

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 GAMBAR

DAFTAR TABEL

¹

BAB 1

²

PENDAHULUAN

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

⁷ **1.1 Latar Belakang**

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

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

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

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

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

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

³⁴ peta dan daftar.

³⁵ 1.2 Rumusan Masalah

³⁶ Sehubung dengan latar belakang diatas timbul permasalahan sebagai berikut:

- ³⁷ • Bagaimana membuat aplikasi di Windows Phone?
- ³⁸ • Bagaimana mengintegrasikan Kiri API dengan aplikasi pencari rute kendaraan umum di Windows Phone?

⁴⁰ 1.3 Tujuan

⁴¹ Berdasarkan rumusan masalah pada sub bab 1.2, maka tujuan dari pembuatan tugas akhir ini adalah:

- ⁴³ • Mempelajari cara pembuatan perangkat lunak di Windows Phone lalu mengembangkan aplikasi yang akan dibuat.
- ⁴⁵ • Membuat aplikasi di Windows Phone yang memanfaatkan Kiri API.

⁴⁶ 1.4 Batasan Masalah

⁴⁷ Ruang lingkup pengembangan aplikasi Pencari Rute Kendaraan untuk Windows Phone ini dibatasi hal berikut:

- ⁴⁹ • Aplikasi ini akan berjalan di sistem operasi Windows Phone 8.
- ⁵⁰ • Aplikasi ini membutuhkan koneksi internet.
- ⁵¹ • Aplikasi ini akan menampilkan rute jalur angkot, bus umum dan travel di tiga kota besar yaitu Bandung, Jakarta, dan Surabaya.

⁵³ 1.5 Metode Penelitian

⁵⁴ Metode Penelitian yang penulis gunakan dalam membuat tugas akhir ini adalah sebagai berikut:

- ⁵⁶ • Melakukan studi pustaka mengenai XAML, kontrol dan navigasi di Windows Phone, Peta di Windows Phone, GPS di Windows Phone dan Kiri API.
- ⁵⁸ • Melakukan analisis terhadap aplikasi lain yang menggunakan Kiri API.
- ⁵⁹ • Melakukan analisis terhadap dasar teori untuk pembangunan aplikasi Pencarian Rute Kendaraan Umum untuk Windows Phone.
- ⁶¹ • Melakukan perancangan aplikasi Pencarian Rute Kendaraan Umum untuk Windows Phone.
- ⁶³ • Implementasi dari aplikasi Pencarian Rute Kendaraan Umum untuk Windows Phone.

- 64 • Menguji aplikasi Pencarian Rute Kendaraan Umum untuk Windows Phone.
- 65 • Membuat kesimpulan.

66 1.6 Sistematika Penulisan

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

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

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

81

BAB 2

82

DASAR TEORI

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

87 **2.1 Windows Phone**

88 Windows Phone merupakan sistem operasi untuk perangkat bergerak yang dikembangkan
89 Microsoft. Pengembangan aplikasi Windows Phone membutuhkan Windows Desktop
90 8 sebagai media pengembangan. Bahasa pemrograman yang digunakan untuk membuat
91 perangkat lunak di Windows Phone yaitu C# atau Visual Basic[?].

92 Pada sub bab ?? sampai ?? akan membahas pemrograman di Windows Phone. Pembahasan akan dimulai dengan apa itu Windows Phone dan fitur di Windows Phone yang akan
93 digunakan dalam pembangunan perangkat lunak Pencarian Rute Kendaraan di Windows
94 Phone.

96 **2.1.1 Lingkungan Kerja**

97 Microsoft .NET framework merupakan sebuah perangkat lunak yang dibangun untuk
98 membantu dalam pembangunan aplikasi di Windows, Windows Phone, Windows Server,
99 and Microsoft Azure[?]. Microsoft .NET framework terdiri dari runtime bahasa umum dan
100 perpustakaan kelas .NET Framework, yang meliputi kelas, interface, dan jenis nilai yang
101 mendukung berbagai teknologi. Microsoft .NET Framework menyediakan lingkungan yang
102 mudah dikelola, pengembangan yang disederhanakan, dan integrasi dengan berbagai bahasa
103 pemrograman, termasuk Visual Basic dan Visual C#.

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

107 Untuk masalah kehandalan keduanya menawarkan kehandalan yang baik. Kelebihan dari
108 Visual Basic adalah bahasa pemrograman berorientasi objek yang kuat dan memiliki banyak
109 pengembangan fitur di inheritance, polymorphism, interfaces, and overloading[?]. Kelebihan
110 dari C# yang merupakan pengembangan dari C/C++ adalah sederhana, modern, aman dan
111 berorientasi objek[?]. Satu hal yang dirasakan penulis adalah kenyamanan ketika memilih
112 bahasa .NET tersebut. Akan lebih mudah bagi developer yang menggunakan Visual Basic
113 6.0 untuk menggunakan Visual

114 Basic .NET. Tetapi bagi developer yang menggunakan C++ atau java sebelumnya akan
115 lebih mudah menggunakan C#.

116 2.1.2 XAML

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

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

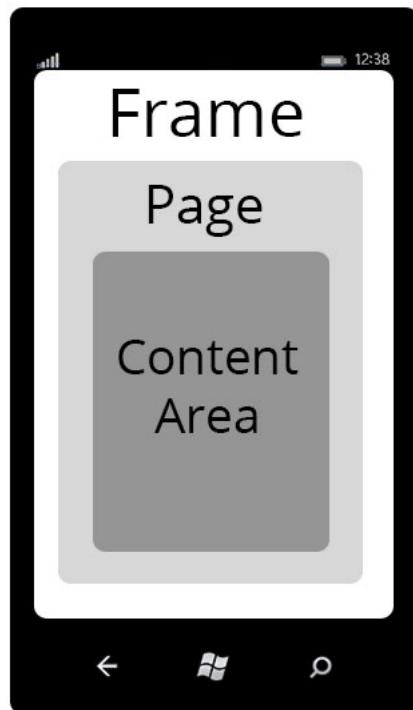
- 128 • x:Key : sebuah nama unik untuk menunjuk referensi ke suatu resource atau berkas
129 lain. Nilai ini dapat dipanggil kembali untuk menggunakan resource tersebut.
- 130 • x:Class : menunjukkan nama kelas.
- 131 • x:Name : menunjukkan nama sebuah obyek dan untuk membedakan antar obyek yang
132 satu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

136 2.1.3 Kontrol terhadap Ponsel

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

140 2.1.3.1 Navigasi

141 Aplikasi yang dibuat di Windows Phone didasarkan pada model halaman. Model ha-
142 laman adalah pengguna berpindah dari satu halaman ke halaman lain dengan konten yang
143 berbeda dan *frame* sebagai pengontrolnya. Setiap antarmuka aplikasi dibungkus dengan
144 *frame*. *Frame* ini yang akan melakukan kontrol terhadap aplikasi dan memungkinkan per-
145 pindahan dari satu halaman ke halaman lain. Sedangkan halaman merupakan pembungkus
146 dari elemen di dalamnya saja. Untuk lebih jelas mengenai *frame*, halaman, dan area konten
147 dapat dilihat pada gambar ??.



Gambar 2.1: Hirarki navigasi

¹⁴⁸ 2.1.3.2 Kontrol Tata Letak

¹⁴⁹ Kontrol tata letak merupakan penampung pada antarmuka Windows Phone untuk objek
¹⁵⁰ di antarmuka dan kontrol yang lain (tombol radio, *textbox*, dan lain-lain). Kontrol tata
¹⁵¹ letak digunakan untuk meletakan objek-objek di layar. Ketika pertama membuat aplikasi
¹⁵² Windows Phone maka tata letak dasar sebagai penampung akan langsung dibuat berikut
¹⁵³ panel judul dan panel konten. Selanjutnya untuk penambahan kontrol tata letak yang lain
¹⁵⁴ dapat ditambahkan di panel konten.

¹⁵⁵ Terdapat 3 jenis panel yang digunakan untuk menangani tata letak yaitu *Grid*, *Stack-¹⁵⁶*
¹⁵⁷ *Panel*, dan *Canvas*. Perlu diperhatikan bahwa setiap halaman hanya memiliki satu jenis
 panel. Berikut adalah 3 jenis panel pada Windows Phone:

- ¹⁵⁸ • *StackPanel* merupakan panel yang memposisikan element menjadi 1 baris dan beberapa elemen di setiap halaman diposisikan horizontal atau vertical saja.
- ¹⁶⁰ • *Grid* merupakan panel yang mendukung tata letak yang rumit. Panel ini memposisikan elemen di baris dan kolom mana saja di setiap halaman.
- ¹⁶² • *Canvas* memposisikan elemen sebagai absolut kordinat. Jadi setiap elemen di dalam *canvas* dapat diposisikan spesifik sesuai kordinat x dan y.

¹⁶⁴ Kode untuk mengatur jenis panel pada Windows Phone dapat dilihat pada *listing ??*.

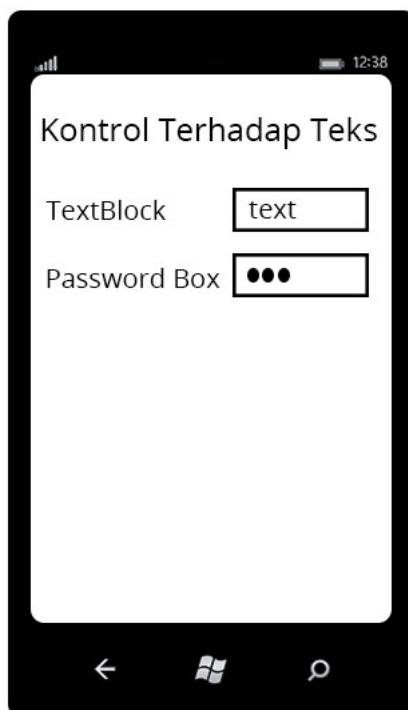
Listing 2.1: Kode tata letak grid

```
165 | <Grid x:Name="ContentPanel">
166 | </Grid>
```

¹⁶⁷ **2.1.3.3 Kontrol Terhadap Teks**

¹⁶⁸ Kontrol terhadap teks akan menampilkan konten yang memiliki tipe *String*. Terdapat
¹⁶⁹ berbagai macam kontrol terhadap teks di Windows Phone yaitu *TextBlock*, *TextBox* dan
¹⁷⁰ *PasswordBox*. Ketiga macam kontrol tersebut dibedakan berdasarkan tujuannya. Berikut
¹⁷¹ keterangan, gambar ??, dan kode pada *listing* ?? kontrol teks.

- ¹⁷² • *TextBlock* merupakan tempat menaruh potongan teks yang hanya bisa dilihat.
¹⁷³ • *TextBox* biasanya digunakan untuk teks masukan yang pendek. Tapi bisa juga dipakai
¹⁷⁴ untuk masukan yang banyak dan beberapa baris.
¹⁷⁵ • *PasswordBox* biasanya digunakan untuk masukan yang bersifat rahasia. Karakter
¹⁷⁶ yang dimasukan langsung disamarkan menjadi bentuk titik.



