisis Peta	36
3.2.5	Analisis Pemanfaatan Sumber Data	37
3.2.6	Analisis Kiri API	38
3.2.7	Diagram <i>Use Case</i> dan Skenario	42
3.2.8	Kelas Diagram	46
4	PERANCANGAN	49
4.1	Diagram <i>Sequence</i>	49
4.2	Perancangan Kelas	52
4.2.1	Kelas <i>PhoneApplicationPage</i>	57
4.2.2	Kelas <i>MainPage</i>	57
4.2.3	Kelas <i>City</i>	58
4.2.4	Kelas <i>BackgroundWorker</i>	59
4.2.5	Kelas <i>Geocoordinate</i>	59
4.2.6	Kelas <i>Geolocator</i>	59
4.2.7	Kelas <i>Geoposition</i>	59
4.2.8	Kelas <i>LocationFinder</i>	60
4.2.9	Kelas <i>Map</i>	61
4.2.10	Kelas <i>HttpClient</i>	62
4.2.11	Kelas <i>JsonConvert</i>	62
4.2.12	Kelas <i>Protocol</i>	62
4.2.13	Kelas <i>RootObjectSearchPlace</i>	64
4.2.14	Kelas <i>SearchResult</i>	64
4.2.15	Kelas <i>RootObjectFindRoute</i>	64
4.2.16	Kelas <i>RoutingResult</i>	64
4.2.17	Kelas <i>Route</i>	65
4.3	Perancangan Antar Muka	67
4.3.1	Antarmuka Kelas <i>MainPage</i>	67
4.3.2	Antarmuka Kelas <i>Map</i>	68
4.3.3	Antarmuka Kelas <i>Route</i>	69
5	IMPLEMENTASI DAN PENGUJIAN APLIKASI	71
5.1	Implementasi	71
5.1.1	Perangkat Keras untuk Implementasi	71
5.1.2	Perangkat Lunak untuk Implementasi	72
5.1.3	Hasil Implementasi	72
5.2	Pengujian	73
5.2.1	Lingkungan Pengujian	73
5.2.2	Pengujian Fungsional	74
5.2.3	Pengujian Eksperimental	77
5.2.4	Perbandingan Waktu pencarian Rute	79
5.2.5	Perbandingan Aplikasi Pencari Rute di Android dengan Aplikasi Pencari Rute di Windows Phone	81
6	KESIMPULAN DAN SARAN	83
6.1	Kesimpulan	83
6.2	Saran	83
DAFTAR REFERENSI		85
A KODE PROGRAM KELAS MAINPAGE		87

B KODE PROGRAM KELAS MAP	93
C KODE PROGRAM KELAS ROUTE	95
D KODE PROGRAM KELAS CITY	101
E KODE PROGRAM KELAS LOCATIONFINDER	103
F KODE PROGRAM KELAS PROTOCOL	107
G KODE PROGRAM OBJEK DARI JSON	111

DAFTAR GAMBAR

2.1	Hierarki Navigasi	6
2.2	Tata Letak <i>StackPanel</i>	7
2.3	Tata Letak <i>Grid</i>	8
2.4	Tata Letak <i>Canvas</i>	8
2.5	<i>TextBlock</i> , <i>Textbox</i> dan <i>Passwordbox</i>	9
2.6	Gambar Kontrol Tombol dan Pilihan pada Windows Phone	10
2.7	Antarmuka <i>ListBox</i>	10
2.8	Gambar siklus Hidup Aplikasi[1]	11
2.9	Tampilan Peta pada Windows Phone	13
2.10	<i>Cartographic</i> [1]	14
2.11	<i>Heading</i> pada Peta[1]	14
2.12	<i>Pitch</i> pada Peta[1]	15
2.13	Keluaran Toolkit Pushpin pada Peta [2]	15
2.14	Tampilan Polyline pada Peta	16
3.1	Tampilan Utama Aplikasi Public Transport	31
3.2	Menunjuk Lokasi pada Peta	32
3.3	Memberikan Daftar Nama Tempat dan Nama Jalan Terkait	32
3.4	Tampilan Rute Kendaraan Umum dalam Bentuk Daftar	33
3.5	Tampilan Rute Kendaraan Umum di Peta	34
3.6	Tampilan <i>Pushpin</i> untuk Angkot	37
3.7	Tampilan <i>Pushpin</i> untuk Jalan Kaki	37
3.8	Diagram <i>Use Case</i>	44
3.9	Diagram Kelas	46
4.1	Diagram <i>sequence</i> Mendapatkan Kordinat Perangkat	49
4.2	Diagram <i>sequence</i> Mendapatkan Kordinat pada Peta	50
4.3	Diagram <i>sequence</i> Mendapatkan Rute	51
4.4	Diagram Kelas	52
4.5	Kelas Mainpage (sub-bab 4.2.1)	53
4.6	Kelas City (sub-bab 4.2.3)	53
4.7	Kelas JsonConvert (sub-bab 4.2.11)	53
4.8	Kelas LocationFinder (sub-bab 4.2.8)	54
4.9	Kelas Map (sub-bab 4.2.1)	54
4.10	<i>Package</i> WindowsPhone (sub-bab 4.2.4 sampai 4.2.7)	54
4.11	Kelas Protocol (sub-bab 4.2.12)	55
4.12	Kelas Route (sub-bab 4.2.17)	55
4.13	Kelas SearchPlace (sub-bab 4.2.13 dan 4.2.14)	56
4.14	Kelas FindRoute (sub-bab 4.2.15 dan 4.2.16)	56
4.15	Antarmuka <i>MainPage</i>	67
4.16	Antarmuka Daftar Tempat	68
4.17	Antarmuka <i>Map</i>	69
4.18	Antarmuka <i>Route</i>	69

4.19	Antarmuka Rute dalam Bentuk Daftar	70
5.1	Gambar antarmuka kelas MainPage	73
5.2	Gambar Antarmuka Splash di Kelas MainPage	74
5.3	Gambar Antarmuka <i>List</i> Asal dan <i>List</i> Tujuan di Kelas MainPage	75
5.4	Gambar Antarmuka Kelas Map	76
5.5	Gambar Antarmuka Menunggu di Kelas Route	77
5.6	Gambar Antarmuka Kelas Route	78
5.7	Gambar Antarmuka <i>List</i> di Kelas Route	79
5.8	Gambar Perbandingan Antarmuka <i>Home</i> di Android(kiri) dan Windows Phone(kanan)	80
5.9	Gambar Perbandingan Antarmuka Pemilihan Lokasi di Android(kiri) dan Windows Phone(kanan)	80
5.10	Gambar Perbandingan Antarmuka Menunggu hasil pencarian rute di Android(kiri) dan Windows Phone(kanan)	81
5.11	Gambar Perbandingan Antarmuka Penentuan Rute di Android(kiri) dan Windows Phone(kanan)	82

DAFTAR TABEL

2.1	Properti <i>Polyline Class</i>	18
2.2	Properti Kelas Map	19
2.3	Kelas pada <i>Namespace Geolocator</i>	21
2.4	Properti pada <i>Geocoordinate</i>	22
2.5	Tabel Parameter Layanan Penentuan Rute	27
2.6	Tabel Parameter Layanan Pencarian Lokasi	29
2.7	Tabel Parameter :ayanan Menemukan Transportasi Terdekat	30
3.1	Skenario Mendapatkan Lokasi untuk Masukan Lokasi Asal dan Lokasi Tujuan	44
3.2	Skenario Memasukan Lokasi Asal dan Lokasi Tujuan pada <i>textbox</i>	45
3.3	Skenario Menunjuk Lokasi Asal dan Lokasi Tujuan pada Peta	45
3.4	Skenario Memilih Alamat	45
3.5	Skenario Menampilkan Rute Kendaraan Umum	46
5.1	Tabel Hasil yang Diharapkan Penulis untuk Pengujian	74
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	77
5.7	Tabel Perbedaan	81
5.8	Tabel Perbedaan	82

BAB 1

PENDAHULUAN

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

1.1 Latar Belakang

Transportasi menjadi bagian yang penting bagi kehidupan manusia. Ada dua jenis transportasi yaitu kendaraan umum dan kendaraan pribadi. Pada saat penelitian ini dilakukan, kebanyakan orang lebih memilih kendaraan pribadi dibanding kendaraan umum. Maraknya penggunaan kendaraan pribadi ditambah penambahan jalur kendaraan yang tidak sebanding dengan banyaknya kendaraan di Indonesia akhirnya menimbulkan kemacetan. Hal yang menimbulkan maraknya penggunaan kendaraan pribadi dikarenakan kurang nyamannya kendaraan umum dan kesulitan dalam menentukan kendaraan umum yang harus dinaiki. Kesulitan seseorang untuk memilih kendaraan umum disebabkan banyaknya rute kendaraan umum membuat orang kebingungan dalam memilih kendaraan umum untuk menuju lokasi yang diinginkan dan kurangnya publikasi mengenai rute angkutan umum. Orang tidak tahu mengenai jenis angkutan umum yang akan dipakai dan cenderung malas untuk bertanya dan mencari rute yang benar dan efisien. Karena hal tersebut, seseorang lebih memilih menggunakan kendaraan pribadi dibandingkan dengan kendaraan umum.

Ide pembuatan aplikasi yang memudahkan seseorang dalam menentukan rute kendaraan umum sudah lebih dulu ada. Ide dalam penentuan rute angkutan umum dicetuskan oleh Kiri. Pembuat Kiri membuat pencarian rute berdasarkan latar belakang tiga masalah besar yaitu pemanasan global, kemacetan, dan harga bahan bakar minyak yang tinggi¹. Kiri pertama dikenal dalam bentuk situs *web* dengan alamat [kiri.travel](http://static.kiri.travel/en-about/). Meskipun Kiri dibuat dalam basis situs *web*, tetapi layanan Kiri dalam menentukan rute kendaraan umum dapat dimanfaatkan untuk pencarian rute selain dari situs *web*. Pemanfaatan layanan Kiri dalam mencari rute kendaraan umum tersebut yaitu dengan menggunakan Kiri API. Kiri API merupakan serangkaian aturan yang dapat dimanfaatkan dari perangkat lain yang dalam kasus ini adalah untuk mencari rute.

Pesatnya perkembangan teknologi informasi dan komunikasi memiliki pengaruh besar terhadap kehidupan manusia. Kebutuhan akan informasi mendorong perkembangan perangkat yang dapat mengakses informasi di manapun dan kapanpun. Teknologi yang dapat membantu manusia dalam hal tersebut adalah perangkat *mobile*. Kemudahan dalam penggunaan yang dimiliki perangkat

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

mobile kian digemari orang-orang terutama di Indonesia. Beberapa contoh sistem operasi pada perangkat *mobile* adalah Android, IOS, dan Windows Phone. Pada sistem operasi Android sudah banyak aplikasi pencarian rute. Selain Android salah satu sistem operasi yang menarik perhatian adalah Windows Phone 8 yang dibuat Microsoft. Antarmuka Windows Phone 8 Microsoft sebut *Metro*. Antarmuka *Metro* memiliki keunggulan, yaitu menarik dan mudah digunakan. Pada sistem operasi Windows Phone belum dijumpai aplikasi pencari rute.

Berdasarkan hal tersebut, aplikasi yang dikembangkan dalam tugas akhir ini adalah aplikasi Pencarian Rute Kendaraan Umum di Windows Phone. Aplikasi yang dikembangkan memungkinkan pengguna menemukan rute kendaraan umum untuk sampai di tujuan. Untuk memudahkan pengguna, aplikasi akan menampilkan hasil pencarian rute dalam dua bentuk yaitu bentuk peta dan bentuk daftar.

1.2 Rumusan Masalah

Sehubungan dengan latar belakang pada sub-bab 1.1 timbul permasalahan sebagai berikut:

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

1.3 Tujuan

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

- Mempelajari cara pembuatan perangkat lunak di Windows Phone lalu mengembangkan aplikasi yang dibuat.
- Membuat aplikasi di Windows Phone yang memanfaatkan Kiri API.

1.4 Batasan Masalah

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

- Aplikasi ini membutuhkan koneksi internet.
- Aplikasi ini menampilkan rute jalur angkot, bus umum dan travel di empat kota besar yaitu Bandung, Jakarta, Malang, dan Surabaya.

1.5 Metode Penelitian

Metode penelitian yang digunakan dalam membuat tugas akhir ini adalah sebagai berikut:

- Melakukan studi pustaka mengenai XAML, kontrol dan navigasi di Windows Phone, Peta di Windows Phone, GPS di Windows Phone dan Kiri API.

- Melakukan analisis terhadap aplikasi lain yang menggunakan Kiri API.
- Melakukan analisis terhadap dasar teori untuk pembangunan aplikasi Pencarian Rute Kendaraan Umum untuk Windows Phone.
- Melakukan perancangan aplikasi Pencarian Rute Kendaraan Umum untuk Windows Phone.
- Implementasi dari aplikasi Pencarian Rute Kendaraan Umum untuk Windows Phone.
- Menguji aplikasi Pencarian Rute Kendaraan Umum untuk Windows Phone.
- Membuat kesimpulan.

1.6 Sistematika Penulisan

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

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

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

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

Bab 5 membahas tentang implementasi dari perancangan di bab 4 lalu pengujian yang sudah dilakukan. Pengujian yang dilakukan ada dua yaitu pengujian fungsional dan eksperimental.

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

BAB 2

DASAR TEORI

Bab ini berisi dasar teori dari pembangunan Aplikasi Pencarian Rute Kendaraan Umum untuk Windows Phone. Beberapa teori yang dibahas dalam bab ini adalah XAML, navigasi, kontrol tata letak, kontrol terhadap teks, tombol dan kontrol pilihan, kontrol daftar, siklus hidup Windows Phone, peta, lokasi , pemanfaatan sumber data, dan Kiri API.

2.1 Windows Phone

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

Sub-bab 2.1.1 sampai 2.1.14 membahas pemrograman di Windows Phone. Pembahasan dimulai dengan apa itu Windows Phone dan fitur di Windows Phone yang digunakan dalam pembangunan perangkat lunak Pencarian Rute Kendaraan di Windows Phone.

2.1.1 Lingkungan Kerja

Microsoft .NET framework merupakan sebuah perangkat lunak yang dibangun untuk membantu dalam pembangunan aplikasi di Windows, Windows Phone, Windows Server, and Microsoft Azure[1]. Microsoft .NET framework terdiri dari *Common Language Runtime* dan *library* .NET Framework, yang meliputi kelas, *interface*, dan jenis nilai yang saling terintegrasi. Microsoft .NET Framework menyediakan lingkungan yang mudah dikelola, pengembangan yang disederhanakan, dan integrasi dengan berbagai bahasa pemrograman, termasuk Visual Basic dan Visual C#.

Seperti yang telah disebutkan ada dua bahasa pemrograman dalam .NET Framework yang dipakai dalam pembangunan aplikasi di Windows Phone 8 yaitu Visual Basic dan Visual C#. Untuk masalah kehandalan keduanya menawarkan kehandalan yang baik. Kelebihan dari Visual Basic adalah bahasa pemrograman berorientasi objek yang kuat dan memiliki banyak pengembangan fitur di *inheritance*, *polymorphism*, *interfaces*, dan *overloading*[1]. Kelebihan dari C# yang merupakan pengembangan dari C/C++ adalah sederhana, moderen, aman dan berorientasi objek[1]. Ketika memilih bahasa pemrograman didasarkan pada kenyamanan penggunaannya. Bahasa pemrograman Visual Basic .NET lebih mudah digunakan bagi seseorang yang sudah pernah menggunakan Visual Basic 6.0, sedangkan bahasa pemrograman C# lebih mudah digunakan bagi seseorang yang sudah pernah menggunakan bahasa pemrograman C++ atau java.

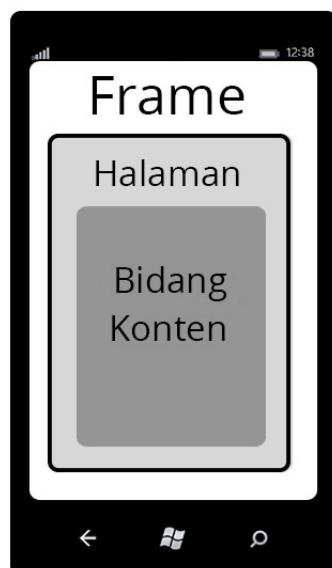
2.1.2 XAML [1]

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

Untuk aturan penulisan sintaks XAML didasarkan pada XML. Setiap XAML menggunakan konvensi bahasa XAML dan ditulis pada *namespace* yang ditandai dengan *prefix* x sebagai elemen paling atas. Setelah itu di baris ke dua dimulai dengan "xmlns" diikuti titik dua, lalu nama dari *namespace*, selanjutnya diikuti dengan tanda sama dengan dan *path* yang merepresentasikan *namespace*[2]. *Prefix* x pada XAML mengandung beberapa struktur program yang sering digunakan yaitu :

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

2.1.3 Navigasi [1]



Gambar 2.1: Hierarki Navigasi

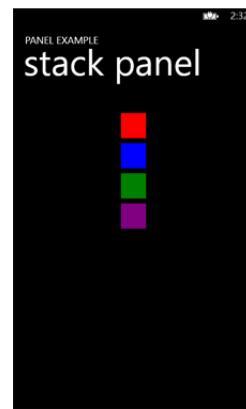
Aplikasi yang dibuat di Windows Phone didasarkan pada model halaman. Model halaman adalah pengguna berpindah dari satu halaman ke halaman lain dengan konten yang berbeda dan *frame* sebagai pengontrolnya. Setiap antarmuka aplikasi dibungkus dengan *frame*. *Frame* ini melakukan kontrol terhadap aplikasi dan memungkinkan perpindahan dari satu halaman ke halaman lain. Sedangkan halaman merupakan pembungkus dari elemen di dalamnya saja. Untuk lebih jelas mengenai *frame*, halaman, dan area konten dapat dilihat pada gambar 2.1.

2.1.4 Kontrol Tata Letak [1]

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. Pada saat 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.

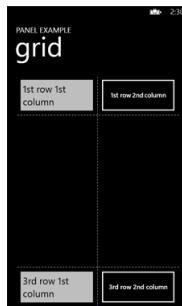
Terdapat 3 jenis panel yang digunakan untuk menangani tata letak yaitu *grid*, *stackPanel*, dan *canvas*. Perlu diperhatikan bahwa setiap halaman hanya memiliki satu jenis panel. Berikut adalah 3 jenis panel pada Windows Phone:

- *StackPanel* merupakan panel yang memposisikan elemen menjadi 1 baris dan beberapa elemen di setiap halaman diposisikan horizontal atau vertical saja. Contoh peletakan elemen untuk tata letak *stackpanel* dapat dilihat pada gambar 2.2.

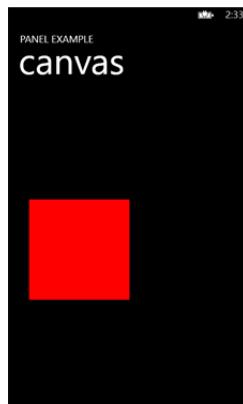


Gambar 2.2: Tata Letak *StackPanel*

- *Grid* merupakan panel yang mendukung tata letak yang rumit. Panel ini memposisikan elemen di baris dan kolom mana saja di setiap halaman. Contoh peletakan elemen untuk tata letak *grid* dapat dilihat pada gambar 2.3.
- *Canvas* memposisikan elemen sebagai absolut koordinat. Jadi setiap elemen di dalam *canvas* dapat diposisikan spesifik sesuai kordinat x dan y. Contoh peletakan elemen untuk tata letak *canvas* dapat dilihat pada gambar 2.4.



Gambar 2.3: Tata Letak *Grid*



Gambar 2.4: Tata Letak *Canvas*

Kode untuk mengatur jenis panel pada Windows Phone dapat dilihat pada *listing 2.1*.

Listing 2.1: Kode Tata Letak Grid

```

1 | <Grid x:Name="ContentPanel">
2 | </Grid>
```

2.1.5 Kontrol Terhadap Teks [1]

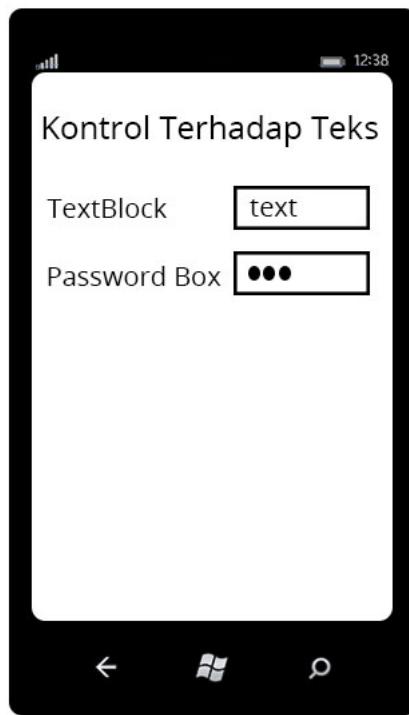
Kontrol terhadap teks akan menampilkan konten yang memiliki tipe *string*. Terdapat berbagai macam kontrol terhadap teks di Windows Phone yaitu *textblock*, *textbox* dan *passwordbox*. Ketiga macam kontrol tersebut dibedakan berdasarkan tujuannya. Berikut keterangan, gambar 2.5, dan kode pada *listing 2.2* kontrol teks.

- *Textblock* merupakan tempat menaruh potongan teks yang hanya bisa dilihat.
- *Textbox* biasanya digunakan untuk teks masukan yang pendek. Tapi bisa juga dipakai untuk masukan yang banyak dan beberapa baris.
- *Passwordbox* biasanya digunakan untuk masukan yang bersifat rahasia. Karakter yang dimasukan langsung disamarkan menjadi bentuk titik.

Listing 2.2: Kode untuk Menampilkan TextBlock dan TextBox

```

1 | <TextBlock x:Name="TextBlock1" Text="TextBlock"/>
2 | <TextBox x:Name="TextBox1" Text="TextBox"/>
```

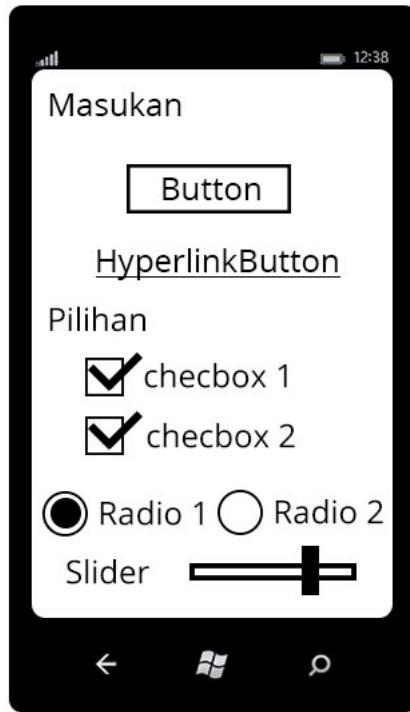


Gambar 2.5: *TextBlock*, *Textbox* dan *Passwordbox*

2.1.6 Tombol dan Kontrol Pilihan [1]

Tombol memungkinkan pengguna untuk berpindah halaman. Sedangkan kontrol pilihan memudahkan dalam memilih. Berikut keterangan dan gambar 2.6 tombol dan kontrol pilihan.

- *Button* merupakan kontrol yang dipakai pengguna untuk mengaktifkan *event* tekan.
- *HyperlinkButton* merupakan kontrol yang menampilkan tautan. Jika *HyperlinkButton* ditekan maka akan berpindah ke halaman yang dituju.
- *CheckBox* merupakan kontrol yang memungkinkan pengguna memilih beberapa item.
- *RadioButton* merupakan kontrol yang memungkinkan pengguna memilih satu pilihan dari beberapa pilihan.
- *Slider* merupakan kontrol yang memungkinkan pengguna memilih nilai kisaran dari jalur yang sudah disediakan.

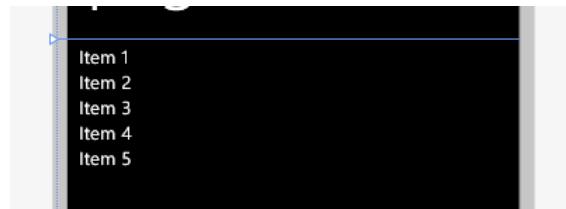


Gambar 2.6: Gambar Kontrol Tombol dan Pilihan pada Windows Phone

2.1.7 Kontrol Daftar [1]

Kontrol yang dipakai untuk menampilkan daftar dari beberapa *item*. Berikut keterangan, gambar 2.7, dan kode pada *listing 2.3* kontrol daftar.

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



Gambar 2.7: Antarmuka *ListBox*

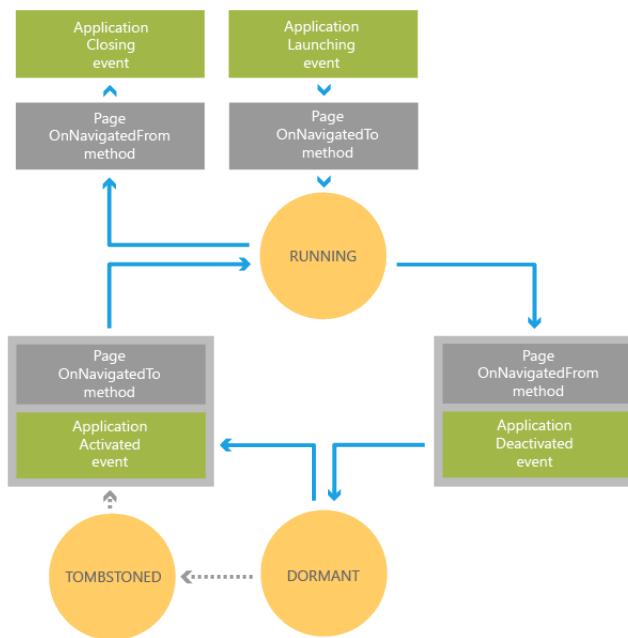
Listing 2.3: Kode untuk menampilkan *listBox*

```

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>
```

2.1.8 Siklus Hidup Aplikasi [1]

Siklus hidup aplikasi merupakan waktu mulai dari aplikasi dijalankan sampai dengan aplikasi diberhentikan dari memori. Siklus hidup aplikasi penting diketahui agar pengguna tidak kecewa (maksud kecewa berarti adanya ketidaksesuaian aksi pengguna dan reaksi perangkat) menggunakan aplikasi serta memastikan sumber daya tersedia (dalam aplikasi ini yaitu sumber daya GPS). Seringkali pengguna tidak berhati-hati dalam menggunakan aplikasi, maka dari itu perlu dipahami kapan aplikasi harus diaktifkan, ditangguhkan, atau bahkan dinonaktifkan karena sudah tidak digunakan. Gambar 2.8 adalah ilustrasi dari siklus hidup pada Windows Phone.



