

SKRIPSI

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



YOHAN

NPM: 2011730048

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

DAFTAR ISI

DAFTAR ISI	iii
DAFTAR GAMBAR	v
DAFTAR TABEL	vi
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	2
1.4 Batasan Masalah	2
1.5 Metode Penelitian	2
1.6 Sistematika Penulisan	3
2 DASAR TEORI	5
2.1 Windows Phone	5
2.1.1 Lingkungan Kerja	5
2.1.2 XAML	6
2.1.3 Kontrol terhadap Ponsel	6
2.1.4 Siklus Hidup Aplikasi	10
2.1.5 Peta di Windows Phone	12
2.1.6 Lokasi	18
2.1.7 Memanfaatkan Sumber Data	21
2.2 Kiri API	23
2.2.1 <i>Web Service</i> Penentuan Rute	24
2.2.2 <i>Web Service</i> Pencarian Lokasi	26
2.2.3 <i>Web Service</i> Menemukan Transportasi Terdekat	27
3 ANALISIS	29
3.1 Analisis Aplikasi Sejenis	29
3.2 Analisis Aplikasi	32
3.2.1 Kebutuhan Aplikasi	32
3.2.2 Analisis Kontrol yang Dipakai	32
3.2.3 Analisis Terhadap Siklus Hidup Aplikasi	33
3.2.4 Analisis Peta	34
3.2.5 Analisis Pemanfaatan Sumber Data	36
3.2.6 Analisis Kiri API	36
3.2.7 Diagram <i>Use Case</i> dan Scenario	39
3.2.8 Kelas Diagram	43
4 PERANCANGAN	45
4.1 Diagram <i>Sequence</i>	45
4.2 Perancangan Kelas	47

4.2.1	Kelas <i>PhoneApplicationPage</i>	47
4.2.2	Kelas <i>MainPage</i>	47
4.2.3	Kelas <i>City</i>	48
4.2.4	Kelas <i>BackgroundWorker</i>	49
4.2.5	Kelas <i>Geocoordinate</i>	49
4.2.6	Kelas <i>Geolocator</i>	49
4.2.7	Kelas <i>Geoposition</i>	49
4.2.8	Kelas <i>LocationFinder</i>	49
4.2.9	Kelas <i>Map</i>	51
4.2.10	Kelas <i>HttpClient</i>	51
4.2.11	Kelas <i>JsonConvert</i>	52
4.2.12	Kelas <i>Protocol</i>	52
4.2.13	Kelas <i>RootObjectSearchPlace</i>	54
4.2.14	Kelas <i>SearchResult</i>	54
4.2.15	Kelas <i>RootObjectFindRoute</i>	54
4.2.16	Kelas <i>RoutingResult</i>	54
4.2.17	Kelas <i>Route</i>	54
4.3	Perancangan Antar Muka	56
4.3.1	Antarmuka Kelas <i>MainPage</i>	56
4.3.2	Antarmuka Kelas <i>Map</i>	57
4.3.3	Antarmuka Kelas <i>Route</i>	57
5	IMPLEMENTASI DAN PENGUJIAN APLIKASI	63
5.1	Implementasi	63
5.1.1	Perangkat Keras untuk Implementasi	63
5.1.2	Perangkat Lunak untuk Implementasi	63
5.1.3	Hasil Implementasi	64
5.2	Pengujian	66
5.2.1	Lingkungan Pengujian	67
5.2.2	Pengujian Fungsional	68
5.2.3	Pengujian Experimental	68
6	KESIMPULAN DAN SARAN	71
6.1	Kesimpulan	71
6.2	Saran	71
BIBLIOGRAFI		73

DAFTAR GAMBAR

2.1	Hirarki navigasi	7
2.2	<i>TextBlock</i> , <i>TextBox</i> dan <i>PasswordBox</i>	8
2.3	Gambar kontrol pada Windows Phone	9
2.4	Antarmuka <i>ListBox</i>	10
2.5	Gambar siklus hidup aplikasi[1]	10
2.6	Tampilan peta pada Windows Phone	12
2.7	<i>cartographic</i>	13
2.8	Keluaran Toolkit Pushpin pada peta [2]	14
2.9	Tampilan polyline pada Peta	15
3.1	Tampilan utama aplikasi Public Transport	29
3.2	Menunjuk lokasi pada peta	30
3.3	Memberikan daftar nama tempat dan nama jalan terkait	30
3.4	Tampilan rute kendaraan umum dalam bentuk daftar	31
3.5	Tampilan rute kendaraan umum di peta	31
3.6	Tampilan <i>pushpin</i> untuk angkot	35
3.7	Tampilan <i>pushpin</i> untuk jalan kaki	35
3.8	Diagram <i>use case</i>	41
3.9	Diagram Kelas	43
4.1	Diagram <i>sequence</i> Mendapatkan Kordinat perangkat	45
4.2	Diagram <i>sequence</i> Mendapatkan Kordinat pada Peta	46
4.3	Diagram <i>sequence</i> Mendapatkan Rute	59
4.4	Diagram Kelas	60
4.5	Antarmuka <i>MainPage</i>	60
4.6	Antarmuka Daftar Tempat	61
4.7	Antarmuka <i>Map</i>	61
4.8	Antarmuka <i>Route</i>	61
4.9	Antarmuka Rute dalam bentuk Daftar	62
5.1	Gambar antarmuka kelas <i>MainPage</i>	64
5.2	Gambar antarmuka <i>Splash</i> di kelas <i>MainPage</i>	65
5.3	Gambar antarmuka <i>list</i> asal dan <i>list</i> tujuan di kelas <i>MainPage</i>	65
5.4	Gambar antarmuka kelas <i>Map</i>	66
5.5	Gambar antarmuka menunggu di kelas <i>Route</i>	66
5.6	Gambar antarmuka kelas <i>Route</i>	67
5.7	Gambar antarmuka <i>list</i> di kelas <i>Route</i>	67

DAFTAR TABEL

2.1	Properti kelas Map	16
2.2	Properti <i>Polyline Class</i>	18
2.3	Kelas pada <i>Namespace Geolocator</i>	20
2.4	Properti pada <i>Geocoordinate</i>	20
2.5	Tabel parameter layanan penentuan rute	24
2.6	Tabel parameter layanan pencarian lokasi	26
2.7	Tabel parameter layanan menemukan transportasi terdekat	27
3.1	Skenario mandapatkan lokasi untuk masukan lokasi asal dan lokasi tujuan	41
3.2	Skenario memasukan lokasi asal dan lokasi tujuan pada <i>TextBox</i>	42
3.3	Skenario menunjuk lokasi asal dan lokasi tujuan pada peta	42
3.4	Skenario memilih alamat	42
3.5	Skenario menampilkan rute kendaraan umum	43
5.1	Tabel Hasil pengujian fungsionalitas	69

BAB 1

PENDAHULUAN

Pada Bab satu akan dibahas pendahuluan dari penelitian yang dilakukan. Bab satu terbagi dalam delapan *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 manusia di saat penelitian ini dilakukan. Ada dua jenis transportasi bagi seseorang yaitu kendaraan umum dan kendaraan pribadi. Kenyataannya pada saat penelitian ini dilakukan banyak yang lebih memilih kendaraan pribadi dibanding kendaraan umum. Maraknya penggunaan kendaraan pribadi dan penambahan jalur kendaraan yang tidak sebanding banyaknya kendaraan menimbulkan kemacetan. Maraknya penggunaan kendaraan pribadi dikarenakan kurang nyamannya kendaraan umum dan kesulitan dalam menentukan kendaraan umum yang harus dinaiki. Banyaknya rute kendaraan umum membuat orang kebingungan dalam memilih kendaraan umum menuju lokasi yang diinginkan. Seseorang cenderung malas untuk bertanya dan mencari rute yang efisien. Karena hal tersebut, seseorang lebih memilih menggunakan kendaraan pribadi ketimbang kendaraan umum.

Ide pembuatan aplikasi yang memudahkan seseorang dalam menentukan rute kendaraan umum sudah lebih dulu ada yang dikenal dengan nama Kiri. Kiri dibuat dengan latar belakang tiga masalah besar yaitu pemanasan global, kemacetan, dan harga bahan bakar minyak yang tinggi¹. Meskipun Kiri pertama dibuat di web tetapi Kiri dapat dimanfaatkan untuk pencarian kendaraan selain di web. Pemanfaatan Kiri tersebut dalam mencari rute kendaraan umum dengan menggunakan Kiri API.

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

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

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

1.2 Rumusan Masalah

Sehubung dengan latar belakang diatas 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 linak di Windows Phone lalu mengembangkan aplikasi yang akan 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 akan berjalan di sistem operasi Windows Phone 8.
- Aplikasi ini membutuhkan koneksi internet.
- Aplikasi ini akan menampilkan rute jalur angkot, bus umum dan travel di tiga kota besar yaitu Bandung, Jakarta, dan Surabaya.

1.5 Metode Penelitian

Metode Penelitian yang penulis gunakan 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 akan digunakan dalam tugas akhir ini. Bahasan yang dijelasakan 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 dijelasakan 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 akan 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 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, kontrol terhadap ponsel, siklus hidup Windows Phone, peta, lokasi , pemanfaatan sumber data, dan Kiri API.

2.1 Windows Phone

Windows Phone merupakan sistem operasi untuk perangkat bergerak yang dikembangkan 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].