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

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

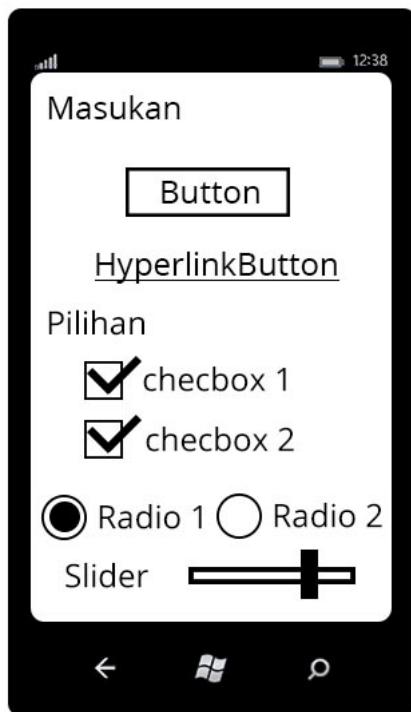
```
177 | <TextBlock x:Name="TextBlock1" Text="TextBlock"/>
178 | <TextBox x:Name="TextBox1" Text="TextBox"/>
```

¹⁷⁹ **2.1.3.4 Tombol dan Kontrol Pilihan**

¹⁸⁰ Tombol memungkinkan pengguna untuk berpindah halaman. Sedangkan kontrol pilihan
¹⁸¹ memudahkan dalam memilih. Berikut keterangan, gambar ??, dan kode pada *listing* ??
¹⁸² tombol dan kontrol pilihan.

- ¹⁸³ • *Button* merupakan kontrol yang dipakai pengguna untuk mengaktifkan *event* tekan.

- 184 • *HyperlinkButton* merupakan kontrol yang menampilkan tautan. Jika *HyperlinkButton*
 185 ditekan maka akan berpindah ke halaman yang akan dituju.
- 186 • *CheckBox* merupakan kontrol yang memungkinkan pengguna memilih beberapa item.
- 187 • *RadioButton* merupakan kontrol yang memungkinkan pengguna memilih satu pilihan
 188 dari beberapa pilihan.
- 189 • *Slider* merupakan kontrol yang memungkinkan pengguna memilih nilai kisaran dari
 190 jalur yang sudah disediakan.



Gambar 2.3: Gambar kontrol pada Windows Phone

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

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

192 2.1.3.5 Kontrol Daftar

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

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

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

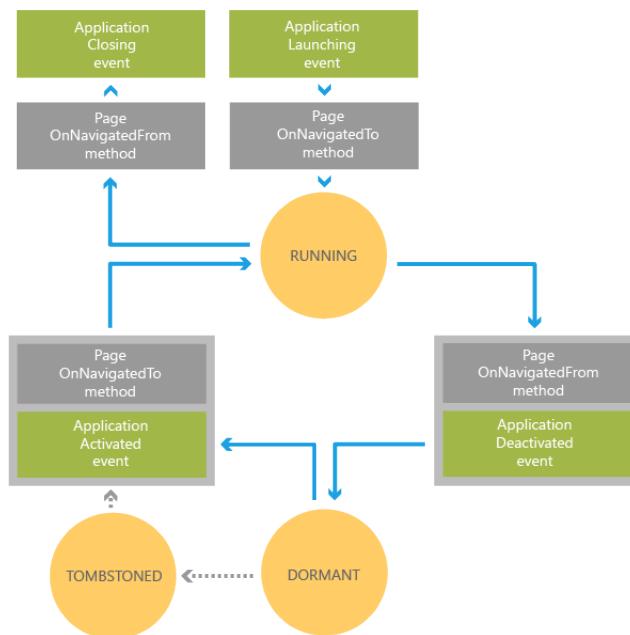
```

198 <ListBox>
199     <ListBoxItem Content="Item 1" />
200     <ListBoxItem Content="Item 2" />
201     <ListBoxItem Content="Item 3" />
202     <ListBoxItem Content="Item 4" />
203     <ListBoxItem Content="Item 5" />
204 </ListBox>

```

205 2.1.4 Siklus Hidup Aplikasi

206 Siklus hidup aplikasi merupakan waktu mulai dari aplikasi dijalankan sampai dengan
 207 aplikasi diberhentikan dari memori. Siklus hidup aplikasi penting diketahui agar pengguna
 208 tidak kecewa (dalam aplikasi ini yaitu kecewa karena aplikasi keluar saat layar perangkat
 209 dimatikan) menggunakan aplikasi serta memastikan sumber daya tersedia (dalam aplikasi ini
 210 yaitu sumber daya GPS). Seringkali pengguna tidak berhati-hati dalam menggunakan aplikasi
 211 kasi, maka dari itu penulis harus memahami kapan aplikasi harus diaktifkan, ditangguhkan,
 212 atau bahkan dinonaktifkan karena sudah tidak digunakan. Gambar ?? adalah ilustrasi dari
 213 siklus hidup pada Windows Phone.



Gambar 2.5: Gambar siklus hidup aplikasi[?]

214 Sesuai gambar ?? lingkaran melambangkan keadaan aplikasi, persegi panjang menun-
 215 jukan peristiwa aplikasi atau tingkat peristiwa di halaman. Berikut ini adalah keterangan
 216 untuk siklus hidup Windows Phone pada gambar ??.

- 217 • *The Launching Event*

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

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

223 ● *Running*

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

229 ● *method OnNavigatedFrom*

230 Merupakan *method* yang dipanggil ketika berpindah ke halaman lain aplikasi. Ketika
231 *method* ini dipanggil maka aplikasi akan menyimpan keadaan dari halaman sebelum
232 ditinggalkan. Hal tersebut dibutuhkan agar halaman tersebut bisa dikembalikan ke
233 keadaan sebelum ditinggalkan saat pengguna ingin kembali ke halaman tersebut. Pe-
234 manggilan dilakukan ketika berpindah antara halaman di aplikasi atau ketika berpin-
235 dah aplikasi.

236 ● *The Deactivated Event*

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

243 ● *Dormant*

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

249 ● *Tombstoned*

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

252 ● *The Activated Event*

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

255 ● *The OnNavigatedTo Method*

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

259 ● *The Closing Event*

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

262 **2.1.5 Peta di Windows Phone**

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

267 **2.1.5.1 Penambahan Peta Ke Aplikasi**

268 Untuk penambahan peta pada Windows Phone menggunakan kontrol peta. Kontrol peta
269 merupakan bagian dari *library* Windows Phone. Dengan begitu untuk dapat menggunakan-
270 nya perlu direferensikan. Untuk dapat menggunakannya juga harus ditambah *capability*
271 ID_CAP_MAP. Setelah hal tersebut dilakukan barulah peta dapat ditampilkan. Berikut
272 gambar ??, kode XAML pada *listing* ??, dan kode program pada *listing* ?? peta.



Gambar 2.6: Tampilan peta pada Windows Phone

Listing 2.5: Menampilkan peta dengan nama MyMap dari XAML

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

Listing 2.6: Menampilkan peta dengan nama MyMap dari kode program

```
274 | public void mapFrom()
275 | {
276 |     InitializeComponent();
277 |     Map MyMap = new Map();
278 |     ContentPanel.Children.Add(MyMap);
279 | }
```

280 **2.1.5.2 Tampilan Peta di Windows Phone**

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

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

291

Listing 2.7: Mengatur tingkat zoom dari XAML

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

Listing 2.8: Mengatur tingkat zoom dari dari kode program

```
293     |     public mapFrom()
294     |     {
295        |     InitializeComponent();
296        |     Map MyMap = new Map();
297        |
298        |     //Mengatur titik tengah peta
299        |     MyMap.Center = new GeoCoordinate(47.6097, -122.3331);
300        |
301        |     //mengatur tingkat zoom
302        |     MyMap.ZoomLevel = 10;
303        |     ContentPanel.Children.Add(MyMap);
304     | }
```

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

307

- *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*

- 311 ● Mode warna yang disediakan Windows Phone ada dua yaitu terang dan gelap. Secara
 312 *default* mode pada peta di Windows Phone adalah terang.

- 313 ● Tampilan pada Peta di Windows Phone dapat berubah karena hasil diputar, di-
 314 miningkan, ditarik, dan diturunkan. Berikut beberapa hal yang dapat diatur sebagai
 315 tampilan di peta.

316

- *Heading* merupakan representasi dari derajat secara geometri. Derajat ini dide-
 317 finisikan dalam 0 sampai 360 yang dipakai untuk memutar peta. Contoh, 0 atau
 318 360 ke arah utara, 90 ke arah barat, 180 ke arah selatan, dan 270 derajat ke arah
 319 timur.

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

323 **2.1.5.3 Pushpin ke Peta**

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



Gambar 2.8: Keluaran Toolkit Pushpin pada peta [?]

Listing 2.9: Kode untuk menampilkan pushpin

```
330 MapOverlay overlay = new MapOverlay
331 {
332     GeoCoordinate = map.Center,
333     Content = new Border
334     {
335         BorderBrush = new SolidColorBrush(Colors.FromArgb(120, 255, 0, 0)),
336         Child = new TextBlock(){Text="Pushpin"},
337         BorderThickness = new Thickness(1),
338         Background = new SolidColorBrush(Colors.FromArgb(120, 255, 0, 0)),
339         Width = 80,
340         Height = 60
341     }
342 };
343 MapLayer layer = new MapLayer();
344 layer.Add(overlay);
345
346 map.Layers.Add(layer);
```

347 **2.1.5.4 Polyline pada Peta**

348 Dalam menentukan arah dibutuhkan dua titik yaitu titik awal dan titik tujuan. Tentu
 349 saja arah tersebut butuh ditandai dengan garis. *Polyline* merupakan tentetan garis lurus
 350 yang saling terhubung satu sama lain. Dengan *polyline* arah pada peta dapat ditandai
 351 dengan warna maupun tebal atau tipisnya garis. Contoh keluaran *polyline* dapat dilihat
 352 pada Gambar ?? dan kode untuk menampilkannya dapat dilihat pada *listing* ??.

Listing 2.10: Kode untuk menampilkan polyline

```
353 MapPolyline line = new MapPolyline();
354 line.StrokeColor = Colors.Red;
```



Gambar 2.9: Tampilan polyline pada Peta

```

355     line.StrokeThickness = 10;
356     line.Path.Add(new GeoCoordinate(-6.8619546, 107.614441));
357     line.Path.Add(new GeoCoordinate(-6.908693, 107.611185));

```

358 2.1.5.5 Namespace Control Map

359 *Namespace* merupakan nama yang dipakai untuk mengatur kelas-kelas. Windows Phone
 360 8 sudah menyediakan *namespace* bawaan untuk mengatur peta. *Namespace* yang disediakan
 361 adalah *Maps.Controls*. *Namespace* ini yang berisi kelas-kelas yang paling sering digunakan
 362 untuk mengatur peta pada Windows Phone. Agar dapat menggunakan kelas pada *na-*
 363 *mepace* tersebut perlu ditambahkan *namespace* dan *capabilities*. *Namespace* yang harus
 364 ditambahkan pada baris awal XAML adalah *Microsoft.Phone.Maps.Controls*. Selanjutnya
 365 ada penambahan *capabilities ID_CAP_MAP*. Penambahan *capabilities* ditambahkan pada
 366 *WMAppManifest.xml*.