Gambar 2.8: Gambar siklus Hidup Aplikasi[1]

Sesuai gambar 2.8 lingkaran melambangkan keadaan aplikasi, persegi panjang menunjukkan peristiwa aplikasi atau tingkat peristiwa di halaman. Berikut ini adalah keterangan untuk siklus hidup Windows Phone pada gambar 2.8.

- *Launching Event*

Merupakan tampilan awal saat aplikasi dipilih yang memberitahukan pengguna bahwa aplikasi sedang dijalankan. *Event* ini akan dipanggil ketika aplikasi dijalankan pertama kali. *Event* ini akan berjalan di belakang (*background processing*) ketika aplikasi ditutup sementara atau sedang berada pada keadaan *Dormant* atau *Tombstoned* menjadi *running*.

- Keadaan *Running*

Setelah dijalankan, aplikasi akan masuk ke keadaan *running*. Hal ini akan terus berlangsung sampai pengguna berpindah ke halaman depan, atau mundur melewati halaman utama aplikasi. Aplikasi keluar dari keadaan *running* jika perangkat di kunci. Keadaan *running* masih dapat terjadi saat perangkat di kunci dengan menonaktifkan *idle detection* pada aplikasi.

- *Method OnNavigatedFrom*

Merupakan *method* yang dipanggil ketika berpindah ke halaman lain aplikasi. Ketika *method* ini dipanggil maka aplikasi akan menyimpan keadaan dari halaman sebelum ditinggalkan. Hal tersebut dibutuhkan agar halaman tersebut bisa dikembalikan ke keadaan sebelum ditinggalkan saat pengguna ingin kembali ke halaman tersebut. Pemanggilan dilakukan ketika berpindah antara halaman di aplikasi atau ketika berpindah aplikasi.

- *The Deactivated Event*

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

- Keadaan *Dormant* (Tidak Aktif)

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

- Keadaan *Tombstoned* (Batu Nisan)

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

- *Activated Event*

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

- *Method OnNavigatedTo*

Method ini dipanggil ketika pengguna berpindah ke halaman yang sebelumnya ditinggalkan. *method* ini akan memeriksa keadaan aplikasi dan memulihkannya jika keadaan sebelumnya pernah disimpan.

- *Closing Event*

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

2.1.9 Peta di Windows Phone [1][2]

Peta yang dipakai pada Windows Phone adalah *Windows Phone Maps*. *Windows Phone Maps* menawarkan beberapa pilihan dalam tampilan peta yang terdiri dari *cartographic*, pencahayaan dan pandangan. Selain tampilan pada sub-bab ini dibahas mengenai mendapatkan lokasi, petunjuk arah, *map polyline* dan *pushpin*[1].

Penambahan peta pada Windows Phone harus memanfaatkan kontrol peta. Kontrol peta merupakan bagian dari *library* Windows Phone. Dengan begitu penggunaan peta di Windows Phone perlu direferensikan. Untuk dapat menggunakan peta, perlu ditambahkan *capability*



Gambar 2.9: Tampilan Peta pada Windows Phone

ID_CAP_MAP. Setelah hal tersebut dilakukan peta dapat ditampilkan di layar perangkat. Berikut gambar 3.5, kode XAML pada *listing 2.4*, dan kode program pada *listing 2.5* peta.

Listing 2.4: Menampilkan Peta dengan Nama MyMap dari XAML

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

Listing 2.5: Menampilkan Peta dengan Nama MyMap dari Kode Program

```
1| public mapFrom()
2| {
3|     InitializeComponent();
4|     Map MyMap = new Map();
5|     ContentPanel.Children.Add(MyMap);
6| }
```

Dalam tampilannya ada beberapa hal yang perlu diperhatikan agar pengguna merasa nyaman saat melihat peta di Windows Phone. Beberapa tampilan yang bisa ditampilkan dibuat untuk hal yang berbeda-beda. Berikut pembahasan cara untuk menentukan pusat dan tingkat 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 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 2.6* dan *listing 2.7*.

Listing 2.6: Mengatur tingkat zoom dari XAML

```
1| <Controls:Map x:Name="MyMap" ZoomLevel="10"/>
```

Listing 2.7: Mengatur Tingkat Zoom dari Kode Program

```
1| public mapFrom()
2| {
3|     InitializeComponent();
4|     Map MyMap = new Map();
5|
6|     // Mengatur titik tengah peta
7|     MyMap.Center = new GeoCoordinate(47.6097, -122.3331); // Parameter 1 bernilai latitude
8|                                         // bertipe double, parameter 2 bernilai longitude bertipe double
```

```

9   // mengatur tingkat zoom
10  MyMap.ZoomLevel = 10; //Rentang nilai zoom 0 sampai 20
11  ContentPanel.Children.Add(MyMap);
12 }

```

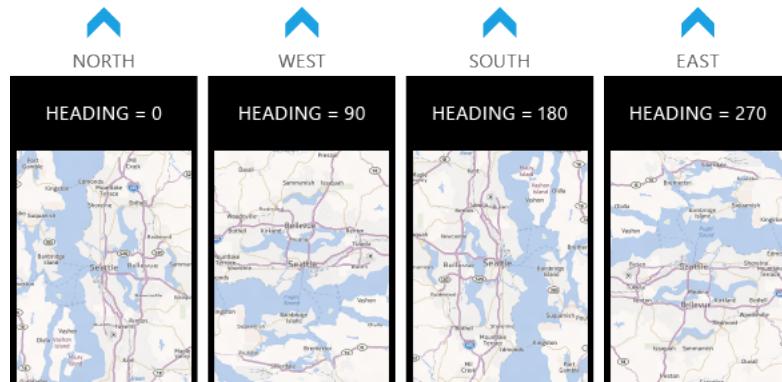
- *Cartographic* peta di Windows Phone merupakan cara pandang dalam melihat dan menerjemahkan peta. Terdapat empat jenis *cartographic*, yaitu:

- *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.



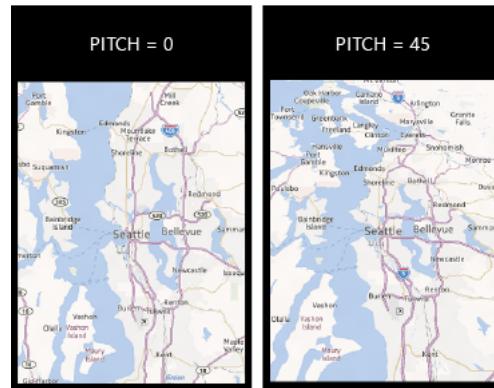
Gambar 2.10: *Cartographic*[1]

- 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.



Gambar 2.11: *Heading* pada Peta[1]

- *Pitch* merupakan derajat kemiringan dari peta yang dilihat dari sudut pandang pengguna. Contoh, *Pitch* = 0 berarti melihat dari atas ke bawah sedangkan *Pitch* = 45 berarti melihat dari samping ke bawah dengan sudut 45 derajat.



Gambar 2.12: *Pitch* pada Peta[1]

2.1.10 *Pushpin* pada Peta[1]

Pushpin merupakan elemen yang dapat ditempatkan pada peta secara spesifik dan dipakai untuk interaksi pada peta. Peta tidak mendukung langsung penggunaan *pushpin* karena merupakan elemen dari *MapOverlay* (bagian/lapisan terpisah dari peta). Windows Phone memiliki Windows Phone 8 Toolkit yang memiliki set objek agar dapat menggunakan *pushpin* pada peta di Windows Phone. Contoh keluaran *pushpin* dapat dilihat pada Gambar 2.13 dan kode untuk menampilkannya dapat dilihat pada listing 2.8.



Gambar 2.13: Keluaran Toolkit Pushpin pada Peta [2]

Listing 2.8: Kode untuk Menampilkan Pushpin

```

1 MapOverlay overlay = new MapOverlay
2 {
3     GeoCoordinate = map.Center,
4     Content = new Border
5     {
6         BorderBrush = new SolidColorBrush( Color.FromArgb(120, 255, 0, 0)),
7         Child = new TextBlock(){Text="Pushpin"},
8         BorderThickness = new Thickness(1),

```

```

9     Background = new SolidColorBrush (Color . FromArgb(120 , 255 , 0 , 0)) ,
10    Width = 80 ,
11    Height = 60
12  }
13 };
14 MapLayer layer = new MapLayer () ;
15 layer . Add (overlay) ;
16
17 map . Layers . Add (layer) ;

```

Kelas *pushpin* 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*:

- *background {set;}*
Untuk mengatur latar belakang.
- *content {set;}*
Untuk mengatur konten pada *pushpin*.
- *margin {set;}*
Untuk mengatur batas pada *pushpin*.
- *GeoCoordinate {set;}*
Untuk mengatur kordinat peletakan *pushpin*.

2.1.11 *Polyline* pada Peta[1]

Dalam menentukan arah dibutuhkan dua titik yaitu titik awal dan titik tujuan. Tentu saja arah tersebut butuh ditandai dengan garis. *Polyline* merupakan serangkaian garis lurus yang terhubung. *Polyline* dapat menandai peta dengan warna maupun ketebalan tertentu. Contoh keluaran *polyline* dapat dilihat pada Gambar 2.14 dan kode untuk menampilkannya dapat dilihat pada listing 2.9.



Gambar 2.14: Tampilan Polyline pada Peta

Listing 2.9: Kode untuk Menampilkan Polyline

```

1 MapPolyline line = new MapPolyline () ;
2 line . StrokeColor = Colors . Red; //Memberikan warna
3 line . StrokeThickness = 10; //Mengatur ketebalan
4 line . Path . Add (new GeoCoordinate (-6.8619546, 107.614441)); //Menambahkan titik pertama
5 line . Path . Add (new GeoCoordinate (-6.908693, 107.611185)); //Menambahkan titik kedua

```

Kelas *polyline* merupakan kelas yang dipakai untuk menggambarkan garis lurus yang saling terhubung. Kelas ini tergabung ke dalam *namespace Microsoft.Phone.Maps.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 2.1: Properti *Polyline Class*

Berikut *method* yang dapat digunakan pada kelas ini.

- *CheckAccess*

method yang menentukan bisa atau tidaknya pemanggilan *thread* untuk mengakses objek.

- *ClearValue*

method yang membersihkan nilai lokal

- *Finalize*

method yang dipakai untuk melakukan pembersihan pada sumber daya yang tidak terpakai sebelum objek dihancurkan.

2.1.12 Namespace Control Map

Namespace merupakan nama yang digunakan untuk menyatakan lingkup yang berisi satu set objek. *Namespace Control Map* merupakan nama dari set objek yang mengatur peta. 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*.

Kelas *Map* Merupakan kelas yang mewakili *Namespace Control 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>PedestrianFeaturesEnabled</i>	Indikasi fitur pejalan kaki ditampilkan.
<i>Pitch</i>	Mengatur dan mendapatkan derajat kemiringan peta.
<i>Width</i>	Mengatur dan mendapatkan lebar.
<i>ZoomLevel</i>	Mengatur dan mendapatkan tingkat zoom pada peta.

Tabel 2.2: Properti Kelas Map

Berikut *method* yang dapat digunakan pada kelas ini.

- SetView(*LocationRectangle*)

method untuk mengatur pandangan di atas peta secara spesifik sesuai wilayah geografis. *Method* ini tidak mengembalikan nilai.
- SetView(*GeoCoordinate, Double*)

method untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah dan tingkat zoom. *Method* ini tidak mengembalikan nilai.
- SetView(*LocationRectangle, MapAnimationKind*)

method untuk mengatur pandangan di atas peta secara spesifik sesuai region geografis dan animasi. *Method* ini tidak mengembalikan nilai.
- SetView(*LocationRectangle, Thickness*)

method untuk mengatur pandangan di atas peta secara spesifik sesuai region geografis dengan batas tertentu. *Method* ini tidak mengembalikan nilai.
- SetView(*GeoCoordinate, Double, MapAnimationKind*)

method untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah, tingkat zoom, dan animasi. *Method* ini tidak mengembalikan nilai.
- SetView(*GeoCoordinate, Double, Double*)

method untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah, tingkat zoom, dan *heading*. *Method* ini tidak mengembalikan nilai.
- SetView(*LocationRectangle, Thickness, MapAnimationKind*)

method untuk mengatur pandangan di atas peta secara spesifik sesuai wilayah geografis dengan batas tertentu, dan animasi. *Method* ini tidak mengembalikan nilai.
- SetView(*GeoCoordinate, Double, Double, MapAnimationKind*)

method untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah, tingkat zoom, *heading*, dan animasi. *Method* ini tidak mengembalikan nilai.

- *SetView(GeoCoordinate, Double, Double, Double)*
method untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah, tingkat zoom, *heading*, dan *pitch*. *Method* ini tidak mengembalikan nilai.
- *SetView(GeoCoordinate, Double, Double, Double, MapAnimationKind)*
method untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah, tingkat zoom, *heading*, *pitch*, dan animasi. *Method* ini tidak mengembalikan nilai.
- *UpdateLayout*
method yang memastikan semua posisi objek turunan mengikuti tata letak.

2.1.13 Lokasi

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

Hal yang perlu diperhatikan dalam menentukan layanan lokasi adalah penangkap GPS, Wi-Fi, dan jaringan seluler. Perangkat tersebut berfungsi sebagai penyedia data lokasi dengan berbagai tingkat akurasi dan konsumsi daya. Perangkat diatas juga berkomunikasi langsung untuk memutuskan sumber mana yang digunakan untuk menentukan lokasi perangkat berdasarkan ketersediaan data lokasi dan prasyarat yang ditentukan aplikasi. Lapisan diatas penyedia data lokasi tersebut adalah pengelola antarmuka. Aplikasi menggunakan antarmuka tersebut untuk memulai dan menghentikan layanan lokasi, mengatur tingkat akurasi, dan menerima data lokasi.

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

Mendapatkan Posisi Pengguna

Windows Phone 8 telah menyediakan kelas *GeoCoordinate* yang digunakan untuk mengetahui posisi pengguna. *Geolocator class* dari *Windows.Devices.Geolocation* dapat mengembalikan posisi saat ini. Untuk menggunakan *Geolocator*, perlu menghidupkan *ID_CAP_LOCATION* di [*/properties/WMApManifest.xml*](#). Dalam mendapatkan posisi perlu diperhatikan status dari GPS karena membutuhkan waktu dari awal pengaktifan hingga mendapatkan lokasi pengguna secara akurat. Untuk lebih jelas mengenai status posisi dapat dilihat pada nilai status dibawah ini.

- *Ready* : Jika lokasi tersedia.

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

Namespace Geolocator

Namespace Geolocator merupakan nama dari set objek yang dipakai untuk mengatur pengaksesan lokasi. Windows Phone 8 menyediakan *namespace* bawaan untuk mengakses lokasi. *Namespace* yang disediakan adalah *namespace* Geolocator. *Namespace* ini dapat 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 *WMAAppManifest.xml*. Kelas yang diatur oleh *namespace* Geolocator dapat dilihat pada tabel 2.3[3].

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

Tabel 2.3: Kelas pada *Namespace Geolocator*

Kelas Geocoordinate

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

Kelas Geolocator

Kelas Geolocator merupakan kelas yang mendukung pengaksesan terhadap lokasi. Berikut *method* yang disediakan *Geolocator*:

- *public IAsyncOperation<Geoposition> GetGeopositionAsync()*
Operator await diatas dimaksudkan untuk meminta posisi lokasi terus menerus sampai selesai dan menunda tugas yang lain. *Method* GetGeopositionAsync() merupakan *method* bawaan

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*

kelas Geolocator. *Method GetGeopositionAsync()* dapat meminta data lokasi dan menangani-nya sampai selesai. Kembalian dari *method GetGeopositionAsync()* adalah objek *Geoposition*.

Berikut Properti yang disediakan kelas Geolocator:

- *public PositionStatus LocationStatus { get; }*

Merupakan properti dari kelas *geolocator* untuk mendapatkan status posisi dengan meng-embalikan kelas *PositionStatus*. Status pada kelas *PositionStatus* adalah *Ready*, *Initializing*, *NoData*, *Disable*, *NotInitialized*, dan *NotAvailable*.

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

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

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

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

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

Merupakan properti untuk mendapatkan selang waktu pembaruan lokasi. Properti ini meng-eluarkan tipe data unit.

Kelas Geoposition

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

- *public CivicAddress CivicAddress { get; }*

Data alamat sipil yang terkait dengan lokasi geografis.

- *public Geocoordinate Coordinate { get; }*

Data latitude dan longitude yang terkait lokasi geografis.

Kelas ReverseGeocodeQuery

ReverseGeocodeQuery merupakan kelas yang digunakan untuk membalikkan objek *query geocode*. Kelas *ReverseGeocodeQuery* dapat mendapatkan alamat dari kordinat *latitude* dan *longitude*. Berikut Properti yang disediakan kelas *ReverseGeocodeQuery*:

- *GeoCoordinate { get; set; }*
Untuk menampung kordinat.

Berikut *method* yang disediakan *ReverseGeocodeQuery*:

- *QueryAsync()*
Memulai permintaan.

Berikut *event* yang disediakan *ReverseGeocodeQuery*:

- *QueryCompleted*
Terjadi ketika permintaan selesai dijalankan.

2.1.14 Memanfaatkan Sumber Data

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

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

Banyak *web service* yang mengembalikan data dalam format JSON. JSON memiliki format *data-interchange* yang ringan ^[4]. Karena hal tersebut JSON mudah diurai dan dihasilkan oleh mesin. Kurung kurawal mengindikasikan objek, kurung siku berarti *array*, dan properti berupa nama dan nilai pasangan yang dipisahkan oleh titik dua. JSON memiliki ukuran data yang kecil dan baik untuk penggunaan perangkat *mobile*. Untuk contoh format JSON dapat dilihat di bagian Kiri API pada Bab dua ini karena Kiri API menggunakan format JSON. *Serializes* menggunakan *DataContractJsonSerializer* membuat *serializes* mudah untuk menerjemahkan *form string* JSON ke objek yang dapat langsung digunakan. *DataContractJsonSerializer* memakai *WriteObject()* untuk *serializes* and *method ReadObject()* untuk *deserializes*.

Kelas HttpClient

Kelas HttpClient merupakan kelas yang dipakai untuk mengirim permintaan HTTP dan menerima kembalian HTTP dari *Uniform Resource Identifier*(URI) yang dapat diidentifikasi. URI merupakan urutan kompak karakter yang mengidentifikasi sumber daya abstrak dan fisik ^[5]. Berikut *method* yang disediakan kelas *HttpClient*.

- *DeleteAsync(Uri)*

Method yang dipakai untuk mengirimkan permintaan DELETE ke URI yang spesifik sebagai operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan aplikasi untuk melanjutkan pekerjaan selagi *method* ini dipanggil². *Method* ini membutuhkan parameter

URI sebagai tujuan dari permintaan. Sedangkan kembalinya berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut memiliki 2 parameter yaitu hasil berupa pesan kembalian dari http dan kemajuan dari data yang dikirim.

- **GetAsync(*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 kembalinya berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut memiliki 2 parameter yaitu hasil berupa pesan kembalian dari http dan kemajuan dari data yang dikirim.

- **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 kembalinya berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut memiliki 2 parameter yaitu hasil berupa pesan kembalian dari HTTP dan kemajuan dari data yang dikirim.

- **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 kembalinya 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 kembalinya 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 kembalinya berupa objek yang mewakili

operasi *asynchronous* disertai kemajuan. Objek tersebut memiliki 2 parameter yaitu hasil berupa pesan kembalian dalam bentuk string dan kemajuan dari data yang dikirim.

- **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 memiliki 2 parameter yaitu hasil berupa pesan kembalian dari HTTP dan kemajuan dari data yang dikirim.

- **SendRequestAsync(*HttpRequestMessage*)**

Method yang dipakai untuk mengirimkan permintaan HTTP sebagai operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan aplikasi untuk melanjutkan pekerjaan selagi *method* ini dipanggil². *Method* ini membutuhkan parameter pesan dari permintaan. Sedangkan 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.

- **SendRequestAsync(*HttpRequestMessage*, *HttpCompletionOption*)**

Method yang dipakai untuk mengirimkan permintaan HTTP sebagai operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan aplikasi untuk melanjutkan pekerjaan selagi *method* ini dipanggil². *Method* ini membutuhkan parameter pesan dari permintaan dan nilai tambahan yang 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.

Json.NET

Json.NET merupakan *library* yang digunakan untuk melakukan *serialize* (proses konversi objek ke *stream of bytes*) dan *deserialize* (proses konversi *stream of bytes* ke bentuk objek) terhadap objek dalam .NET [6]. *Library* ini memiliki 2 kelas yaitu kelas JsonConvert dan kelas JsonSerializer. Berikut merupakan keterangan dari dua kelas tersebut.

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

1. *Method* SerializeObject()

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

2. *Method* DeserializeObject()

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

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

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

2.1.15 Thread

Sebuah aplikasi terdiri dari satu atau lebih proses. Sebuah proses adalah kode program yang sedang dieksekusi. Dalam konteks proses terdapat satu atau lebih thread yang dijalankan. Thread merupakan unit dasar yang sistem operasi alokasikan sebagai waktu yang prosessor kerjakan. Setiap thread dapat menjalankan setiap bagian dari kode program tergantung yang diatur sistem operasi.

BackgroundWorker

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

- *Method RunWorkerAsync()*
method yang digunakan untuk memulai eksekusi operasi di latar belakang.

Berikut *event* yang disediakan kelas *BackgroundWorker*.

- *DoWork*
event yang terjadi ketika *method RunWorkerAsync()* dipanggil.
- *RunWorkerCompleted*
event yang terjadi ketika operasi di latar belakang selesai dilakukan, dibatalkan, atau mencapai *exception*.

2.2 Kiri API

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

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

- <http://preview.kiri.travel/handle.php>
Merupakan URL untuk uji coba. Untuk kemampuannya juga menurut dokumentasi Kiri API masih tidak stabil.
- <http://kiri.travel/handle.php>
Merupakan URL produksi. Ini merupakan URL yang direkomendasikan untuk menangani permintaan pengguna.

Untuk setiap permintaan membutuhkan *API key* yang didapat dengan mendaftar[7]. Penggunaan API memungkinkan pengaksesan di mana saja dengan menggunakan koneksi internet. Pada subbab 2.2.1 sampai sub-bab 2.2.3 dibahas beberapa layanan Kiri API.

Berikut langkah-langkah untuk mendapatkan *API key*.

- Masuk ke situs dev.kiri.travel.

- Daftar dengan memasukan alamat email, nama, dan nama perusahaan.
- Setelah mendaftar maka password dikirimkan ke alamat email. Tentunya password akan dibuat otomatis oleh pihak Kiri.
- *Login* dengan menggunakan password yang dikirim ke alamat email.
- Setelah berhasil *login*, di menu utama pilih *API Keys Managements*.
- Pilih tombol Add lalu masukan deskripsi penggunaan *API key*.
- *API key* didapat dan dapat digunakan.

2.2.1 Web Service Penentuan Rute

Web service penentuan rute merupakan layanan Kiri API yang digunakan untuk mendapatkan langkah perjalanan dari lokasi asal ke lokasi tujuan. Parameter dan keterangan 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

Format kembalian layanan penentuan rute dapat dilihat pada *listing 2.10*:

Listing 2.10: Kode Kembalian Pencarian Rute

```

1 {
2     "status": "ok" or "error"
3     "routingresults": [
4         {
5             "steps": [
6                 [
7                     "walk" or "none" or others,
8                     "walk" or vehicle_id or "none",
9                     ["lat_1,lon_1", "lan_2,lon_2", ... "lat_n,lon_n"],
10                    "human readable description, dependant on locale",
11                    URL for ticket booking or null (future)
12                ],
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            "traveltime": any text string, null if and only if route is not found.
22        },
23        {
24            "steps": [ ... ],
25            "traveltime": "..."
26        },

```

```

27     {
28         "steps": [ ... ],
29         "traveltime": "...",
30     } ,
31     ...
32 }
33 }
```

Berikut maksud dari *listing 2.10*:

Ketika pencarian rute sukses dilakukan maka status memberitahukan "ok" seperti di baris 2. Selanjutnya setiap langkah dari posisi awal ke posisi tujuan ditampung di elemen *array* untuk menampung langkah. Pencarian dapat bernilai "error" jika terdapat kesalahan dalam pengiriman parameter yang diterima Kiri. Berikut keterangan dari setiap *array* yang menampung langkah.

- Indeks ke 0 atau baris 7 pada *listing 2.10* dapat berisi "walk" atau "none" atau "others". Baris tersebut berarti jika "walk" untuk berjalan kaki, "none" jika rute tidak ditemukan dan "others" untuk menggunakan kendaraan.
- Indeks ke 1 atau baris 8 pada *listing 2.10* 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 2.10* 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 2.10* berisi bentuk yang akan ditampilkan kepada pengguna. Informasi yang disampaikan dapat berupa:
 - %fromicon = untuk menunjukkan ikon "from". Biasanya untuk mode presentasi pada perangkat *mobile*.
 - %toicon = untuk menunjukkan ikon "to". Biasanya untuk mode presentasi pada perangkat *mobile*.
- Indeks ke 4 atau bari 11 pada *listing 2.10* berisi URL untuk pemesanan tiket jika tersedia. Jika tidak tersedia maka bernilai *null*.

