

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 | 3 |
| 1.5 Metode Penelitian | 3 |
| 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 | 26 |
| 3 ANALISIS | 29 |
| 3.1 Analisis Aplikasi Sejenis | 29 |
| 3.2 Analisis Aplikasi | 32 |
| 3.2.1 Kebutuhan Aplikasi | 32 |
| 3.2.2 Analisis Kontrol yang Dipakai | 32 |
| 3.2.3 Analisis Terhadap Siklus Hidup Aplikasi | 33 |
| 3.2.4 Analisis Peta | 34 |
| 3.2.5 Analisis Pemanfaatan Sumber Data | 36 |
| 3.2.6 Analisis Kiri API | 36 |
| 3.2.7 Diagram <i>Use Case</i> dan Scenario | 39 |
| 3.2.8 Kelas Diagram | 43 |
| 4 PERANCANGAN | 45 |
| 4.1 Perancangan Kelas | 45 |
| 4.1.1 Kelas <i>PhoneApplicationPage</i> | 46 |

| | | |
|--------|------------------------------------|----|
| 4.1.2 | Kelas <i>MainPage</i> | 46 |
| 4.1.3 | Kelas <i>Geocoordinate</i> | 46 |
| 4.1.4 | Kelas <i>Geolocator</i> | 46 |
| 4.1.5 | Kelas <i>Geoposition</i> | 46 |
| 4.1.6 | Kelas <i>LocationFinder</i> | 46 |
| 4.1.7 | Kelas <i>PointFromMap</i> | 47 |
| 4.1.8 | Kelas <i>HttpClient</i> | 47 |
| 4.1.9 | Kelas <i>Protocol</i> | 47 |
| 4.1.10 | Kelas <i>ResponsHandler</i> | 48 |
| 4.1.11 | Kelas <i>RootObjectSearchPlace</i> | 48 |
| 4.1.12 | Kelas <i>SearchResult</i> | 48 |
| 4.1.13 | Kelas <i>RootObjectFindRoute</i> | 48 |
| 4.1.14 | Kelas <i>RoutingResult</i> | 49 |

BIBLIOGRAFI

51

DAFTAR GAMBAR

| | | |
|-----|--|----|
| 1.1 | Statistik Pengguna Windows Phone | 2 |
| 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 [3] | 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 Kelas | 45 |

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 mendapatkan 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 | 42 |

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*.

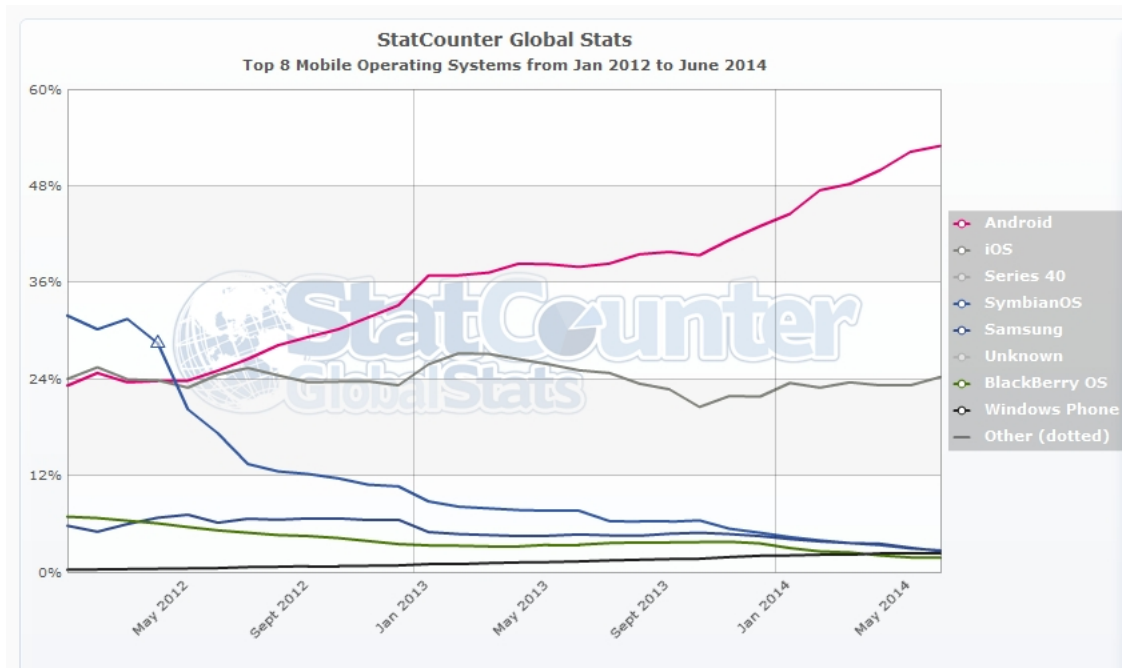
1.1 Latar Belakang

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. Meskipun jumlah penggunanya masih belum sebanyak pengguna Android dan IOS tapi jumlah penggunanya terus naik di tahun 2014 ini. Statistik peningkatan jumlah pengguna di Windows Phone dari tahun 2012 hingga 2014² dapat dilihat pada gambar 1.1.