367 2.1.5.6 Kelas Map

368 Merupakan kelas yang mewakili kontrol map.

Berikut properti yang dapat digunakan pada kelas ini.

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

Tabel 2.1: Properti kelas Map

369

370 Berikut *method* yang dapat digunakan pada kelas ini.

- 371 • *SetView(LocationRectangle)*
372 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai wilayah geogra-
373 fis. *method* ini tidak mengembalikan nilai.
- 374 • *SetView(GeoCoordinate, Double)*
375 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah dan
376 tingkat zoom. *method* ini tidak mengembalikan nilai.
- 377 • *SetView(LocationRectangle, MapAnimationKind)*
378 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai region geografis
379 dan animasi. *method* ini tidak mengembalikan nilai.
- 380 • *SetView(LocationRectangle, Thickness)*
381 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai region geografis
382 dengan batas tertentu. *method* ini tidak mengembalikan nilai.
- 383 • *SetView(GeoCoordinate, Double, MapAnimationKind)*
384 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah,
385 tingkat zoom, dan animasi. *method* ini tidak mengembalikan nilai.
- 386 • *SetView(GeoCoordinate, Double, Double)*
387 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah,
388 tingkat zoom, dan *heading*. *method* ini tidak mengembalikan nilai.
- 389 • *SetView(LocationRectangle, Thickness, MapAnimationKind)*
390 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai wilayah geografis
391 dengan batas tertentu, dan animasi. *method* ini tidak mengembalikan nilai.
- 392 • *SetView(GeoCoordinate, Double, Double, MapAnimationKind)*
393 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah,
394 tingkat zoom, *heading*, dan animasi. *method* ini tidak mengembalikan nilai.
- 395 • *SetView(GeoCoordinate, Double, Double, Double)*
396 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah,
397 tingkat zoom, *heading*, *pitch*. *method* ini tidak mengembalikan nilai.
- 398 • *SetView(GeoCoordinate, Double, Double, Double, MapAnimationKind)*
399 *method* untuk mengatur pandangan di atas peta secara spesifik sesuai titik tengah,
400 tingkat zoom, *heading*, *pitch*, dan animasi. *method* ini tidak mengembalikan nilai.
- 401 • *UpdateLayout*
402 *method* yang memastikan semua posisi objek turunan mengikuti tata letak.

403 **2.1.5.7 Polyline Class**

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

Berikut properti yang dapat digunakan pada kelas ini.

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

Tabel 2.2: Properti *Polyline Class*

406

407 Berikut *method* yang dapat digunakan pada kelas ini.

- *CheckAccess*

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

- *ClearValue*

412 *method* yang akan membersihkan nilai lokal

- *Finalize*

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

416 2.1.5.8 *Pushpin Class*

417 Merupakan kelas yang dipakai untuk menggambarkan elemen terpisah diatas peta. Mes-
418kipun pushpin merupakan bawaan pada peta untuk menunjuk suatu lokasi tetapi *pushpin*
419 dari peta tidak dapat diubah-ubah. *Pushpin* pada Windows Phone 8 dapat dibuat sesuai
420 kebutuhan. Namun ada cara lain dengan menambahkan Windows Phone Toolkit. Windows
421 Phone Toolkit mempunyai komponen untuk menggambar pushpin diatas peta.

422 2.1.6 Lokasi

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

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

436 diatas penyedia data lokasi tersebut adalah pengelola antarmuka. Aplikasi akan mengunakan
437 antarmuka tersebut untuk memulai dan menghentikan layanan lokasi, mengatur tingkat
438 akurasi, dan menerima data lokasi.

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

446 **2.1.6.1 Mendapatkan Posisi Pengguna**

447 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.

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

463 **2.1.6.2 Namespace Geolocator**

464 *Namespace* merupakan nama yang dipakai untuk mengatur kelas-kelas. Windows Phone
465 8 sudah menyediakan *namespace* bawaan untuk mengakses lokasi. *Namespace* yang disediakan
466 adalah *namespace geolocator*. *Namespace* ini akan mengakses lokasi geografis dari
467 perangkat dan mendukung pelacakan lokasi dari waktu ke waktu. Agar dapat menggunakan
468 kelas pada *namespace* tersebut perlu ditambahkan *namespace* dan *capabilities*. *Namespace* yang
469 harus ditambahkan pada baris awal XAML adalah **Windows.Device.Geolocator**.
470 Selanjutnya ada penambahan *capabilities ID_CAP_LOCATION*. Penambahan *capabilities*
471 ditambahkan pada *WMAppManifest.xml*. Kelas yang diatur oleh *namespace geolocator*
472 dapat dilihat pada tabel ??[?].

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***473 2.1.6.3 Geocoordinate**

474 *Geocoordinate* adalah kelas yang menunjukkan lokasi sebagai kordinat geografis. Kelas
 475 ini hanya menyediakan properti yang hanya bisa dibaca. Kelas ini menyediakan properti
 476 yang ditunjukan pada tabel ??.

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***477 2.1.6.4 Geolocator**

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

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

486 Berikut Properti yang disediakan kelas Geolocator:

- 487 • *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*.
- 491 • *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*.

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

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

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

500 Merupakan properti untuk mendapatkan selang waktu pembaruan lokasi. Properti ini
501 mengeluarkan tipe data unit.

502 **2.1.6.5 Geoposition**

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

504 Properti yang disediakan kelas *Geoposition*:

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

506 Data alamat sipil yang terkait dengan lokasi geografis.

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

508 Data latitude dan longitude yang terkait lokasi geografis.

509 **2.1.7 Memanfaatkan Sumber Data**

510 Hal yang penting dari sebuah aplikasi adalah informasi. Windows Phone 8 memiliki

511 kemampuan dalam menghubungkan aplikasi dengan sumber data lainnya. Memanfaatkan

512 sumber data ada dua cara yaitu yang lokal atau berada di perangkat dan *web service*. *Web*

513 *Service* merupakan *method* komunikasi antara dua perangkat melalui jaringan.

514 Sebelum data dapat dikirim antar perangkat perlu dilakukan *Serialization*. *Serialization*

515 disini merupakan proses mentransformasikan objek ke format yang bisa dengan mudah di-

516 kirim melewati jaringan atau disimpan di database. Formatnya disini berupa string yang

517 direpresentasikan sebagai objek di XML atau JSON(Javascript Object Notation). Ada beberapa

518 objek yang dapat melakukan serialisasi, tetapi yang akan dibahas penulis disini hanya

519 serialisasi JSON[?].

520 Banyak *web service* yang mengembalikan data dalam format JSON. JSON memiliki

521 format *data-interchange* yang ringan [?]. Karena hal tersebut JSON mudah diurai dan di-

522 hasilkan oleh mesin. Kurung kurawal mengindikasikan objek, kurung siku berarti array, dan

523 properti berupa nama dan nilai pasangan yang dipisahkan oleh titik dua. JSON format

524 memiliki ukuran data yang kecil dan baik untuk penggunaan perangkat bergerak. Untuk

525 contoh format JSON dapat dilihat di bagian Kiri API pada Bab dua ini karena Kiri API

526 menggunakan format JSON. Serialisasi menggunakan *DataContractJsonSerializer* membuat

527 serialisasi mudah untuk menerjemahkan form String JSON ke objek yang dapat langsung

528 digunakan. *DataContractJsonSerializer* memakai *WriteObject()* untuk serialisasi and *Rea-*

529 *dObject()* untuk de-serialisasi.

530 **2.1.7.1 HttpClient**

531 Merupakan Kelas yang dipakai untuk mengirim permintaan HTTP dan menerima kembali

532 HTTP dari *Uniform Resource Identifier*(URI) yang dapat diidentifikasi. *Uniform Re-*

533 *source Identifier*(URI) merupakan urutan kompak karakter yang mengidentifikasi sumber

534 daya abstrak dan fisik [?]. Berikut *method* yang disediakan kelas *HttpClient*.

535 ● *DeleteAsync(Uri)*

536 *method* yang dipakai untuk mengirimkan permintaan DELETE ke URI yang spesifik
537 sebagai operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memung-
538 kinkan aplikasi untuk melanjutkan pekerjaan selagi *method* ini dipanggil². *method* ini
539 membutuhkan parameter URI sebagai tujuan dari permintaan. Sedangkan kembali-
540 annya berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek
541 tersebut memiliki 2 parameter yaitu hasil berupa pesan kembalian dari http dan ke-
542 majuan dari data yang dikirim.

543 ● *GetAsync(Uri)*

544 *method* yang dipakai untuk mengirimkan permintaan GET ke URI yang spesifik seba-
545 gi operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan
546 aplikasi untuk melanjutkan pekerjaan selagi *method* ini dipanggil². *method* ini mem-
547 butuhkan parameter URI sebagai tujuan dari permintaan. Sedangkan kembalinya
548 berupa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut
549 memiliki 2 parameter yaitu hasil berupa pesan kembalian dari http dan kemajuan dari
550 data yang dikirim.

551 ● *GetAsync(Uri,HttpCompletionOption)*

552 *method* yang dipakai untuk mengirimkan permintaan GET ke URI yang spesifik seba-
553 gi operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan
554 aplikasi untuk melanjutkan pekerjaan selagi *method* ini dipanggil². *method* ini mem-
555 butuhkan parameter URI sebagai tujuan dari permintaan dan nilai tambahan yang
556 dimaksukan sebagai indikasi operasi dianggap selesai. Sedangkan kembalinya ber-
557 upa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut
558 memiliki 2 parameter yaitu hasil berupa pesan kembalian dari http dan kemajuan
559 dari data yang dikirim.

560 ● *GetBufferAsync(Uri)*

561 *method* yang dipakai untuk mengirimkan permintaan GET ke URI yang spesifik seba-
562 gi operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan
563 aplikasi untuk melanjutkan pekerjaan selagi *method* ini dipanggil². *method* ini mem-
564 butuhkan parameter URI sebagai tujuan dari permintaan. Sedangkan kembalinya ber-
565 upa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut
566 memiliki 2 parameter yaitu hasil berupa pesan kembalian yang dikirimkan secara
567 buffer(disimpan dalam memori) dan kemajuan dari data yang dikirim.

568 ● *GetInputStreamAsync(Uri)*

569 *method* yang dipakai untuk mengirimkan permintaan GET ke URI yang spesifik seba-
570 gi operasi *asynchronous*. Maksud dari operasi *asynchronous* adalah memungkinkan
571 aplikasi untuk melanjutkan pekerjaan selagi *method* ini dipanggil². *method* ini mem-
572 butuhkan parameter URI sebagai tujuan dari permintaan. Sedangkan kembalinya ber-
573 upa objek yang mewakili operasi *asynchronous* disertai kemajuan. Objek tersebut
574 memiliki 2 parameter yaitu hasil berupa pesan kembalian yang dikirimkan secara
575 stream(langsung sesuai waktu) dan kemajuan dari data yang dikirim.