Pada sub bab 2.1.1 sampai 2.1.7 akan membahas pemrograman di Windows Phone. Pembahasan akan dimulai dengan apa itu Windows Phone dan fitur di Windows Phone yang akan 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 runtime bahasa umum dan perpustakaan kelas .NET Framework, yang meliputi kelas, interface, dan jenis nilai yang mendukung berbagai teknologi. 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, and overloading[1]. Kelebihan dari C# yang merupakan pengembangan dari C/C++ adalah sederhana, modern, aman dan berorientasi objek[1]. Satu hal yang dirasakan penulis adalah kenyamanan ketika memilih bahasa .NET tersebut. Akan lebih mudah bagi developer yang menggunakan Visual Basic 6.0 untuk menggunakan Visual Basic .NET. Tetapi bagi developer yang menggunakan C++ atau java sebelumnya akan lebih mudah menggunakan C#.

2.1.2 XAML

Extensible Application Markup Language (XAML) merupakan bahasa deklaratif 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 sintak XAML didasarkan pada XML. Setiap XAML Windows Runtime 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*, diikuti tanda sama dengan dan *path* yang merepresentasikan *namespace*. *Prefix* x pada XAML mengandung beberapa struktur program yang sering kita gunakan yaitu :

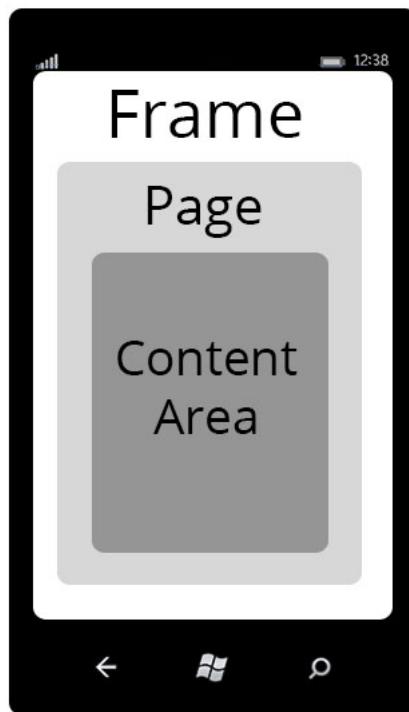
- x:Key : sebuah nama unik untuk menunjuk referensi ke suatu resource atau berkas lain. Nilai ini dapat dipanggil kembali untuk menggunakan resource tersebut.
- x:Class : menunjukkan nama kelas.
- x:Name : menunjukkan nama sebuah obyek dan untuk membedakan antar obyek yang satu dengan obyek yang lain.
- 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 Kontrol terhadap Ponsel

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

2.1.3.1 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 yang akan 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.



Gambar 2.1: Hirarki navigasi

2.1.3.2 Kontrol Tata Letak

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

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

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

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

Listing 2.1: Kode tata letak grid

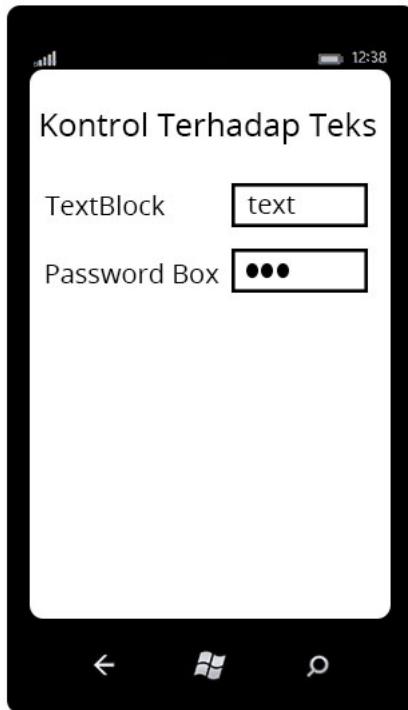
```

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

2.1.3.3 Kontrol Terhadap Teks

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 2.2, 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.



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

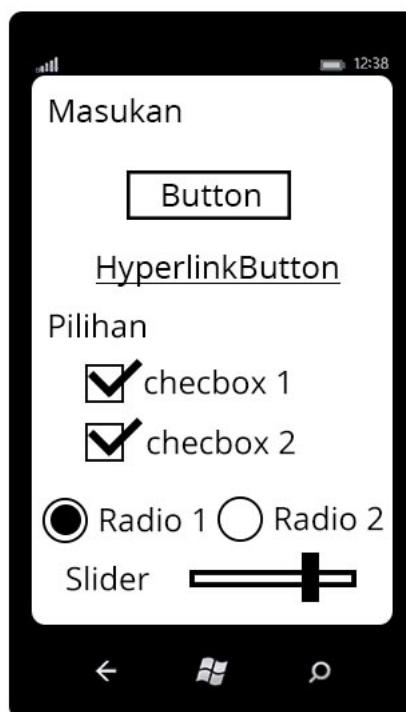
Listing 2.2: Kode untuk menampilkan *TextBlock* dan *TextBox*

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

2.1.3.4 Tombol dan Kontrol Pilihan

Tombol memungkinkan pengguna untuk berpindah halaman. Sedangkan kontrol pilihan memudahkan dalam memilih. Berikut keterangan, gambar 2.3, dan kode pada *listing 2.3* 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 akan 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 2.3: Gambar kontrol pada Windows Phone

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

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

2.1.3.5 Kontrol Daftar

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

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



Gambar 2.4: Antarmuka *ListBox*

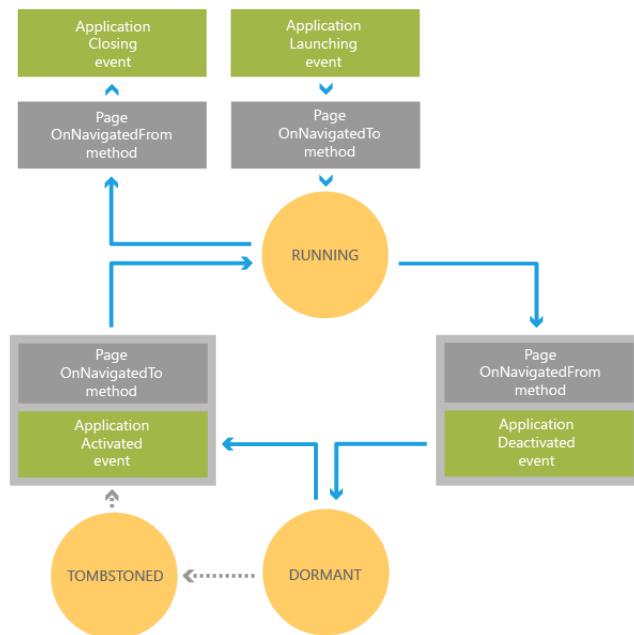
Listing 2.4: Kode untuk menampilkan *listBox*

```

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

2.1.4 Siklus Hidup Aplikasi

Siklus hidup aplikasi merupakan waktu mulai dari aplikasi dijalankan sampai dengan aplikasi diberhentikan dari memori. Siklus hidup aplikasi penting diketahui agar pengguna tidak kecewa (dalam aplikasi ini yaitu kecewa karena aplikasi keluar saat layar perangkat dimatikan) 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 penulis harus memahami kapan aplikasi harus diaktifkan, ditangguhkan, atau bahkan dinonaktifkan karena sudah tidak digunakan. Gambar 2.5 adalah ilustrasi dari siklus hidup pada Windows Phone.



Gambar 2.5: Gambar siklus hidup aplikasi[1]

Sesuai gambar 2.5 lingkaran melambangkan keadaan aplikasi, persegi panjang menunjukan peristiwa aplikasi atau tingkat peristiwa di halaman. Berikut ini adalah keterangan untuk siklus hidup Windows Phone pada gambar 2.5.

- *The Launching Event*

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

- *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.

- *Dormant*

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.

- *Tombstoned*

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

- *The Activated Event*

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

- *The OnNavigatedTo Method*

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

- *The 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.

2.1.5 Peta di Windows Phone

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

2.1.5.1 Penambahan Peta Ke Aplikasi

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



Gambar 2.6: Tampilan peta pada Windows Phone

Listing 2.5: Menampilkan peta dengan nama MyMap dari XAML

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

Listing 2.6: Menampilkan peta dengan nama MyMap dari kode program

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

2.1.5.2 Tampilan Peta di Windows Phone

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 akan dibahas 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 akan ditampilkan di peta. *Zoom* memiliki nilai yang bisa diatur dari satu hingga dua puluh. Kode untuk mengatur titik tengah peta dan tingkat *zoom* dapat dilihat pada *listing 2.7* dan *listing 2.8*.

Listing 2.7: Mengatur tingkat zoom dari XAML

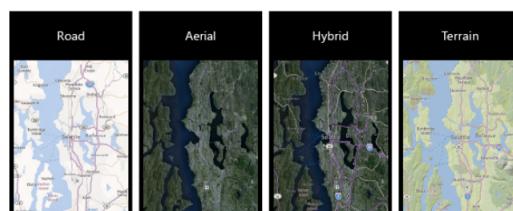
```
1| <Controls:Map x:Name="MyMap" ZoomLevel="10" Margin="-25,0,-16,0"/>
```

Listing 2.8: Mengatur tingkat zoom dari 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);
8|
9|     //mengatur tingkat zoom
10|    MyMap.ZoomLevel = 10;
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 2.7: *cartographic*

- Mode warna yang disediakan Windows Phone ada dua yaitu terang dan gelap. Secara *default* mode pada peta di Windows Phone adalah terang.
- Tampilan pada Peta di Windows Phone dapat berubah karena hasil diputar, dimiringkan, ditarik, dan diturunkan. Berikut beberapa hal yang dapat diatur sebagai tampilan di peta.