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

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

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

2.2.2 Web Service Pencarian Lokasi

Merupakan layanan Kiri API yang digunakan untuk mencari lokasi beserta koordinat *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 dicari tempatnya
<i>querystring</i>	teks apa saja dengan minimum text satu karakter	<i>Query string</i> yang 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 2.11*.

Listing 2.11: 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 }
```

Berikut maksud dari *listing 2.11*:

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

- "searchresult" (pada baris 4 sampai 7, 8 sampai 11, dan seterusnya) berisi *array* dari tempat:
 - placename: nama tempat
 - location: *latitude* dan *longitude* dari tempat
- "attributions" berisi kumpulan nilai yang berisikan atribut tambahan untuk dimunculkan.

2.2.3 Web Service Menemukan Transportasi Terdekat

Merupakan Kiri API yang digunakan untuk menemukan rute transportasi terdekat sesuai titik yang diinginkan pengguna. Parameter dan keterangan untuk layanan ini dapat dilihat 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>	<i>Latitude</i> dan <i>longitude</i> yang dicari tempatnya
<i>apikey</i>	16-digit hexadesimal	<i>API key</i> yang digunakan

Tabel 2.7: Tabel Parameter :ayanan Menemukan Transportasi Terdekat

Format kembalian layanan menemukan transportasi terdekat dapat dilihat pada *listing 2.12*.

Listing 2.12: Kode Kembalian Menemukan Lokasi Terdekat

```

1 {
2     "status": "ok" or "error"
3     "nearbytransports": [
4         [
5             "walk" or "none" or others ,
6             "walk" or vehicle_id or "none",
7             text string ,
8             decimal value
9         ],
10        [
11            "walk" or "none" or others ,
12            "walk" or vehicle_id or "none",
13            text string ,
14            decimal value
15        ],
16        ...
17    ]
18 }
```

Berikut maksud dari *listing 2.12*:

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

- Indeks ke 0 atau baris 5 pada *listing 2.12* dapat berisi "walk" atau "none" atau "others". Artinya jika "walk" berarti berjalan kaki, "none" jika rute tidak ditemukan dan "others" berarti menggunakan kendaraan.
- Indeks ke 1 atau baris 6 pada *listing 2.12* merupakan detail dari indeks 0. Artinya jika indeks 0 "walk" berarti indeks 1 harus "walk", "none" berarti indeks 1 harus "none" dan selain itu menyatakan kendaraan yang bisa dipakai untuk ditampilkan gambarnya.
- Indeks ke 2 atau baris 7 pada *listing 2.12* berisi nama kendaraan.
- Indeks ke 3 atau baris 8 pada *listing 2.12* berisi jarak dengan satuan kilometer.

BAB 3

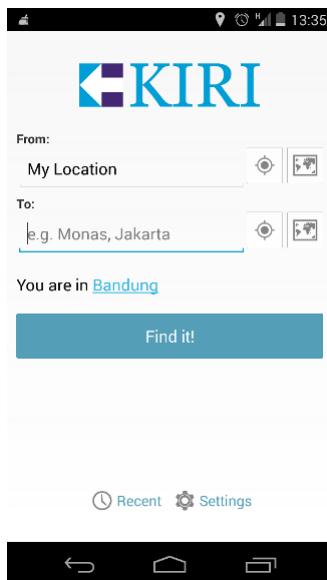
ANALISIS

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

3.1 Analisis Aplikasi Sejenis

Aplikasi sejenis yang ditemui bernama Public Transport¹. Namun aplikasi Public Transport tersebut hanya dapat dijalankan pada sistem aplikasi android. Aplikasi Public Transport ini memanfaatkan Kiri API yang penggunaannya sederhana. Di halaman awal pengguna dapat mengetik lokasi asal dan tujuan. Selain dengan mengetik pengguna juga dapat menunjuk lokasi pada peta. Setelah lokasi dipilih sistem memastikan dengan memberi daftar nama jalan dan tempat terkait. Jika sudah memilih maka sistem mengeluarkan hasil pencarian rute.

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



Gambar 3.1: Tampilan Utama Aplikasi Public Transport

Gambar 3.1 menunjukkan halaman utama aplikasi Public Transport. Di halaman ini pengguna dapat memasukan lokasi asal dan lokasi tujuan. Cara memasukan lokasi ada 2 macam yaitu dengan

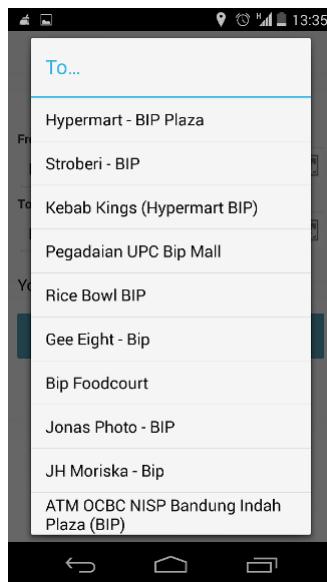
¹<https://play.google.com/store/apps/details?id=travel.kiri.smartransportapp>

mengetik dan menunjuk pada peta dengan mengetuk tombol peta. Bila pengguna ingin menunjuk lokasi pengguna berada dapat dilakukan dengan menunjuk lokasi pada peta lalu menekan tombol untuk memilih koordinat. Tersedia juga pilihan kota yang dapat dipilih oleh pengguna.



Gambar 3.2: Menunjuk Lokasi pada Peta

Gambar 3.2 jika pengguna sudah mengetahui lokasi namun tidak tahu nama lokasi. Pada halaman ini pengguna diarahkan untuk menemukan lokasi pada peta memilihnya.



Gambar 3.3: Memberikan Daftar Nama Tempat dan Nama Jalan Terkait

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

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



Gambar 3.4: Tampilan Rute Kendaraan Umum dalam Bentuk Daftar

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

Aplikasi 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 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 dibangun.

3.2.1 Kebutuhan Aplikasi

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

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

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



Gambar 3.5: Tampilan Rute Kendaraan Umum di Peta

dipilih.

Berdasarkan kebutuhan aplikasi diatas maka aplikasi yang dibangun didasarkan pada kebutuhan sebagai berikut.

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

3.2.2 Analisis Kontrol yang Dipakai

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

Untuk kontrol tata letak dibutuhkan pengaturan yang tertata rapih dan beberapa elemen dalam satu baris atau kolom. Penggunaan kontrol tata letak yang rumit tidak diharapkan dalam aplikasi karena akan membingungkan pengguna. Dari hasil pengamatan kontrol tata letak yang cocok adalah tata letak *grid*. Kontrol tata letak ini dipilih karena mudah diposisikan sesuai baris dan kolom. Kelebihan tata letak *grid* adalah menyesuaikan jika layar diputar dari posisi pemandangan ke posisi potret dan sebaliknya.

Kontrol terhadap teks juga diperlukan untuk aplikasi. Kebutuhan yang diperlukan adalah mengeluarkan potongan teks yang dapat dibaca dan tempat pengguna memasukan teks. Untuk menge luarkan teks yang dapat dilihat pengguna diperlukan penggunaan *textblock*. *textblock* digunakan untuk menampilkan tulisan "from" dan "to" pada halaman utama aplikasi. Untuk masukan pengguna, aplikasi menyediakan *textbox* sebagai tempat masukan teks. *textbox* digunakan sebagai masukan untuk lokasi asal dan lokasi tujuan.

Suatu aplikasi tentunya tidak hanya mempunyai satu halaman, aplikasi yang dibuat memiliki beberapa halaman yang mempunyai tugas berbeda. Karena hal tersebut dibutuhkan kontrol untuk berpindah dari satu halaman ke halaman lain. Kontrol yang dibutuhkan yaitu kontrol tombol. Kontrol tombol mengeksekusi *event click* yang memungkinkan pindah halaman dan melakukan perintah. Kontrol tombol digunakan untuk berpindah ke halaman peta, menemukan lokasi pengguna, dan pencarian rute. Pada halaman utama terdapat 5 buah dengan penjelasan sebagai berikut.

- Tombol "map" pada bagian from

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

- Tombol "here" pada bagian from

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

- Tombol "map" pada bagian to

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

- Tombol "here" pada bagian to

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

- Tombol find

Tombol ini akan mencari rute angkutan umum dan menampilkannya pada halaman peta.

Pada aplikasi ini ditampilkan daftar tempat dan daftar rute angkutan umum yang dipakai. Bentuk daftar digunakan karena hasil tempat dan rute angkutan umum yang ditampilkan cukup banyak jumlahnya. Bentuk daftar yang dapat dipakai di Windows Phone adalah menggunakan *ListBox*. *ListBox* menampilkan daftar tempat dan daftar rute satu per satu menurun ke bawah.

3.2.3 Analisis Terhadap Siklus Hidup Aplikasi

Aplikasi pada Windows Phone memiliki siklus hidup yang dijelaskan pada bab 2.1.8. Pengaturan aplikasi ini diatur sesuai konfigurasi awal sistem operasi Windows Phone. Tetapi pengaturan ini dapat diatur sesuai kebutuhan aplikasi. Karena di aplikasi ini terdapat keadaan yang berbeda dengan konfigurasi awal sistem operasi Windows Phone maka perlu dilakukan pengaturan ulang siklus hidup.

Saat aplikasi dijalankan, pengguna memasukkan lokasi asal dan lokasi tujuan. Setelah memasukkan lokasi pengguna lalu mencari rute. Ketika rute berhasil ditemukan maka aplikasi berada di keadaan Running. Tetapi ada kemungkinan pengguna berpindah aplikasi atau mematikan layar untuk menghemat daya. Dalam kasus tersebut sistem operasi Windows Phone menganggap

aplikasi tidak aktif dan aplikasi masuk pada keadaan *dormant*. Untuk menangani kasus tersebut maka aplikasi harus menyimpan keadaan dan informasi sesaat sebelum aplikasi menjadi tidak aktif. Penanganan yang digunakan adalah menyimpan keadaan sebelumnya di memori.

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

Ketika aplikasi sudah berada pada keadaan *Dormant* atau *Tombstoned*, pengguna masih dapat memulihkan keadaan aplikasi saat aplikasi berada di keadaan *Running* sebelumnya. Penanganan yang dilakukan untuk hal tersebut adalah menggunakan *method* *OnNavigatedTo()*. Menggunakan *method* tersebut bertujuan memulihkan informasi halaman pada keadaan *Running* sebelumnya.

3.2.4 Analisis Peta

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

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

Untuk cara pandang peta terdapat 4 pandangan yang disediakan peta di Windows Phone yaitu *Road*, *Aerial*, *Hybrid*, dan *Terrain*. Aplikasi ini adalah aplikasi pencari rute dan pandangan lebih banyak diarahkan ke jalanan perkotaan. Kebutuhan pengguna adalah nama jalan, kondisi jalan, dan kondisi sekitar. Dari dasar pandangan tersebut pandangan yang dipilih untuk aplikasi ini adalah *Road*. Tambahan setelah mengatur pandangan peta yaitu mengatur warna dan menggunakan mode warna terang sesuai bawaan peta di Windows Phone.

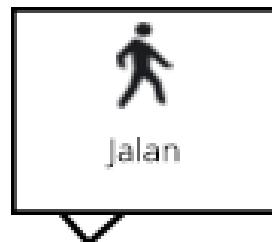
Tampilan awal peta di Windows Phone akan mengeluarkan peta dengan pandangan dunia. Karena aplikasi pencarian rute ini masih terbatas di Pulau Jawa, Indonesia terutama Jawa Barat maka tingkat *zoom* harus diatur agar mengikuti lokasi pengguna dan di satu daerah saja. Jika pengguna berada di daerah Bandung maka tingkat *zoom* pada peta disesuaikan pada daerah tersebut. Tingkat *zoom* dapat dapat diatur dari kode dan XAML. Tingkat *zoom* yang digunakan adalah 14. Tingkat *zoom* 14 menampilkan satu kota dengan jelas.

Di setiap kota ada satu angkutan umum yang banyak dipakai yaitu angkutan kota(angkot). Namun bagi yang baru pertama mengunjungi suatu daerah dan mencari angkot mungkin akan mengalami 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 peta tempat gambar tersebut. Untuk hal tersebut dimanfaatkan *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 digunakan untuk aplikasi yaitu dengan memakai Kiri API. Kiri API memberikan kembalian berupa titik-titik rute perjalanan dari lokasi asal ke lokasi tujuan. Karena hal itu aplikasi harus menggambar rute tersebut sesuai jalan pada peta. Untuk hal tersebut digunakan *polyline* pada Windows Phone untuk menggambarnya. *Polyline* yang digambar harus terlihat dengan jelas dan diberi warna yang kontras dengan tampilan peta. Warna *polyline* yang dipilih adalah merah dengan ketebalan 4.

3.2.5 Analisis Pemanfaatan Sumber Data

Aplikasi yang dibuat memanfaatkan sumber data dari luar. Sumber data yang didapatkan 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 perlu memanfaatkan kelas *HttpClient.Method* yang digunakan adalah *GetStringAsync()*. *Method* ini mengirimkan permintaan dengan parameter URI dan hasil yang dikembalikan memiliki tipe data *string*. Karena *method* ini mengembalikan hasil dalam tipe data *string* maka mudah disesuaikan dengan kebutuhan tugas akhir ini. Selanjutnya agar data dalam bentuk *string* tersebut dapat dimanfaatkan maka diperlukan proses *Deserialization*. Proses *Deserialization* dimaksudkan untuk mengubah bentuk *string* ke objek. Untuk melakukan *Deserialization* perlu memanfaatkan *library Json.NET*. Aplikasi memanfaatkan

library Json.NET karena memiliki performa yang baik. Untuk hal tersebut aplikasi yang dibuat perlu menggunakan *method* DeserializeObject().

3.2.6 Analisis Kiri API

Kiri API menyediakan 2 jenis permintaan yaitu *POST* dan *GET*. Dalam tugas akhir ini digunakan protokol *GET* untuk melakukan permintaan. Permintaan **GET** dipilih karena dalam tugas akhir ini aplikasi banyak mendapatkan data dan tidak ada data sensitif yang dikirimkan. Untuk hal ini maka mengirim ke URI <http://kiri.travel/handle.php>.

Untuk setiap permintaan terhadap Kiri API dibutuhkan *API key*. Kegunaan *API key* adalah password untuk mengakses Kiri API. *API key* dapat didapatkan di <dev.kiri.travel>. *API key* yang digunakan pada tugas akhir ini adalah 97A7A1157A05E***.

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

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

- *version*: 2

Karena acuan yang digunakan adalah Kiri API versi 2 maka di parameter *version* aplikasi menggunakan versi 2.

- *mode*: "searchplace"

Mode "searchplace" digunakan untuk mencari lokasi terkait.

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

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

- *querystring*: merupakan kata kunci lokasi

- *apikey*: 16 digit heksadesimal

Format layanan yang dikirim melalui URL adalah kiri.travel/handle.php?version=2&mode=searchplace®ion=cgk/bdo/sub&querystring="string"&apikey=97A7A1157A05E***.

Percobaan mencari lokasi bip dari kata kunci "bip" yang berada di bandung. Layanan dikirimkan ke URL kiri.travel/handle.php. Berikut format layanan yang dikirimkan:

http://kiri.travel/handle.php?version=2&mode=searchplace®ion=bdo&querystring=bip&apikey=97A7A1157A05E***

Berikut hasil kembalian dari Kiri API:

Listing 3.1: Kode Kembalian dari Pencarian Rute

```

1 {
2     "status ":"ok",
3     "searchresult ":[
4         {
5             "placename ":"Hypermart - BIP Plaza",
6             "location ":"-6.90864,107.61108"
7         },
8         {
9             "placename ":"Stroberi - BIP",
10            "location ":"-6.90834,107.61115"
11        },
12        {
13            "placename ":"Kebab Kings (Hypermart BIP)",
14            "location ":"-6.91503,107.61017"
15        },
16        {
17            "placename ":"Pegadaian UPC Bip Mall",
18            "location ":"-6.90916,107.61052"
19        },
20        {
21            "placename ":"Rice Bowl BIP",
22            "location ":"-6.90873,107.61088"
23        },
24        {
25            "placename ":"Gee Eight - Bip",
26            "location ":"-6.90817,107.61080"
27        },
28        {
29            "placename ":"Jonas Photo - BIP",
30            "location ":"-6.91066,107.61016"
31        },
32        {
33            "placename ":"Bip Foodcourt",
34            "location ":"-6.91081,107.61015"
35        },
36        {
37            "placename ":"Mister Baso BIP",
38            "location ":"-6.90348,107.61709"
39        },
40        {
41            "placename ":"JH Moriska - Bip",
42            "location ":"-6.90868,107.61070"
43        }
44    ],
45    "attributions ":null
46 }
```

Hasil dari kembalian berupa kumpulan *placename* dan *location*. Hasil tersebut ditampung aplikasi, namun yang ditampilkan ke pengguna hanya *placename*. Menampilkan *location* tidak efektif karena akan membingungkan pengguna. Dari percobaan yang dilakukan, nilai dari *attributions* selalu bernilai "null". Karena hal tersebut maka nilai *attributions* diabaikan.

Layanan lainnya yang dimanfaatkan adalah layanan penentuan rute. Pemanfaatan layanan penentuan rute digunakan mendapatkan langkah-langkah yang harus ditempuh pengguna untuk menuju lokasi tujuan dari lokasi asal. Pemanfaatan layanan ini yaitu dengan permintaan *GET* melalui protokol HTTP. Berikut parameter yang harus dikirim:

- *version*: 2

Karena acuan adalah Kiri API versi 2 maka pada parameter versi ditulis 2.

- *mode*: "findroute"

Mode "findroute" digunakan untuk mendapatkan langkah yang harus ditempuh menuju lokasi

tujuan.

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

Karena aplikasi ini bertujuan untuk memudahkan pemakaian transportasi di Indonesia maka diputuskan untuk menggunakan bahasa Indonesia.

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

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

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

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

- *presentation*: "mobile" untuk perangkat *mobile* dan "desktop" untuk komputer.

Karena aplikasi ini dirancang untuk Windows Phone 8, presentasi yang dipilih adalah "desktop". Pemilihan ini didasarkan karena deskripsi yang ditampilkan tidak ada tulisan "image from" dan "image to", selain itu presentasi "desktop" juga memungkinkan rute alternatif.

- *apikey*: 16 digit heksadesimal.

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

Percobaan pernentuan rute menuju Jalan Merdeka dari Jalan Ciumbuleuit. Layanan dikirimkan ke URL <http://kiri.travel/handle.php?version=2&mode=findroute&locale=id&start=-6.8747337,107.6048829&finish=-6.9114646,107.6104887&presentation=mobile&apikey=97A7A1157A05ED6F>.

Berikut hasil kembalian dari Kiri API:

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

```

1 {
2     "status ":"ok",
3     "routingresults ":[
4         {
5             "steps ":[
6                 [
7                     "walk",
8                     "walk",
9                     ["-6.8747337,107.6048829","-6.87445,107.60465"],
10                    "Jalan dari lokasi mulai Anda \%fromicon ke Jalan Ciumbuleuit \%toicon sejauh kurang
11                    lebih 41 meter.",
12                    null,
13                    null
14                ],
15                [
16                    "angkot",
17                    "ciumbuleuitsthalilurus",
18                    ["-6.87445,107.60465","-6.87541,107.60443","-6.87637,107.60421","-6.87734,107.60400",
19                    "-6.87830,107.60378","-6.87926,107.60356","-6.87926,107.60356","-6.87963,107.60352",
20                    "-6.87978,107.60352","-6.88093,107.60392","-6.88209,107.60433","-6.88209,107.60433",
21                    "-6.88328,107.60490","-6.88328,107.60490","-6.88347,107.60481","-6.88452,107.60459",
22                    "-6.88556,107.60436","-6.88660,107.60413","-6.88764,107.60390","-6.88764,107.60391",
23                    "-6.88782,107.60392","-6.88887,107.60404","-6.88991,107.60416","-6.88991,107.60416",
24                    "-6.89161,107.60428","-6.89161,107.60428","-6.89166,107.60421","-6.89275,107.60424",
25                    "-6.89275,107.60424","-6.89405,107.60408","-6.89405,107.60408","-6.89496,107.60400"],
```

```

25      "Naik angkot Ciumbuleuit - St. Hall (lurus) di Jalan Ciumbuleuit \%fromicon, dan turun
26      di Jalan Cihampelas \%toicon kurang lebih setelah 3,3 kilometer.", 
27      null,
28      https:////angkot.web.id/go/route/640?ref=kiri
29  ],
30  [
31      "walk",
32      "walk",
33      [ "-6.90424,107.60433", "-6.90429,107.60440"],
34      "Jalan dari Jalan Cihampelas \%fromicon ke Jalan Abdul Rivai \%toicon sejauh kurang
35      lebih 10 meter.",
36      null,
37      null
38  ],
39  [
40      "angkot",
41      "kalapaledeng",
42      [ "-6.89501,107.60403", "-6.89562,107.60398", "-6.89623,107.60395", "-6.89732,107.60401",
43      "-6.89732,107.60401", "-6.89882,107.60414", "-6.89882,107.60414", "-6.89969,107.60418",
44      "-6.90071,107.60426", "-6.90173,107.60433", "-6.90173,107.60433", "-6.90297,107.60437",
45      "-6.90420,107.60440", "-6.90426,107.60440", "-6.90426,107.60456", "-6.90422,107.60481",
46      "-6.90399,107.60546", "-6.90406,107.60617", "-6.90454,107.60697", "-6.90454,107.60697",
47      "-6.90512,107.60745", "-6.90618,107.60778", "-6.90618,107.60778", "-6.90643,107.60787",
48      "-6.90651,107.60807", "-6.90675,107.60914", "-6.90675,107.60914", "-6.90694,107.60939",
49      "-6.90723,107.60939", "-6.90891,107.60943", "-6.90891,107.60943", "-6.90909,107.60934",
50      "-6.90914,107.60857", "-6.90933,107.60846", "-6.91021,107.60887", "-6.91021,107.60887",
51      "-6.91030,107.60897", "-6.91028,107.60927", "-6.90986,107.61040", "-6.90986,107.61040"],
52      "Naik angkot Kalapa - Ledeng di Jalan Abdul Rivai \%fromicon, dan turun di Jalan Aceh
53      \%toicon kurang lebih setelah 1,1 kilometer.",
54      https:////angkot.web.id/go/route/156?ref=kiri"
55  ],
56  [
57      "walk",
58      "walk",
59      [ "-6.90986,107.61040", "-6.9114646,107.6104887"],
60      "Walk about 178 meter from Jalan Aceh \%fromicon to your destination \%toicon.",
61      null,
62      null
63  ],
64  ],
65 }

```

Format layanan yang dikirim dengan *presentation desktop* adalah

<http://kiri.travel/handle.php?version=2&mode=findroute&locale=id&start=-6.8747337,107.6048829&finish=-6.9114646,107.6104887&presentation=desktop&apikey=97A7A1157A05ED6F>.

Berikut hasil kembalian dari Kiri API:

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

```

1 {
2     "status": "ok",
3     "routingresults": [
4         {
5             "steps": [
6                 [
7                     "walk",
8                     "walk",
9                     [ "-6.8747337,107.6048829", "-6.87445,107.60464"],
10                    "Jalan dari lokasi mulai Anda ke Jalan Ciumbuleuit sejauh kurang lebih 41 meter.",
11                    null,
12                    null
13                ],
14                [
15                    "angkot",
16                    "ciumbuleuitsthallurus",
17                    [ "-6.87445,107.60464", "-6.87541,107.60443", "-6.87541,107.60443", "-6.87637,107.60422",
18                    "-6.87637,107.60422", "-6.87734,107.60400", "-6.87734,107.60400", "-6.87830,107.60378",
19                    "-6.87830,107.60378", "-6.87926,107.60356", "-6.87926,107.60356", "-6.87926,107.60356",
20                    "-6.87963,107.60352", "-6.87978,107.60352", "-6.88093,107.60393", "-6.88093,107.60393",
21                    "-6.88209,107.60433", "-6.88209,107.60433", "-6.88209,107.60433", "-6.88328,107.60490",
22                    "-6.88328,107.60490", "-6.88328,107.60490", "-6.88347,107.60481", "-6.88452,107.60458",
23                    "-6.88452,107.60458", "-6.88556,107.60435", "-6.88556,107.60435", "-6.88660,107.60413",
24                    "-6.88660,107.60413", "-6.88764,107.60390", "-6.88764,107.60390", "-6.88764,107.60390",
25                    "-6.88782,107.60392", "-6.88887,107.60404", "-6.88887,107.60404", "-6.88991,107.60416",
26                    "-6.88991,107.60416", "-6.88991,107.60416", "-6.89161,107.60428", "-6.89161,107.60428",
27                    "-6.89161,107.60428", "-6.89166,107.60420", "-6.89275,107.60424", "-6.89275,107.60424"

```