576 ● *GetStringAsync(Uri)*

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

584 • *PostAsync(Uri)*

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

592 • *SendRequestAsync(HttpRequestMessage)*

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

599 • *SendRequestAsync(HttpRequestMessage, HttpCompletionOption)*

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

607 **2.1.7.2 Json.NET**

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

611 •

612 **2.1.8 Thread**

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

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

615 **2.1.8.1 *BackgroundWorker***

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

- 618 • *RunWorkerAsync()*

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

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

- 621 • *DoWork*

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

- 623 • *RunWorkerCompleted*

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

626 **2.2 Kiri API**

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

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

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

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

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

638 Merupakan URL produksi. Ini merupakan URL yang direkomendasikan untuk mena-
639 ngani permintaan pengguna.

640 Untuk setiap permintaan membutuhkan *API key* yang didapat dengan mendaftar[?]. Peng-
641 gunaan API memungkinkan pengaksesan di mana saja dengan menggunakan koneksi inter-
642 net. Pada sub bab ?? sampai sub bab ?? penulis akan membahas beberapa layanan Kiri
643 API.

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

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

646 • Register dengan memasukan alamat email, nama, dan nama perusahaan.

647 • Password akan dikirimkan ke alamat email. Tentunya password akan dibuat otomatis
648 oleh pihak Kiri.

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

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

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

653 **2.2.1 Web Service Penentuan Rute**

654 *Web service* penentuan rute merupakan layanan Kiri API yang digunakan untuk men-
 655 dapatkan langkah perjalanan dari lokasi asal ke lokasi tujuan. Parameter dan keterangan
 656 untuk layanan ini dapat dilihat pada tabel ??.

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

Tabel 2.5: Tabel parameter layanan penentuan rute

657 Format kembalian layanan penentuan rute dapat dilihat pada *listing* ??:

Listing 2.11: Kode kembalian pencarian rute

```
658 {
  "status": "ok" or "error"
659 "routingresults": [
660   {
661     "steps": [
662       [
663         "walk" or "none" or others,
664         "walk" or vehicle_id or "none",
665         ["lat_1,lon_1", "lan_2,lon_2", ... "lat_n,lon_n"],
666         "human readable description, dependant on locale",
667         URL for ticket booking or null (future)
668       ],
669     ],
670     [
671       "walk" or "none" or others,
672       "walk" or vehicle_id or "none",
673       ["lat_1,lon_1", "lan_2,lon_2", ... "lat_n,lon_n"],
674       "human readable description, dependant on locale",
675       URL for ticket booking or null (future)
676     ],
677     "traveltime": any text string, null if and only if route is not found.
678   },
679   {
680     "steps": [ ... ],
681     "traveltime": "..."
682   },
683   {
684     "steps": [ ... ],
685     "traveltime": "..."
686   },
687   ...
688 }
689 ]
690 }
```

691 Berikut maksud dari *listing* ??:

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

- 696 • Indeks ke 0 atau baris 7 pada *listing ??* dapat berisi "walk" atau "none" atau "others".
697 Baris tersebut berarti jika "walk" untuk berjalan kaki, "none" jika rute tidak ditemuk-
698 an dan "others" untuk menggunakan kendaran.
- 699 • Indeks ke 1 atau baris 8 pada *listing ??* merupakan detail dari indeks 0. Artinya jika
700 indeks 0 menyatakan "walk" berarti indeks 1 harus "walk", "none" berarti indeks 1
701 harus "none", dan selain itu menyatakan id kendaraan yang mana bisa dipakai untuk
702 ditampilkan gambarnya.
- 703 • Indeks ke 2 atau baris 9 pada *listing ??* adalah deretan nilai tipe *String* yang berisi
704 jalur dalam format "lat,lng". Maksud dari "lat,lng" disini adalah titik awal dan titik
705 akhir dari setiap jalur yang dilewati.
- 706 • Indeks ke 3 atau baris 10 pada *listing ??* berisi bentuk yang akan ditampilkan kepada
707 pengguna. Informasi yang disampaikan dapat berupa:
 - 708 – %fromicon = untuk menunjukan ikon "from". Biasanya untuk mode presentasi
709 di perangkat bergerak.
 - 710 – %toicon = untuk menunjukan ikon "to". Biasanya untuk mode presentasi di
711 perangkat bergerak.
- 712 • Indeks ke 4 atau bari 11 pada *listing ??* berisi URL untuk pemesanan tiket jika tersedia.
713 Jika tidak tersedia akan bernilai *null*.

714 Kiri telah menyediakan gambar untuk setiap angkutan umum. Gambar tersebut
715 dapat di akses di URL:

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

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

720 2.2.2 Web Service Pencarian Lokasi

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

<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

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

Listing 2.12: Kode kembalian pencarian lokasi

```

725 {
726     "status": "ok" or "error"
727     "searchresult": [
728         {
729             "placename": "place_name"
730             "location": "lat,lon"
731         },
732         {
733             "placename": "place_name"
734             "location": "lat,lon"
735         },
736         ...
737     ]
738     "attributions": [
739         "attribution_1", "attribution_2", ...
740     ]
741 }

```

742 Berikut maksud dari *listing ??*:

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

- 746 • "searchresult" (pada baris 4 sampai 7, 8 sampai 11, dan seterusnya) berisi array dari
747 tempat:
- 748 — placename: nama tempat
749 — location: latitude dan longitude dari tempat
- 750 • "attributions" berisi kumpulan nilai yang berisikan atribut tambahan untuk dimun-
751 culkan.

752 2.2.3 Web Service Menemukan Transportasi Terdekat

753 Merupakan Kiri API yang digunakan untuk menemukan rute transportasi terdekat sesuai
754 titik yang diinginkan pengguna. Parameter dan keterangan untuk layanan ini dapat dilihat
755 pada tabel ??.

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

Tabel 2.7: Tabel parameter layanan menemukan transportasi terdekat

756 Format kembalian layanan menemukan transportasi terdekat dapat dilihat pada *lis-*
757 *ting ??*.

Listing 2.13: Kode kembalian menemukan lokasi terdekat

```

758 {
759     "status": "ok" or "error"
760     "nearbytransports": [
761         [
762             "walk" or "none" or others,
763             "walk" or vehicle_id or "none",
764             text_string,
765             decimal_value
766         ],
767         [

```

```
768     "walk" or "none" or others ,  
769     "walk" or vehicle_id or "none" ,  
770     text string ,  
771     decimal value  
772     ],  
773     ...  
774 } ]  
775 }
```

776 Berikut maksud dari *listing ??*:

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

- 780 • Indeks ke 0 atau baris 5 pada *listing ??* dapat berisi "walk" atau "none" atau "others".
781 Artinya jika "walk" berarti berjalan kaki, "none" jika rute tidak ditemukan dan "others"
782 berarti menggunakan kendaraan.
- 783 • Indeks ke 1 atau baris 6 pada *listing ??* merupakan detail dari indeks 0. Artinya jika
784 indeks 0 "walk" berarti indeks 1 harus "walk", "none" berarti indeks 1 harus "none"
785 dan selain itu menyatakan id kendaraan yang mana bisa dipakai untuk ditampilkan
786 gambarnya.
- 787 • Indeks ke 2 atau baris 7 pada *listing ??* berisi nama kendaraan.
- 788 • Indeks ke 3 atau baris 8 pada *listing ??* berisi jarak dengan satuan kilometer.

BAB 3

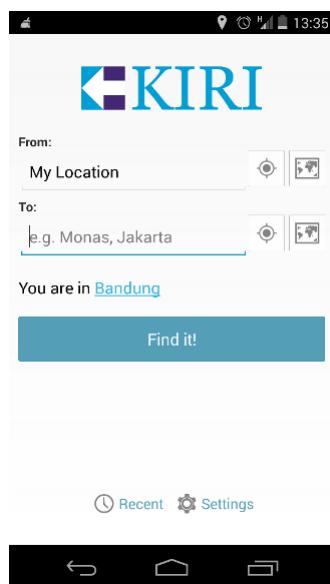
ANALISIS

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

794 **3.1 Analisis Aplikasi Sejenis**

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

802 Berikut adalah tampilan dari aplikasi Public Transport (Gambar ?? sampai ??):



Gambar 3.1: Tampilan utama aplikasi Public Transport

803 Gambar ?? menunjukkan halaman utama aplikasi Public Transport. Di halaman ini
 804 pengguna dapat memasukan lokasi asal dan lokasi tujuan. Cara memasukan lokasi ada 2
 805 macam yaitu dengan mengetik dan menunjuk pada peta dengan mengetuk tombol peta.

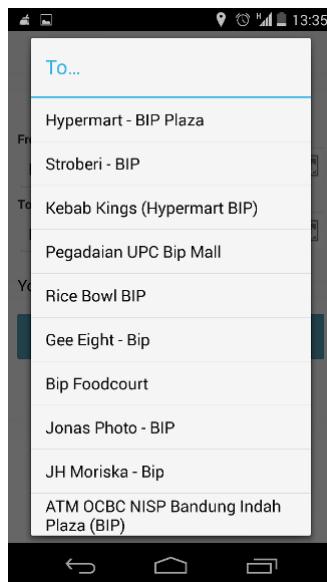
¹<https://play.google.com/store/apps/details?id=travel.kiri.smarttransportapp>

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



Gambar 3.2: Menunjuk lokasi pada peta

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

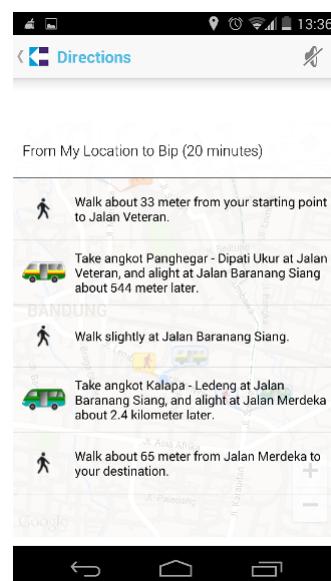


Gambar 3.3: Memberikan daftar nama tempat dan nama jalan terkait

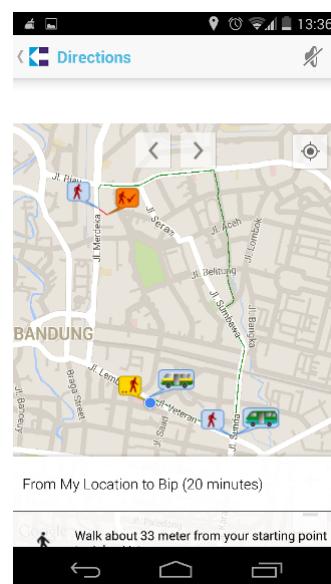
811 Pada gambar ?? pengguna dapat memilih nama tempat terkait. Pemilihan didasarkan
 812 sesuai masukan pengguna untuk memastikan tempat asal maupun tempat tujuan. Jika
 813 nama tempat sudah jelas maka tidak akan ada halaman ini.