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



Gambar 1.1: Statistik Pengguna Windows Phone

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.

1.2 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 lunak di Windows Phone lalu mengembangkan aplikasi yang akan dibuat.
- Membuat aplikasi di di Windows Phone yang memanfaatkan Kiri API.

²http://gs.statcounter.com/#mobile_os-ww-monthly-201201-201406

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 dijelaskan pada bab ini adalah Windows Phone dan Kiri API. Teori Windows Phone yang dijelaskan meliputi lingkungan kerja, XAML, kontrol terhadap ponsel, siklus hidup aplikasi, peta di Windows Phone, lokasi, dan memanfaatkan sumber data. Teori Kiri API yang dijelaskan meliputi *web service* penentuan rute, *web service* pencarian lokasi, dan *web service* menemukan transportasi terdekat.

Bab 3 membahas tentang analisis pembangunan aplikasi Pencarian Rute Kendaraan Umum untuk Windows Phone. Pada Bab 3 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[3].

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[3]. 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 :

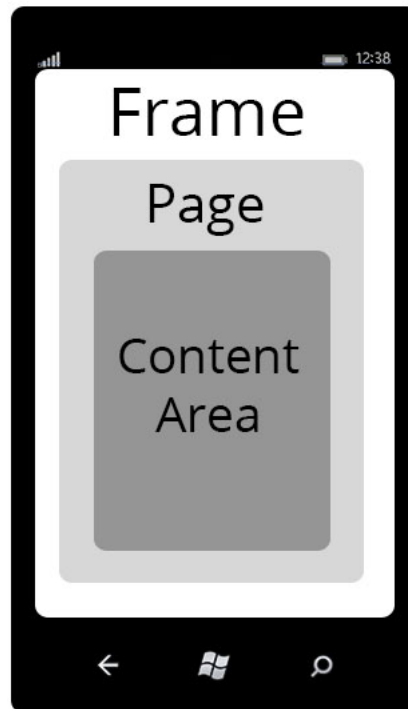
- *x:Key* : sebuah nama unik untuk menunjuk referensi ke suatu resource atau berkas lain. Nilai ini dapat dipanggil kembali untuk menggunakan resource tersebut.
- *x:Class* : menunjukkan nama kelas.
- *x>Name* : menunjukkan nama sebuah obyek dan untuk membedakan antar obyek yang satu dengan obyek yang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

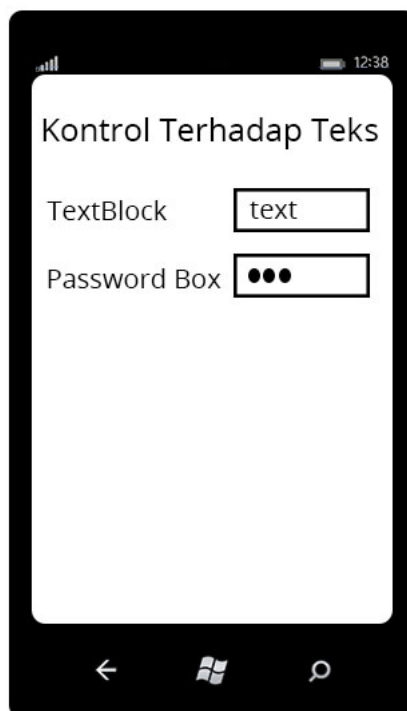
Listing 2.1: Kode tata letak grid

```
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*

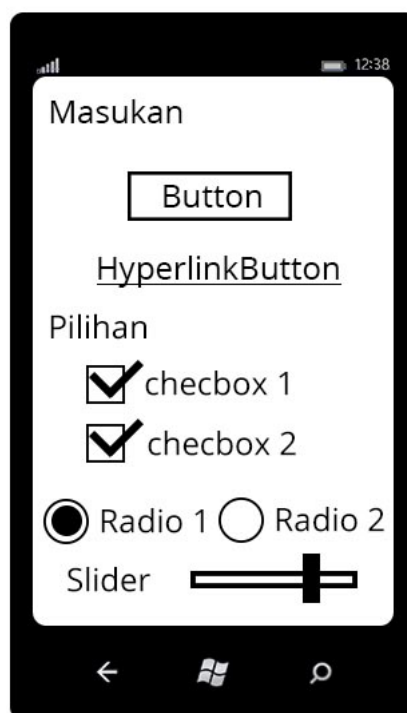
Listing 2.2: Kode untuk menampilkan *TextBlock* dan *TextBox*

```
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 penggulangan terhadap daftar yang panjang.