```

28      " -6.89275,107.60424", " -6.89405,107.60407", " -6.89405,107.60407", " -6.89405,107.60407",
29      " -6.89496,107.60400", " -6.89496,107.60400", " -6.89586,107.60392", " -6.89586,107.60392",
30      " -6.89586,107.60392", " -6.89759,107.60397", " -6.89759,107.60397", " -6.89759,107.60397",
31      " -6.89895,107.60406", " -6.89895,107.60406", " -6.89895,107.60406", " -6.89970,107.60413",
32      " -6.89999,107.60416", " -6.90114,107.60426", " -6.90114,107.60426", " -6.90114,107.60426",
33      " -6.90218,107.60428", " -6.90218,107.60428", " -6.90321,107.60431", " -6.90321,107.60431",
34      " -6.90424,107.60433",
35      "Naik angkot Ciumbuleuit – St. Hall (lurus) di Jalan Ciumbuleuit , dan turun di Jalan
36      Cihampelas kurang lebih setelah 3,3 kilometer.", null,
37      "https:// angkot.web.id/go/route/640?ref=kiri"
38  ],
39  [
40      "walk",
41      "walk",
42      [" -6.90424,107.60433", " -6.90429,107.60440"],
43      "Jalan dari Jalan Cihampelas ke Jalan Abdul Rivai sejauh kurang lebih 10 meter.", null,
44      null
45  ],
46  [
47      "angkot",
48      "kalapaledeng",
49      [" -6.90429,107.60440", " -6.90434,107.60490", " -6.90398,107.60558", " -6.90417,107.60619",
50      " -6.90465,107.60702", " -6.90465,107.60702", " -6.90465,107.60702", " -6.90521,107.60748",
51      " -6.90600,107.60771", " -6.90646,107.60775", " -6.90755,107.60760", " -6.90755,107.60760",
52      " -6.90755,107.60760", " -6.90866,107.60789", " -6.90866,107.60789", " -6.90866,107.60789",
53      " -6.90911,107.60826", " -6.91034,107.60892", " -6.91034,107.60892", " -6.91034,107.60892",
54      " -6.91007,107.60985"],
55      "Naik angkot Kalapa – Ledeng di Jalan Abdul Rivai , dan turun di Jalan Aceh kurang
56      lebih setelah 1,1 kilometer.", null,
57      "https:// angkot.web.id/go/route/156?ref=kiri"
58  ],
59  [
60      "walk",
61      "walk",
62      [" -6.91007,107.60985", " -6.9114646,107.6104887"],
63      "Jalan dari Jalan Aceh ke tujuan akhir Anda sejauh kurang lebih 171 meter.", null,
64      null
65  ],
66  ],
67  ],
68  "traveltime ":"30 menit"}]}

```

Setiap langkah kembalian dari Kiri API ditampung dalam elemen *array*. Untuk keterangan dan jenis angkutan umum aplikasi tampilkan dalam bentuk *pushpin* pada peta atau daftar. Sedangkan untuk titik-titik kordinat digambarkan pada peta. Dari analisa didapatkan bahwa setiap langkah menunjukkan perpindahan angkutan umum yang dipakai, berpindah dari angkutan umum atau jalan, dan dari jalan untuk menaiki angkutan umum. Keterangan yang ditambahkan harus berada antara setiap *steps* tersebut. Dari analisa didapatkan kata "%fromicon" dan "%toicon" yang tidak menunjukkan sesuatu untuk keluaran jika melakukan permintaan dengan *presentation mobile*. Karena hal tersebut mode *presentation* yang digunakan adalah *desktop*. Gambar angkutan kota dan gambar jalan yang sudah disediakan dari Kiri diambil dengan memanfaatkan URL yang disediakan.

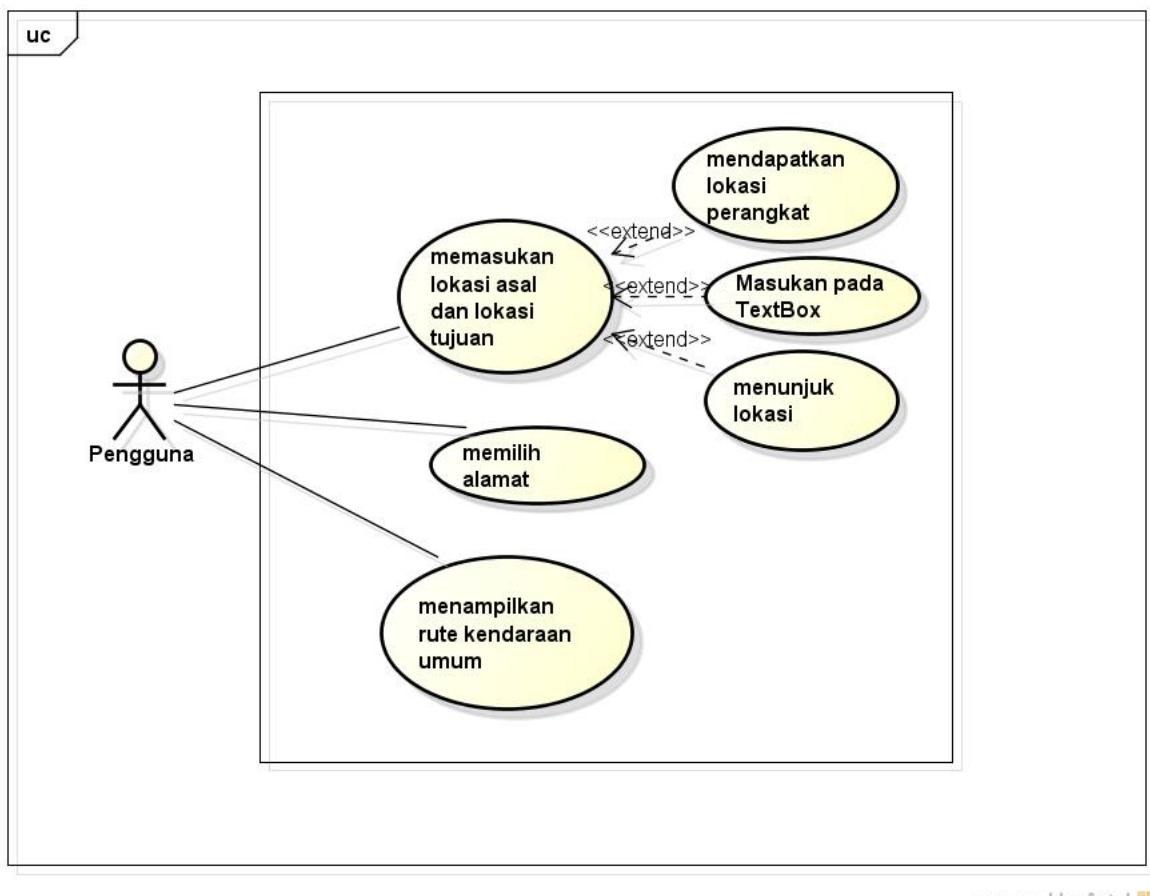
3.2.7 Diagram *Use Case* dan Skenario

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

- Mendapatkan lokasi pengguna berada.
- Memasukan lokasi asal dan lokasi tujuan.
- Menunjuk langsung lokasi asal dan tujuan pada peta.
- Memilih alamat atau tempat dari pilihan yang disediakan.

- Menampilkan rute kendaraan umum dalam bentuk titik dan *pushpin* pada peta atau bentuk daftar dari tempat asal ke tempat tujuan.

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



powered by Astah

Gambar 3.8: Diagram *Use Case*

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

Nama	Mendapatkan lokasi perangkat
Aktor	Pengguna
Deskripsi	Mendapatkan lokasi perangkat berada
Kondisi awal	textbox masih kosong dan pengguna menekan tombol lokasi
Kondisi akhir	Lokasi ditemukan dan textbox berisi "here"
Skenario utama	Pengguna menekan tombol lalu perangkat mencari lokasi perangkat dan textbox 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

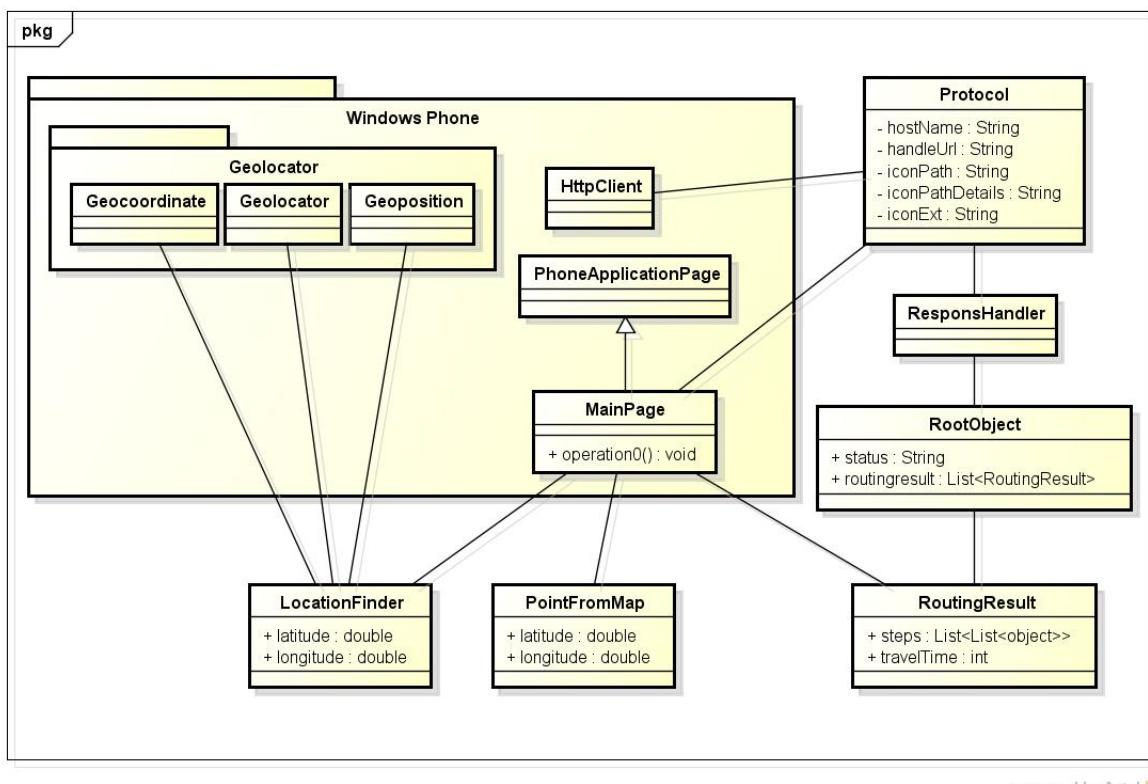
Tabel 3.4: Skenario Memilih Alamat

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 lalu istem akan memproses data masukan. Sistem akan mengembalikan hasil rute kendaraan umum pada peta dan dalam bentuk daftar
Ekspsi	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 ditangani oleh kelas ini.

- Kelas ResponsHandler

Merupakan kelas yang menangani masukan dari pemanggilan layanan.

- Kelas RootObject

Merupakan kelas untuk menampung status dan daftar dari layanan *routing* Kiri API. Hasil kembalian dipisahkan di kelas ini untuk selanjutnya ditambahkan di kelas *RoutingResult*.

- Kelas RoutingResult

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

- Kelas PointFromMap

Merupakan kelas yang dapat mengetahui lokasi yang ditunjuk pengguna pada peta. Kelas ini menyimpan lokasi yang ditunjuk pengguna dalam bentuk *latitude* dan *longitude*.

- Kelas LocationFinder

Merupakan kelas yang digunakan untuk mencari lokasi. Kelas ini memanfaatkan kelas *Geocoordinate* untuk mendapatkan lokasi. Setelah lokasi didapatkan dalam bentuk kelas *Geoposition* maka diubah ke *latitude* dan *longitude*.

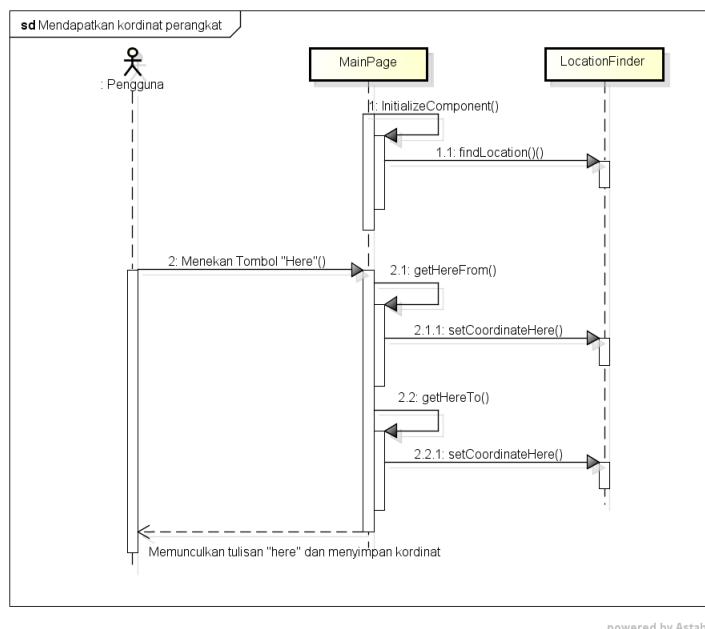
BAB 4

PERANCANGAN

Pada bab 4 dibahas mengenai perancangan aplikasi mulai dari diagram *sequence*, diagram kelas secara rinci, deskripsi artibut dan *method* dari setiap kelas, dan perancangan antarmuka.

4.1 Diagram Sequence

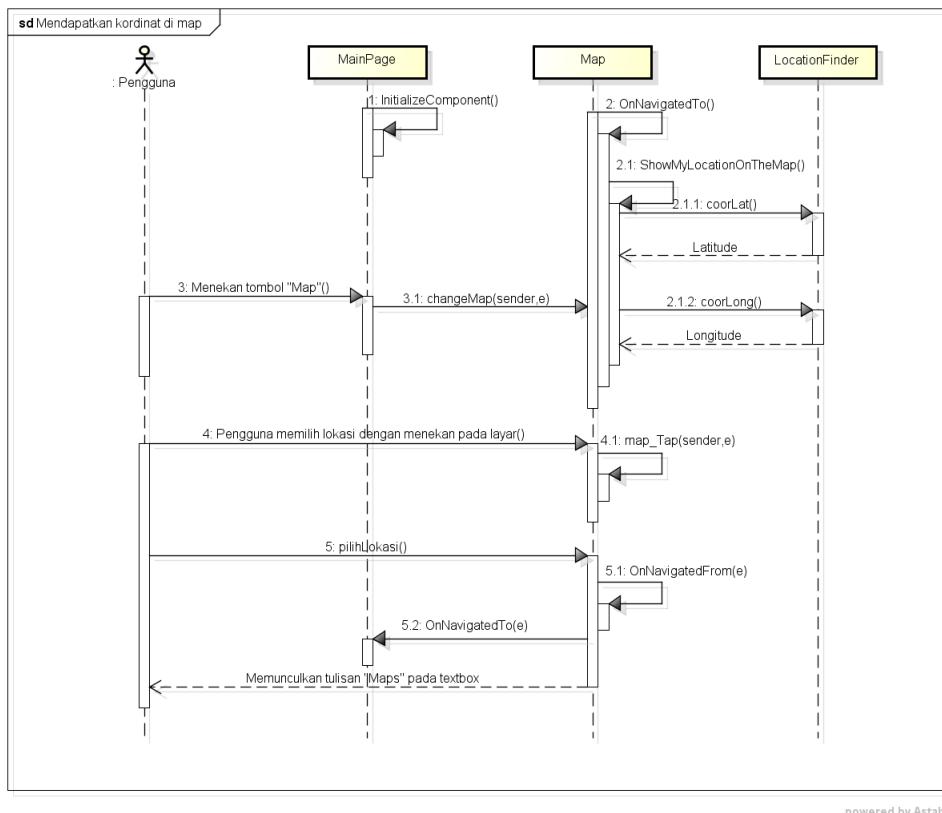
Diagram *sequence* merupakan diagram yang menggambarkan interaksi antar objek dalam suatu skenario. Gambar diagram *sequence* dapat dilihat pada gambar 4.1 sampai 4.3.



Gambar 4.1: Diagram *sequence* Mendapatkan Kordinat Perangkat

Diagram 4.1 merupakan diagram *sequence* untuk penentuan lokasi asal/tujuan dengan masukan lokasi perangkat. Diagram menunjukkan bahwa setelah aplikasi dibuka maka aplikasi akan mencari dahulu lokasi perangkat dengan memanfaatkan kelas LocationFinder. Lalu setelah aplikasi terbuka jika pengguna ingin memilih lokasi tersebut sebagai lokasi asal maka pengguna harus menekan tombol "here". Setelah tombol "here" ditekan maka kelas MainPage akan mengambil nilai *Latitude* dan nilai *Longitude* dari kelas LocatonFinder. Setelah lokasi didapatkan maka aplikasi memunculkan tulisan "here" pada *textbox* di kelas MainPage.

Diagram 4.2 merupakan diagram *sequence* untuk memilih lokasi. Diagram menunjukkan

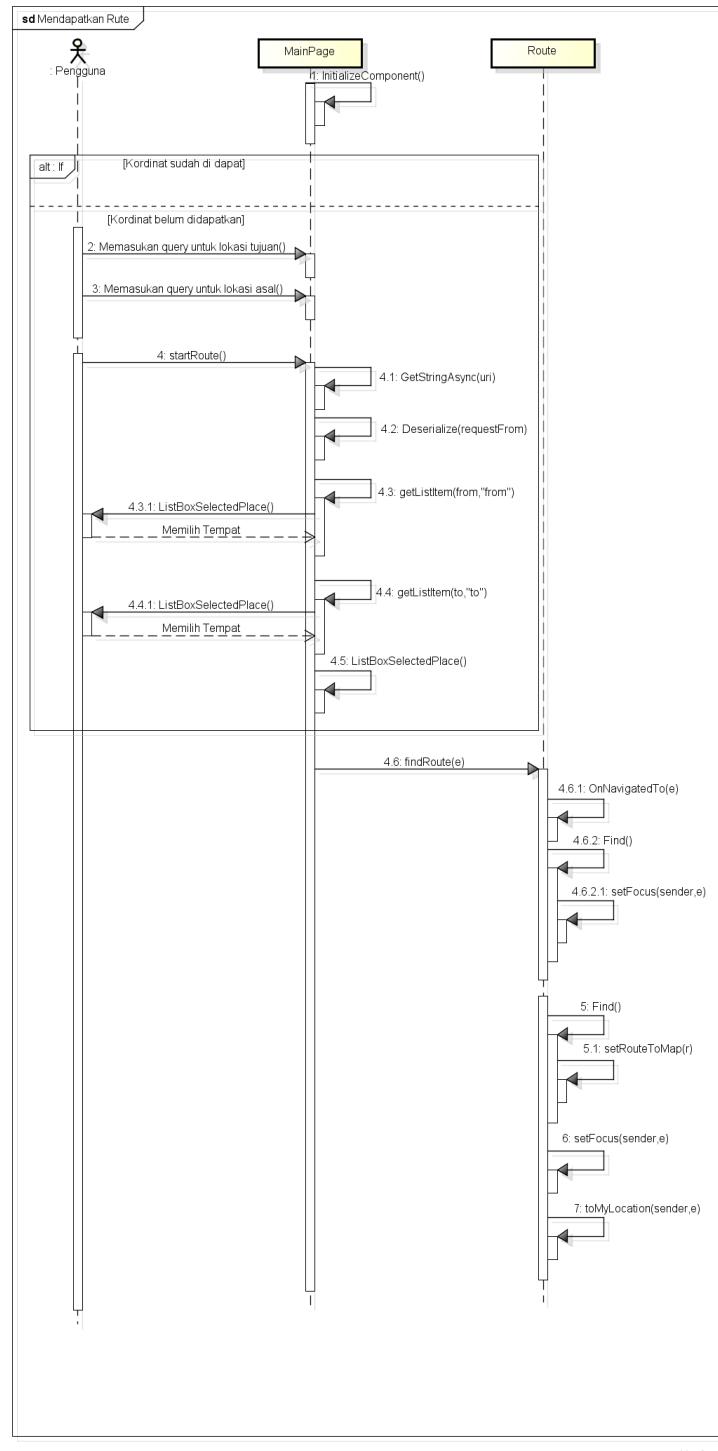


Gambar 4.2: Diagram *sequence* Mendapatkan Kordinat pada Peta

bahwa setelah aplikasi dibuka maka pengguna dapat menekan tombol "map". Setelah tombol "map" ditekan maka halaman akan dialihkan ke kelas Map. Saat kelas Map terbuka maka lokasi yang ditunjukkan adalah lokasi dimana perangkat berada. Untuk mengetahui lokasi kelas Map mengambil kordinat *latitude* dan *longitude* dari kelas LocationFinder. Di kelas "Map" pengguna dapat memilih lokasi dengan memilih lokasi pada peta dan memanggil *method map_tap()*, lalu setelah pengguna memilih tempat pengguna akan menekan tombol Pilih Lokasi yang akan memanggil *method pilih-Lokasi()*. Lokasi yang dipilih pengguna akan 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 koordinat, namun jika pengguna memasukkan kata kunci perlu didapat koordinat *latitude* dan *longitude* dari kata kunci tersebut. Jika masukan yang didapat berupa kata kunci maka akan dilakukan pemeriksaan apakah koordinat 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 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 hasil-

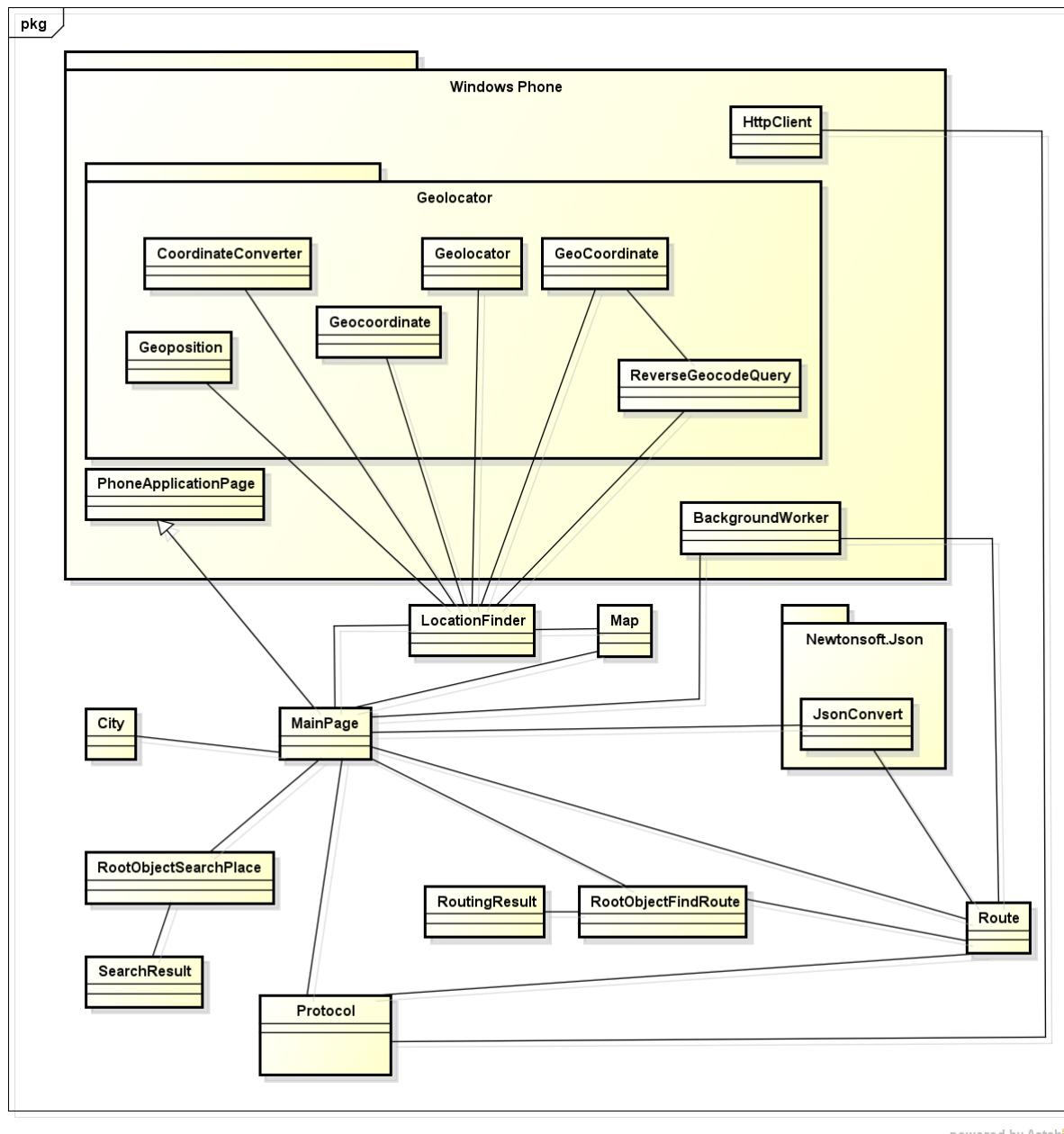
nya. Kelas Route akan memanggil *method OnNavigatedTo()* yang bertujuan untuk mendapatkan lokasi asal dan lokasi tujuan. Pada kelas Route, kelas Route akan memanggil *method Find()* lalu mengembalikan rute yang ditemukan kepada pengguna.



Gambar 4.3: Diagram *sequence* Mendapatkan Rute

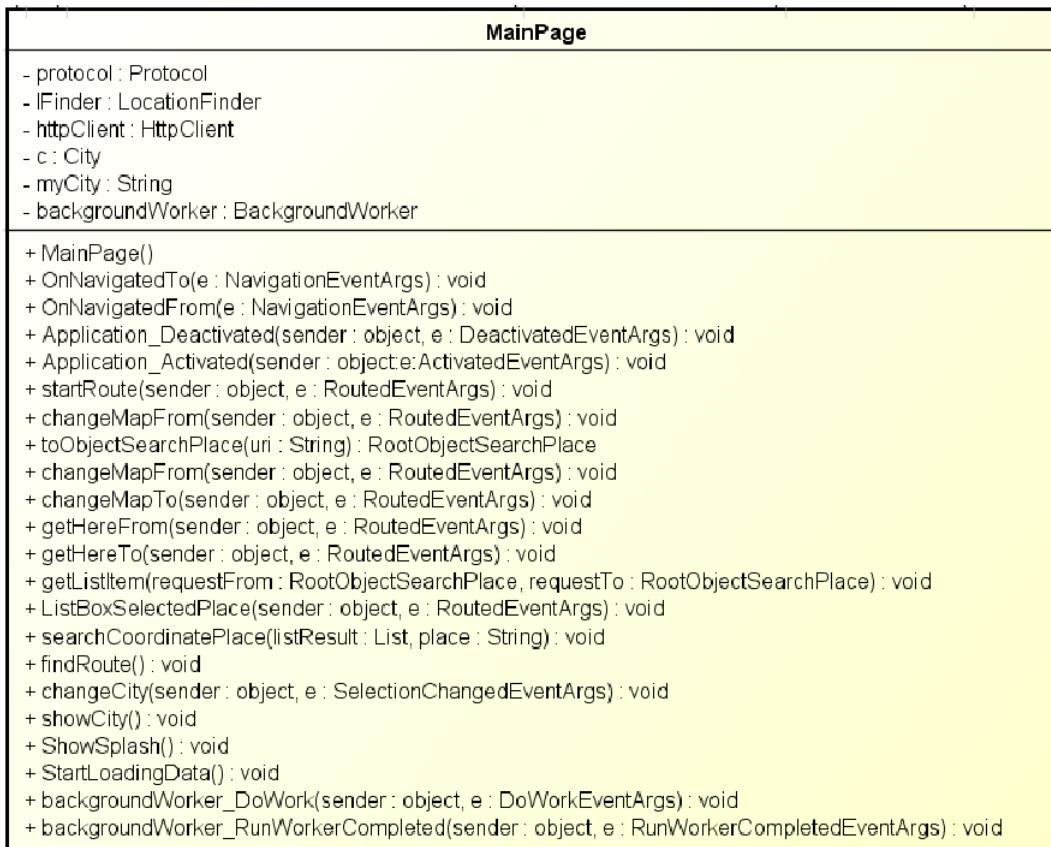
4.2 Perancangan Kelas

Pada sub-bab ini 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, penyajian gambar diagram kelas yang dapat dilihat pada gambar 4.4.

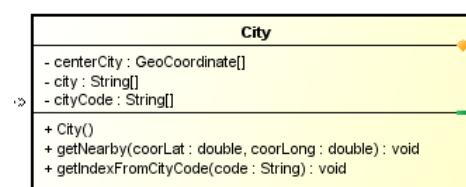


Gambar 4.4: Diagram Kelas

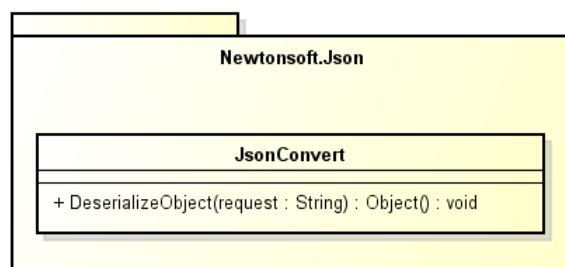
Berikut penjelasan kelas diagram secara rinci dengan setiap kelas disertai atribut dan *method*.



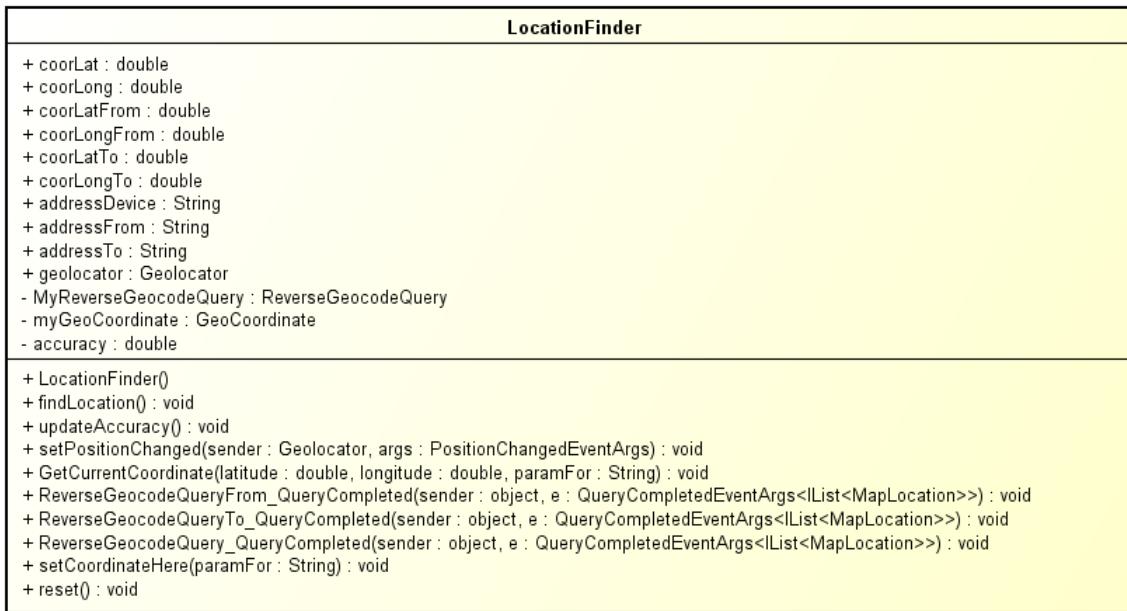
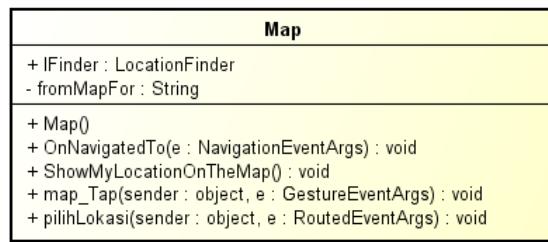
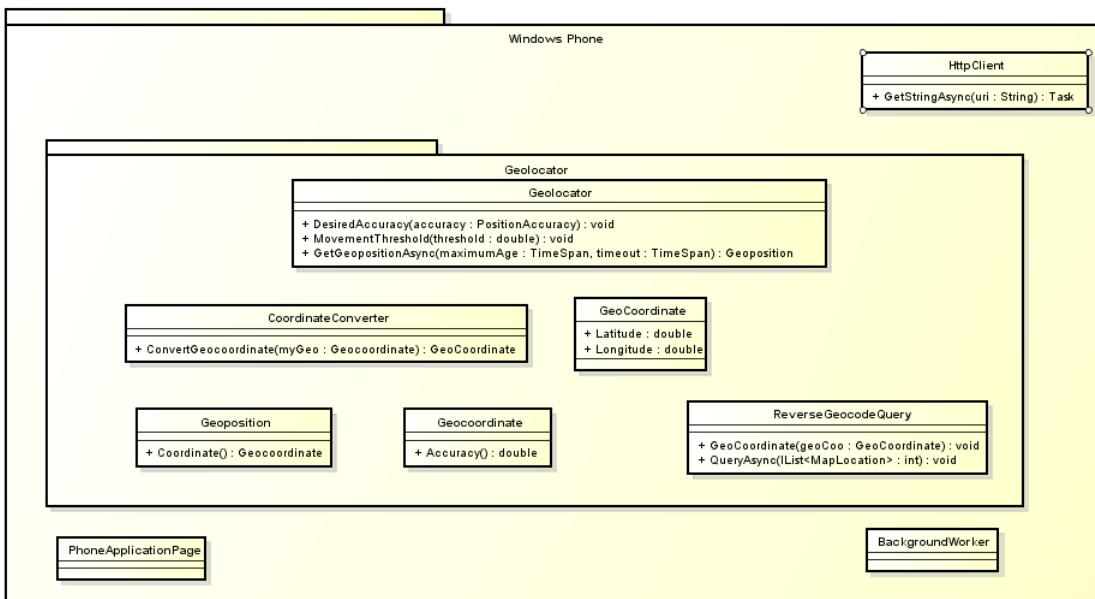
Gambar 4.5: Kelas Mainpage (sub-bab 4.2.1)

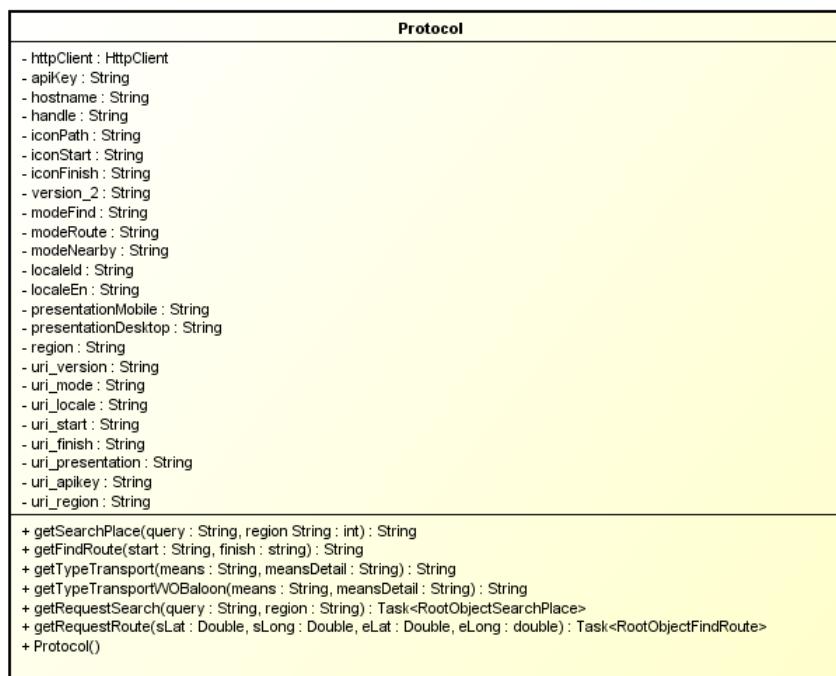
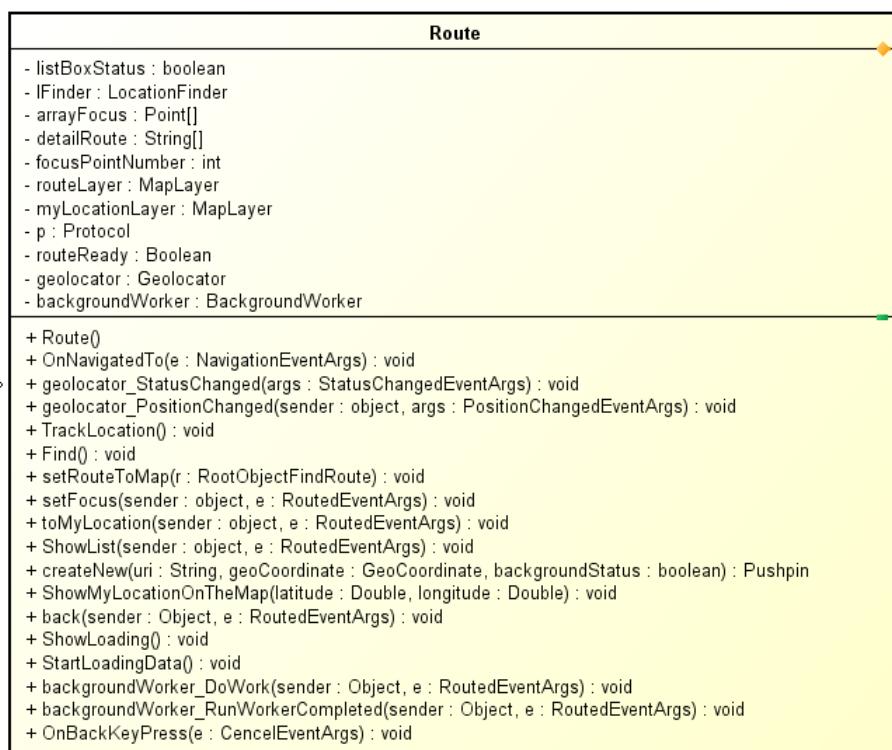


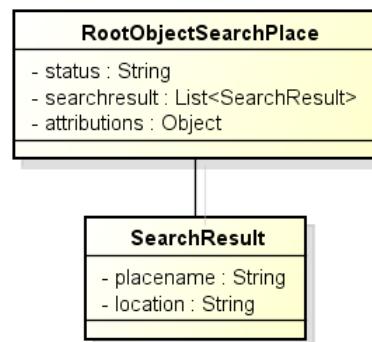
Gambar 4.6: Kelas City (sub-bab 4.2.3)



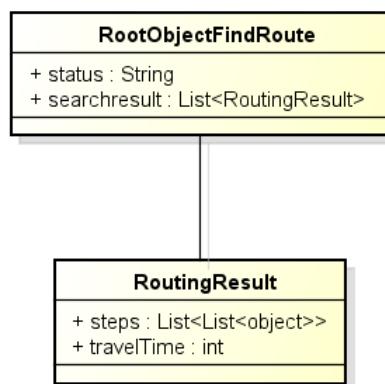
Gambar 4.7: Kelas JsonConvert (sub-bab 4.2.11)

Gambar 4.8: Kelas LocationFinder (sub-bab [4.2.8](#))Gambar 4.9: Kelas Map (sub-bab [4.2.1](#))Gambar 4.10: *Package WindowsPhone* (sub-bab [4.2.4](#) sampai [4.2.7](#))

Gambar 4.11: Kelas Protocol (sub-bab [4.2.12](#))Gambar 4.12: Kelas Route (sub-bab [4.2.17](#))



Gambar 4.13: Kelas SearchPlace (sub-bab [4.2.13](#) dan [4.2.14](#))



Gambar 4.14: Kelas FindRoute (sub-bab [4.2.15](#) dan [4.2.16](#))

4.2.1 Kelas *PhoneApplicationPage*

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

4.2.2 Kelas *MainPage*

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

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

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

1. Konstruktor MainPage digunakan untuk memuat komponen yang ada di halaman MainPage dan mendapatkan kota terdekat dari lokasi perangkat.
2. *method* OnNavigatedTo digunakan untuk mendapatkan lokasi dari peta dan mendapatkan objek LocationFinder. *Method* ini secara otomatis dipanggil jika pengguna kembali ke kelas MainPage. *Method* ini memiliki parameter NavigationEventArgs.
3. *Method* Application_Deactivated digunakan untuk menyimpan *state* objek saat meninggalkan aplikasi. *method* ini memiliki parameter object dan DeactivatedEventArgs.
4. *Method* Application_Activated digunakan untuk mengambil *state* objek saat kembali ke aplikasi. *method* ini memiliki parameter object dan ActivatedEventArgs.
5. *Method* startRoute digunakan untuk mendapatkan masukan pengguna. Jika masukan didapat dari peta atau lokasi perangkat berarti sudah dalam kordinat, namun jika masukan didapat dari *query* maka akan dicari lokasi yang terkait sesuai *query* tersebut. Lokasi terkait yang didapatkan lalu dikembalikan ke pengguna untuk dipilih. Setelah pengguna lokasi dalam bentuk kordinat didapatkan maka kelas akan diarahkan ke kelas Route. *method* ini memiliki parameter objek tombol dan event.

6. *Method* changeMapFrom digunakan untuk berpindah ke halaman mapFrom. *Method* ini memiliki parameter objek tombol dan event.
7. *Method* changeMapTo digunakan untuk berpindah ke halaman mapTo. *method* ini memiliki parameter objek tombol dan event.
8. *Method* getHereFrom digunakan untuk mendapatkan kordinat perangkat lalu menyimpan nilainya di kordinat asal di kelas LocationFinder dan menulisakan "Here" pada *TextBox* lokasi asal. *Method* ini memiliki parameter objek tombol dan event.
9. *Method* getHereTo digunakan untuk mendapatkan kordinat perangkat lalu menyimpan nilainya di kordinat tujuan di kelas LocationFinder dan menulisakan "Here" pada *TextBox* lokasi tujuan. *Method* ini memiliki parameter objek tombol dan event.
10. *Method* getListItem digunakan untuk membuat *listBox* lalu menampilkan ke pengguna. *method* ini memiliki parameter RootObjectSearchPlace dan string yang menunjukan *list* yang ditampilkan untuk lokasi asal dan lokasi tujuan.
11. *Method* ListBoxSelectedPlace digunakan untuk mendapatkan tempat asal yang dipilih pengguna. *method* ini memiliki parameter objek dan *event SelectionChangedEventArgs*.