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

817 Pada gambar ?? menampilkan rute kendaraan umum dan jalur yang harus dilalui pada



Gambar 3.4: Tampilan rute kendaraan umum dalam bentuk daftar



Gambar 3.5: Tampilan rute kendaraan umum di peta

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

819 3.2 Analisis Aplikasi

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

826 3.2.1 Kebutuhan Aplikasi

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

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

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

843 Aplikasi yang penulis bangun didasarkan pada kebutuhan sebagai berikut.

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

848 3.2.2 Analisis Kontrol yang Dipakai

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

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

858 Kontrol terhadap teks juga diperlukan untuk aplikasi. Kebutuhan yang diperlukan adalah mengeluarkan potongan teks yang dapat dibaca dan tempat pengguna memasukan teks.
859 Untuk mengeluarkan teks yang dapat dilihat penulis akan menggunakan *TextBlock*. Te-
860 xtBlock digunakan untuk menampilkan tulisan "from" dan "to" pada halaman utama apli-
861 kasi. Untuk masukan pengguna terhadap aplikasi penulis akan menyediakan *TextBox* sebagai
862 tempat teks. *TextBox* digunakan sebagai masukan untuk lokasi asal dan lokasi tujuan.
863

864 Suatu aplikasi tentunya tidak hanya mempunyai satu halaman. Sama hal dengan aplikasi
865 yang penulis buat memiliki beberapa halaman yang mempunyai tugas berbeda. Karena hal
866 tersebut dibutuhkan kontrol untuk berpindah dari satu halaman ke halaman lain. Kontrol
867 yang dibutuhkan yaitu kontrol tombol. Kontrol tombol akan mengeksekusi *event click* yang
868 memungkinkan pindah halaman dan melakukan perintah. Kontrol tombol akan penulis
869 gunakan untuk berpindah ke halaman peta, menemukan lokasi pengguna, dan pencarian
870 rute. Pada Gambar ?? terdapat 5 tombol yaitu tombol map pada bagian from, tombol here
871 pada bagian from, tombol map pada bagian to, tombol here pada bagian to, dan tombol
872 find. Berikut kegunaan dari tombol-tombol tersebut.

- 873 • Tombol map pada bagian from

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

- 877 • Tombol map pada bagian from

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

- 879 • Tombol map pada bagian to

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

- 883 • Tombol map pada bagian to

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

- 885 • Tombol find Tombol ini akam mencari rute angkutan umum dan menampilkannya
886 pada halaman peta.

887 Pada aplikasi ini penulis akan menampilkan daftar tempat dan daftar rute angkutan
888 umum yang dipakai. Bentuk daftar digunakan penulis karena hasil tempat dan rute ang-
889 kutan umum akan banyak. Bentuk daftar yang dapat dipakai di Windows Phone adalah
890 menggunakan *ListBox*. *ListBox* akan menampilkan daftar tempat dan daftar rute satu per
891 satu menurun ke bawah.

896 **3.2.3 Analisis Terhadap Siklus Hidup Aplikasi**

897 Aplikasi pada Windows Phone memiliki siklus hidup yang dijelaskan pada bab ??.
898 Pengaturan aplikasi ini diatur sesuai konfigurasi awal sistem operasi Windows Phone. Tetapi

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

902 Saat aplikasi dijalankan, pengguna memasukan lokasi asal dan lokasi tujuan. Setelah
903 memasukan lokasi pengguna akan mencari rute. Ketika rute berhasil ditemukan aplikasi akan
904 berada di keadaan *Running*. Tetapi ada kemungkinan pengguna berpindah aplikasi atau
905 mematikan layar untuk menghemat daya. Dalam kasus tersebut sistem operasi Windows
906 Phone akan menganggap aplikasi tidak aktif dan aplikasi akan masuk pada keadaan *dormant*.
907 Untuk menangani kasus tersebut maka penulis harus menyimpan keadaan dan informasi
908 saat sebelum aplikasi menjadi tidak aktif. Penanganan yang penulis akan lakukan adalah
909 menggunakan *method OnNavigatedFrom()*. Dengan *method* tersebut keadaan aplikasi akan
910 disimpan di memori.

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

918 Ketika aplikasi sudah berada pada keadaan *Dormant* atau *Tombstoned*, pengguna masih
919 dapat memulihkan keadaan aplikasi saat aplikasi berada di keadaan *Running* sebelumnya.
920 Penanganan yang penulis akan lakukan untuk hal tersebut adalah menggunakan *method*
921 *OnNavigatedTo()*. Menggunakan *method* tersebut akan memulihkan informasi halaman pada
922 keadaan *Running* sebelumnya.

923 3.2.4 Analisis Peta

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

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

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

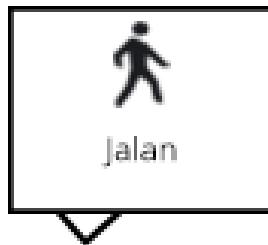
937 Tampilan awal peta di Windows Phone akan mengeluarkan peta dengan pandangan du-
938 nia. Karena aplikasi pencarian rute ini masih terbatas di Pulau Jawa, Indonesia terutama

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

944 Di setiap kota ada satu angkutan umum yang banyak dipakai yaitu angkutan kota(angkot).
 945 Namun bagi yang baru pertama mengunjungi suatu daerah dan mencari angkot mungkin
 946 akan kesulitan membaca trayek dari angkot tersebut. Namun ada satu cara yang mudah
 947 untuk membedakan angkot di setiap rute yaitu dari warna dan coraknya. Agar dapat me-
 948 mudahkan pengguna dan menghindari pengguna dari kesalahan naik angkot maka Kiri API
 949 sudah menyediakan gambar angkot yang seusai dengan setiap rute. Gambar angkot terse-
 950 but akan ditempatkan di peta dengan suatu penampung beserta keterangannya. Salah satu
 951 teknik untuk menempatkan gambar tersebut adalah dengan membuat lapisan terpisah di
 952 atas peta tempat gambar tersebut. Untuk hal tersebut penulis akan memanfaatkan Pushpin
 953 sebagai lapisan terpisah untuk menaruh gambar dan keterangan angkutan umum. Berikut
 954 tampilan *pushpin* untuk angkot ?? dan *pushpin* untuk jalan kaki ??.



Gambar 3.6: Tampilan *pushpin* untuk angkot



Gambar 3.7: Tampilan *pushpin* untuk jalan kaki

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

962 3.2.5 Analisis Pemanfaatan Sumber Data

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

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

972 3.2.6 Analisis Kiri API

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

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

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

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

- 988 • *version*: 2

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

- 991 • *mode*: "searchplace"

992 Mode "searchplace" digunakan untuk mencari lokasi terkait.

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

994 Karena Kiri API baru tersedia di 3 kota yaitu Jakarta, Bandung, dan Surabaya maka
995 region harus dimasukan untuk pencarian. Region harus dipilih antara "cgk"/"bdo"/"sub"
996 sebagai parameter. Pengguna dapat menentukan masukan region jika menuliskannya
997 pada lokasi asal atau lokasi tujuan. Tetapi, jika pengguna tidak menuliskannya maka
998 sistem yang akan menentukan. Cara penentuan region oleh sistem adalah sistem akan
999 menampung titik tengah dari ketiga region tersebut lalu membandingkannya dengan
1000 lokasi pengguna berada. Jarak terdekat antara lokasi pengguna dan salah satu region
1001 menandakan pengguna berada di region tersebut.

- 1002 • *querystring*: merupakan kata kunci lokasi

1003 • *apikey*: 16 digit heksadesimal

1004 Format layanan yang dikirim melalui URL adalah `kiri.travel/handle.php?version=2&mode=searchplace®ion=cgk/bdo/sub&querystring="string"&apikey=97A7A1157A05ED6F.`

1006

1007 Penulis mencoba mencari lokasi bip dari kata kunci "bip" yang berada di bandung. La-

1008 yan dan dikirimkan ke URL `kiri.travel/handle.php`. Berikut format layanan yang penulis

1009 kirim:

1010 `http://kiri.travel/handle.php?version=2&mode=searchplace®ion=bdo&querystring=`

1011 `bip&apikey=97A7A1157A05ED6F`

1012

1013 Berikut hasil kembalian dari Kiri API:

Listing 3.1: Kode kembalian dari pencarian rute

```

1014 {
1015     "status ":"ok",
1016     "searchresult ":[
1017         {
1018             "placename ":"Hypermart - BIP Plaza",
1019             "location ":"-6.90864,107.61108"
1020         },
1021         {
1022             "placename ":"Stroberi - BIP",
1023             "location ":"-6.90834,107.61115"
1024         },
1025         {
1026             "placename ":"Kebab Kings (Hypermart BIP)",
1027             "location ":"-6.91503,107.61017"
1028         },
1029         {
1030             "placename ":"Pegadaian UPC Bip Mall",
1031             "location ":"-6.90916,107.61052"
1032         },
1033         {
1034             "placename ":"Rice Bowl BIP",
1035             "location ":"-6.90873,107.61088"
1036         },
1037         {
1038             "placename ":"Gee Eight - Bip",
1039             "location ":"-6.90817,107.61080"
1040         },
1041         {
1042             "placename ":"Jonas Photo - BIP",
1043             "location ":"-6.91066,107.61016"
1044         },
1045         {
1046             "placename ":"Bip Foodcourt",
1047             "location ":"-6.91081,107.61015"
1048         },
1049         {
1050             "placename ":"Mister Baso BIP",
1051             "location ":"-6.90348,107.61709"
1052         },
1053         {
1054             "placename ":"JH Moriska - Bip",
1055             "location ":"-6.90868,107.61070"
1056         }
1057     ],
1058     "attributions":null
1059 }
```

1060 Hasil dari kembalian berupa kumpulan *placename* dan *location*. Hasil tersebut akan

1061 aplikasi tampung namun yang akan ditampilkan ke pengguna hanya *placename*. Menam-

1062 pilkan *location* tidak efektif menurut penulis karena akan membingungkan pengguna. Dari

1063 percobaan yang penulis lakukan, nilai dari *attributions* selalu bernilai "null". Karena hal

1064 tersebut maka nilai *attributions* akan penulis abaikan.