- *Heading* merupakan representasi dari derajat secara geometri. Derajat ini didefinisikan dalam 0 sampai 360 yang dipakai untuk memutar peta. Contoh, 0 atau 360 ke arah utara, 90 ke arah barat, 180 ke arah selatan, dan 270 derajat ke arah timur.
- *Pitch* merupakan derajat kemiringan dari peta dari sudut pengguna. Contoh, $Pitch = 0$ berarti melihat dari atas ke bawah sedangkan $Pitch = 45$ berarti melihat dari samping ke bawah dengan sudut 45 derajat.

2.1.5.3 *Pushpin* ke Peta

Pushpin merupakan elemen yang dapat ditempatkan pada peta secara spesifik dan bisa dipakai untuk interaksi pada peta. Peta tidak mendukung langsung penggunaan *pushpin* karena merupakan elemen dari *MapOverlay* (bagian/lapisan terpisah dari peta). Untungnya di 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 2.8 dan kode untuk menampilkannya dapat dilihat pada listing 2.9.



Gambar 2.8: Keluaran Toolkit Pushpin pada peta [2]

Listing 2.9: Kode untuk menampilkan pushpin

```

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);

```

2.1.5.4 *Polyline* pada Peta

Dalam menentukan arah dibutuhkan dua titik yaitu titik awal dan titik tujuan. Tentu saja arah tersebut butuh ditandai dengan garis. *Polyline* merupakan tentetan garis lurus yang saling terhubung satu sama lain. Dengan *polyline* arah pada peta dapat ditandai dengan warna maupun

tebal atau tipisnya garis. Contoh keluaran *polyline* dapat dilihat pada Gambar 2.9 dan kode untuk menampilkannya dapat dilihat pada *listing 2.10*.



Gambar 2.9: Tampilan polyline pada Peta

Listing 2.10: Kode untuk menampilkan polyline

```

1 MapPolyline line = new MapPolyline();
2 line.StrokeColor = Colors.Red;
3 line.StrokeThickness = 10;
4 line.Path.Add(new GeoCoordinate(-6.8619546, 107.614441));
5 line.Path.Add(new GeoCoordinate(-6.908693, 107.611185));

```

2.1.5.5 Namespace Control Map

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

2.1.5.6 Kelas Map

Merupakan kelas yang mewakili kontrol map.

Berikut properti yang dapat digunakan pada kelas ini.

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.

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

Tabel 2.1: Properti kelas Map

- *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*, *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.

2.1.5.7 *Polyline Class*

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

Berikut properti yang dapat digunakan pada kelas ini.

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

Tabel 2.2: Properti *Polyline Class*

Berikut *method* yang dapat digunakan pada kelas ini.

- *CheckAccess*

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

- *ClearValue*

method yang akan membersihkan nilai lokal

- *Finalize*

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

2.1.5.8 *Pushpin Class*

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. *Pushpin* pada Windows Phone 8 dapat dibuat sesuai kebutuhan. Namun ada cara lain dengan menambahkan Windows Phone Toolkit. Windows Phone Toolkit mempunyai komponen untuk menggambar pushpin diatas peta.

2.1.6 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. Ada 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 menyediakan data lokasi tersebut adalah pengelola antarmuka. Aplikasi akan 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 akan 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*.

2.1.6.1 Mendapatkan Posisi Pengguna

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

- *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 *GetGeopositionAsync()* atau *register*.
- *Disable* : Status mengindikasikan tidak diperbolehkannya pengaksesan lokasi.
- *NotInitialized* : Data lokasi belum tersedia. Status ini muncul jika aplikasi belum memanggil *GetGeopositionAsync()* atau *register*.
- *NotAvailable* : Jika sensor arah mata angin dan lokasi tidak tersedia.

2.1.6.2 Namespace Geolocator

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

2.1.6.3 Geocoordinate

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

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

Tabel 2.3: Kelas pada *Namespace Geolocator*

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

Tabel 2.4: Properti pada *Geocoordinate*

2.1.6.4 Geolocator

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.

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

Berikut Properti yang disediakan kelas *Geolocator*:

- *public PositionStatus LocationStatus { get; }*

Merupakan properti dari kelas *geolocator* untuk mendapatkan status posisi dengan mengembalikan 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 mengeluarkan tipe data unit.

2.1.6.5 *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.

2.1.7 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 ada dua cara yaitu yang lokal atau berada di perangkat dan *web service*. *Web Service* merupakan *method* komunikasi antara dua perangkat melalui jaringan.

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

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 format memiliki ukuran data yang kecil dan baik untuk penggunaan perangkat bergerak. Untuk contoh format JSON dapat dilihat di bagian Kiri API pada Bab dua ini karena Kiri API menggunakan format JSON. Serialisasi menggunakan *DataContractJsonSerializer* membuat serialisasi mudah untuk menerjemahkan form String JSON ke objek yang dapat langsung digunakan. *DataContractJsonSerializer* memakai *WriteObject()* untuk serialisasi and *ReadObject()* untuk de-serialisasi.

2.1.7.1 *HttpClient*

Merupakan Kelas yang dipakai untuk mengirim permintaan HTTP dan menerima kembalian HTTP dari *Uniform Resource Identifier*(URI) yang dapat diidentifikasi. 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 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.

- *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 dimaksukan sebagai indikasi operasi dianggap selesai. Sedangkan kembalinya berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut memiliki 2 parameter yaitu hasil berupa pesan kembalian dari http dan kemajuan dari data yang dikirim.

- *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 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.

- *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 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.

- *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 dimaksukan sebagai indikasi operasi dianggap selesai. Sedangkan kembalinya berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut memiliki 2 parameter yaitu hasil berupa pesan kembalian dari http dan kemajuan dari data yang dikirim.

2.1.7.2 Json.NET

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

- JsonConvert merupakan kelas yang berfungsi untuk konversi dari JSON String ke objek dan sebaliknya. Kelas ini memiliki dua buah *method* yaitu SerializeObject() dan DeserializeObject().
- JsonSerializer merupakan kelas yang berfungsi untuk konversi dari JSON String ke objek dan sebaliknya. Kelebihan yang dimiliki kelas ini adalah dapat membaca dan menulis JSON dari *file text*.

2.2 Kiri API

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

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

Pemanfaatan Kiri API dengan melakukan permintaan dengan parameter *POST* atau *GET* lalu Kiri akan 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^[5]. Penggunaan API memungkinkan pengaksesan di mana saja dengan menggunakan koneksi internet. Pada sub bab 2.2.1 sampai sub bab 2.2.3 penulis akan membahas beberapa layanan Kiri API.

Berikut langkah-langkah untuk mendapatkan *API key*.

- Masuk ke situs dev.kiri.travel.
- Register dengan memasukan alamat email, nama, dan nama perusahaan.
- Password akan 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 versi 2
<i>mode</i>	"findroute"	Menginstruksikan 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 2.11*:

Listing 2.11: 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 2.11*:

Ketika pencarian rute sukses dilakukan maka status akan memberitahukan "ok" seperti di baris 2. Selanjutnya setiap langkah dari posisi awal ke posisi tujuan akan ditampung di elemen *array* untuk menampung langkah. Berikut keterangan dari setiap *array* yang menampung langkah.