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

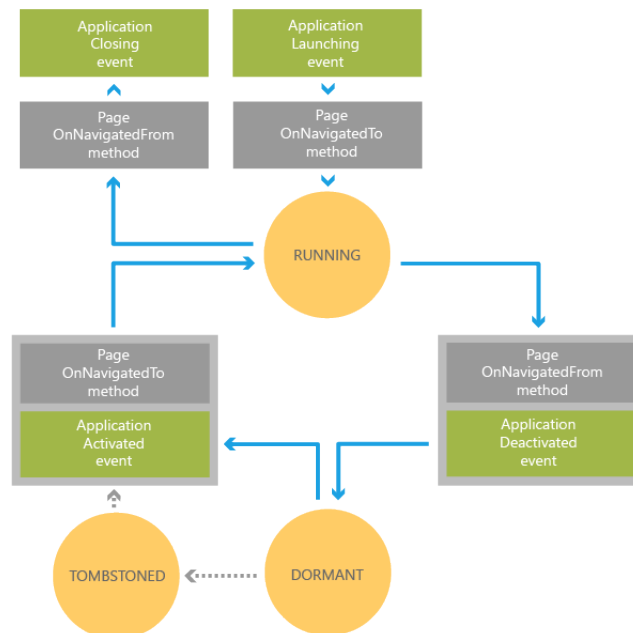
```

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 menunjukkan 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.

- Metode *OnNavigatedFrom*

Merupakan metode yang dipanggil ketika berpindah ke halaman lain aplikasi. Ketika metode 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*

Metode ini dipanggil ketika pengguna berpindah ke halaman yang sebelumnya ditinggalkan. Metode 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 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 [3]

Listing 2.9: Kode untuk menampilkan pushpin

```

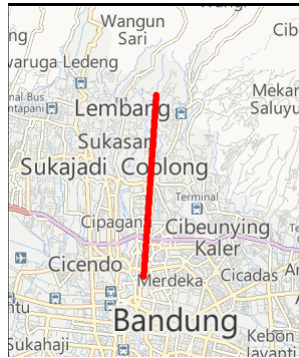
1  MapOverlay overlay = new MapOverlay
2  {
3      GeoCoordinate = map.Center,
4      Content = new Border
5      {
6          BorderBrush = new SolidColorBrush(Color.FromArgb(120, 255, 0, 0)),
7          Child = new TextBlock() {Text="Pushpin"},
8          BorderThickness = new Thickness(1),
9     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

- *SetView(LocationRectangle)*

Metode untuk mengatur pandangan di atas peta secara spesifik sesuai wilayah geografis. Metode ini tidak mengembalikan nilai.

- *SetView(GeoCoordinate, Double)*

Metode untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah dan tingkat zoom. Metode ini tidak mengembalikan nilai.

- *SetView(LocationRectangle, MapAnimationKind)*

Metode untuk mengatur pandangan di atas peta secara spesifik sesuai region geografis dan animasi. Metode 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)*
Metode untuk mengatur pandangan di atas peta secara spesifik sesuai region geografis dengan batas tertentu. Metode ini tidak mengembalikan nilai.
- *SetView(GeoCoordinate, Double, MapAnimationKind)*
Metode untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah, tingkat zoom, dan animasi. Metode ini tidak mengembalikan nilai.
- *SetView(GeoCoordinate, Double, Double)*
Metode untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah, tingkat zoom, dan *heading*. Metode ini tidak mengembalikan nilai.
- *SetView(LocationRectangle, Thickness, MapAnimationKind)*
Metode untuk mengatur pandangan di atas peta secara spesifik sesuai wilayah geografis dengan batas tertentu, dan animasi. Metode ini tidak mengembalikan nilai.
- *SetView(GeoCoordinate, Double, Double, MapAnimationKind)*
Metode untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah, tingkat zoom, *heading*, dan animasi. Metode ini tidak mengembalikan nilai.
- *SetView(GeoCoordinate, Double, Double, Double)*
Metode untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah, tingkat zoom, *heading*, *pitch*. Metode ini tidak mengembalikan nilai.
- *SetView(GeoCoordinate, Double, Double, Double, MapAnimationKind)*
Metode untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah, tingkat zoom, *heading*, *pitch*, dan animasi. Metode ini tidak mengembalikan nilai.
- *UpdateLayout*
Metode 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.Maps.Controls*.

Berikut properti yang dapat digunakan pada kelas ini.

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

Tabel 2.2: Properti *Polyline Class*

Berikut metode yang dapat digunakan pada kelas ini.

- *CheckAccess*
Metode yang menentukan bisa atau tidaknya pemanggilan *thread* untuk mengakses objek.
- *ClearValue*
Metode yang akan membersihkan nilai lokal
- *Finalize*
Metode 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 penyedia 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 di lihat pada tabel 2.3.

2.1.6.3 Geocoordinate

Geocoordinate adalah kelas yang menunjukan lokasi sebagai kordinat geografis. Kelas ini hanya menyediakan properti yang hanya bisa dibaca. Kelas ini menyediakan properti yang ditunjukkan 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 metode 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 menangannya 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 *uint*.

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 metode 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[3].

Banyak *web service* yang mengembalikan data dalam format JSON. JSON memiliki struktur yang mudah dipahami. 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 metode yang disediakan kelas *HttpClient*.