1065 Pemanfaatan layanan penentuan rute untuk mendapatkan langkah yang harus ditempuh

1066 pengguna untuk mencapai lokasi tujuan dari lokasi asal. Pemanfaatan layanan ini yaitu

1067 dengan parameter *GET* melalui protokol HTTP. Berikut parameter yang harus dikirim:

- 1068 ● *version*: 2
 1069 Karena acuan penulis adalah Kiri API versi maka di parameter *version* penulis akan
 1070 menggunakan 2.
- 1071 ● *mode*: "findroute"
 1072 Mode "findroute" digunakan untuk mendapatkan langkah yang harus ditempuh me-
 1073 nuju lokasi tujuan.
- 1074 ● *locale*: "en" untuk bahasa Inggris dan "id" untuk bahasa Indonesia.
 1075 Karena aplikasi ini memungkinkan dipakai orang banyak maka penulis putuskan untuk
 1076 menggunakan bahasa Inggris.
- 1077 ● *start*: koordinat lokasi awal dalam berupa latitude dan longitude.
 1078 Masukan untuk lokasi awal harus dalam bentuk koordinat. Jika masukan dari peng-
 1079 guna adalah alamat atau tempat maka perlu dicari kordinatnya dahulu.
- 1080 ● *finish*: koordinat lokasi tujuan dalam berupa latitude dan longitude.
 1081 Masukan untuk lokasi tujuan harus dalam bentuk koordinat. Jika masukan dari peng-
 1082 guna adalah alamat atau tempat maka perlu dicari kordinatnya dahulu.
- 1083 ● *presentation*: "mobile" untuk perangkat bergerak dan "desktop" untuk komputer.
 1084 Karena aplikasi ini dirancang untuk Windows Phone 8, presentasi yang penulis pi-
 1085 ilih adalah "desktop". Pemilihan ini didasarkan karena deskripsi yang ditampilkan
 1086 tidak ada tulisan "image from" dan "image to", selain itu presentasi "desktop" juga
 1087 memungkinkan rute alternatif.
- 1088 ● *apikey*: 16 digit heksadesimal.

1089 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

1090 Penulis mencoba menuju jalan merdeka dari jalan ciumbuleuit. Layanan dikirimkan ke
 1091 URL <http://kiri.travel/handle.php?version=2&mode=findroute&locale=id&start=-6.8747337,107.6048829&finish=-6.9114646,107.6104887&presentation=mobile&apikey=97A7A1157A05ED6F>.

1097

1098 Berikut hasil kembalian dari Kiri API:

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

```
1099 {
1100   "status ":"ok",
1101   "routingresults": [
1102     {
1103       "steps": [
1104         [
1105           "walk",
1106           "walk",
1107           "-6.8747337,107.6048829", "-6.87445,107.60465"],
1108           "Jalan dari lokasi mulai Anda \%fromicon ke Jalan Ciumbuleuit \%toicon sejauh
1109           kurang lebih 41 meter.",
1110           null,
1111           null
1112         ],
1113         [
1114           "angkot",
```

```

1115     " ciumbuleuitsthallurus " ,
1116     [ "-6.87445,107.60465", "-6.87541,107.60443", "-6.87637,107.60421", "-6.87734,107.60400",
1117     "-6.87830,107.60378",
1118     " -6.87926,107.60356", "-6.87926,107.60356", "-6.87963,107.60352",
1119     "-6.87978,107.60352", "-6.88093,107.60392", "-6.88209,107.60433", "-6.88209,107.60433",
1120     "-6.88328,107.60490", "-6.88328,107.60490", "-6.88347,107.60481", "-6.88452,107.60459",
1121     "-6.88556,107.60436", "-6.88660,107.60413", "-6.88764,107.60390", "-6.88764,107.60391",
1122     "-6.88782,107.60392", "-6.88887,107.60404", "-6.88991,107.60416", "-6.88991,107.60416",
1123     "-6.89161,107.60428", "-6.89161,107.60428", "-6.89166,107.60421", "-6.89275,107.60424",
1124     "-6.89275,107.60424", "-6.89405,107.60408", "-6.89405,107.60408", "-6.89496,107.60400"],
1125     "Naik angkot Ciumbuleuit – St. Hall (lurus) di Jalan Ciumbuleuit \%fromicon ,
1126     dan turun di Jalan Cihampelas \%toicon kurang lebih setelah 3,3 kilometer
1127     .",
1128     null ,
1129     https:// angkot.web.id/go/route/640?ref=kiwi
1130   ],
1131   [
1132     "walk",
1133     "walk",
1134     [ "-6.90424,107.60433", "-6.90429,107.60440"],
1135     "Jalan dari Jalan Cihampelas \%fromicon ke Jalan Abdul Rivai \%toicon sejauh
1136     kurang lebih 10 meter.",
1137     null ,
1138     null
1139   ],
1140   [
1141     "angkot",
1142     "kalapaledeng",
1143     [ "-6.89501,107.60403", "-6.89562,107.60398", "-6.89623,107.60395", "-6.89732,107.60401",
1144     "-6.89732,107.60401", "-6.89882,107.60414", "-6.89882,107.60414", "-6.89969,107.60418",
1145     "-6.90071,107.60426", "-6.90173,107.60433", "-6.90173,107.60433", "-6.90297,107.60437",
1146     "-6.90420,107.60440", "-6.90420,107.60440", "-6.90426,107.60456", "-6.90422,107.60481",
1147     "-6.90399,107.60546", "-6.90406,107.60617", "-6.90454,107.60697", "-6.90454,107.60697",
1148     "-6.90512,107.60745", "-6.90618,107.60778", "-6.90618,107.60778", "-6.90643,107.60787",
1149     "-6.90651,107.60807", "-6.90675,107.60914", "-6.90675,107.60914", "-6.90694,107.60939",
1150     "-6.90723,107.60939", "-6.90891,107.60943", "-6.90891,107.60943", "-6.90909,107.60934",
1151     "-6.90914,107.60857", "-6.90933,107.60846", "-6.91021,107.60887", "-6.91021,107.60887",
1152     "-6.91030,107.60897", "-6.91028,107.60927", "-6.90986,107.61040", "-6.90986,107.61040"],
1153     "Naik angkot Kalapa – Ledeng di Jalan Abdul Rivai \%fromicon , dan turun di
1154     Jalan Aceh \%toicon kurang lebih setelah 1,1 kilometer.",
1155     "https:// angkot.web.id/go/route/156?ref=kiwi"
1156   ],
1157   [
1158     "walk",
1159     "walk",
1160     [ "-6.90986,107.61040", "-6.9114646,107.6104887"],
1161     "Walk about 178 meter from Jalan Aceh \%fromicon to your destination \%toicon
1162     .",
1163     null ,
1164     null
1165   ],
1166   [
1167     "traveltime ":"30 minutes"
1168   ]
1169 }
1170 }
```

1188 Berikut format layanan yang penulis kirim dengan *presentation mobile* <http://kiwi.travel/handle.php?version=2&mode=findroute&locale=id&start=-6.8747337,107.6048829&finish=-6.9114646,107.6104887&presentation=desktop&apikey=97A7A1157A05ED6F>.

1191

1192

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

```

1193 {
1194     "status ":"ok",
1195     "routingresults ":[
1196         {
1197             "steps ":[
1198                 [
1199                     "walk",
1200                     "walk",
1201                     ["-6.8747337,107.6048829","-6.87445,107.60464"],
1202                     "Jalan dari lokasi mulai Anda ke Jalan Ciumbuleuit sejauh kurang lebih 41
1203                     meter.",
1204                     null,
1205                     null
1206                 ],
1207                 [
1208                     "angkot",
1209                     "ciumbuleuitsthallurus",
1210                     ["-6.87445,107.60464","-6.87541,107.60443","-6.87541,107.60443","-6.87637,107.60422",
1211                     "-6.87637,107.60422","-6.87734,107.60400","-6.87734,107.60400","-6.87830,107.60378",
1212                     "-6.87830,107.60378","-6.87926,107.60356","-6.87926,107.60356","-6.87926,107.60356",
1213                     "-6.87963,107.60352","-6.87978,107.60352","-6.88093,107.60393","-6.88093,107.60393",
1214                     "-6.88209,107.60433","-6.88209,107.60433","-6.88209,107.60433","-6.88328,107.60490",
1215                     "-6.88328,107.60490","-6.88328,107.60490","-6.88347,107.60481","-6.88452,107.60458",
1216                     "-6.88452,107.60458","-6.88556,107.60435","-6.88556,107.60435","-6.88660,107.60413",
1217                     "-6.88660,107.60413","-6.88764,107.60390","-6.88764,107.60390","-6.88764,107.60390",
1218                     "-6.88782,107.60392","-6.88887,107.60404","-6.88887,107.60404","-6.88991,107.60416",
1219                     "-6.88991,107.60416","-6.88991,107.60416","-6.89161,107.60428","-6.89161,107.60428",
1220                     "-6.89161,107.60428","-6.89166,107.60420","-6.89275,107.60424","-6.89275,107.60424",
1221                     "-6.89275,107.60424","-6.89405,107.60407","-6.89405,107.60407","-6.89405,107.60407",
1222                     "-6.89496,107.60400","-6.89496,107.60400","-6.89586,107.60392","-6.89586,107.60392",
1223                     "-6.89586,107.60392","-6.89759,107.60397","-6.89759,107.60397","-6.89759,107.60397",
1224                     "-6.89895,107.60406","-6.89895,107.60406","-6.89895,107.60406","-6.89970,107.60413",
1225                     "-6.89999,107.60416","-6.90114,107.60426","-6.90114,107.60426","-6.90114,107.60426",
1226                     "-6.90218,107.60428","-6.90218,107.60428","-6.90321,107.60431","-6.90321,107.60431",
1227                     "-6.90424,107.60433"],
1228                     "Naik angkot Ciumbuleuit – St. Hall (lurus) di Jalan Ciumbuleuit , dan turun di
1229                     Jalan Cihampelas kurang lebih setelah 3,3 kilometer.",
1230                     null,
1231                     "https:// angkot.web.id/go/route/640?ref=kiri"
1232                 ],
1233                 [
1234                     "walk",
1235                     "walk",
1236                     ["-6.90424,107.60433","-6.90429,107.60440"],
1237                     "Jalan dari Jalan Cihampelas ke Jalan Abdul Rivai sejauh kurang lebih 10 meter
1238                     .",
1239                     null,
1240                     null
1241                 ],
1242                 [
1243                     "angkot",
1244                     "kalapaledeng",
1245                     ["-6.90429,107.60440","-6.90434,107.60490","-6.90398,107.60558","-6.90417,107.60619",
1246                     "-6.90465,107.60702","-6.90465,107.60702","-6.90465,107.60702","-6.90521,107.60748",
1247                     "-6.90600,107.60771","-6.90646,107.60775","-6.90755,107.60760","-6.90755,107.60760",
1248                     "-6.90755,107.60760","-6.90866,107.60789","-6.90866,107.60789","-6.90866,107.60789",
1249                     "-6.90911,107.60826","-6.91034,107.60892","-6.91034,107.60892","-6.91034,107.60892",
1250                     "-6.91007,107.60985"],
1251                     "Naik angkot Kalapa – Ledeng di Jalan Abdul Rivai , dan turun di Jalan Aceh
1252                     kurang lebih setelah 1,1 kilometer.",
1253                     null,
1254                     "https:// angkot.web.id/go/route/156?ref=kiri"
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276