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

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

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

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

2.2.2 Web Service Pencarian Lokasi

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

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

Tabel 2.6: Tabel parameter layanan pencarian lokasi

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

Listing 2.12: Kode kembalian pencarian lokasi

```

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

Berikut maksud dari *listing 2.12*:

Ketika pencarian lokasi sukses dilakukan maka status akan memberitahukan "ok" seperti di baris 2. Selanjutnya akan 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>	Kota yang akan dicari tempatnya
<i>apikey</i>	16-digit hexadesimal	<i>API key</i> yang digunakan

Tabel 2.7: Tabel parameter layanan menemukan transportasi terdekat

Format kembalian layanan menemukan transportasi terdekat dapat dilihat pada *listing 2.13*.

Listing 2.13: 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 2.13*:

Ketika pencarian rute sukses dilakukan maka status akan memberitahukan "ok" seperti di baris 2. Selanjutnya akan 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 2.13* 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 2.13* 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 id kendaraan yang mana bisa dipakai untuk ditampilkan gambarnya.
- Indeks ke 2 atau baris 7 pada *listing 2.13* berisi nama kendaraan.
- Indeks ke 3 atau baris 8 pada *listing 2.13* berisi jarak dengan satuan kilometer.

BAB 3

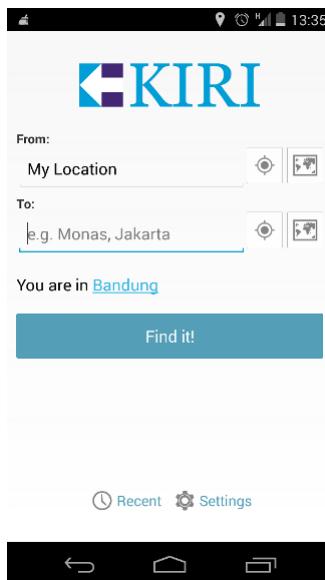
ANALISIS

Pada bab ini akan 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 penulis temui bernama Public Transport¹. Namun aplikasi Public Transport tersebut hanya dapat dijalankan di sistem aplikasi android. Aplikasi Public Transport ini memanfaatkan Kiri API. Aplikasi tersebut penggunaannya sederhana. Di halaman awal pengguna dapat mengetikan lokasi awal dan tujuan. Selain dengan mengetik pengguna juga dapat menunjuk lokasi pada peta. Setelah lokasi dipilih sistem akan memastikan dengan memberi daftar nama jalan dan tempat terkait. Jika sudah memilih maka sistem akan 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

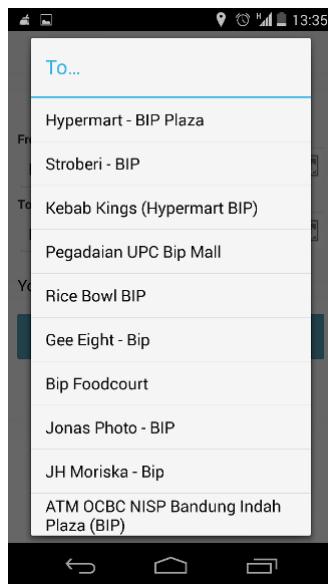
¹<https://play.google.com/store/apps/details?id=travel.kiri.smartransapp>

mengetik dan menunjuk pada peta dengan mengetuk tombol peta. Bila pengguna ingin menunjuk lokasi pengguna berada dapat dilakukan dengan mengetuk tombol kordinat. Tersedia juga pelihan kota yang dapat dipilih oleh pengguna.



Gambar 3.2: Menunjuk lokasi pada peta

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



Gambar 3.3: Memberikan daftar nama tempat dan nama jalan terkait

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

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



Gambar 3.4: Tampilan rute kendaraan umum dalam bentuk daftar lokasi tujuan.



Gambar 3.5: Tampilan rute kendaraan umum di peta

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

3.2.1 Kebutuhan Aplikasi

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

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

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

Aplikasi yang penulis bangun didasarkan pada kebutuhan sebagai berikut.

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

3.2.2 Analisis Kontrol yang Dipakai

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

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

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 penulis akan menggunakan *TextBlock*. *TextBlock* digunakan untuk menampilkan tulisan "from" dan "to" pada halaman utama aplikasi. Untuk masukan pengguna ter hadap aplikasi penulis akan menyediakan *TextBox* sebagai tempat teks. *TextBox* digunakan sebagai masukan untuk lokasi asal dan lokasi tujuan.

Suatu aplikasi tentunya tidak hanya mempunyai satu halaman. Sama hal dengan aplikasi yang penulis buat 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 akan mengeksekusi *event click* yang memungkinkan pindah halaman dan melakukan perintah. Kontrol tombol akan penulis gunakan untuk berpindah ke halaman peta, menemukan lokasi pengguna, dan pencarian rute. Pada Gambar ?? terdapat 5 tombol yaitu tombol map pada bagian from, tombol here pada bagian from, tombol map pada bagian to, tombol here pada bagian to, dan tombol find. Berikut kegunaan dari tombol-tombol tersebut.

- 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 "lokasi dari peta".

- Tombol map 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 "lokasi dari peta".

- Tombol map 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 akam mencari rute angkutan umum dan menampilkannya pada halaman peta.

Pada aplikasi ini penulis akan menampilkan daftar tempat dan daftar rute angkutan umum yang dipakai. Bentuk daftar digunakan penulis karena hasil tempat dan rute angkutan umum akan banyak. Bentuk daftar yang dapat dipakai di Windows Phone adalah menggunakan *ListBox*. *ListBox* akan 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 2.1.4. 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 memasukan lokasi asal dan lokasi tujuan. Setelah memasukan lokasi pengguna akan mencari rute. Ketika rute berhasil ditemukan aplikasi akan berada di keadaan *Running*. Tetapi ada kemungkinan pengguna berpindah aplikasi atau mematikan layar untuk menghemat daya. Dalam kasus tersebut sistem operasi Windows Phone akan menganggap aplikasi tidak aktif dan aplikasi akan masuk pada keadaan *dormant*. Untuk menangani kasus tersebut maka penulis harus menyimpan keadaan dan informasi saat sebelum aplikasi menjadi tidak aktif. Penanganan yang penulis akan lakukan adalah menggunakan *method OnNavigatedFrom()*. Dengan *method* tersebut keadaan aplikasi akan disimpan di memori.

Pada saat aplikasi masuk keadaan *Dormant* semua *thread* dan proses akan dihentikan. Pada saat tersebut juga GPS Windows Phone akan 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 penulis buat akan 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 penulis akan lakukan untuk hal tersebut adalah menggunakan *method OnNavigatedTo()*. Menggunakan *method* tersebut akan 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 penulis pilih untuk aplikasi ini adalah *Road*. Tambahan setelah mengatur pandangan peta yaitu mengatur warna dan penulis akan menggunakan mode warna terang sesuai bawaan peta di Windows Phone.

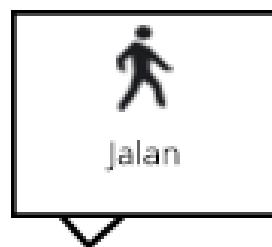
Tampilan awal peta di Windows Phone akan mengeluarkan peta dengan pandangan dunia. Karena aplikasi pencarian rute ini masih terbatas di Pulau Jawa, Indonesia terutama Jawa Barat maka tingkat zoom harus diatur agar mengikuti lokasi pengguna dan di satu daerah saja. Jika pengguna

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 penulis akan gunakan adalah 14. Tingkat zoom 14 akan 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 kesulitan membaca trayek 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 seusai 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 penulis akan memanfaatkan Pushpin sebagai lapisan terpisah untuk menaruh gambar dan keterangan angkutan umum. Berikut tampilan *pushpin* untuk angkot [3.6](#) dan *pushpin* untuk jalan kaki [3.7](#).



Gambar 3.6: Tampilan *pushpin* untuk angkot



Gambar 3.7: Tampilan *pushpin* untuk jalan kaki

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

3.2.5 Analisis Pemanfaatan Sumber Data

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

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

3.2.6 Analisis Kiri API

Kiri API menyediakan 2 parameter untuk permintaan yaitu *POST* dan *GET*. Dalam tugas akhir ini penulis akan menggunakan parameter *GET*. Parameter **GET** penulis pilih karena dalam tugas akhir ini penulis akan banyak mendapatkan data dan tidak ada data sensitif yang dikirimkan. Untuk hal ini penulis akan mengirim ke URL <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 penulis gunakan pada tugas akhir ini adalah 97A7A1157A05ED6F.

Untuk tugas akhir ini penulis akan menggunakan 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 penulis adalah Kiri API versi maka di parameter *version* penulis akan menggunakan 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=97A7A1157A05ED6F](http://kiri.travel/handle.php?version=2&mode=searchplace®ion=cgk/bdo/sub&querystring=\).

Penulis mencoba mencari lokasi bip dari kata kunci "bip" yang berada di bandung. Layanan dikirimkan ke URL [kiri.travel/handle.php](http://kiri.travel/handle.php?version=2&mode=searchplace®ion=bdo&querystring=bip&apikey=97A7A1157A05ED6F). Berikut format layanan yang penulis kirim:

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

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 akan aplikasi tampung namun yang akan ditampilkan ke pengguna hanya *placename*. Menampilkan *location* tidak efektif menurut penulis karena akan membingungkan pengguna. Dari percobaan yang penulis lakukan, nilai dari *attributions* selalu bernilai "null". Karena hal tersebut maka nilai *attributions* akan penulis abaikan.

Pemanfaatan layanan penentuan rute untuk mendapatkan langkah yang harus ditempuh pengguna untuk mencapai lokasi tujuan dari lokasi asal. Pemanfaatan layanan ini yaitu dengan parameter *GET* melalui protokol HTTP. Berikut parameter yang harus dikirim:

- *version*: 2

Karena acuan penulis adalah Kiri API versi maka di parameter *version* penulis akan menggunakan 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 memungkinkan dipakai orang banyak maka penulis putuskan untuk menggunakan bahasa Inggris.

- *start*: koordinat lokasi awal dalam berupa latitude dan longitude.

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

- *finish*: koordinat lokasi tujuan dalam berupa latitude dan longitude.

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

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

Karena aplikasi ini dirancang untuk Windows Phone 8, presentasi yang penulis pilih adalah "mobile".

- *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

Penulis mencoba menuju jalan merdeka dari jalan ciumbuleuit. Layanan dikirimkan ke URL <http://kiri.travel/handle.php?version=2&mode=findroute&locale=en&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

```

1 {
2     "status ":"ok",
3     "routingresults ":[
4         {
5             "steps ":[
6                 [
7                     "walk",
8                     "walk",
9                     ["-6.8747337,107.6048829","-6.87445,107.60465"],
10                     "Walk about 41 meter from your starting point \%fromicon to Jalan Ciumbuleuit \%toicon
11                     .",
12                 ],
13             ],
14         }
15     ]
16 }
```