- *DeleteAsync(Uri)*
Metode yang dipakai untuk mengirimkan permintaan DELETE ke URI yang spesifik sebagai operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan aplikasi untuk melanjutkan pekerjaan selagi metode ini dipanggil². Metode 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)*

Metode yang dipakai untuk mengirimkan permintaan *GET* ke URI yang spesifik sebagai operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan aplikasi untuk melanjutkan pekerjaan selagi metode ini dipanggil². Metode 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,HttpCompletionOption)*

Metode yang dipakai untuk mengirimkan permintaan *GET* ke URI yang spesifik sebagai operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan aplikasi untuk melanjutkan pekerjaan selagi metode ini dipanggil². Metode ini membutuhkan parameter URI sebagai tujuan dari permintaan dan nilai tambahan yang dimaksudkan sebagai indikasi operasi dianggap selesai. Sedangkan kembaliannya berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut memiliki 2 parameter yaitu hasil berupa pesan kembalian dari http dan kemajuan dari data yang dikirim.

- *GetBufferAsync(Uri)*

Metode yang dipakai untuk mengirimkan permintaan *GET* ke URI yang spesifik sebagai operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan aplikasi untuk melanjutkan pekerjaan selagi metode ini dipanggil². Metode ini membutuhkan parameter URI sebagai tujuan dari permintaan. Sedangkan kembaliannya berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut memiliki 2 parameter yaitu hasil berupa pesan kembalian yang dikirimkan secara buffer(disimpan dalam memori) dan kemajuan dari data yang dikirim.

- *GetInputStreamAsync(Uri)*

Metode yang dipakai untuk mengirimkan permintaan *GET* ke URI yang spesifik sebagai operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan aplikasi untuk melanjutkan pekerjaan selagi metode ini dipanggil². Metode ini membutuhkan parameter URI sebagai tujuan dari permintaan. Sedangkan kembaliannya berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut memiliki 2 parameter yaitu hasil berupa pesan kembalian yang dikirimkan secara stream(langsung sesuai waktu) dan kemajuan dari data yang dikirim.

- *GetStringAsync(Uri)*

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

- *PostAsync(Uri)*

Metode yang dipakai untuk mengirimkan permintaan *POST* ke URI yang spesifik sebagai

operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan aplikasi untuk melanjutkan pekerjaan selagi metode ini dipanggil². Metode ini membutuhkan parameter URI sebagai tujuan dari permintaan. Sedangkan kembaliannya berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut memiliki 2 parameter yaitu hasil berupa pesan kembalian dari http dan kemajuan dari data yang dikirim.

- *SendRequestAsync(HttpRequestMessage)*

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

- *SendRequestAsync(HttpRequestMessage, HttpCompletionOption)*

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

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.

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[2]. 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.

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

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 veris 2 |
| <i>mode</i> | "findroute" | Mengintruksikan layanan untuk mencari rute |
| <i>locale</i> | "en" or "id" | Bahasa yang digunakan untuk balasan |
| <i>start</i> | lat,lng | Titik awal <i>latitude</i> dan <i>longitude</i> |
| <i>finish</i> | lat,lng | Titik akhir <i>latitude</i> dan <i>longitude</i> |
| <i>presentation</i> | "mobile" or "desktop" | Menentukan tipe presentasi untuk keluaran. Contoh, jika tipe presentasi "mobile", maka link "tel:" akan ditambahkan di hasil. |
| <i>apikey</i> | 16-digit hexadecimal | <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 menunjukan ikon "from". Biasanya untuk mode presentasi di perangkat bergerak.
 - %toicon = untuk menunjukan ikon "to". Biasanya untuk mode presentasi di perangkat bergerak.
- Indeks ke 4 atau baris 11 pada *listing 2.11* berisi URL untuk pemesanan tiket jika tersedia. Jika tidak tersedia akan bernilai *null*.

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

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

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

2.2.2 Web Service Pencarian Lokasi

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

| | | |
|--------------------|---|---|
| <i>version</i> | 2 | Memberitahukan bahwa layanan yang dipakai adalah protokol 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 veris 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 hexadecimal | <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

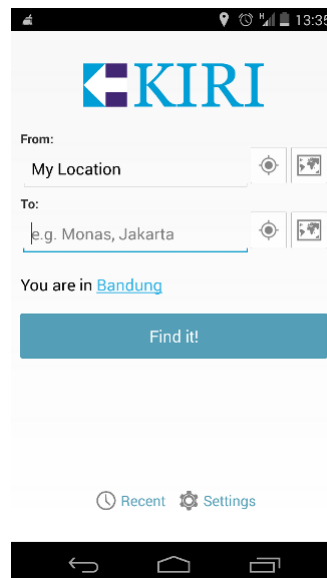
ANALISIS

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 mengetikkan 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