12. *Method* searchCoordinatePlace digunakan untuk mencari kordinat dari tempat pilihan pengguna. *method* ini memiliki parameter *ListBox* dan tempat yang dipilih dalam bentuk *string*.
13. *Method* findRoute digunakan untuk berpindah ke kelas Route jika lokasi asal dan lokasi tujuan sudah ditentukan.
14. *Method* changeCity digunakan untuk mengubah kota tujuan dari pencarian. *method* ini memiliki parameter objek dan *event SelectionChangedEventArgs*.
15. *Method* showCity digunakan untuk mencari kota yang paling dekat dengan lokasi perangkat.
16. *Method* ShowSplash digunakan untuk menampilkan tampilan awal untuk proses inisialisasi aplikasi.
17. *Method* StartLoadingData digunakan untuk memanggil BackgroundWorker. BackgroundWorker digunakan untuk melakukan aksi di belakang layar.
18. *Method* backgroundWorker_DoWork digunakan untuk melakukan pemanggilan aksi di belakang layar. *method* ini memiliki parameter objek dan *DoWorkEventArgs*.
19. *Method* backgroundWorker_RunWorkerCompleted digunakan untuk melakukan pemanggilan saat BackgroundWorker selesai melakukan tugasnya. *method* ini memiliki parameter objek dan *RunWorkerCompletedEventArgs*.

4.2.3 Kelas *City*

City merupakan kelas yang menyimpan kota-kota yang mendukung pencarian rute kendaraam umum dengan bantuan Kiri. Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

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

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

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

4.2.4 Kelas *BackgroundWorker*

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

1. *Event* DoWork
2. *Event* RunWorkerCompleted

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

1. *Method* RunWorkerAsync() digunakan untuk memulai operasi di belakang layar.

4.2.5 Kelas *Geocoordinate*

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

4.2.6 Kelas *Geolocator*

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

4.2.7 Kelas *Geoposition*

Geoposition merupakan kelas yang menampung lokasi sesuai kembalian *Geolocator*.

4.2.8 Kelas *LocationFinder*

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

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

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

1. Konstruktor LocationFinder berfungsi mengatur atribut geolocator dan mencari lokasi perangkat.
2. *Method* findLocation berfungsi inisialisasi GPS lalu mendapat kordinat dan menampungnya di atribut.
3. *Method* updateAccuracy berfungsi untuk mengubah nilai akurasi dari perangkat.
4. *Method* setPositionChanged berfungsi mengubah atribut coorLat, atribut coorLong, dan akurasi jika terdapat perubahan lokasi. *method* ini memiliki parameter Geolocator dan PositionChangedEventArgs yang akan menjalankan *method* jika terdapat perubahan yang diberitahukan melalui kelas Geolocator.

5. *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. *Method* ReverseGeocodeQueryFrom_QueryCompleted berfungsi untuk mencari alamat lokasi asal. *method* ini memiliki parameter objek dan QueryCompletedEventArgs< IList<MapLocation>>.
7. *Method* ReverseGeocodeQueryTo_QueryCompleted berfungsi untuk mencari alamat lokasi tujuan. *method* ini memiliki parameter objek dan QueryCompletedEventArgs< IList<MapLocation>>.
8. *Method* ReverseGeocodeQuery_QueryCompleted berfungsi untuk mencari alamat lokasi perangkat. *method* ini memiliki parameter objek dan QueryCompletedEventArgs< IList<MapLocation>>.
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 digunakan sebagai masukan disimpannya lokasi perangkat.
10. *Method* reset berfungsi untuk memasang kembali lokasi asal dan lokasi tujuan.

4.2.9 Kelas Map

Map merupakan kelas yang digunakan untuk mendapatkan titik yang ditunjuk pengguna pada peta lalu menerjemahkannya dalam bentuk titik kordinat. Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

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

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

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

5. *Method* map_Tap berfungsi untuk menandai lokasi yang ditunjuk pengguna lalu menerjemahkan 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*.

4.2.10 Kelas *HttpClient*

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

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

4.2.11 Kelas *JsonConvert*

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

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

4.2.12 Kelas *Protocol*

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

1. *uri_version* bertipe *string* digunakan untuk menyimpan nama dari parameter uri.
2. *uri_mode* bertipe *string* digunakan untuk menyimpan nama dari parameter mode.
3. *uri_locale* bertipe *string* digunakan untuk menyimpan nama dari parameter locale.
4. *uri_start* bertipe *string* digunakan untuk menyimpan nama dari parameter start.
5. *uri_finish* bertipe *string* digunakan untuk menyimpan nama dari parameter finish.
6. *uri_presentation* bertipe *string* digunakan untuk menyimpan nama dari parameter presentation.
7. *uri_apikey* bertipe *string* digunakan untuk menyimpan nama dari parameter apikey.
8. *uri_region* bertipe *string* digunakan untuk menyimpan nama dari parameter region.
9. *uri_query* bertipe *string* digunakan untuk menyimpan nama dari parameter query.
10. *apiKey* bertipe *string* digunakan untuk menyimpan nilai kunci API untuk mengirim perintahan ke Kiri.

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

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

1. *Method* getTypeTransport merupakan *method* yang akan mengembalikan alamat dari gambar transportasi dengan bingkai. *Method* ini memiliki 2 parameter yaitu means sebagai tipe transportasi dan meansDetail sebagai nama kendaraan.
2. *Method* getTypeTransportWOBaloon merupakan *method* yang mengembalikan alamat dari gambar transportasi tanpa bingkai tambahan. *Method* ini memiliki 2 parameter yaitu means sebagai tipe transportasi dan meansDetail sebagai nama kendaraan.
3. *Method* getSearchPlace merupakan *method* yang mengembalikan URI pencarian lokasi sesuai parameter. Parameter yang dimaksud adalah kata kunci masukan pengguna.
4. *method* getFindRoute merupakan *method* yang mengembalikan URI pencarian rute sesuai parameter. Parameter yang dimaksud adalah kordinat lokasi asal dan kordinat lokasi tujuan yang bertipe *string*.

5. *Method* `getRequestSearch` digunakan untuk mendapatkan lokasi terkait sesuai masukan pengguna. *Method* ini mengembalikan `Task<RootObjectSearchPlace>` karena menggunakan operasi *asynchronous*. *Method* ini memiliki parameter kata kunci masukan pengguna dan kota yang masing-masing parameter bertipe *string*.
6. *Method* `getRequestRoute` digunakan untuk mendapatkan rute sesuai lokasi asal dan lokasi tujuan. *Method* ini mengembalikan `Task<RootObjectFindRoute>` karena menggunakan operasi *asynchronous*. *method* ini memiliki parameter *latitude* lokasi asal, *longitude* lokasi asal, *latitude* lokasi tujuan, dan *longitude* lokasi tujuan yang masing-masing bertipe *double*.

4.2.13 Kelas *RootObjectSearchPlace*

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

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

4.2.14 Kelas *SearchResult*

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

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

4.2.15 Kelas *RootObjectFindRoute*

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

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

4.2.16 Kelas *RoutingResult*

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

1. `steps` bertipe *list* digunakan untuk menampung tiap langkah dari lokasi asal ke lokasi tujuan. Tiap langkah yang dimaksud adalah kendaraan yang digunakan, jalur yang dilewati, dan informasi yang dibutuhkan pengguna.
2. `traveltime` bertipe *string* digunakan untuk menampung waktu yang dibutuhkan dari lokasi asal ke lokasi tujuan.

4.2.17 Kelas *Route*

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

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

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

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

5. *Method* TrackLocation digunakan untuk inisialisasi GeoLocator dan memulai pelacakan lokasi terus menerus.
6. *Method* geolocator_StatusChanged digunakan untuk mengetahui status GeoLocator.
7. *Method* geolocator_PositionChanged digunakan untuk mengetahui perubahan lokasi yang terjadi dan menyimpannya di kelas LocationFinder.
8. *Method* Find digunakan untuk mencari rute yang dicari pengguna.
9. *Method* setRouteToMap digunakan untuk menggambar rute dan lokasi pada *layer* peta. *method* ini memiliki parameter RootObjectFindRoute yang merupakan objek untuk pencarian rute.
10. *Method* setFocus digunakan untuk mengarahkan pusat pandangan ke titik lokasi pergantian jenis transportasi. *method* ini memiliki parameter objek dan RoutedEventArgs.
11. *Method* toMyLocation digunakan untuk mengarahkan pusat pandangan ke titik lokasi perangkat berada. *method* ini memiliki parameter objek dan RoutedEventArgs.
12. *Method* ShowList digunakan untuk membuka dan menutup rute dalam bentuk daftar. *method* ini memiliki parameter objek dan RoutedEventArgs.
13. *Method* createNew digunakan untuk membuat objek Pushpins. *method* ini memiliki parameter uri bertipe String, transport bertipe String yang menandakan jenis transportasi, dan geoCoordinate bertipe GeoCoordinate sebagai lokasi dari Pushpins.
14. *Method* drawMyLocationOnTheMap digunakan untuk membuat penanda lokasi di lapisan myLocationLayer pada peta. *method* ini memiliki parameter latitude bertipe double dan longitude bertipe double.
15. *Method* back digunakan untuk konfirmasi ke pengguna jika pengguna ingin meninggalkan aplikasi. *method* ini memiliki dua buah parameter sender bertipe objek dan e bertipe RoutedEventArgs.
16. *Method* ShowLoading digunakan untuk memunculkan *popup* menunggu.
17. *Method* StartLoadingData digunakan untuk pemanggilan BackgroundWorker.
18. *Method* backgroundWorker_DoWork digunakan untuk eksekusi *method* dengan BackgroundWorker. *method* ini memiliki dua buah parameter sender bertipe objek dan e bertipe DoWorkEventArgs.
19. *Method* backgroundWorker_RunWorkerCompleted digunakan untuk menutup *popup* menunggu jika semua *method* sudah selesai dijalankan. *method* ini memiliki dua buah parameter sender bertipe objek dan e bertipe RunWorkerCompletedEventArgs.
20. *Method* OnBackKeyPress digunakan untuk konfirmasi ke pengguna jika pengguna ingin meninggalkan aplikasi dengan menekan tombol "back". *method* ini memiliki parameter e bertipe CancelEventArgs.

4.3 Perancangan Antar Muka

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

4.3.1 Antarmuka Kelas MainPage



Gambar 4.15: Antarmuka *MainPage*

Antarmuka Kelas Map pada gambar 4.15 merupakan tampilan awal saat aplikasi dijalankan. Antarmuka Kelas Map memiliki dua buah masukan, lima buah tombol, dan satu menu daftar. Berikut adalah detailnya.

Dua buah masukan yaitu.

- Masukan lokasi asal

Merupakan masukan lokasi asal mula pengguna ingin melakukan perjalanan.

- Masukan lokasi tujuan

Merupakan masukan lokasi tujuan berhentinya perjalanan.

Lima buah tombol yaitu.

- Tombol map untuk lokasi asal

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

- Tombol here untuk lokasi asal

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

- Tombol map untuk lokasi tujuan

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

- Tombol here untuk lokasi tujuan