```

11           null
12       ],
13   [
14     "angkot",
15     "ciumbuleuitsthallurus",
16     ["-6.87445,107.60465", "-6.87541,107.60443", "-6.87637,107.60421", "-6.87734,107.60400",
17     "-6.87830,107.60378", "-6.87926,107.60356", "-6.87926,107.60356", "-6.87963,107.60352",
18     "-6.87978,107.60352", "-6.88093,107.60392", "-6.88209,107.60433", "-6.88209,107.60433",
19     "-6.88328,107.60490", "-6.88328,107.60490", "-6.88347,107.60481", "-6.88452,107.60459",
20     "-6.88556,107.60436", "-6.88660,107.60413", "-6.88764,107.60390", "-6.88764,107.60391",
21     "-6.88782,107.60392", "-6.88887,107.60404", "-6.88991,107.60416", "-6.88991,107.60416",
22     "-6.89161,107.60428", "-6.89161,107.60428", "-6.89166,107.60421", "-6.89275,107.60424",
23     "-6.89275,107.60424", "-6.89405,107.60408", "-6.89405,107.60408", "-6.89496,107.60400"],
24     "Take angkot Ciumbuleuit – St. Hall (lurus) at Jalan Ciumbuleuit \%fromicon, and
25     alight at Jalan Cihampelas
26     \%toicon about 2.3 kilometer later.",
27   ],
28   [
29     "angkot",
30     "kalapaledeng",
31     ["-6.89501,107.60403", "-6.89562,107.60398", "-6.89623,107.60395", "-6.89732,107.60401",
32     "-6.89732,107.60401", "-6.89882,107.60414", "-6.89882,107.60414", "-6.89969,107.60418",
33     "-6.90071,107.60426", "-6.90173,107.60433", "-6.90173,107.60433", "-6.90297,107.60437",
34     "-6.90420,107.60440", "-6.90420,107.60440", "-6.90426,107.60456", "-6.90422,107.60481",
35     "-6.90399,107.60546", "-6.90406,107.60617", "-6.90454,107.60697", "-6.90454,107.60697",
36     "-6.90512,107.60745", "-6.90618,107.60778", "-6.90618,107.60778", "-6.90643,107.60787",
37     "-6.90651,107.60807", "-6.90675,107.60914", "-6.90675,107.60914", "-6.90694,107.60939",
38     "-6.90723,107.60939", "-6.90891,107.60943", "-6.90891,107.60943", "-6.90909,107.60934",
39     "-6.90914,107.60857", "-6.90933,107.60846", "-6.91021,107.60887", "-6.91021,107.60887",
40     "-6.91030,107.60897", "-6.91028,107.60927", "-6.90986,107.61040", "-6.90986,107.61040"],
41     "Take angkot Kalapa – Ledeng at Jalan Cihampelas \%fromicon, and alight at Jalan Aceh
42     \%toicon about 2.3 kilometer later.",
43   ],
44   [
45     [
46       "walk",
47       "walk",
48       ["-6.90986,107.61040", "-6.9114646,107.6104887"],
49       "Walk about 178 meter from Jalan Aceh \%fromicon to your destination \%toicon.",
50     ],
51   ],
52   [
53     "traveltime":"30 minutes"
54   ]
55 ]
56 }

```

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

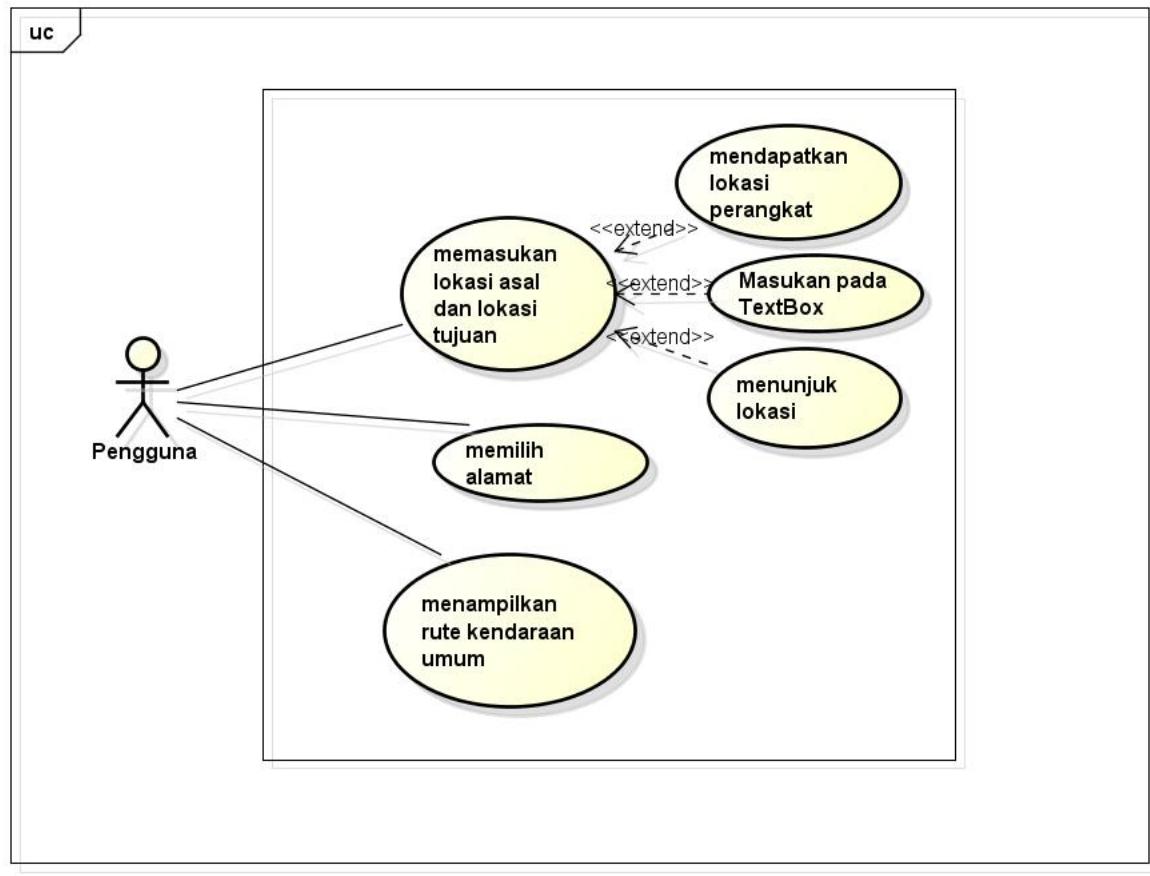
3.2.7 Diagram Use Case dan Scenario

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 awal
Kondisi akhir	Lokasi ditemukan dan TextBox berisi "here"
Skenario utama	Pengguna menekan tombol lalu perangkat akan mencari lokasi perangkat dan TextBox berisi "here"
Eksespsi	lokasi tidak ditemukan jika GPS perangkat tidak aktif

Tabel 3.1: Skenario 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	TextBox masih dalam keadaan belum terisi
Kondisi akhir	Lokasi awal dan tujuan sudah dimasukan
Skenario utama	Pengguna mengetikan lokasi awal dan tujuan pada <i>TextBox</i> yang sudah disediakan
Eksespsi	tidak ada

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

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

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

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

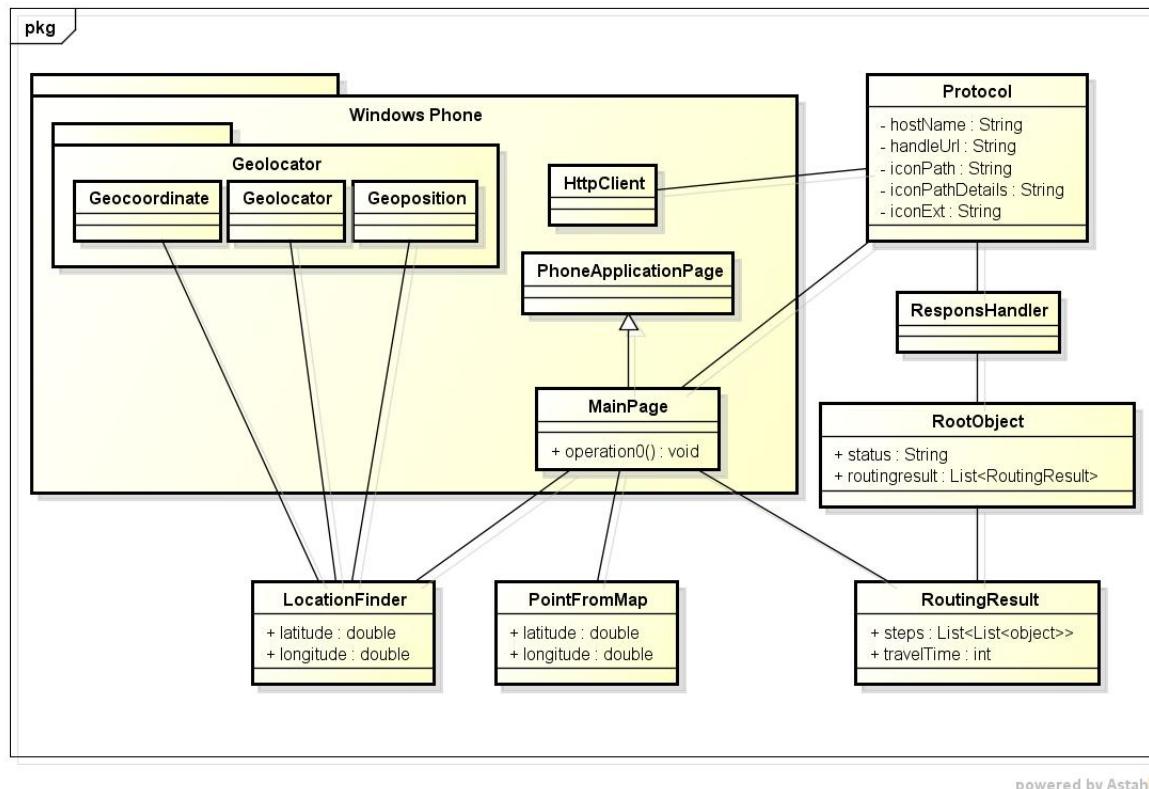
Tabel 3.4: Skenario memilih alamat

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

Tabel 3.5: Skenario menampilkan rute kendaraan umum

3.2.8 Kelas Diagram

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



Gambar 3.9: Diagram Kelas

Berikut deskripsi kelas pada gambar 4.4.

- Kelas *Protocol*

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

- Kelas *ResponsHandler*

Merupakan kelas yang menangani masukan dari pemanggilan layanan.

- Kelas *RootObject*

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

- Kelas *RoutingResult*

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

- Kelas *PointFromMap*

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

- Kelas *LocationFinder*

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

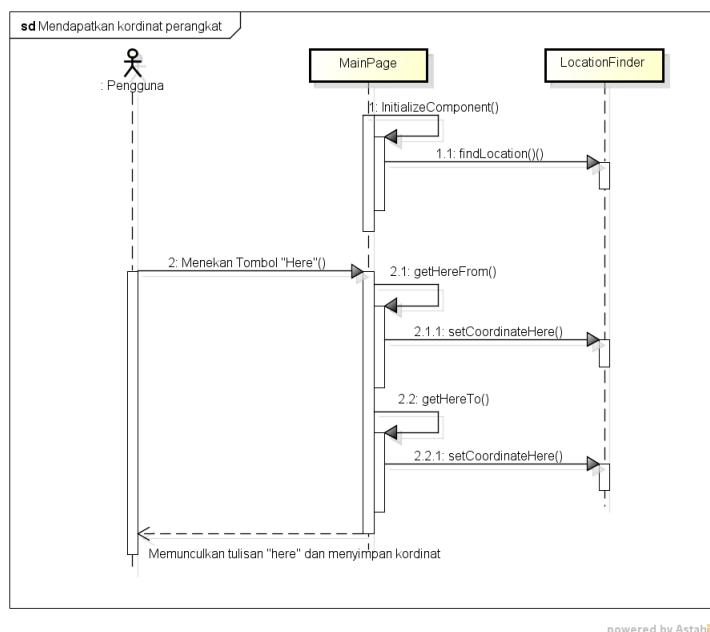
BAB 4

PERANCANGAN

Pada bab 4 akan dibahas mengenai perancangan seperti diagram *sequence* kelas secara rinci dan perancangan antarmuka.

4.1 Diagram *Sequence*

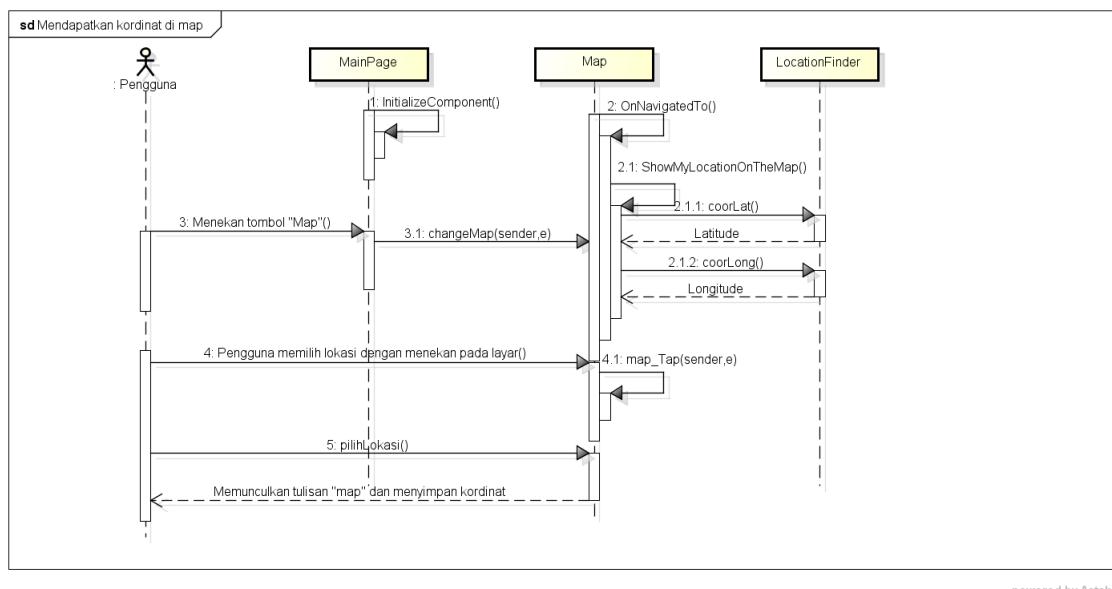
Diagram *sequence* merupakan diagram yang menggambarkan interaksi antar objek dalam suatu skenario. Gambar diagram *sequence* dapat dilihat pada gambar reffig:sequence lokasi perangkat sampai reffig:sequence rute.