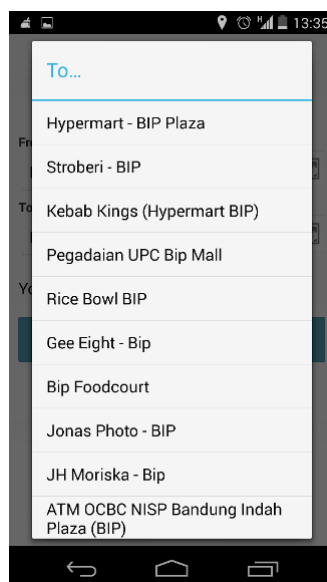
¹<https://play.google.com/store/apps/details?id=travel.kiri.smarttransportapp>

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



Gambar 3.2: Menunjuk lokasi pada peta

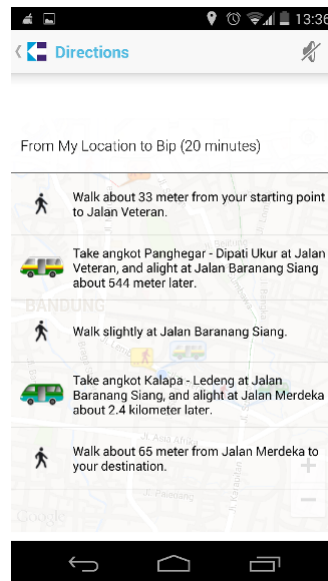
Gambar 3.2 jika pengguna sudah mengetahui lokasi namun tidak tahu nama lokasi. Pada halaman ini pengguna diarahkan untuk menemukan lokasi pada peta dan mengetuk lokasi tersebut untuk memilihnya.



Gambar 3.3: Memberikan daftar nama tempat dan nama jalan terkait

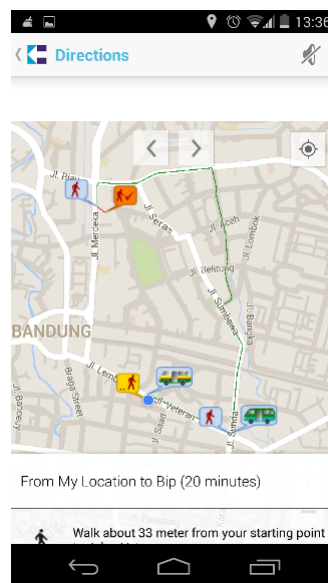
Pada gambar 3.3 pengguna dapat memilih nama tempat terkait. Pemilihan didasarkan sesuai masukan pengguna untuk memastikan tempat asal maupun tempat tujuan. Jika nama tempat sudah jelas maka tidak akan ada halaman ini.

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 diposisisikan 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 mengeluarkan teks yang dapat dilihat penulis akan menggunakan *TextBlock*. *TextBlock* digunakan untuk menampilkan tulisan "from" dan "to" pada halaman utama aplikasi. Untuk masukan pengguna terhadap 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 akan 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 memasukkan lokasi asal dan lokasi tujuan. Setelah memasukkan 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 metode *OnNavigatedFrom()*. Dengan metode 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 metode *OnNavigatedTo()*. Menggunakan metode 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.

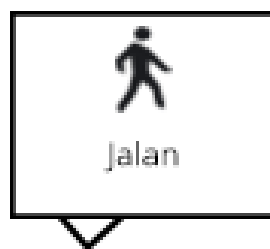
Tampilan awal peta di Windows Phone akan mengeluarkan peta dengan pandangan dunia. Karena aplikasi pencarian rute ini masih terbatas di Pulau Jawa, Indonesia terutama Jawa Barat maka tingkat *zoom* harus diatur agar mengikuti lokasi pengguna dan di satu daerah saja. Jika pengguna

berada di daerah Bandung maka tingkat zoom pada peta disesuaikan pada daerah tersebut. Tingkat zoom dapat 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 sesuai dengan setiap rute. Gambar angkot tersebut akan ditempatkan di peta dengan suatu penampung beserta keterangannya. Salah satu teknik untuk menempatkan gambar tersebut adalah dengan membuat lapisan terpisah di atas peta tempat gambar tersebut. Untuk hal tersebut 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 menggambarinya. *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*.

Metode yang penulis gunakan adalah *GetStringAsync()*. Metode ini akan mengirimkan permintaan melalui URI dan mengembalikan hasilnya dalam tipe data *String* dan kemajuan data. Karena metode 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.