```

```

1277   ],
1278   [
1279     "walk",
1280     "walk",
1281     ["-6.91007,107.60985", "-6.9114646,107.6104887"],
1282     "Jalan dari Jalan Aceh ke tujuan akhir Anda sejauh kurang lebih 171 meter.",
1283     null,
1284     null
1285   ],
1286   "traveltime":"30 menit"}]}

```

Setiap langkah akan aplikasi tampung dalam elemen *array*. Untuk keterangan dan jenis angkutan umum akan aplikasi tampilkan dalam bentuk *pushpin* pada peta atau daftar. Sedangkan untuk titik-titik kordinat akan digambarkan pada peta. Dari analisa penulis setiap langkah menunjukkan perpindahan angkutan umum yang dipakai, berpindah dari angkutan umum atau jalan, dan dari jalan untuk menaiki angkutan umum. Keterangan yang penulis akan tambahkan harus berada antara setiap *steps* tersebut. Dari analisa penulis juga terdapat kata "%fromicon" dan "%toicon" yang tidak 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 Skenario

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

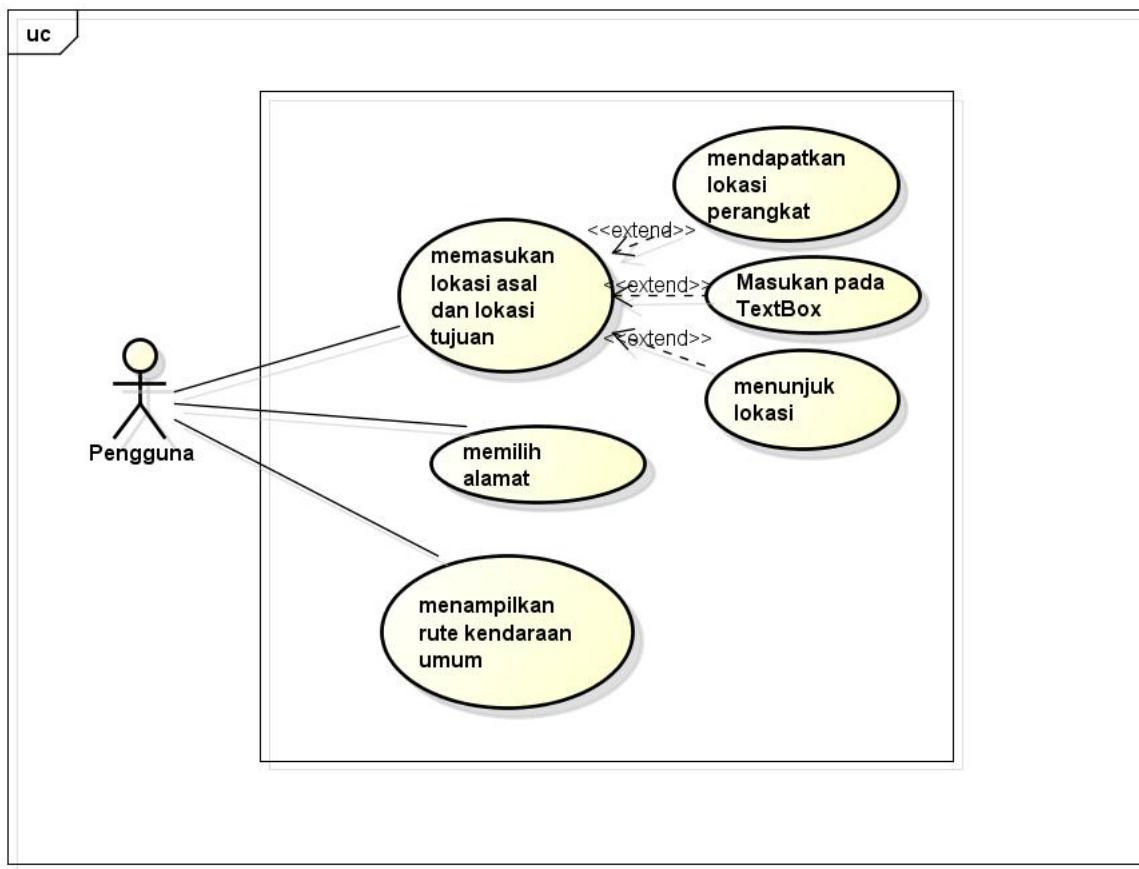
- Mendapatkan lokasi pengguna berada.
- Memasukan lokasi asal dan lokasi tujuan.
- Menunjuk langsung lokasi asal dan tujuan pada peta.
- Memilih alamat atau tempat dari pilihan yang disediakan.
- Menampilkan rute kendaraan umum dalam bentuk titik dan *pushpin* pada peta atau bentuk daftar dari tempat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

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

Skenario pencarian rute kendaraan umum dapat dilihat pada tabel ?? sampai tabel ??.

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

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

Gambar 3.8: Diagram *use case*

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

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

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

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

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

Tabel 3.4: Skenario memilih alamat

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

Tabel 3.5: Skenario menampilkan rute kendaraan umum

3.2.8 Kelas Diagram

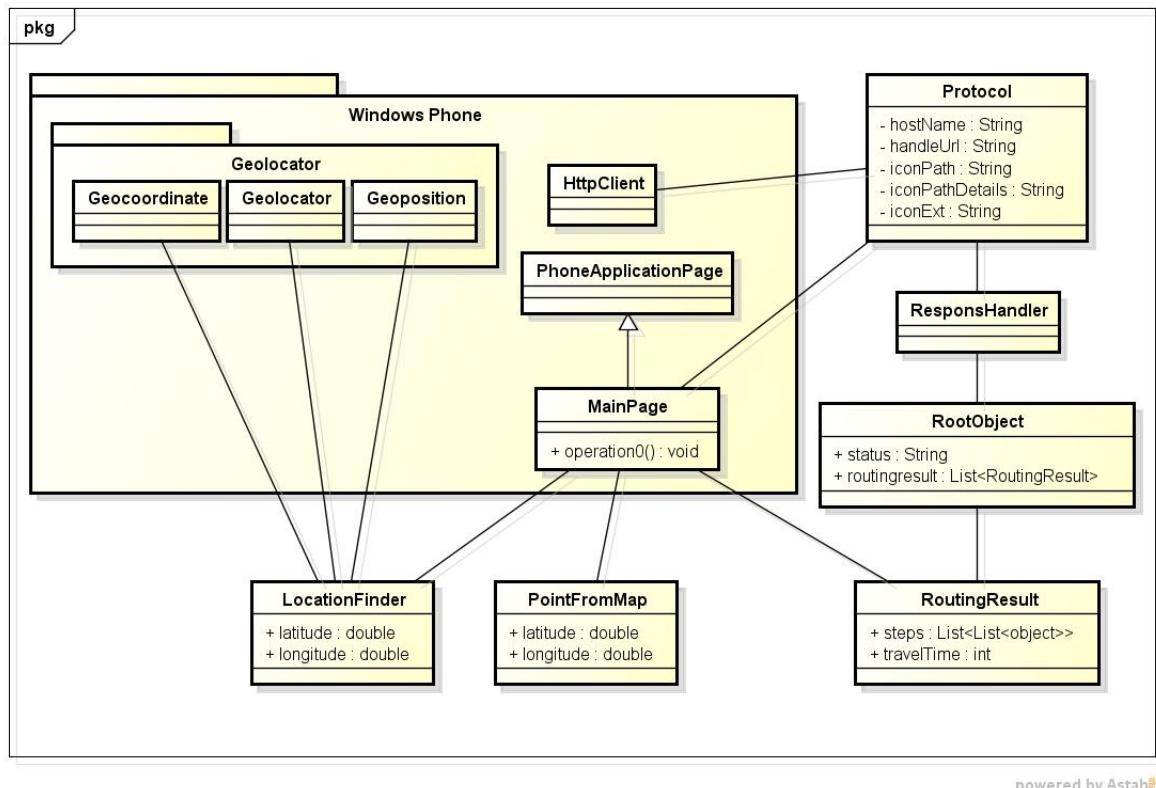
1309 Pembuatan kelas diagram didasarkan pada skenario pada sub bab ???. Kelas diagram
1310 dapat dilihat pada gambar ??.

1311 Berikut deskripsi kelas pada gambar ??.

- 1312 • Kelas *Protocol*

1313 Merupakan kelas yang menampung semua alamat URL yang berhubungan dengan Kiri
1314 API. Semua pemanggilan akan ditangani oleh kelas ini.

1315



Gambar 3.9: Diagram Kelas

1316 • Kelas *ResponsHandler*

1317 Merupakan kelas yang menangani masukan dari pemanggilan layanan.

1318 • Kelas *RootObject*

1319 Merupakan kelas untuk menampung status dan daftar dari layanan *routing* Kiri API.
1320 Hasil kembalian akan dipisahkan di kelas ini untuk selanjutnya ditambahkan di kelas
1321 *RoutingResult*.

1322 • Kelas *RoutingResult*

1323 Merupakan kelas untuk menampung setiap langkah dari rute sesuai masukan pengguna.
1324 Pada kelas ini juga rute akan digambarkan pada peta.

1325 • Kelas *PointFromMap*

1326 Merupakan kelas yang dapat mengetahui lokasi yang ditunjuk pengguna pada peta.
1327 Kelas ini akan menyimpan lokasi yang ditunjuk pengguna dalam bentuk *latitude* dan
1328 *longitude*.

1329 • Kelas *LocationFinder*

1330 Merupakan kelas yang digunakan untuk mencari lokasi. kelas ini akan memanfaatkan
1331 kelas *Geocoordinate* untuk mendapatkan lokasi. Setelah lokasi didapatkan dalam
1332 bentuk kelas *Geoposition* maka akan diubah ke *latitude* dan *longitude*.

1333

BAB 4

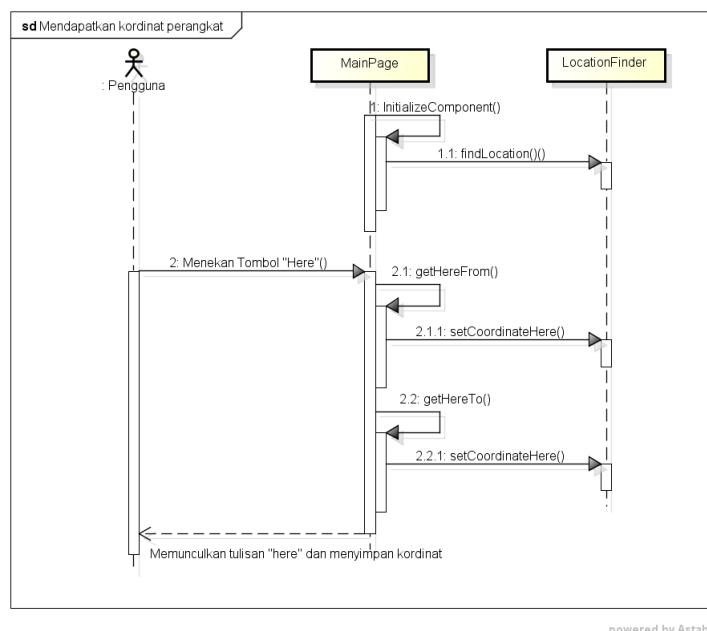
1334

PERANCANGAN

1335 Pada bab 4 akan dibahas mengenai perancangan seperti diagram *sequence*, diagram kelas
1336 secara rinci, deskripsi atribut dan *method* dari setiap kelas, dan perancangan antarmuka.