Gambar 4.1: Diagram *sequence* Mendapatkan Kordinat perangkat

Diagram 4.1 merupakan diagram *sequence* untuk memilih lokasi dengan lokasi perangkat berada. Diagram menunjukkan bahwa setelah aplikasi dibuka maka aplikasi akan mencari dahulu lokasi perangkat. 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 akan muncul tulisan "here" pada masukan 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. 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 pilihLokasi*. Lokasi yang dipilih pengguna akan disimpan di kelas MainPage dan pada masukan akan tertulis "map".

Diagram 4.3 merupakan diagram *sequence* untuk mencari rute. Diagram menunjukkan bahwa setelah aplikasi dibuka maka aplikasi akan melakukan inisialisasi. Selanjutnya pengguna juga dapat memasukan kata kunci untuk masukan asal dan tujuan (masukan dapat juga dari peta atau lokasi perangkat sesuai diagram 4.1 dan 4.2) lalu memanggil *method startRoute*. Jika masukan yang didapat berupa kata kunci maka akan dilakukan pemeriksaan apakah kordinat untuk kata kunci tersebut tersedia. Pemeriksaan dilakukan dengan melakukan pemanggilan Kiri API. Tahap pemanggilan meliputi pemanggilan *method GetStringAsync* lalu mengjadikan objek kembalinya dengan *method Deserialize*. Jika sudah didapat dan hasilnya lebih dari satu maka akan dipanggil *method getListItem* yang akan menampilkan daftar pilihan ke pengguna untuk dipilih. Pengguna dapat memilih tempat sesuai tempat asal maupun tujuan yang diinginkan. Setelah lokasi asal dan lokasi tujuan didapat maka kelas MainPage akan mengarahkan ke kelas Route untuk menampilkan hasilnya. Kelas Route akan memanggil *method OnNavigatedTo* yang bertujuan untuk mendapatkan lokasi asal dan lokasi tujuan. Setelah itu akan memanggil *method Find* lalu mengembalikan rute yang ditemukan kepada pengguna.

4.2 Perancangan Kelas

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

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 akan ditaruh kontrol yang diperlukan. Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

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

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

1. Konstruktor MainPage digunakan untuk memuat komponen yang ada di halaman MainPage dan mendapatkan kota terdekat dari lokasi perangkat.
2. *method* OnNavigatedTo digunakan untuk mendapatkan lokasi dari peta dan mendapatkan objek LocationFinder. *method* ini memiliki parameter NavigationEventArgs.
3. *method* startRoute digunakan untuk mendapatkan masukan pengguna. Jika masukan didapat dari peta atau lokasi perangkat berarti sudah dalam kordinat, namun jika masukan didapat dari *query* maka akan dicari lokasi yang terkait sesuai *query* tersebut. Lokasi terkait yang didapatkan akan dikembalikan ke pengguna untuk dipilih. Setelah pengguna lokasi dalam bentuk kordinat didapatkan maka kelas akan diarahkan ke kelas Route. *method* ini memiliki parameter objek tombol dan event.
4. *method* changeMapFrom digunakan untuk berpindah ke halaman mapFrom. *method* ini memiliki parameter objek tombol dan event.

5. *method* changeMapTo digunakan untuk berpindah ke halaman mapTo. *method* ini memiliki parameter objek tombol dan event.
6. *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.
7. *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.
8. *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.
9. *method* ListBoxSelectedPlace digunakan untuk mendapatkan tempat asal yang dipilih pengguna. *method* ini memiliki parameter objek dan *event SelectionChangedEventArgs*.
10. *method* searchCoordinatePlace digunakan untuk mencari kordinat dari tempat pilihan pengguna. *method* ini memiliki parameter ListBox dan tempat yang dipilih dalam bentuk string.
11. *method* findRoute digunakan untuk berpindah ke kelas Route jika lokasi asal dan lokasi tujuan sudah ditentukan.
12. *method* changeCity digunakan untuk mengubah kota tujuan dari pencarian. *method* ini memiliki parameter objek dan *event SelectionChangedEventArgs*.
13. *method* showCity digunakan untuk mencari kota yang paling dekat dengan lokasi perangkat.
14. *method* ShowSplash digunakan untuk menampilkan tampilan awal untuk proses inisialisasi aplikasi.
15. *method* StartLoadingData digunakan untuk memanggil BackgroundWorker. BackgroundWorker digunakan untuk melakukan aksi di belakang layar.
16. *method* backgroundWorker_DoWork digunakan untuk melakukan pemanggilan aksi di belakang layar. *method* ini memiliki parameter objek dan DoWorkEventArgs.