Penulis mencoba mencari lokasi bip dari kata kunci "bip" yang berada di bandung. Layanan dikirimkan ke URL kiri.travel/handle.php. 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 [kiri.travel/handle.php](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 format layanan yang penulis kirim <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 |         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 |         null
28 |     ],
29 |     [
30 |         "angkot",
31 |         "kalapaledeng",
32 |         ["-6.89501,107.60403","-6.89562,107.60398","-6.89623,107.60395","-6.89732,107.60401",
33 |         "-6.89732,107.60401","-6.89882,107.60414","-6.89882,107.60414","-6.89969,107.60418",
34 |         "-6.90071,107.60426","-6.90173,107.60433","-6.90173,107.60433","-6.90297,107.60437",
35 |         "-6.90420,107.60440","-6.90420,107.60440","-6.90426,107.60456","-6.90422,107.60481",
36 |         "-6.90399,107.60546","-6.90406,107.60617","-6.90454,107.60697","-6.90454,107.60697",
37 |         "-6.90512,107.60745","-6.90618,107.60778","-6.90618,107.60778","-6.90643,107.60787",
38 |         "-6.90651,107.60807","-6.90675,107.60914","-6.90675,107.60914","-6.90694,107.60939",
39 |         "-6.90723,107.60939","-6.90891,107.60943","-6.90891,107.60943","-6.90909,107.60934",
40 |         "-6.90914,107.60857","-6.90933,107.60846","-6.91021,107.60887","-6.91021,107.60887",
41 |         "-6.91030,107.60897","-6.91028,107.60927","-6.90986,107.61040","-6.90986,107.61040"],
42 |         "Take angkot Kalapa - Ledeng at Jalan Cihampelas \"%fromicon, and alight at Jalan Aceh
43 |         \"%toicon about 2.3 kilometer later.",
44 |         null
45 |     ],
46 |     [
47 |         "walk",
48 |         "walk",
49 |         ["-6.90986,107.61040","-6.9114646,107.6104887"],
50 |         "Walk about 178 meter from Jalan Aceh \"%fromicon to your destination \"%toicon.",
51 |         null
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.

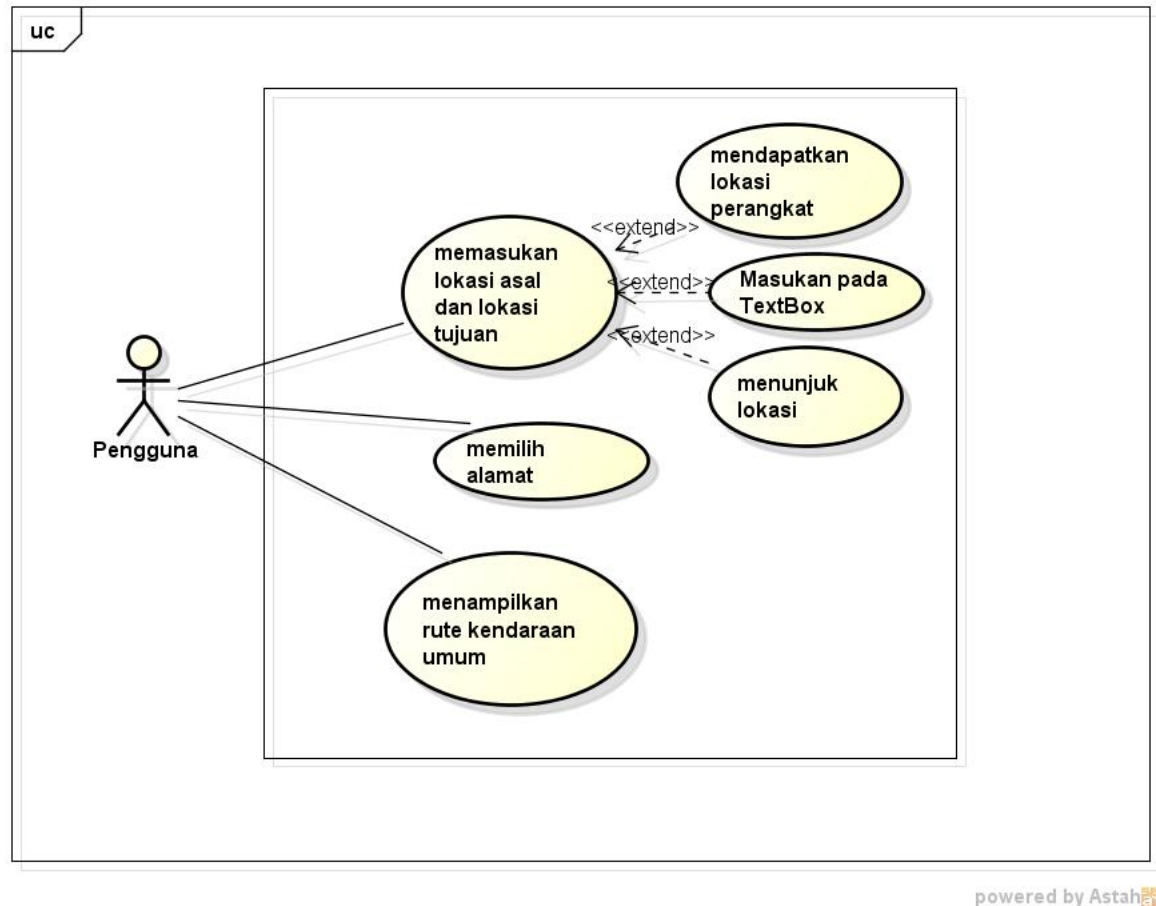
3.2.7 Diagram *Use Case* dan Scenario

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):



Gambar 3.8: Diagram *use case*

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

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

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

| | |
|----------------|---|
| 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 TextBox 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 |