Jika tombol ditekan maka lokasi tujuan adalah lokasi perangkat saat tombol ditekan dan masukan lokasi tujuan menjadi "here".

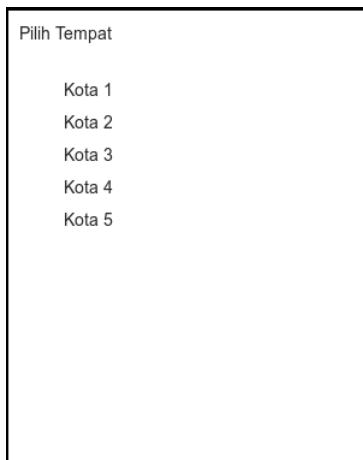
- Tombol find

Jika tombol ditekan maka aplikasi akan menampilkan daftar tempat asal dan tempat tujuan lalu mengarahkan ke Kelas Route.

Satu buah daftar yaitu.

- Daftar kota yang tersedia

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.

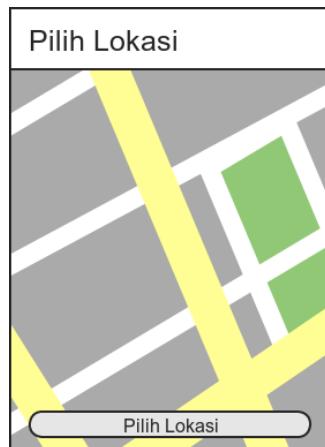


Gambar 4.16: Antarmuka Daftar Tempat

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

4.3.2 Antarmuka Kelas Map

Antarmuka Kelas Map pada gambar 4.17 merupakan antarmuka yang menunjuk lokasi pada peta. Terdapat satu buah tombol yang dimunculkan jika pengguna sudah memilih lokasi. Jika tombol ditekan maka kordinat lokasi akan di simpan dan dikirim pada kelas MainPage dan halaman akan diarahkan ke kelas MainPage.



Gambar 4.17: Antarmuka *Map*



Gambar 4.18: Antarmuka *Route*

4.3.3 Antarmuka Kelas Route

Antarmuka Kelas Route pada gambar 4.18 merupakan antarmuka untuk melihat rute dari lokasi asal ke lokasi tujuan dalam bentuk daftar maupun peta. Terdapat empat buah tombol pada antarmuka Kelas Route. Berikut tombol yang terdapat pada Kelas Route.

- Tombol prev

Jika tombol ditekan maka akan menunjuk titik sebelumnya pada rute peta.

- Tombol next

Jika tombol ditekan maka akan menunjuk titik setelahnya pada rute peta.

- Tombol here

Jika tombol ditekan maka akan menunjuk lokasi perangkat berada pada peta.

- Tombol Show List

Jika tombol ditekan maka akan menunjuk atau menyembunyikan daftar rute.

Antarmuka rute dalam bentuk daftar pada gambar 4.19 merupakan antarmuka untuk melihat rute secara lebih jelas dengan keterangan tahap demi tahap disertai jarak dan waktu perjalanan.



Gambar 4.19: Antarmuka Rute dalam Bentuk Daftar

Antarmuka daftar dapat dilihat atau disembunyikan sesuai keinginan pengguna namun saat kelas Rute dibuka antarmuka daftar rute akan disembunyikan.

BAB 5

IMPLEMENTASI DAN PENGUJIAN APLIKASI

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

5.1 Implementasi

Pada sub-bab ini dijelaskan mengenai lingkungan yang digunakan untuk membangun aplikasi Pencari Rute Kendaraan Umum untuk Windows Phone dan hasil dari implementasi. Lingkungan implementasi yang dibahas meliputi perangkat lunak untuk implementasi dan perangkat keras untuk implementasi. Hasil dari implementasi juga disampaikan dalam bentuk *screenshoot* ketika aplikasi dijalankan.

5.1.1 Perangkat Keras untuk Implementasi

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

1. Komputer
 - (a) *Processor*: intel Core i7-2620M CPU 2,7 GHz
 - (b) RAM: 4 GB
 - (c) Hardisk: 640 GB
 - (d) VGA: Intel HD 3000
2. Perangkat *Mobile*
 - (a) *Processor*: 1,2 GHz
 - (b) RAM: 1 GB
 - (c) ROM: 8 GB
 - (d) Layar: 720 x 1280 pixel, 4,7 inch
 - (e) GPS
 - (f) Sensor: kompas, *accelerometer*

5.1.2 Perangkat Lunak untuk Implementasi

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

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

5.1.3 Hasil Implementasi

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

1. Kode program
Kode Program pada perangkat lunak ditulis dengan menggunakan bahasa C#. Bahasa C# dipilih berdasarkan analisa pada bab 3.
2. Hasil kompilasi program
Hasil dari kompilasi program berupa *file* Kiri_Debug_AnyCPU.xap. *File* ini dapat dipasang pada perangkat dengan sistem operasi Windows Phone versi 8 atau lebih tinggi.
3. Antarmuka aplikasi
Berikut merupakan hasil implementasi antarmuka aplikasi Pencari Rute Kendaraan Umum untuk Windows phone.



Gambar 5.1: Gambar antarmuka kelas MainPage

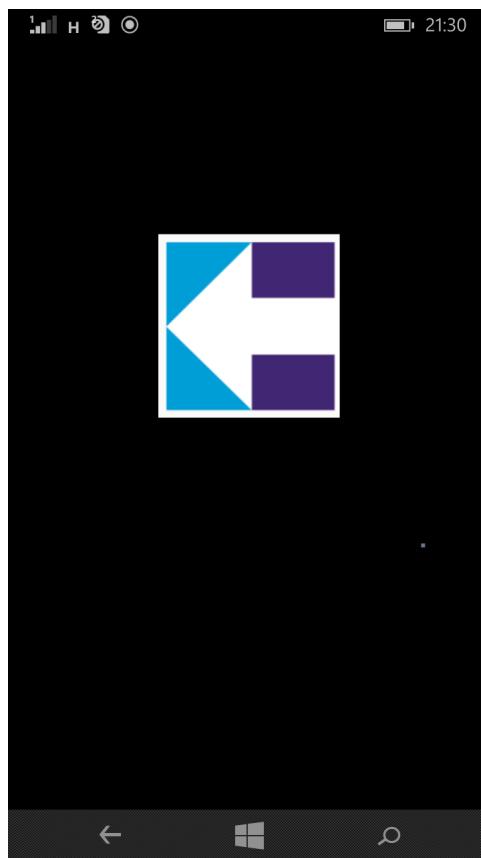
5.2 Pengujian

Pada bagian ini dibahas mengenai hasil pengujian yang telah dilakukan terhadap aplikasi yang telah dibangun. 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 digunakan 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



Gambar 5.2: Gambar Antarmuka Splash di Kelas MainPage

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.

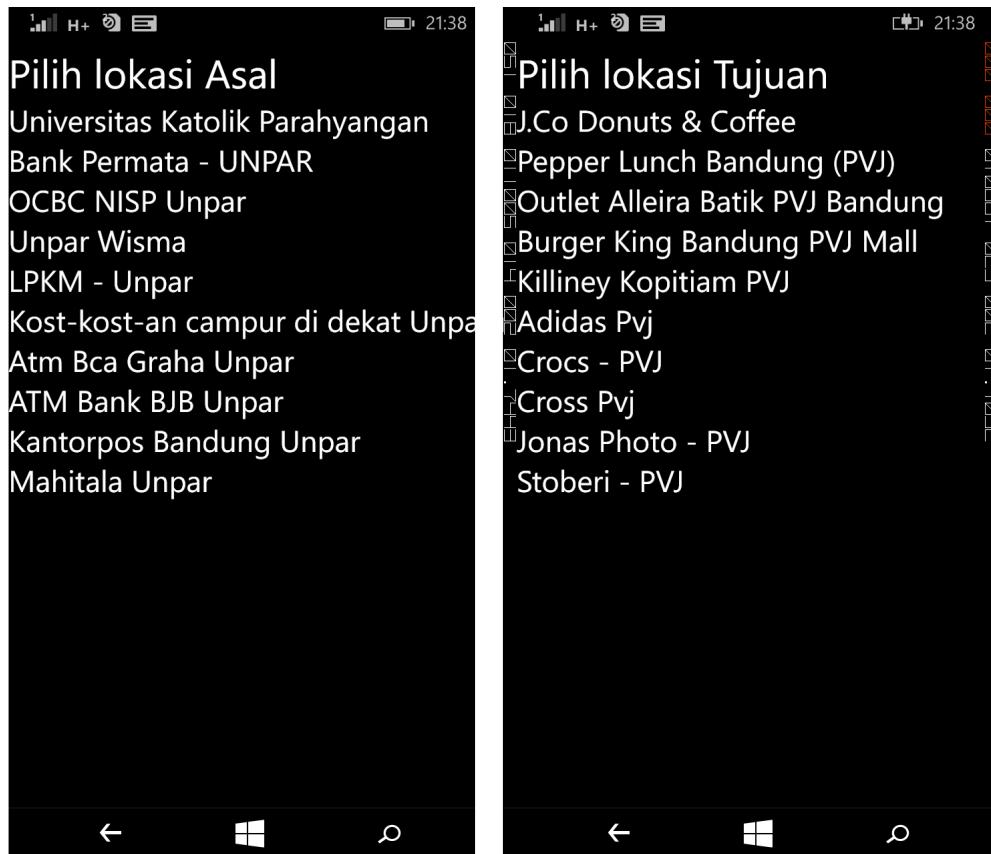
1. Carilah rute dari sini ke BIP!
2. Carilah rute dari BIP ke Ciwalk!
3. Carilah rute dari sini ke ITB pintu jalan Ganesa!

Dari tiga perintah berikut, penulis mengharapkan hasil sebagai berikut.

Perintah	Aksi Pengguna
1	Mencari lokasi dengan 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	Mencari lokasi dengan tombol "Here" dan menunjuk pintu masuk ITB yang di jalan Ganesa

Tabel 5.1: Tabel Hasil yang Diharapkan Penulis untuk Pengujian

Dari tiga perintah berikut didapat hasil sebagai berikut.



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

Perintah	Aksi Pengguna	Kesesuaian
1	Mencari lokasi dengan 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	Mencari lokasi dengan tombol "Here" dan menulis "jl Ganesa" pada <i>textbox</i>	tidak sesuai

Tabel 5.2: Tabel Hasil Pengujian Fungsionalitas Terhadap Pengguna 1

Perintah	Aksi Pengguna	Kesesuaian
1	Mencari lokasi dengan 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	Mencari lokasi dengan tombol "Here" dan menunjuk pada peta	sesuai

Tabel 5.3: Tabel Hasil Pengujian Fungsionalitas Terhadap Pengguna 2

Hasil pengujian menunjukkan 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



Gambar 5.4: Gambar Antarmuka Kelas Map

Perintah	Aksi Pengguna	Kesesuaian
1	Mencari lokasi dengan 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	Mencari lokasi dengan tombol "Here" dan menulis "ITB Ganeca" pada TextBox lalu menyadari tombol map dan menggunakannya	sesuai

Tabel 5.4: Tabel Hasil Pengujian Fungsionalitas Terhadap Pengguna 3

Perintah	Aksi Pengguna	Kesesuaian
1	Mencari lokasi dengan 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	Mencari lokasi dengan tombol "Here" dan menulis "Institut Teknologi Bandung" pada <i>textbox</i>	tidak sesuai

Tabel 5.5: Tabel Hasil Pengujian Fungsionalitas Terhadap Pengguna 4

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.



Gambar 5.5: Gambar Antarmuka Menunggu di Kelas Route

Perintah	Aksi Pengguna	Kesesuaian
1	Mencari lokasi dengan 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	Mencari lokasi dengan tombol "Here" dan menulis "jalan ganesa" pada <i>textbox</i>	tidak sesuai

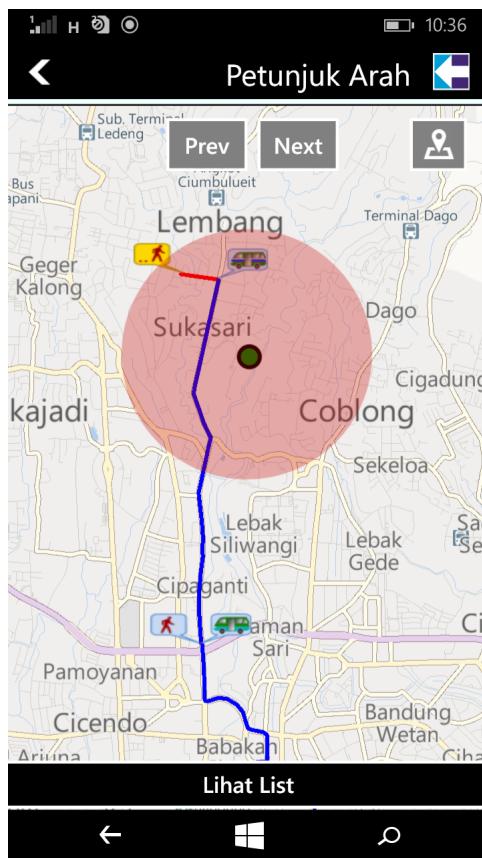
Tabel 5.6: Tabel Hasil Pengujian Fungsionalitas Terhadap Pengguna 5

5.2.3 Pengujian Eksperimental

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

- Tempat : Jalan Kejaksaan menuju Unpar dan kembali ke jalan Kejaksaan
- Waktu : Pukul 9 pada tanggal 7 April 2014
- Pengalaman :

Pengujian dilakukan dengan memasukan lokasi asal dengan tombol "here" (dilakukan di dalam rumah) menuju Unpar dengan memasukan kata kunci "unpar" pada lokasi tujuan. Sesaat setelah memasukan kata kunci "unpar" muncul daftar tempat sesuai kata kunci "Unpar", lalu penulis memilih "Universitas Katolik Parahyangan". Saat rute ditampilkan ada ketidaktepatan lokasi yang



Gambar 5.6: Gambar Antarmuka Kelas Route

ditunjukkan untuk lokasi asal(rumah) sedangkan untuk lokasi tujuan(Unpar) ditunjukkan dengan tepat. Kaitidaktepatan tersebut kira-kira seratus meter dan lokasi ditandai lingkaran merah dengan ukuran besar. Hal tersebut dikarenakan tempat tertutup yang membuat GPS tidak menunjukkan lokasi yang akurat. Tapi saat pengujian penulis berusaha mengikuti titik naik angkutan umum.

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 ditunjukkan aplikasi sudah sesuai dengan rute angkutan kota dan penunjuk lokasi menunjukkan 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.

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



Gambar 5.7: Gambar Antarmuka *List* di Kelas Route

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.

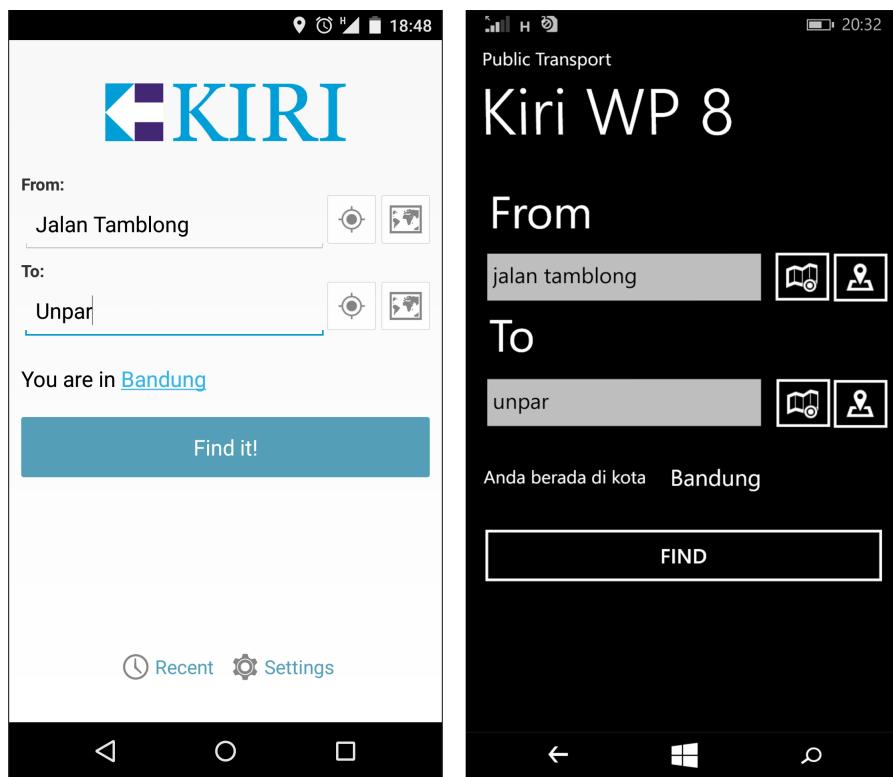
Selain pengujian langsung penulis juga coba mampandingkan hasilnya pencarian di Android dan di Windows Phone. Hasil perbandingan dapat dilihat pada gambar [5.8](#) sampai [5.11](#).

Dari hasil perbandingan aplikasi pencarian rute pada Android dan Windows Phone didapatkan kesimpulan sebagai berikut.

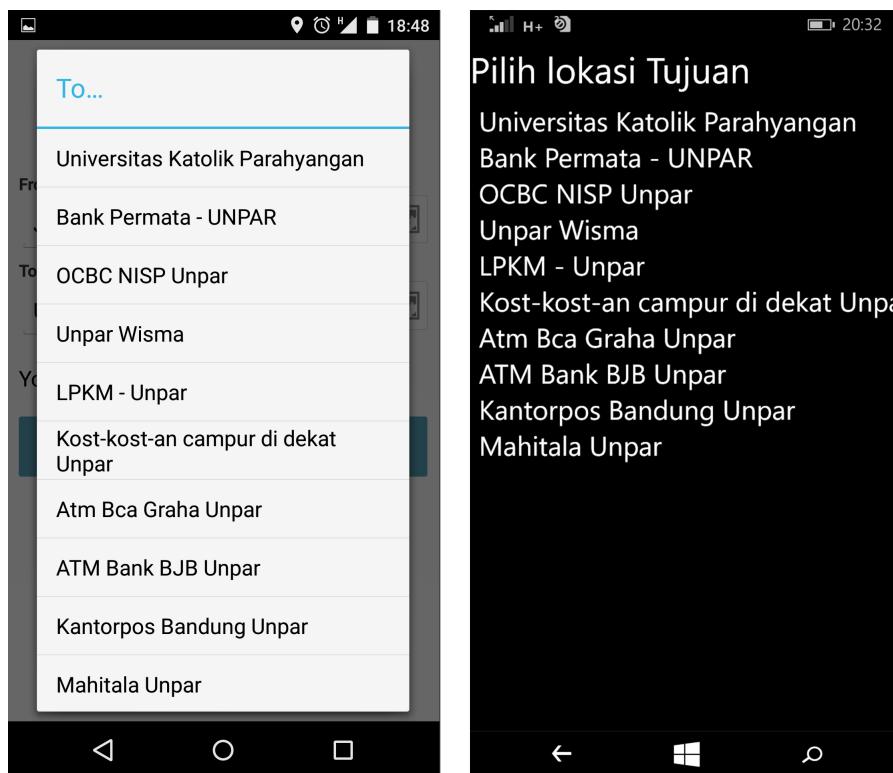
- Penunjuk lokasi perangkat menunjuk lokasi yang hampir sama ketika berada di ruangan terbuka.
- Hasil penentuan lokasi dari kata kunci memiliki nilai yang sama.
- Hasil penentuan rute sama-sama menunjukkan keakuratan yang sama.

5.2.4 Perbandingan Waktu pencarian Rute

Waktu menjadi hal yang penting bagi seseorang untuk menggunakan aplikasi. Pada pengujian ini dibandingkan waktu yang dibutuhkan dalam pencarian rute antara jaringan *mobile* dengan Wi-Fi. Pengukur waktu dilakukan mulai dari aplikasi dibuka sampai rute berhasil didapatkan. Tempat pengujian dilakukan di rumah dan di Unpar. Berikut hasil perbandingannya.

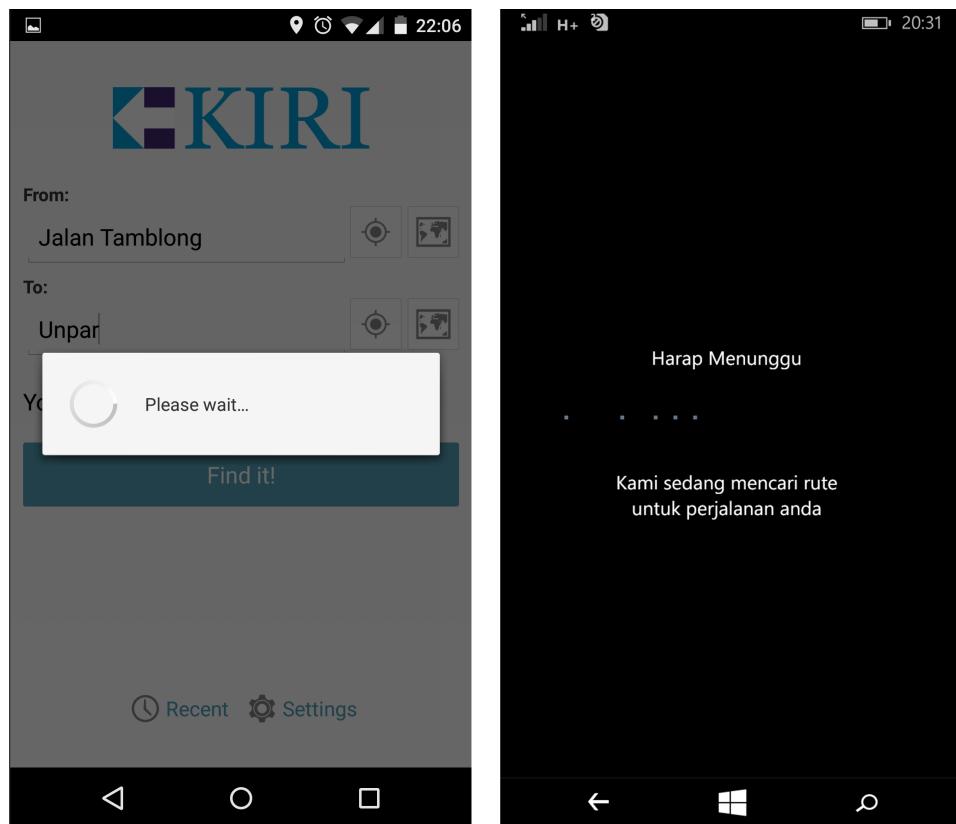


Gambar 5.8: Gambar Perbandingan Antarmuka *Home* di Android(kiri) dan Windows Phone(kanan)



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

Dari hasil pengujian didapat hasil bahwa hasil pencarian rute lebih cepat didapat jika meng-



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

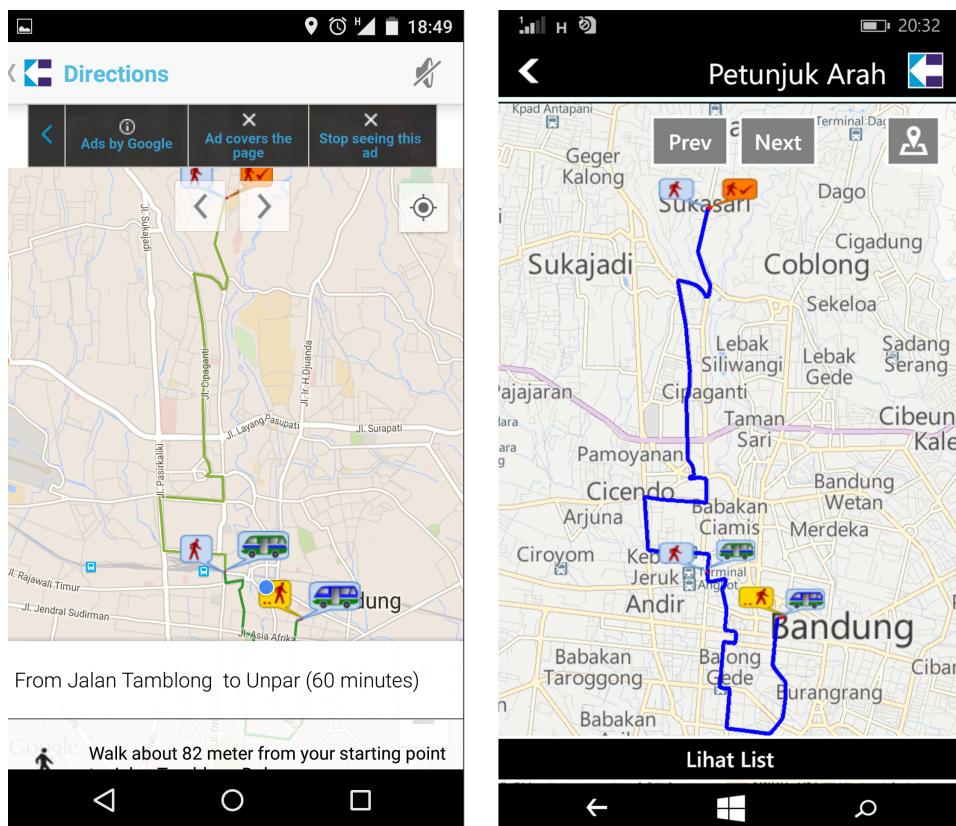
Keadaan	Jaringan <i>mobile</i> (detik)	Wi-Fi (detik)
Dalam Rumah	36	34
Luar Rumah	34	33
Di Unpar	37	35
Rata-rata	35.6	34

Tabel 5.7: Tabel Perbedaan

gunakan jaringan Wi-Fi. Hal tersebut dikarenakan kecepatan jaringan Wi-Fi lebih cepat dari pada jaringan *mobile* yang menyebabkan data lebih cepat diterima perangkat. Hasil pengujian juga menunjukan hasil pencarian rute di dalam ruangan tertutup lebih lambat dari pada di luar ruangan. Pencarian rute di dalam ruangan yang lebih lambat dikarenakan GPS yang lebih lama mendapatkan lokasi dibandingkan jika perangkat berada di luar ruangan.

5.2.5 Perbandingan Aplikasi Pencari Rute di Android dengan Aplikasi Pencari Rute di Windows Phone

Aplikasi Pencari rute yang dibuat dengan yang ada di Android memiliki beberapa perbedaan. Berikut perbedaan antara aplikasi pencari rute di Android dengan Windows Phone.



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

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 memperbarui 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 5.8: Tabel Perbedaan

BAB 6

KESIMPULAN DAN SARAN

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

6.1 Kesimpulan

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

1. Pengguna dapat memasukan lokasi berdasarkan peta, lokasi perangkat, dan dengan kata kunci.
2. Penggunaan Kiri API sudah dapat digunakan untuk mencari rute angkutan umum.
3. Kecepatan internet dan GPS mempengaruhi performansi mendapatkan rute. Kecepatan akses internet yang cepat dan keadaan perangkat yang berada di luar ruangan menyebabkan pencarian rute lebih cepat dilakukan.
4. Hasil pengujian menunjukan orang-orang lebih memilih untuk mengetikan kata kunci dari pada memilih lokasi pada peta.

6.2 Saran

Berdasarkan hasil kesimpulan yang telah dipaparkan, untuk pengembangan selanjutnya disaran untuk:

1. Memanfaatkan tombol "enter" untuk memanggil *method* startRoute. Masukan dari pengguna karena setelah pengguna menentukan tujuan tombol "Find" terhalang *keyboard* maka pengguna harus menekan tombol "back" terlebih dahulu untuk menekan tombol "Find".
2. Memanfaatkan pencarian lokasi pada peta agar pengguna dapat memilih lokasi dengan lebih mudah.
3. Menambahkan *gesture* dan animasi yang akan meningkatkan interaksi pengguna terhadap aplikasi.
4. Mengubah beberapa tampilan yang menjadi ciri khas Windows Phone seperti menggunakan *application bar*.

DAFTAR REFERENSI

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

LAMPIRAN A

KODE PROGRAM KELAS MAINPAGE

Listing A.1: MainPage.xaml.cs

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