17. *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 kendaraa 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 index 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 akan 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 sesuak kembalian *Geolocator*.

4.2.8 Kelas *LocationFinder*

LocationFinder merupakan kelas yang akan 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 akan 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 akan 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 akan ditentukan dari map untuk kemudian ditampung di Objek LocationFinder. *method* ini memiliki sebuah parameter NavigationEventArgs.
3. *method* ShowMyLocationOnTheMap digunakan untuk memberitahu dan menandai lokasi perangkat.
4. *method* map _ Tap berfungsi untuk menandai lokasi yang ditunjuk pengguna lalu menerjemahkan lokasi yang ditunjuk pengguna pada peta dan mengirimnya ke kelas LocationFinder.
5. *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 mengkonversi 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 permitaan 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* getTypeTransportWOBalloon merupakan *method* yang akan mengembalikan alamat dari gambar transportasi tanpa bingkai tambahan. *method* ini memiliki 2 parameter yaitu means sebagai tipe transportasi dan meansDetail sebagai nama kendaraan.
3. getSearchPlace merupakan *method* yang akan mengembalikan URI pencarian lokasi sesuai parameter. Parameter yang dimaksud adalah kata kunci masukan pengguna.
4. *method* getFindRoute merupakan *method* yang akan mengembalikan URI pencarian rute sesuai parameter. Parameter yang dimaksud adalah kordinat lokasi asal dan kordinat lokasi tujuan yang bertipe string.
5. *method* getRequestSearch digunakan untuk mendapatkan lokasi terkait sesuai masukan pengguna. *method* ini akan mengembalikan Task<RootObjectSearchPlace> karena menggunakan operasi *asynchronous*. *method* ini memiliki parameter kata kunci masukan pengguna dan kota yang masing-masing parameter bertipe String.
6. *method* getRequestRoute digunakan untuk mendapatkan rute sesuai lokasi asal dan lokasi tujuan. *method* ini akan mengembalikan Task<RootObjectFindRoute> karena menggunakan operasi *asynchronous*. *method* ini memiliki parameter *latitude* lokasi asal, *longitude* lokasi asal, *latitude* lokasi tujuan, dan *longitude* lokasi tujuan yang masing-masing bertipe double.

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
2. routingresults

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
2. traveltime

4.2.17 Kelas *Route*

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

1. listBoxStatus bertipe boolean digunakan untuk status rute dalam bentuk daftar sedang ter-tutup atau terbuka.
2. IFinder 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. 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* TrackLocation digunakan untuk inisislisasi GeoLocator dan dan memulai pelacakan lokasi terus menerus.
4. *method* geolocator_StatusChanged digunakan untuk mengetahui status GeoLocator.
5. *method* geolocator_PositionChanged digunakan untuk mengetahui perubahan lokasi yang terjadi dan menyimpannya di kelas LocationFinder.
6. *method* Find digunakan untuk mencari rute yang dicari pengguna.
7. *method* setRouteToMap digunakan untuk menggambar rute dan lokasi pada *layer* peta. *method* ini memiliki parameter RootObjectFindRoute yang merupakan objek untuk pencarian rute.
8. *method* setFocus digunakan untuk mengarahkan pusat pandangan ke titik lokasi pergantian jenis transportasi. *method* ini memiliki parameter objek dan RoutedEventArgs.
9. *method* toMyLocation digunakan untuk mengarahkan pusat pandangan ke titik lokasi perangakt berada. *method* ini memiliki parameter objek dan RoutedEventArgs.
10. *method* ShowList digunakan untuk membuka dan menutup rute dalam bentuk daftar. *method* ini memiliki parameter objek dan RoutedEventArgs.
11. *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.

12. *method* drawMyLocationOnTheMap digunakan untuk membuat penanda lokasi di lapisan myLocationLayer pada peta. *method* ini memiliki parameter latitude bertipe double dan longitude bertipe double.
13. *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.
14. *method* ShowLoading digunakan untuk memunculkan *popup* menunggu.
15. *method* StartLoadingData digunakan untuk pemanggilan BackgroundWorker.
16. *method* backgroundWorker_DoWork digunakan untuk eksekusi *method* dengan BackgroundWorker. *method* ini memiliki dua buah parameter sender bertipe objek dan e bertipe DoWorkEventArgs.
17. *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.