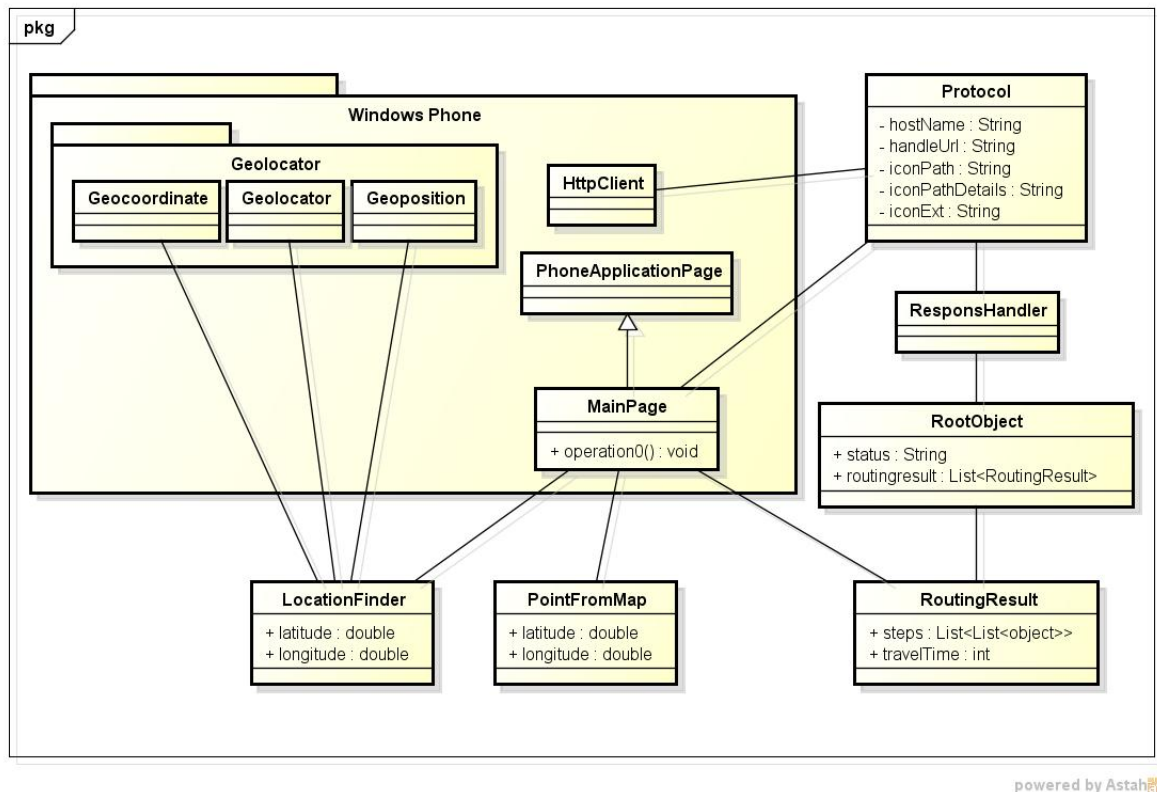
Tabel 3.4: Skenario memilih alamat

| | |
|----------------|--|
| 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 4.1.



Gambar 3.9: Diagram Kelas

Berikut deskripsi kelas pada gambar 4.1.

- 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*.