```

72         }
73     if (lFinder.coorLatTo == 0.0 && lFinder.coorLongTo == 0.0 && !lFinder.addressTo.Equals("Maps"))
74     {
75         toBox.Text = lFinder.addressTo;
76     }
77     else if (lFinder.coorLatTo != 0.0 && lFinder.coorLongTo != 0.0)
78     {
79         if (!lFinder.addressTo.Equals("Here"))
80         {
81             toBox.Text = "Here";
82         }
83         else
84         {
85             toBox.Text = "Maps";
86         }
87     }
88 }
89 string forMaps = "";
90 if (NavigationContext.QueryString.TryGetValue("for", out forMaps));
91 if (forMaps!=null)
92 {
93     if (forMaps.Equals("from"))
94     {
95         fromBox.Text = "Maps";
96     }
97     else
98     {
99         toBox.Text = "Maps";
100    }
101 }
102 }
103 }
104 protected override void OnNavigatedFrom(NavigationEventArgs e)
105 {
106     base.OnNavigatedFrom(e);
107 }
108 }
109 private void Application_Deactivated(object sender, DeactivatedEventArgs e){
110     PhoneApplicationService.Current.State["location"] = lFinder;
111 }
112 }
113 private void Application_Activated(object sender, ActivatedEventArgs e) {
114     if (PhoneApplicationService.Current.State.ContainsKey("location"))
115     {
116         this.lFinder = (LocationFinder)PhoneApplicationService.Current.State["location"];
117         this.findRoute();
118     }
119 }
120 }
121 private async void startRoute(object sender, RoutedEventArgs e)
122 {
123
124     String queryFrom = fromBox.Text;
125     String queryTo = toBox.Text;
126     RootObjectSearchPlace from = null; //Untuk Asal
127     RootObjectSearchPlace to = null; //Untuk Tujuan
128
129     //Check form
130     if (!queryFrom.Equals("") && !queryTo.Equals(""))
131     {
132         //Validate From
133         Boolean routeStatus = true;
134         progressFindPlace.IsIndeterminate = true;
135         //Get place from query
136         if ((!queryFrom.Equals("Here") || !queryFrom.Equals("Maps")) && (lFinder.coorLatFrom ==
137             0.0 && lFinder.coorLongFrom == 0.0)) //Check get location from GPS
138         {
139             from = await protocol.getRequestSearch(queryFrom, myCity); //Mengubah String menjadi
140             objek
141             if (from.searchresult.Count() == 0)
142             {
143                 MessageBox.Show("Pencarian_untuk_kata_" + fromBox.Text + "_tidak_ditemukan");
144                 routeStatus = false;
145             }
146         }
147         if ((!queryTo.Equals("Here") || !queryTo.Equals("Maps")) && (lFinder.coorLatTo == 0.0 &&
148             lFinder.coorLongTo == 0.0)) //Check get location from GPS
149         {
150             to = await protocol.getRequestSearch(queryTo, myCity); //Mengubah String menjadi objek
151             if (to.searchresult.Count() == 0)
152             {
153                 MessageBox.Show("Pencarian_untuk_kata_" + toBox.Text + "_tidak_ditemukan");
154             }
155         }
156         //Check Query
157         if (routeStatus == true)
158         {
159             if ((lFinder.coorLatFrom == 0.0 && lFinder.coorLongFrom == 0.0))
160             {
161                 getListItem(from, "from"); //Show Listbox for location From
162             }
163             if ((lFinder.coorLatTo == 0.0 && lFinder.coorLongTo == 0.0))
164             {
165                 getListItem(to, "to"); //Show Listbox for location To
166             }
167         }
168     }
169     this.findRoute();
170 }
171 
```

```

167             }
168             progressFindPlace.IsIndeterminate = false;
169         }
170     else {
171         MessageBox.Show("Harap melengkapi tempat asal dan tempat tujuan");
172     }
173 }
174
175 private void changeMapFrom(object sender, RoutedEventArgs e)
176 {
177     NavigationService.Navigate(new Uri("/Map.xaml?fromMapFor=from", UriKind.Relative));
178 }
179 private void changeMapTo(object sender, RoutedEventArgs e)
180 {
181     NavigationService.Navigate(new Uri("/Map.xaml?fromMapFor=to", UriKind.Relative));
182 }
183
184 private void getHereFrom(object sender, RoutedEventArgs e)
185 {
186     this.lFinder.setCoordinateHere("from");
187     fromBox.Text = "Here";
188     lFinder.addressFrom = "Here";
189 }
190
191 private void getHereTo(object sender, RoutedEventArgs e)
192 {
193     this.lFinder.setCoordinateHere("to");
194     toBox.Text = "Here";
195     lFinder.addressTo = "Here";
196 }
197
198 private void getListItem(RootObjectSearchPlace request, String forRequest)
199 {
200     Dispatcher.BeginInvoke(new Action(delegate
201     {
202         if (request.status.ToString().Equals("ok")) //check status
203         {
204             if (request.searchresult.Count() == 1)
205             {
206                 if (forRequest.Equals("from"))
207                 {
208                     String locFrom = request.searchresult[0].location.ToString();
209                     string[] coordinate = locFrom.Split(',');
210                     lFinder.coorLatFrom = Double.Parse(coordinate[0]);
211                     lFinder.coorLongFrom = Double.Parse(coordinate[1]);
212                 }
213                 else
214                 {
215                     String locTo = request.searchresult[0].location.ToString();
216                     string[] coordinate = locTo.Split(',');
217                     lFinder.coorLatTo = Double.Parse(coordinate[0]);
218                     lFinder.coorLongTo = Double.Parse(coordinate[1]);
219                 }
220                 this.findRoute();
221             }
222         }
223     });
224     if (forRequest.Equals("from"))
225     {
226         for (int c = 0; c < request.searchresult.Count; c++)
227         {
228             listPlaceFrom.Items.Add(request.searchresult[c].placename);
229         }
230         panelFrom.Visibility = Visibility.Visible;
231         listPlaceFrom.DataContext = request.searchresult;
232         listPlaceFrom.SelectionChanged += ListBoxSelectedPlace;
233     }
234     else
235     {
236         panelTo.Visibility = Visibility.Visible;
237         for (int c = 0; c < request.searchresult.Count; c++)
238         {
239             listPlaceTo.Items.Add(request.searchresult[c].placename);
240         }
241         listPlaceTo.DataContext = request.searchresult;
242         listPlaceTo.SelectionChanged += ListBoxSelectedPlace;
243     }
244 }
245 }
246
247 {
248     MessageBox.Show("Error!");
249 }
250 }));
251 }
252
253 private void ListBoxSelectedPlace(object sender, SelectionChangedEventArgs e)
254 {
255     if (panelFrom.Visibility.Equals(Visibility.Visible))
256     { //Check visibility
257         if (null != (sender as ListBox).SelectedItem)
258         {
259             if ((sender as ListBox).SelectedIndex >= 0)
260             {
261                 String locFrom = searchCoordinatePlace((List<Searchresult>)listPlaceFrom.
262                     DataContext, listPlaceFrom.SelectedItem.ToString());
263                 string[] coordinate = locFrom.Split(',');
264                 lFinder.coorLatFrom = Double.Parse(coordinate[0]);
265                 lFinder.coorLongFrom = Double.Parse(coordinate[1]);
266             }
267         }
268     }
269 }
270
271 
```

```

265             lFinder.addressFrom = listPlaceFrom.SelectedItem.ToString();
266         }
267     }
268     panelFrom.Children.Clear();
269     panelFrom.Visibility = Visibility.Collapsed;
270 }
271 else {
272     if (null != (sender as ListBox).SelectedItem)
273     {
274         if ((sender as ListBox).SelectedIndex >= 0)
275         {
276             String locTo = searchCoordinatePlace((List<Searchresult>)listPlaceTo.DataContext,
277                                                 listPlaceTo.SelectedItem.ToString());
278             string[] coordinate = locTo.Split(',');
279             lFinder.coorLatTo = Double.Parse(coordinate[0]);
280             lFinder.coorLongTo = Double.Parse(coordinate[1]);
281             lFinder.addressTo = listPlaceTo.SelectedItem.ToString();
282         }
283         panelTo.Children.Clear();
284         panelTo.Visibility = Visibility.Collapsed;
285     }
286     this.findRoute();
287 }
288
289 public string searchCoordinatePlace(List<Searchresult> listResult, string place) {
290     String coordinate = "0.0";
291     for (int c = 0; c < listResult.Count; c++)
292     {
293         if (place.Equals(listResult[c].placename))
294         {
295             coordinate = listResult[c].location;
296             c = listResult.Count;
297         }
298     }
299     return coordinate;
300 }
301
302 public void findRoute()
303 {
304     PhoneApplicationService.Current.State["location"] = lFinder;
305     if ((lFinder.coorLatFrom != 0.0 && lFinder.coorLongFrom != 0.0) && (lFinder.coorLatTo != 0.0
306     && lFinder.coorLongTo != 0.0))
307     {
308         NavigationService.Navigate(new Uri("/Route.xaml", UriKind.Relative));
309     }
310
311 private void changeCity(object sender, SelectionChangedEventArgs e)
312 {
313     if (!cmbCurrFrom.SelectedItem.Equals(null))
314     {
315         switch (cmbCurrFrom.SelectedItem.ToString())
316         {
317             case "Bandung":
318                 this.myCity="bdo";
319                 break;
320             case "Jakarta":
321                 this.myCity="cgk";
322                 break;
323             case "Malang":
324                 this.myCity="mlg";
325                 break;
326             case "Surabaya":
327                 this.myCity="sub";
328                 break;
329             default:
330                 this.myCity="bdo"; //Finder Auto
331                 break;
332         }
333     }
334 }
335
336 public void showCity() {
337     int indexCity = -1;
338     while (indexCity == -1)
339     {
340         indexCity = c.getNearby(lFinder.coorLat, lFinder.coorLong);
341     }
342     this.myCity = c.cityCode[indexCity];
343 }
344
345 private void ShowSplash()
346 {
347     this.popup.IsOpen = true;
348     StartLoadingData();
349 }
350
351 private void StartLoadingData()
352 {
353     backgroundWorker = new BackgroundWorker();
354     backgroundWorker.DoWork += new DoWorkEventHandler(backgroundWorker_DoWork);
355     backgroundWorker.RunWorkerCompleted += new RunWorkerCompletedEventHandler(
356         backgroundWorker_RunWorkerCompleted);
357     backgroundWorker.RunWorkerAsync();
358 }
359
360 private void backgroundWorker_DoWork(object sender, DoWorkEventArgs e)
{

```

```
361         showCity() ;
362     }
363
364     private void backgroundWorker_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)
365     {
366         this.Dispatcher.BeginInvoke(() =>
367         {
368             this.cmbCurrFrom.SelectedIndex = c.getIndexFromCityCode(myCity) ;
369             this.popup.isOpen = false ;
370         });
371     }
372 }
373 }
```

Listing A.2: MainPage.xaml

```

1  <!--Logo kiri from kiri.travel
2  Icon from modernuiicons.com
3  -->
4  <phone:PhoneApplicationPage
5  x:Class="Kiri.MainPage"
6  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
7  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
8  xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
9  xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
10 xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
11 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
12 xmlns:toolkit="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
13 mc:Ignorable="d"
14 FontFamily="{StaticResource PhoneFontFamilyNormal}"
15 FontSize="{StaticResource PhoneFontSizeNormal}"
16 Foreground="{StaticResource PhoneForegroundBrush}"
17 SupportedOrientations="Portrait" Orientation="Portrait"
18 shell:SystemTray.IsVisible="True">
19
20 <!--LayoutRoot is the root grid where all page content is placed-->
21 <Grid x:Name="LayoutRoot" Background="Transparent" Margin="0,0,0,0">
22     <Grid.RowDefinitions>
23         <RowDefinition Height="158"/>
24         <RowDefinition Height="3"/>
25         <RowDefinition Height="*"/>
26     </Grid.RowDefinitions>
27
28     <!-- LOCALIZATION NOTE:
29         To localize the displayed strings copy their values to appropriately named
30         keys in the app's neutral language resource file (AppResources.resx) then
31         replace the hard-coded text value between the attributes' quotation marks
32         with the binding clause whose path points to that string name.
33
34     For example:
35
36         Text="{Binding Path=LocalizedResources.ApplicationTitle, Source={StaticResource
37             LocalizedStrings}}"
38
39         This binding points to the template's string resource named "ApplicationTitle".
40
41         Adding supported languages in the Project Properties tab will create a
42         new resx file per language that can carry the translated values of your
43         UI strings. The binding in these examples will cause the value of the
44         attributes to be drawn from the .resx file that matches the
45         CurrentUICulture of the app at run time.
46
47     <!-- TitlePanel contains the name of the application and page title -->
48     <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="6,10,6,0" RenderTransformOrigin="0.5,0.5">
49         <StackPanel.RenderTransform>
50             <CompositeTransform SkewX="0.781" TranslateX="0.736"/>
51         </StackPanel.RenderTransform>
52         <TextBlock Text="Public Transport" Style="{StaticResource PhoneTextNormalStyle}" Margin
53             ="12,0"/>
54         <TextBlock Text="Kiri_WP_8" Margin="9,-7,0,0" Style="{StaticResource PhoneTextTitle1Style}">
55             </TextBlock>
56         <PopUp x:Name="popup" Margin="6,0,-6,10" Grid.RowSpan="3">
57             <Grid Background="Black" Height="756" Width="484">
58                 <Image Source="/icons/kiri_logo.png" Margin="144,175,159,358" RenderTransformOrigin
59                     ="1.509,0.516"/>
60                 <ProgressBar Height="34" Width="481" IsIndeterminate="True" Margin="-7,489,10,233"/>
61             </Grid>
62         </PopUp>
63         <!-- ContentPanel -- place additional content here -->
64         <Grid x:Name="ContentPanel" Margin="0,0,0,0" Grid.Row="1" Grid.RowSpan="2">
65             <TextBlock Text="From" Style="{StaticResource PhoneTextNormalStyle}" Margin="26,0,278,484">
66                 <TextBlock.Text>From</TextBlock.Text>
67             <TextBlock Text="To" Style="{StaticResource PhoneTextNormalStyle}" Margin="26,137,361,274">
68                 <TextBlock.Text>To</TextBlock.Text>
69             <TextBlock x:Name="fromBox" HorizontalAlignment="Left" Margin="12,70,0,0" TextWrapping="Wrap">
70                 <TextBlock.Text>From</TextBlock.Text>
71             <TextBlock x:Name="toBox" HorizontalAlignment="Left" Margin="12,209,0,0" TextWrapping="Wrap">
72                 <TextBlock.Text>To</TextBlock.Text>
73             <Button Content="FIND" HorizontalAlignment="Left" Margin="10,376,0,0" VerticalAlignment="Top">
74                 <Button.Click>startRoute</Button.Click>
75             <Button HorizontalAlignment="Left" Margin="330,209,0,0" VerticalAlignment="Top" Click="
76                 changeMapTo">
77                 <Image Source="/icons/map.png" Margin="-14,-12,0,0" Width="60" Height="60"/>
78             </Button>
79             <Button HorizontalAlignment="Left" Margin="329,69,0,0" VerticalAlignment="Top" Click="
80                 changeMapFrom">
81                 <Image Source="/icons/map.png" Margin="-14,-12,0,0" Width="60" Height="60"/>
82             </Button>

```

```

72 | <Image _Source="icons/map.png" _Margin="-14,-12,0,0" _Width="60" _Height="60"></Image>
73 | </Button>
74 | <Button _HorizontalAlignment="Left" _Margin="393,210,0,0" _VerticalAlignment="Top" _Click="
75 |     getHereTo" _Height="77" _Width="83">
76 | </Button>
77 | <Button _HorizontalAlignment="Left" _Margin="393,70,0,0" _VerticalAlignment="Top" _Click="
78 |     getHereFrom" _Height="77" _Width="83">
79 | </Button>
80 | <TextBlock _HorizontalAlignment="Left" _Margin="20,315,0,0" _TextWrapping="Wrap" _Text="Anda_
81 |     berada di kota" _VerticalAlignment="Top"/>
82 | </toolkit:ListPicker _BorderThickness="0" _ScrollViewer.VerticalScrollBarVisibility="Auto" _Margin
83 |     ="217,298,74,-45" _Name="cmbCurrFrom" _SelectionChanged="changeCity" _Visibility="Visible">
84 | </toolkit:ListPicker>
85 | <ProgressBar _x:Name="progressFindPlace" _Background="Black" _HorizontalAlignment="Left" _Height
86 |     ="19" _Margin="0,-24,0,0" _VerticalAlignment="Top" _Width="456" _RenderTransformOrigin="0.493,1.056"/>
87 | <TextBlock _x:Name="textFrom" _TextWrapping="Wrap" _Text="Pilih lokasi Asal" _FontSize="40"/>
88 | <ListBox _x:Name="listPlaceFrom" _Padding="10" _Height="700" _FontSize="30"></ListBox>
89 | <StackPanel _x:Name="panelFrom" _HorizontalAlignment="Stretch" _Canvas.ZIndex="10" _Margin
90 |     ="0,-150,0,5" _VerticalAlignment="Stretch" _Visibility="Collapsed" _Background="Black">
91 | <TextBlock _x:Name="textTo" _TextWrapping="Wrap" _Text="Pilih lokasi Tujuan" _FontSize="40"/>
92 | <ListBox _x:Name="listPlaceTo" _Padding="10" _Height="700" _FontSize="30"></ListBox>
93 | </StackPanel>
94 |
95 |
96 |
97 | <!-- Uncomment to see an alignment grid to help ensure your controls are
98 |     aligned on common boundaries. The image has a top margin of -32px to
99 |     account for the System Tray. Set this to 0 (or remove the margin altogether)
100 |    if the System Tray is hidden.
101 |
102 |    Before shipping remove this XAML and the image itself.-->
103 | <!--<Image _Source="/Assets/AlignmentGrid.png" _VerticalAlignment="Top" _Height="800" _Width="480" _Margin="0,-32,0,0" _Grid.Row="0" _Grid.RowSpan="2" _IsHitTestVisible="False" />-->
104 | </Grid>
105 |
106 | </phone:PhoneApplicationPage>
```

LAMPIRAN B

KODE PROGRAM KELAS MAP

Listing B.1: Map.xaml.cs

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

```

74     myCircle.Opacity = 50;
75
76     MapOverlay myLocationOverlay = new MapOverlay();
77     myLocationOverlay.Content = myCircle;
78     myLocationOverlay.PositionOrigin = new Point(0, 0);
79     myLocationOverlay.GeoCoordinate = s;
80
81     if (fromMapFor.Equals("from"))
82     {
83         this.lFinder.GetCurrentCoordinate(s.Latitude, s.Longitude, "from");
84     }
85     else
86     {
87         this.lFinder.GetCurrentCoordinate(s.Latitude, s.Longitude, "to");
88     }
89     MapLayer myLocationLayer = new MapLayer();
90     myLocationLayer.Add(myLocationOverlay);
91
92     MyMap.Layers.Add(myLocationLayer);
93 }
94
95 private void pilihLokasi(object sender, RoutedEventArgs e)
96 {
97     if (fromMapFor.Equals("from"))
98     {
99         NavigationService.Navigate(new Uri("/ MainPage.xaml?for=from", UriKind.Relative));
100    }
101    else
102    {
103        NavigationService.Navigate(new Uri("/ MainPage.xaml?for=to", UriKind.Relative));
104    }
105 }
106 }
```

Listing B.2: Map.xaml

```

1 <phone:PhoneApplicationPage
2   x:Class="Kiri.Map"
3   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
4   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
5   xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
6   xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
7   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
8   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
9   xmlns:Controls="clr-namespace:Microsoft.Phone.Maps.Controls;assembly=Microsoft.Phone.Maps"
10  xmlns:toolkit="clr-namespace:Microsoft.Phone.Maps.Toolkit;assembly=Microsoft.Phone.Controls.Toolkit"
11  FontFamily="{StaticResource PhoneFontFamilyNormal}"
12  FontSize="{StaticResource PhoneFontSizeNormal}"
13  Foreground="{StaticResource PhoneForegroundBrush}"
14  SupportedOrientations="Portrait" Orientation="Portrait"
15  mc:Ignorable="d"
16  shell:SystemTray.IsVisible="true">
17
18  <!--LayoutRoot is the root grid where all page content is placed-->
19  <Grid x:Name="LayoutRoot" Background="Transparent">
20      <Grid.RowDefinitions>
21          <RowDefinition Height="Auto"/>
22          <RowDefinition Height="*"/>
23      </Grid.RowDefinitions>
24
25  <!--TitlePanel contains the name of the application and page title-->
26  <StackPanel Grid.Row="0" Margin="12,17,0,28">
27      <TextBlock Text="Kiri" Style="{StaticResource PhoneTextNormalStyle}"/>
28      <TextBlock Text="Pilih_lokasi" FontSize="40" Margin="9,-7,0,0" Style="{StaticResource PhoneTextTitle1Style}"/>
29  </StackPanel>
30
31  <!--ContentPanel - place additional content here-->
32  <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
33
34      <Controls:Map x:Name="MyMap" ZoomLevel="14" Margin="0,10,-16,0">
35          </Controls:Map>
36          <Button x:Name="ButtonPilih" Content="Pilih_lokasi" HorizontalAlignment="Left" Margin="-12,586,-16,-9" VerticalAlignment="Top" Width="484" Click="pilihLokasi" Visibility="Collapsed" Background="Black"/>
37      </Grid>
38      <ProgressBar x:Name="loadingFrom" Background="Black" HorizontalAlignment="Left" Height="16" Margin="0,96,0,0" VerticalAlignment="Top" Width="480"/>
39  </Grid>
40
41 </phone:PhoneApplicationPage>
```

LAMPIRAN C

KODE PROGRAM KELAS ROUTE

Listing C.1: Route.xaml.cs

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

```

74    {
75        this.lFinder = (LocationFinder)PhoneApplicationService.Current.State["location"];
76    }
77    this.TrackLocation();
78
79    //detect location
80    if (IsolatedStorageSettings.ApplicationSettings.Contains("LocationConsent"))
81    {
82        return;
83    }
84}
85
86    protected override void OnNavigatedFrom(NavigationEventArgs e)
87    {
88        base.OnNavigatedFrom(e);
89    }
90
91    private void Application_Deactivated(object sender, DeactivatedEventArgs e)
92    {
93        PhoneApplicationService.Current.State["location"] = lFinder;
94    }
95
96    private void TrackLocation()
97    {
98        geolocator = lFinder.geolocator;
99
100       geolocator.StatusChanged += geolocator_StatusChanged;
101      geolocator.PositionChanged += geolocator_PositionChanged;
102    }
103
104    void geolocator_PositionChanged(Geolocator sender, PositionChangedEventArgs args)
105    {
106        Dispatcher.BeginInvoke(() =>
107        {
108            drawMyLocationOnTheMap(args.Position.Coordinate.Latitude, args.Position.Coordinate.
109                Longitude);
110            lFinder.setPositionChanged(sender, args);
111        });
112    }
113
114    public async void Find()
115    {
116        Boolean status = true;
117        HttpClient httpClient = new HttpClient();
118
119        RootObjectFindRoute r = await p.getRequestRoute(lFinder.coorLatFrom, lFinder.coorLongFrom,
120            lFinder.coorLatTo, lFinder.coorLongTo);
121        if (r.status.Equals("ok"))
122        {
123            if (!r.routingresults[0].steps[0][3].Equals("Maaf, kami tidak dapat menemukan route_
124                transportasi publik untuk perjalanan Anda.")) //Check ditemukan atau tidak
125            {
126                this.setRouteToMap(r);
127            }
128            else {
129                MessageBox.Show("Maaf, kami tidak dapat menemukan route transportasi publik untuk_
130                    perjalanan Anda.");
131                status = false;
132            }
133
134            if (status == false)
135            {
136                this.lFinder.reset();
137                PhoneApplicationService.Current.State["location"] = lFinder;
138                NavigationService.Navigate(new Uri("/ MainPage.xaml", UriKind.Relative));
139            }
140        }
141    }
142
143    private void setRouteToMap(RootObjectFindRoute r) {
144        MapPolyline routeRoad = new MapPolyline();
145        for (int i = 0; i < 1; i++) // Ambil Routing result yg pertama
146        {
147            this.arrayFocus = new Point[r.routingresults[i].steps.Count + 1];
148            this.detailRoute = new String[r.routingresults[i].steps.Count + 1];
149            for (int j = 0; j < r.routingresults[i].steps.Count; j++)
150            {
151                routeRoad = new MapPolyline();
152                if (r.routingresults[i].steps[j][0].ToString().Equals("walk"))
153                {
154                    routeRoad.StrokeColor = Color.FromArgb(255, 255, 0, 0);
155                }
156                else
157                {
158                    routeRoad.StrokeColor = Color.FromArgb(255, 0, 0, 255);
159                }
160                routeRoad.StrokeThickness = 4;
161                GeoCoordinate geoCoo = new GeoCoordinate();
162                //Trim character
163                String temp = r.routingresults[i].steps[j][2].ToString();
164                char[] charsToTrim = { '[', ']', ',', ',' };
165                String result = temp.Trim(charsToTrim);
166                result = result.Replace("\", " ");
167                //Split string coordinate to array
168                string[] coordinate = result.Split(',');

```