18. *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 akan dibahas mengenai antarmuka pada aplikasi Pencari Rute Kendaraan Umum untuk Windows Phone. Antarmuka berfungsi sebagai jembatan yang menghubungkan antara aplikasi dengan pengguna. Berikut ini akan dijelaskan mengenai rancangan antarmuka aplikasi Pencari Rute Kendaraan Umum untuk Windows Phone.

4.3.1 Antarmuka Kelas *MainPage*

Antarmuka Kelas Map pada gambar 4.5 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 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 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 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.

Antarmuka Daftar Tempat pada gambar [4.6](#) merupakan daftar yang akan 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 [4.7](#) merupakan antarmuka untuk menunjuk lokasi pada peta. Terdapat satu buah tombol yang akan dimunculkan jika pengguna sudah memilih lokasi. Jika tombol ditekan maka kordinat lokasi akan di simpan dan dikirim pada kelas MainPage dan halaman akan diarahkan ke kelas MainPage.

4.3.3 Antarmuka Kelas *Route*

Antarmuka Kelas Route pada gambar [4.8](#) 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.

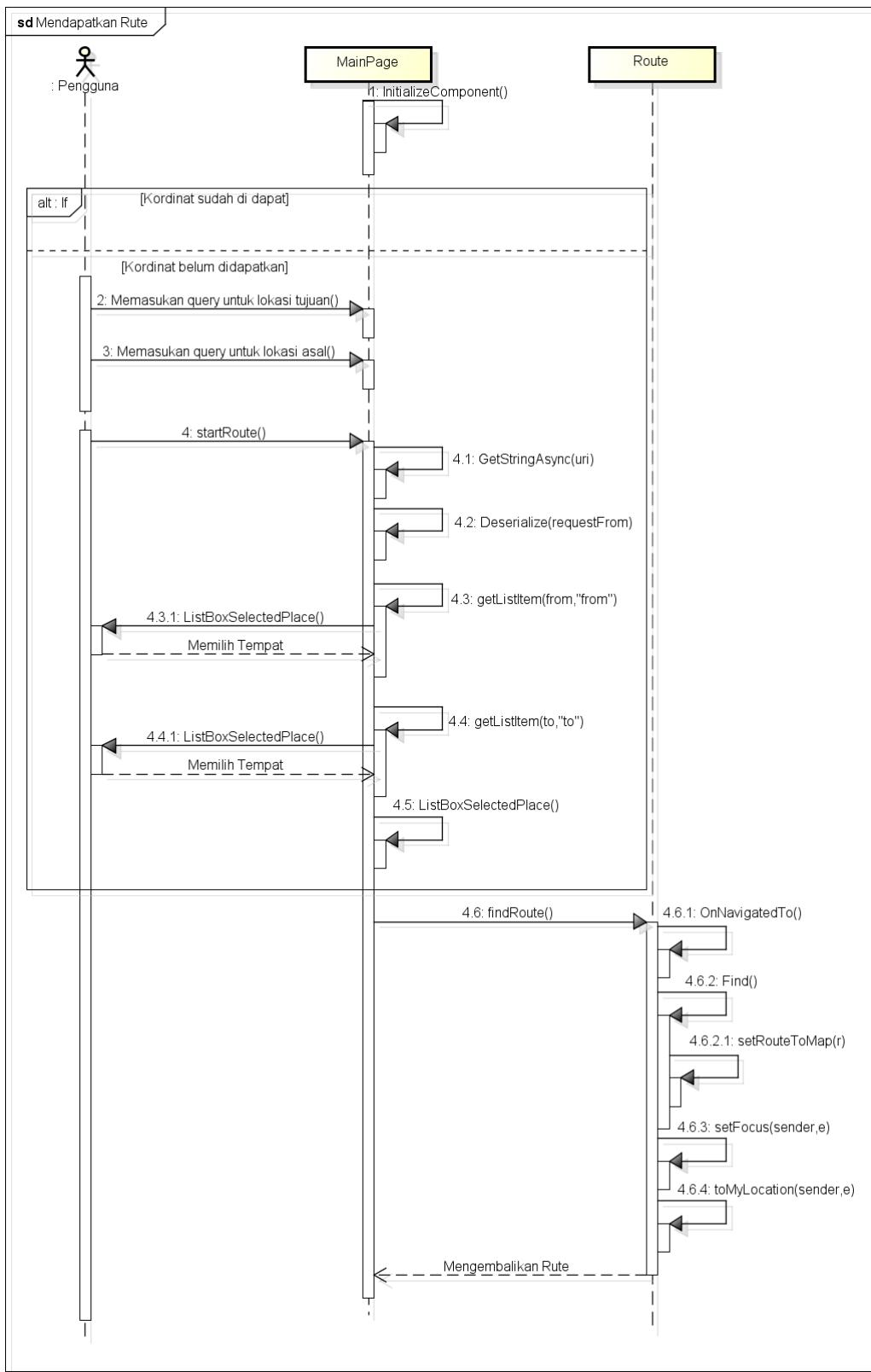
- Tombol here

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

- Tombol Show List

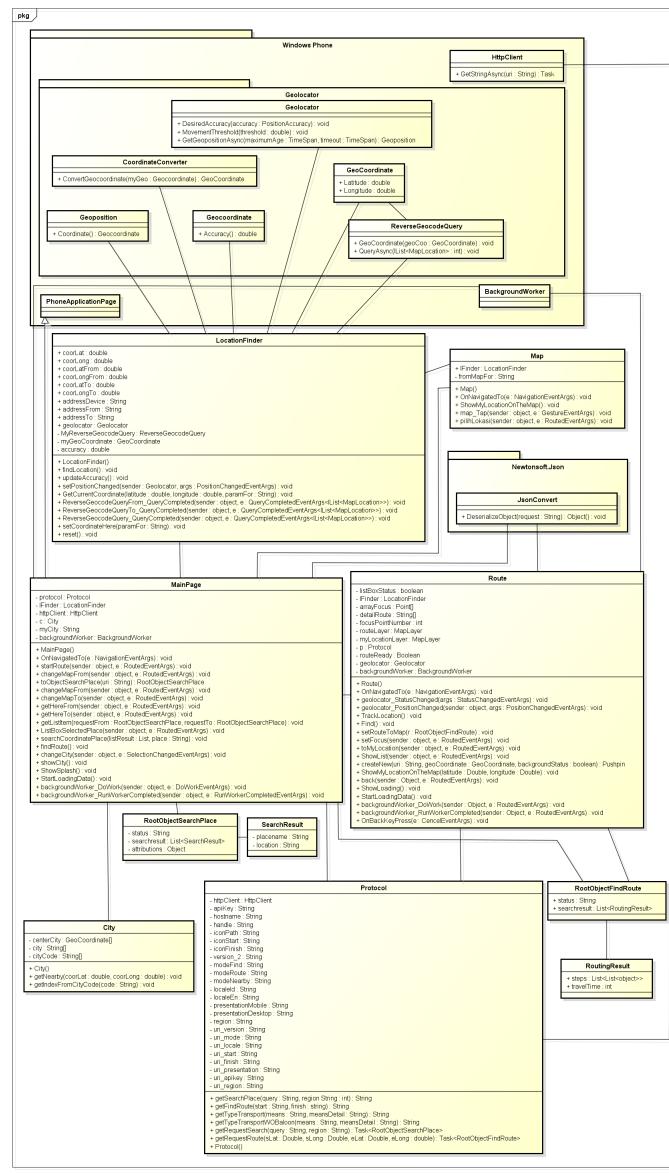
Jika tombol ditekan maka akan menunjuk atau menyembunyikan daftar rute.

Antarmuka rute dalam bentuk daftar pada gambar 4.9 merupakan antarmuka untuk melihat rute secara lebih jelas dengan keterangan tahap demi tahap disertai jarak dan waktu perjalanan. Antarmuka daftar dapat dilihat atau disembunyikan sesuai keinginan pengguna namun saat kelas Rute dibuka antarmuka daftar rute akan disembunyikan.



powered by Astah

Gambar 4.3: Diagram *sequence* Mendapatkan Rute



Gambar 4.4: Diagram Kelas

Gambar 4.5: Antarmuka *MainPage*

Pilih Tempat

Kota 1

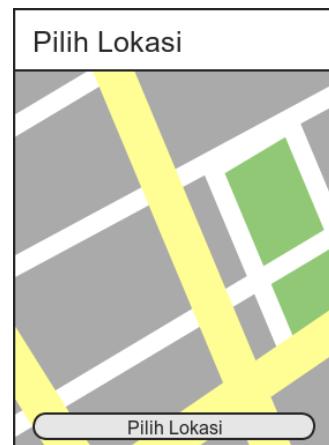
Kota 2

Kota 3

Kota 4

Kota 5

Gambar 4.6: Antarmuka Daftar Tempat



Gambar 4.7: Antarmuka Map



Gambar 4.8: Antarmuka Route



Gambar 4.9: Antarmuka Rute dalam bentuk Daftar

BAB 5

IMPLEMENTASI DAN PENGUJIAN APLIKASI

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

5.1 Implementasi

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

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 Bergerak
 - (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 Bergerak

- (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 dan kemampuan penulis.

2. Hasil kompilasi program

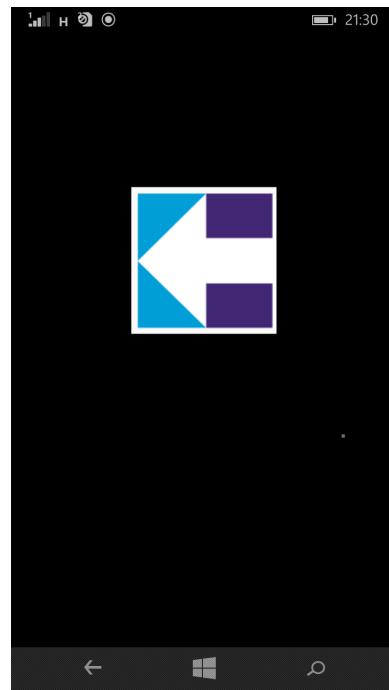
Hasil dari kompilasi program berupa *file* Kiri_Debug_AnyCPU.xap. *File* ini dapat dipasang pada perangkat dengan sistem operasi Windows Phone versi 8 atau lebih tinggi.

3. Antarmuka Aplikasi

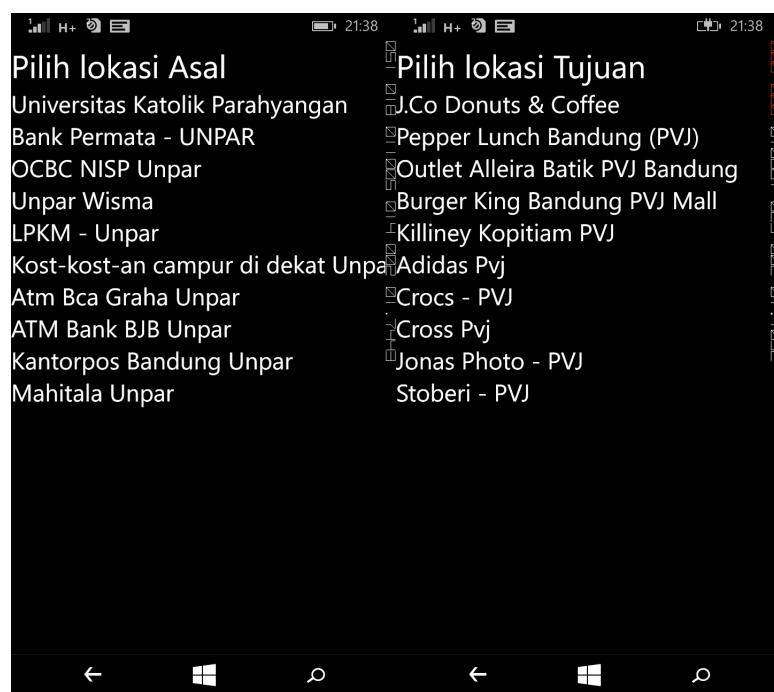
Berikut merupakan hasil implementasi antarmuka aplikasi Pencari Rute Kendaraan Umum untuk Windows phone.



Gambar 5.1: Gambar antarmuka kelas MainPage



Gambar 5.2: Gambar antarmuka Splash di kelas MainPage



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



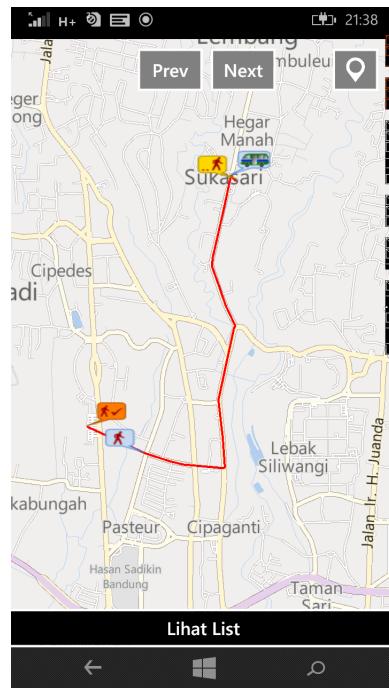
Gambar 5.4: Gambar antarmuka kelas Map



Gambar 5.5: Gambar antarmuka menunggu di kelas Route

5.2 Pengujian

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



Gambar 5.6: Gambar antarmuka kelas Route



Gambar 5.7: Gambar antarmuka *list* di kelas Route

etahui keberhasilan proses kerja dari aplikasi.

5.2.1 Lingkungan Pengujian

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

1. Processor : 1.2 Ghz Quad Core

2. RAM : 1 GB
3. Layar : 1280 x 720 *pixels*, 4,7 inch
4. GPS : A-GPS, GLONASS, Beidou

5.2.2 Pengujian Fungsional

Pengujian fungsional dilakukan untuk menguji kesesuaian reaksi yang terjadi dengan reaksi yang diharapkan. Hasil pengujian tersebut ditunjukan pada tabel 5.1.

5.2.3 Pengujian Experimental

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

1. Pengujian 1

- Tempat : Rumah(Jalan Kejaksaan menuju Unpar)
- Waktu : Pukul 9 pada tanggal 7 April 2014 Pada pengujian 1 penulis memasukan alamat di dalam rumah untuk menuju Unpar. Saat penulis memasukan kata Unpar muncul daftar tempat sesuai kata kunci "Unpar", lalu penulis memilih "Universitas Katolik Parahyangan".

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

Tabel 5.1: Tabel Hasil pengujian fungsionalitas

BAB 6

KESIMPULAN DAN SARAN

6.1 Kesimpulan

6.2 Saran

BIBLIOGRAFI

- [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] “Kiri api v2 documentation.” https://bitbucket.org/projectkiri/kiri_api/wiki/KIRI%20API%20v2%20Documentation, 2014. Accessed: 2014-8-17.