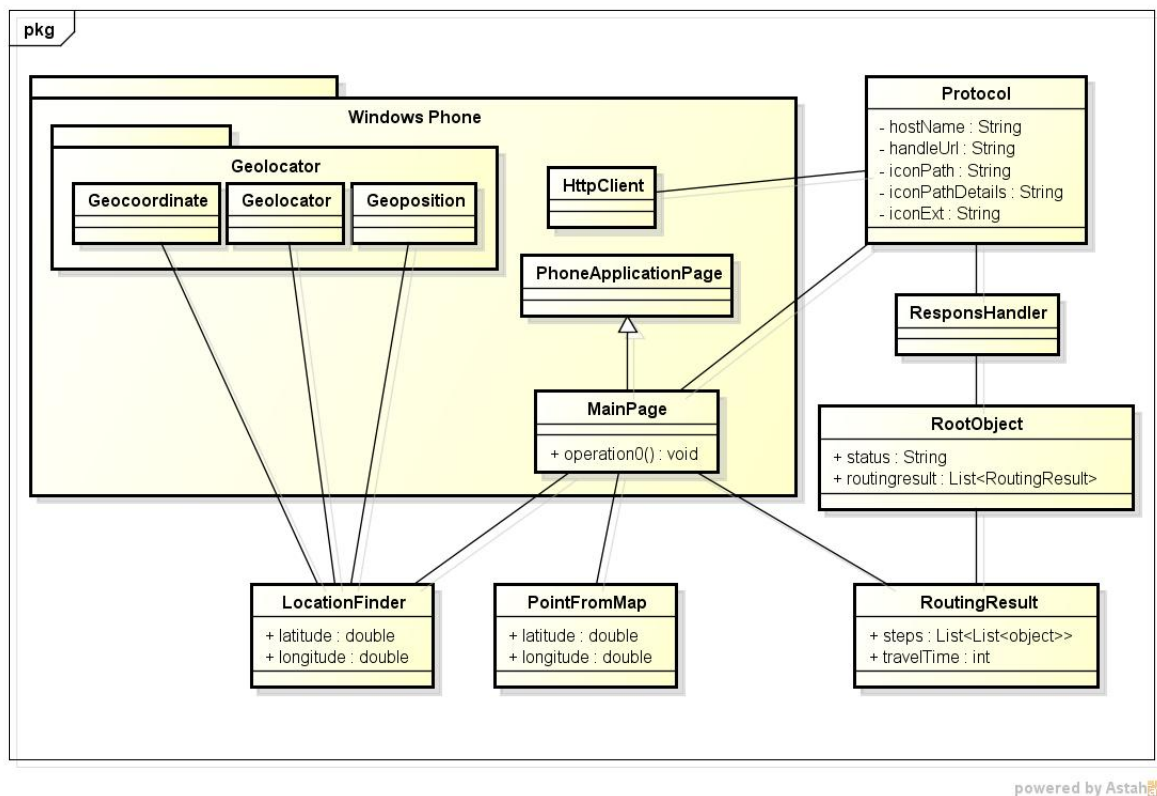
BAB 4

PERANCANGAN

Pada bab 4 akan dibahas mengenai perancangan seperti kelas secara rinci dan perancangan antar-muka.

4.1 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 4.1.



powered by Astah

Gambar 4.1: Diagram Kelas

4.1.1 Kelas *PhoneApplicationPage*

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

4.1.2 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. *listPlaceFrom* bertipe *ListBox* untuk menampung dan menampilkan hasil pencarian lokasi terkait dari lokasi asal.
2. *listPlaceTo* bertipe *ListBox* untuk menampung dan menampilkan hasil pencarian lokasi terkait dari lokasi tujuan
3. *locFrom* bertipe *string* untuk menampung kordinat asal.
4. *locTo* bertipe *string* untuk menampung kordinat tujuan.

4.1.3 Kelas *Geocoordinate*

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

4.1.4 Kelas *Geolocator*

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

4.1.5 Kelas *Geoposition*

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

4.1.6 Kelas *LocationFinder*

LocationFinder merupakan kelas yang akan menampung lokasi perangkat. Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

1. *status* bertipe *boolean* sebagai penanda apakah GPS perangkat sudah siap.
2. *latitude* bertipe *long* untuk menampung kordinat latitude.
3. *longitude* bertipe *long* untuk menampung kordinat longitude.

Berikut adalah penjelasan metode-metode yang dimiliki kelas ini:

1. Metode *onInit* berfungsi untuk inisialisasi GPS perangkat dan memastikan bahwa GPS perangkat siap untuk menemukan lokasi.
2. Metode *getCoordinate* berfungsi untuk mendapatkan kordinat latitude dan longitude dengan memanfaatkan kelas *Geocoordinate*, *Geolocator*, dan *Geoposition*.

4.1.7 Kelas *PointFromMap*

PointFromMap 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. latitude
2. longitude

Berikut adalah penjelasan metode-metode yang dimiliki kelas ini:

1. Metode `getPoint` berfungsi mengambil titik yang ditunjuk lalu menerjemahkan dalam bentuk latitude dan longitude.

4.1.8 Kelas *HttpClient*

HttpClient merupakan kelas bawaan Windows Phone untuk mengatur pengiriman dan kembalian menggunakan protokol HTTP. Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

1. a
2. b

Berikut adalah penjelasan metode-metode yang dimiliki kelas ini:

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

4.1.9 Kelas *Protocol*

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

1. `hostname` bertipe string digunakan untuk 'http://kiri.travel/'.
2. `apiKey` bertipe string digunakan untuk menyimpan kunci untuk mengirim permintaan ke Kiri.
3. `iconPath` bertipe string digunakan untuk
4. `iconStart`
5. `iconFinish`
6. `version`
7. `modeFind`
8. `modeRoute`
9. `modeNearby`
10. `localeEn`

11. localeId
12. start
13. finish
14. presentation
15. region
16. query

Berikut adalah penjelasan metode-metode yang dimiliki kelas ini:

1. get

4.1.10 Kelas *ResponsHandler*

ResponsHandler merupakan kelas untuk menangani kembalian dari server Kiri. Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

1. a

4.1.11 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.

4.1.12 Kelas *SearchResult*

SearchResult merupakan kelas untuk menampung nama tempat dan koordinat 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.

4.1.13 Kelas *RootObjectFindRoute*

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

1. status
2. routingresults

4.1.14 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

BIBLIOGRAFI

- [1] Microsoft *Windows Phone Silverlight development* 2014 : <http://msdn.microsoft.com/library/windows/apps/ff402535.aspx>.
- [2] Kiri Team *KIRI API v2 Documentation* 2014 : https://bitbucket.org/projectkiri/kiri_api/wiki/KIRI%20API%20v2%20Documentation.
- [3] Manning, Paul *Pro Windows Phone App Development* 2013: Apress.
- [4] Szostak, Tomasz *Windows Phone 8 Application Development Essentials* 2013: PACKT.