```

169         for (int c = 0; c < coordinate.Length; c = c + 2)
170     {
171         geoCoo = new GeoCoordinate();
172         geoCoo.Latitude = double.Parse(coordinate[c]);
173         geoCoo.Longitude = double.Parse(coordinate[c + 1]);
174         routeRoad.Path.Add(geoCoo);
175         MapOverlay overlay1 = new MapOverlay();
176         //Process add icon/pushpin
177         if (j == 0 && c == 0)
178         {
179             overlay1.Content = createNew(p.iconStart, geoCoo, "start");
180             this.route.Center = geoCoo;
181             this.route.ZoomLevel = 14;
182         }
183         else if (j == r.routingresults[i].steps.Count - 1 && c == coordinate.Length - 2)
184         {
185             overlay1.Content = createNew(p.iconFinish, geoCoo, "finish");
186         }
187         else if (c == 0)
188         {
189             String iconLoc = p.getTypeTransport(r.routingresults[i].steps[j][0].ToString()
190             , r.routingresults[i].steps[j][1].ToString());
191             if (r.routingresults[i].steps[j][0].ToString().Equals("walk"))
192             {
193                 overlay1.Content = createNew(iconLoc, geoCoo, "walk");
194             }
195             else
196             {
197                 overlay1.Content = createNew(iconLoc, geoCoo, "umum");
198             }
199             overlay1.GeoCoordinate = geoCoo;
200             routeLayer.Add(overlay1);
201         }
202         this.arrayFocus[j] = new Point(double.Parse(coordinate[0]), double.Parse(coordinate
203             [1]));
204         this.detailRoute[j] = r.routingresults[i].steps[j][3].ToString();
205         if (j == r.routingresults[i].steps.Count - 1) // Untuk yg terakhir/sampai di tujuan
206         {
207             this.arrayFocus[j + 1] = new Point(double.Parse(coordinate.Length - 2)
208             , double.Parse(coordinate.Length - 1));
209             this.detailRoute[j + 1] = "Sampai_di_tujuan!";
210         }
211         //Add image
212         Uri imgUri = new Uri(p.getTypeTransportWOBaloon(r.routingresults[i].steps[j][0].
213             ToString(), r.routingresults[i].steps[j][1].ToString()), UriKind.
214             RelativeOrAbsolute);
215         BitmapImage imgSourceR = new BitmapImage(imgUri);
216         Image image = new Image();
217         image.Source = imgSourceR;
218         image.Height = 30;
219         image.Width = 50;
220         //add description
221         listRoute.Items.Add(image);
222         listRoute.Items.Add(r.routingresults[i].steps[j][3].ToString()); // Add text to
223         listBox
224         routeRoad.Path.Add(geoCoo);
225         route.MapElements.Add(routeRoad);
226     }
227     addFindRoute.Text = lFinder.addressFrom + " ke " + lFinder.addressTo + " (" + r.
228         routingresults[i].traveltime + ")";
229     // Add the list box to a parent container in the visual tree.
230     route.Layers.Add(routeLayer);
231     this.routeReady = true;
232 }
233 public void setFocus(object sender, RoutedEventArgs e)
234 {
235     routePanel.Visibility = Visibility.Collapsed;
236     detailPanel.Visibility = Visibility.Visible;
237     this.route.ZoomLevel = 18;
238     if ((sender as Button).Name.Equals("prev"))
239     {
240         this.focusPointNumber--;
241         if (this.focusPointNumber < 0)
242         {
243             this.focusPointNumber = arrayFocus.Length - 1;
244         }
245         this.route.Center = new GeoCoordinate(arrayFocus[focusPointNumber].X, arrayFocus[
246             focusPointNumber].Y);
247     }
248     else
249     {
250         this.focusPointNumber++;
251         if (this.focusPointNumber > arrayFocus.Length - 1)
252         {
253             this.focusPointNumber = 0;
254         }
255         this.route.Center = new GeoCoordinate(arrayFocus[focusPointNumber].X, arrayFocus[
256             focusPointNumber].Y);
257     }
258     detailShows.Text = detailRoute[focusPointNumber];
259 }
260 public void toMyLocation(object sender, RoutedEventArgs e)
261 {
262     this.route.Center = new GeoCoordinate(lFinder.coorLat, lFinder.coorLong);

```

```

259         this.route.ZoomLevel = 15;
260     }
261
262     private void ShowList(object sender, RoutedEventArgs e)
263     {
264         detailPanel.Visibility = Visibility.Collapsed;
265         if (listBoxStatus == false)
266         {
267             routePanel.Visibility = Visibility.Visible;
268             buttonList.Content = "Tutup_List";
269             this.listBoxStatus = true;
270         }
271         else
272         {
273             routePanel.Visibility = Visibility.Collapsed;
274             buttonList.Content = "Lihat_List";
275             this.listBoxStatus = false;
276         }
277     }
278
279     public Pushpin createNew(string uri, GeoCoordinate geoCoordinate, String transport) {
280         Pushpin p = new Pushpin();
281         Uri imgUri = new Uri(uri, UriKind.RelativeOrAbsolute);
282         BitmapImage imgSourceR = new BitmapImage(imgUri);
283         ImageBrush imgBrush = new ImageBrush() { ImageSource = imgSourceR };
284         p.Background = new SolidColorBrush(Color.FromArgb(0, 0, 0, 0));
285         p.Content = new Rectangle()
286         {
287             Fill = imgBrush,
288             Height = 30,
289             Width = 50,
290         };
291         if (transport.Equals("start"))
292         {
293             p.Margin = new Thickness(-51, -35, 0, 0);
294         }
295         else if(transport.Equals("finish")){
296             p.Margin = new Thickness(-6, -34, 0, 0);
297         }else if(transport.Equals("walk")){
298             p.Margin = new Thickness(-55, -35, 0, 0);
299         }else{
300             p.Margin = new Thickness(-5, -35, 0, 0);
301         }
302         p.GeoCoordinate = geoCoordinate;
303         return p;
304     }
305
306     private void drawMyLocationOnTheMap(Double latitude, Double longitude)
307     {
308         this.route.Layers.Remove(myLocationLayer);
309         GeoCoordinate myGeoCoordinate = new GeoCoordinate(latitude, longitude);
310         double myAccuracy = lFinder.accuracy;
311         double metersPerPixels = (Math.Cos(myGeoCoordinate.Latitude * Math.PI / 180) * 2 * Math.PI *
312             6378137) / (256 * Math.Pow(2, this.route.ZoomLevel));
313         double radius = myAccuracy / metersPerPixels;
314
315         Ellipse ellipse = new Ellipse();
316         ellipse.Width = radius * 2;
317         ellipse.Height = radius * 2;
318         ellipse.Margin = new Thickness(-radius+10, -radius+10, 0, 0);
319         ellipse.Fill = new SolidColorBrush(Color.FromArgb(75, 200, 0, 0));
320
321         Ellipse myCircle = new Ellipse();
322         myCircle.Stroke = new SolidColorBrush(Colors.Black);
323         myCircle.StrokeThickness = 4;
324         myCircle.Fill = new SolidColorBrush(Colors.Green);
325         myCircle.Height = 25;
326         myCircle.Width = 25;
327         myCircle.Opacity = 50;
328
329         MapOverlay myLocationOverlayPosition = new MapOverlay();
330         myLocationOverlayPosition.Content = myCircle;
331         myLocationOverlayPosition.PositionOrigin = new Point(0, 0);
332         myLocationOverlayPosition.GeoCoordinate = myGeoCoordinate;
333
334         MapOverlay myLocationOverlayAccuracy = new MapOverlay();
335         myLocationOverlayAccuracy.Content = ellipse;
336         myLocationOverlayAccuracy.PositionOrigin = new Point(0, 0);
337         myLocationOverlayAccuracy.GeoCoordinate = myGeoCoordinate;
338
339         myLocationLayer = new MapLayer();
340         myLocationLayer.Add(myLocationOverlayPosition);
341         myLocationLayer.Add(myLocationOverlayAccuracy);
342         this.route.Layers.Add(myLocationLayer);
343     }
344
345     private void back(object sender, RoutedEventArgs e)
346     {
347         MessageBoxResult result = MessageBox.Show("Anda_yakin_mengakhiri_Navigasi?", "Kembali_ke_Menu_Utama", MessageBoxButton.OKCancel);
348
349         if (result == MessageBoxResult.OK)
350         {
351             this.lFinder.reset();
352             PhoneApplicationService.Current.State["location"] = lFinder;
353             NavigationService.Navigate(new Uri("/ MainPage.xaml", UriKind.Relative));
354         }
355     }

```

```

356     private void ShowLoading()
357     {
358         this.popup.IsOpen = true;
359         StartLoadingData();
360     }
361
362     private void StartLoadingData()
363     {
364         backgroundWorker = new BackgroundWorker();
365         backgroundWorker.RunWorkerAsync();
366         backgroundWorker.DoWork += new DoWorkEventHandler(backgroundWorker_DoWork);
367         backgroundWorker.RunWorkerCompleted += new RunWorkerCompletedEventHandler(
368             backgroundWorker_RunWorkerCompleted);
369     }
370
371     private void backgroundWorker_DoWork(object sender, DoWorkEventArgs e)
372     {
373         this.Dispatcher.BeginInvoke(() =>
374         {
375             this.Find();
376         });
377
378         while (this.routeReady == false)
379         {
380             if (this.timeOut == true)
381             {
382                 this.routeReady = true;
383             }
384         }
385     }
386
387     void OnTimerTick(Object sender, EventArgs args)
388     {
389         newTimer.Stop();
390         this.timeOut = true;
391     }
392
393     private void backgroundWorker_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)
394     {
395         this.Dispatcher.BeginInvoke(() =>
396         {
397             this.popup.IsOpen = false;
398             if (this.timeOut == true)
399             {
400                 MessageBox.Show("Maaf saat ini kami tidak dapat memproses rute anda!");
401                 this.lFinder.reset();
402                 PhoneApplicationService.Current.State["location"] = lFinder;
403                 NavigationService.Navigate(new Uri("/ MainPage.xaml", UriKind.Relative));
404             }
405         });
406     }
407
408     protected override void OnBackKeyPress(BackPressedEventArgs e)
409     {
410         if (MessageBox.Show("Anda yakin mengakhiri Navigasi?", "Kembali ke Menu Utama",
411             MessageBoxButton.OKCancel) == MessageBoxResult.OK)
412         {
413             this.lFinder.reset();
414             PhoneApplicationService.Current.State["location"] = lFinder;
415             NavigationService.Navigate(new Uri("/ MainPage.xaml", UriKind.Relative));
416         }
417         else
418         {
419             e.Cancel = true;
420         }
421     }
422
423     //JSON data to c# using JSON.NET
424     //package from zhttp://www.nuget.org/packages/newtonsoft.json
425     //dok http://www.newtonsoft.com/json/help/html/SerializingJSON.htm
426 }

```

Listing C.2: MainPage.xaml

```

1 <phone:PhoneApplicationPage
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
5     xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
6     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
7     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
8
9     xmlns:Controls="clr-namespace:Microsoft.Phone.Maps.Controls;assembly=Microsoft.Phone.Maps"
10    xmlns:toolkit="clr-namespace:Microsoft.Phone.Maps.Toolkit;assembly=Microsoft.Phone.Controls.Toolkit"
11    xmlns:Maps="clr-namespace:Microsoft.Phone.Controls.Maps;assembly=Microsoft.Phone.Controls.Maps"
12    x:Class="Kiri.Route"
13    FontFamily="{StaticResource PhoneFontFamilyNormal}"
14    FontSize="{StaticResource PhoneFontSizeNormal}"
15    Foreground="{StaticResource PhoneForegroundBrush}"
16    SupportedOrientations="Portrait" Orientation="Portrait"
17    mc:Ignorable="d"
18    shell:SystemTray.IsEnabled="True">
19
20    <!--xmlns:maps="clr-namespace:Microsoft.Phone.Maps.Controls;assembly=Microsoft.Phone.Maps"-->
21    <!--LayoutRoot is the root grid where all page content is placed-->
22    <Grid x:Name="LayoutRoot" Background="Transparent">
23        <Grid.RowDefinitions>
24            <RowDefinition Height="Auto"/>

```

```

25     <RowDefinition Height="*"/>
26   </Grid.RowDefinitions>
27   <!--TitlePanel contains the name of the application and page title-->
28   <Grid HorizontalAlignment="Left" Height="57" Margin="10,10,0,0" Grid.RowSpan="2" VerticalAlignment="Top" Width="460">
29     <Border BorderBrush="Azure" BorderThickness="0,0,0,5" HorizontalAlignment="Left" Height="57"
30       VerticalAlignment="Top" Width="511" Margin="-21,-2,-30,0"/>
31     <TextBlock HorizontalAlignment="Left" Margin="207,4,0,0" FontSize="30" TextWrapping="Wrap"
32       Text="Petunjuk_Arah" VerticalAlignment="Top" Width="198"/>
33     <Button BorderThickness="0" HorizontalAlignment="Left" FontSize="30" Margin="-39,-12,0,-2"
34       VerticalAlignment="Top" Height="71" Width="137" Click="back">
35       <Image Source="icons/back.png" Height="60" Width="60" Margin="-14,-12,0,0"/></Image>
36     </Button>
37     <Image Source="/icons/kiri_logo.png" Margin="405,4,0,16" RenderTransformOrigin="1.509,0.516"/>
38   </Grid>
39   <Controls:Map x:Name="route" ZoomLevel="14" Margin="-9,67,-8,0" Grid.RowSpan="2"/>
40   <Button Name="prev" Content="Prev" Background="Gray" HorizontalAlignment="Left" Margin="146,67,0,0
41     " VerticalAlignment="Top" Height="77" Width="105" Click="setFocus" Grid.RowSpan="2"/>
42   <Button Name="next" Content="Next" Background="Gray" HorizontalAlignment="Left" Margin="237,67,0,0
43     " VerticalAlignment="Top" Height="77" Width="105" Click="setFocus" Grid.RowSpan="2"/>
44   <Button HorizontalAlignment="Left" Background="Gray" Margin="389,67,0,0" VerticalAlignment="Top"
45     Height="77" Width="81" Click="toMyLocation" Grid.RowSpan="2">
46     <Image Source="icons/marker.png" Height="60" Width="60" Margin="-14,-12,0,0"/></Image>
47   </Button>
48   <StackPanel x:Name="detailPanel" HorizontalAlignment="Left" Background="Black" Height="111" Margin
49     ="0,610,0,0" Grid.Row="1" VerticalAlignment="Top" Width="480" Visibility="Collapsed">
50     <TextBlock x:Name="detailShows" HorizontalAlignment="Left" Margin="0,0,0,0" Grid.Row="1"
51       TextWrapping="Wrap" Text="" VerticalAlignment="Top" Height="100" Width="478" Padding="4"/>
52   </StackPanel>
53   <Button x:Name="buttonList" Content="Lihat_List" Background="Black" HorizontalAlignment="Left"
54     Margin="-22,709,-21,-13" VerticalAlignment="Top" Height="72" Width="523" Click="ShowList" Grid
55     .Row="1"/>
56   <StackPanel x:Name="routePanel" Background="White" Height="auto" Margin="0,250,0,47" Grid.Row="1"
57     Visibility="Collapsed">
58     <TextBlock x:Name="addFindRoute" Text="Rute_Perjalanan" Foreground="Black" FontSize="25"
59       TextWrapping="Wrap" Margin="3"/>
60     <ListBox x:Name="listRoute" Foreground="White" Height="426" Background="White" FontSize="20"
61       Padding="5" Margin="3">
62       <ListBox.ItemTemplate>
63         <DataTemplate>
64           <Grid>
65             <Image Grid.Column="0" Margin="3" Width="60"/>
66             <Border BorderBrush="Black" BorderThickness="0,0,0,2">
67               <TextBlock Grid.Column="1" Margin="3" Padding="5" Foreground="Black"
68                 TextWrapping="Wrap" Text="{Binding}"/>
69             </Border>
70           </Grid>
71         </DataTemplate>
72       </ListBox.ItemTemplate>
73     </ListBox>
74   </StackPanel>
75   <Popup x:Name="popup" Margin="-9,0,-8,10" Grid.RowSpan="2">
76     <Grid Background="Black" Height="774" Width="498">
77       <ProgressBar Height="34" Width="481" IsIndeterminate="True" Margin="-7,370,10,352"/>
78       <TextBlock HorizontalAlignment="Left" Margin="166,320,0,0" TextWrapping="Wrap" Text="Harap
79         Menunggu" VerticalAlignment="Top"/>
80       <TextBlock HorizontalAlignment="Left" Margin="83,449,0,0" TextWrapping="Wrap"
81         TextAlignment="Center" Text="Kami_sedang_mencari_rute
82         untuk_perjalanan_anda" VerticalAlignment="Top" Width="322"/>
83     </Grid>
84   </Popup>
85 </Grid>
86 </phone:PhoneApplicationPage>

```

LAMPIRAN D

KODE PROGRAM KELAS CITY

Listing D.1: city.cs

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


LAMPIRAN E

KODE PROGRAM KELAS LOCATIONFINDER

Listing E.1: LocationFinder.cs

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

```

72
73     public void setPositionChanged(Geolocator sender, PositionChangedEventArgs args)
74     {
75         Deployment.Current.Dispatcher.BeginInvoke(() =>
76         {
77             coorLat = args.Position.Coordinate.Latitude;
78             coorLong = args.Position.Coordinate.Longitude;
79             updateAccuracy();
80         });
81     }
82
83
84     public void GetCurrentCoordinate(Double latitude, Double longitude, String paramFor)
85     {
86         try
87         {
88             MyCoordinate = new GeoCoordinate(latitude, longitude);
89
90             if (MyReverseGeocodeQuery == null || !MyReverseGeocodeQuery.IsBusy)
91             {
92                 MyReverseGeocodeQuery = new ReverseGeocodeQuery();
93                 MyReverseGeocodeQuery.GeoCoordinate = new GeoCoordinate(MyCoordinate.Latitude,
94                                         MyCoordinate.Longitude);
95                 if (paramFor.Equals("from"))
96                 {
97                     MyReverseGeocodeQuery.QueryCompleted += ReverseGeocodeQueryFrom_QueryCompleted;
98                     this.coorLatFrom = latitude;
99                     this.coorLongFrom = longitude;
100                }
101               else if (paramFor.Equals("to"))
102               {
103                   MyReverseGeocodeQuery.QueryCompleted += ReverseGeocodeQueryTo_QueryCompleted;
104                   this.coorLatTo = latitude;
105                   this.coorLongTo = longitude;
106                }
107               else
108               {
109                   MyReverseGeocodeQuery.QueryCompleted += ReverseGeocodeQuery_QueryCompleted;
110                   this.coorLat = latitude;
111                   this.coorLong = longitude;
112                }
113            MyReverseGeocodeQuery.QueryAsync();
114        }
115        catch (Exception ex)
116        {
117        }
118    }
119
120
121    private void ReverseGeocodeQueryFrom_QueryCompleted(object sender, QueryCompletedEventArgs< IList<
122                                         MapLocation>> e)
123    {
124        if (e.Error == null)
125        {
126            if (e.Result.Count > 0)
127            {
128                MapAddress add = e.Result[0].Information.Address;
129                addressFrom = add.Street;
130            }
131        }
132
133    private void ReverseGeocodeQueryTo_QueryCompleted(object sender, QueryCompletedEventArgs< IList<
134                                         MapLocation>> e)
135    {
136        if (e.Error == null)
137        {
138            if (e.Result.Count > 0)
139            {
140                MapAddress add = e.Result[0].Information.Address;
141                addressTo = add.Street;
142            }
143        }
144
145    private void ReverseGeocodeQuery_QueryCompleted(object sender, QueryCompletedEventArgs< IList<
146                                         MapLocation>> e)
147    {
148        if (e.Error == null)
149        {
150            if (e.Result.Count > 0)
151            {
152                MapAddress add = e.Result[0].Information.Address;
153                addressDevice = add.Street;
154            }
155        }
156
157    public void setCoordinateHere(string paramFor){
158        if (paramFor.Equals("from"))
159        {
160            this.coorLatFrom = this.coorLat;
161            this.coorLongFrom = this.coorLong;
162            this.addressFrom = this.addressDevice;
163        }
164        else {
165            this.coorLatTo = this.coorLat;
166            this.coorLongTo = this.coorLong;

```

```
167         this.addressTo = this.addressDevice;
168     }
169 }
170
171 public void reset() {
172     this.coorLatFrom = 0.0; //from coordinate
173     this.coorLongFrom = 0.0;
174     this.coorLatTo = 0.0; //to coordinate
175     this.coorLongTo = 0.0;
176
177     this.addressDevice = "";
178     this.addressFrom = "";
179 }
180
181 }
182 }
```


LAMPIRAN F

KODE PROGRAM KELAS PROTOCOL

Listing F.1: Protocol.cs

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

```
74         get
75     {
76         return "&querystring=";
77     }
78 }
79
80 private static String apiKey
81 {
82     get
83     {
84         return "97A7A1157A05ED6F";
85     }
86 }
87 private static String hostname
88 {
89     get
90     {
91         return "http://kiri.travel/";
92     }
93 }
94 private static String handle
95 {
96     get
97     {
98         return hostname+"handle.php?";
99     }
100 }
101 private static String iconPath
102 {
103     get
104     {
105         return hostname + "images/means/";
106     }
107 }
108 public String iconStart
109 {
110     get
111     {
112         return hostname + "images/stepicon-walkstart.png";
113     }
114 }
115 public String iconFinish
116 {
117     get
118     {
119         return hostname + "images/stepicon-finish.png";
120     }
121 }
122
123
124 private static String version_2
125 {
126     get
127     {
128         return "2";
129     }
130 }
131
132 private static String modeFind
133 {
134     get
135     {
136         return "searchplace";
137     }
138 }
139
140 private static String modeRoute
141 {
142     get
143     {
144         return "findroute";
145     }
146 }
147 private static String modeNearby
148 {
149     get
150     {
151         return "nearbytransport";
152     }
153 }
154 private static String localeId
155 {
156     get
157     {
158         return "id";
159     }
160 }
161 private static String localeEn
162 {
163     get
164     {
165         return "en";
166     }
167 }
168 private static String presentationMobile
169 {
170     get
171     {
172         return "mobile";
```

```

173         }
174     }
175     private static String presentationDesktop
176     {
177         get
178         {
179             return "desktop";
180         }
181     }
182
183     public string getTypeTransport(string means, string meansDetail)
184     {
185
186         String uri = iconPath + means + "/baloon/" + meansDetail + ".png";
187         return uri;
188     }
189
190     public string getTypeTransportWOBaloon(string means, string meansDetail)
191     {
192
193         String uri = iconPath + means + "/" + meansDetail + ".png";
194         return uri;
195     }
196
197     public string getSearchPlace(string query, string region)
198     {
199
200         String uri = handle + uri_version + version_2 + uri_mode + modeFind + uri_region + region +
201             uri_query + query + uri_apikey + apiKey;
202         return uri;
203     }
204
205     public string getFindRoute(string start, string finish)
206     {
207
208         String uri = handle + uri_version + version_2 + uri_mode + modeRoute + uri_locale + localeId +
209             uri_start + start + uri_finish + finish + uri_presentation + presentationDesktop +
210                 uri_apikey + apiKey;
211         return uri;
212     }
213
214     public async Task<RootObjectSearchPlace> getRequestSearch(string query, string region)
215     {
216
217         String uri = getSearchPlace(query, region);
218         Task<String> requestRouteTask = httpClient.GetStringAsync(new Uri(uri));
219         String request = await requestRouteTask;
220         RootObjectSearchPlace objectRootSearch = JsonConvert.DeserializeObject<RootObjectSearchPlace>(
221             request);
222         return objectRootSearch;
223     }
224
225
226
227
228     }
229 }

```


LAMPIRAN G

KODE PROGRAM OBJEK DARI JSON

Listing G.1: RootObjectFindRoute.cs

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

Listing G.2: RootObjectSearchPlace.cs

```
1| using System;
2| using System.Collections.Generic;
3| using System.Linq;
4| using System.Text;
5| using System.Threading.Tasks;
6|
7| namespace Kiri
8| {
9|     public class RootObjectSearchPlace
10|    {
11|        public string status { get; set; }
12|        public List<Searchresult> searchresult { get; set; }
13|        public object attributions { get; set; }
14|    }
15| }
```

Listing G.3: Routingresult.cs

```
1| using System;
2| using System.Collections.Generic;
3| using System.Linq;
4| using System.Runtime.Serialization;
5| using System.Text;
6| using System.Threading.Tasks;
7|
8| namespace Kiri
9| {
10|     class Routingresult
11|    {
12|        public List<List<object>> steps { get; set; }
13|        public string travertime { get; set; }
14|    }
15| }
```

Listing G.4: Searchresult.cs

```
1| using System;
2| using System.Collections.Generic;
3| using System.Linq;
4| using System.Text;
5| using System.Threading.Tasks;
6|
7| namespace Kiri
8| {
9|     public class Searchresult
10|    {
11|        public string placename { get; set; }
12|        public string location { get; set; }
13|    }
14| }
```