1337 4.1 Diagram *Sequence*

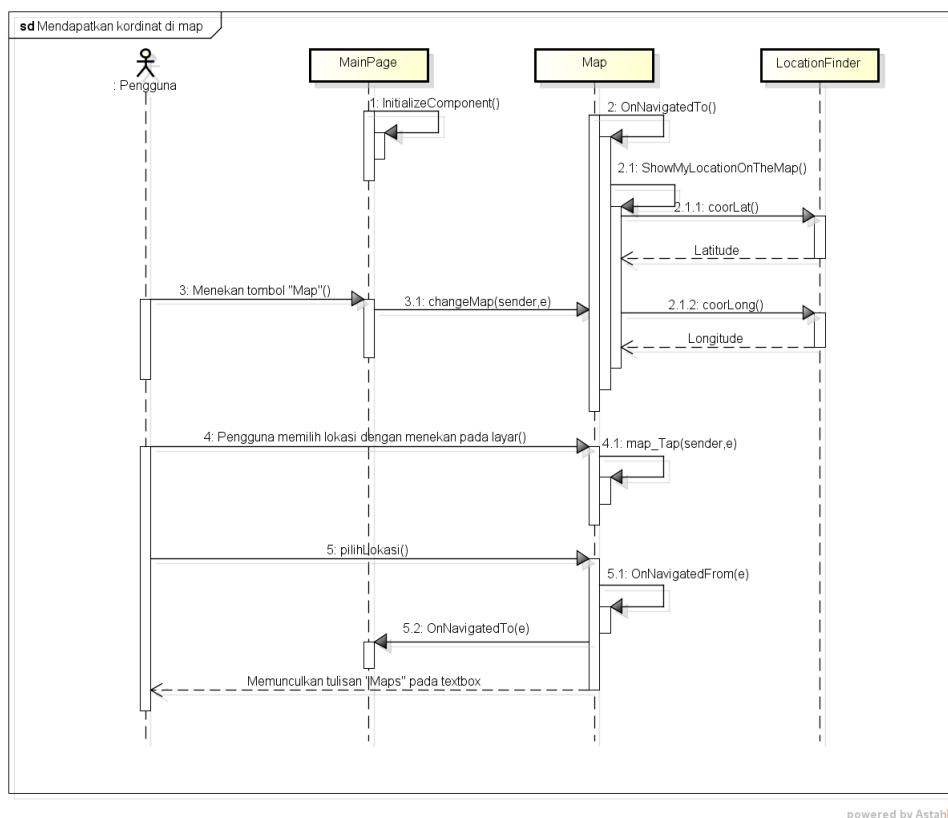
1338 Diagram *sequence* merupakan diagram yang menggambarkan interaksi antar objek dalam
1339 suatu skenario. Gambar diagram *sequence* dapat dilihat pada gambar reffig:sequence lokasi
1340 perangkat sampai reffig:sequence route.



Gambar 4.1: Diagram *sequence* Mendapatkan Kordinat perangkat

1341 Diagram ?? merupakan diagram *sequence* untuk memilih lokasi dengan lokasi per-
1342 angkat berada. Diagram menunjukkan bahwa setelah aplikasi dibuka maka aplikasi akan
1343 mencari dahulu lokasi perangkat dengan memanfaatkan kelas LocationFinder. Lalu setelah
1344 aplikasi terbuka jika pengguna ingin memilih lokasi tersebut sebagai lokasi asal maka peng-
1345 guna harus menekan tombol "here". Setelah tombol "here" ditekan maka kelas MainPage
1346 akan mengambil nilai *Latitude* dan nilai *Longitude* dari kelas LocatonFinder. Setelah lokasi
1347 didapatkan maka akan muncul tulisan "here" pada masukan di kelas MainPage.

1348 Diagram ?? merupakan diagram *sequence* untuk memilih lokasi. Diagram menun-
1349 jukan bahwa setelah aplikasi dibuka maka pengguna dapat menekan tombol "map". Setelah

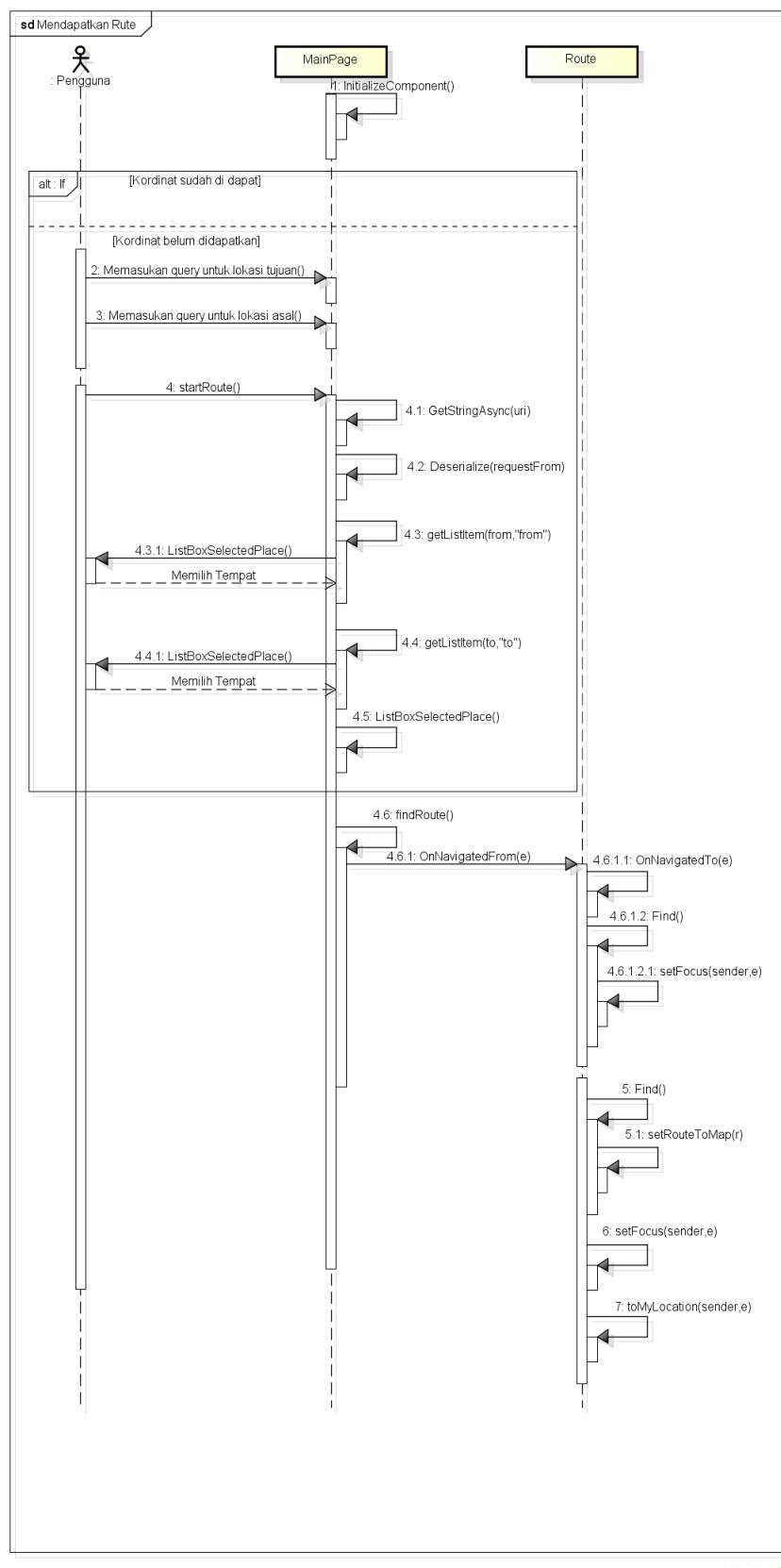


Gambar 4.2: Diagram *sequence* Mendapatkan Kordinat pada Peta

1350 tombol "map" ditekan maka halaman akan dialihkan ke kelas Map. Saat kelas Map terbuka
 1351 maka lokasi yang ditunjukkan adalah lokasi dimana perangkat berada. Untuk mengetahui
 1352 lokasi kelas Map mengambil kordinat *latitude* dan *longitude* dari kelas LocationFinder. Di
 1353 kelas "Map" pengguna dapat memilih lokasi dengan memilih lokasi pada peta dan memang-
 1354 gil *method map_tap*, lalu setelah pengguna memilih tempat pengguna akan menekan tombol
 1355 Pilih Lokasi yang akan memanggil *method pilihLokasi*. Lokasi yang dipilih pengguna akan
 1356 disimpan di kelas MainPage dan pada masukan akan tertulis "map".

1357 Diagram ?? merupakan diagram *sequence* untuk mencari rute. Diagram menunjuk-
 1358 an bahwa setelah aplikasi dibuka maka aplikasi akan melakukan inisialisasi. Untuk mencari
 1359 rute dari lokasi asal ke lokasi tujuan dibutuhkan kordinat *latitude* lokasi asal, *longitude* lo-
 1360 kasi asal, *latitude* lokasi tujuan, dan *longitude* lokasi tujuan. Jika pengguna mendapatkan
 1361 lokasi dari peta atau sesuai lokasi maka yang didapatkan sudah pasti kordinat, namun jika
 1362 pengguna memasukan kata kunci perlu didapat kordinat *latitude* dan *longitude* dari kata
 1363 kunci tersebut. Jika masukan yang didapat berupa kata kunci maka akan dilakukan peme-
 1364 riksaan apakah kordinat untuk kata kunci tersebut tersedia. Pemeriksaan dilakukan dengan
 1365 melakukan pemanggilan Kiri API. Tahap pemanggilan meliputi pemanggilan *method Gets-*
 1366 *tringAsync* lalu mengjadikan objek kembalinya dengan *method Deserialize*. Jika sudah
 1367 didapat dan hasilnya lebih dari satu maka akan dipanggil *method getListItem* yang akan
 1368 menampilkan daftar pilihan ke pengguna untuk dipilih. Pengguna dapat memilih tempat
 1369 sesuai tempat asal maupun tujuan yang diinginkan. Setelah lokasi asal dan lokasi tujuan di-
 1370 dapat maka kelas MainPage akan mengarahkan ke kelas Route untuk menampilkan hasilnya.
 1371 Kelas Route akan memanggil *method OnNavigatedTo* yang bertujuan untuk mendapatkan
 1372 lokasi asal dan lokasi tujuan. Setelah itu akan memanggil *method Find* lalu mengembalikan

¹³⁷³ rute yang ditemukan kepada pengguna.

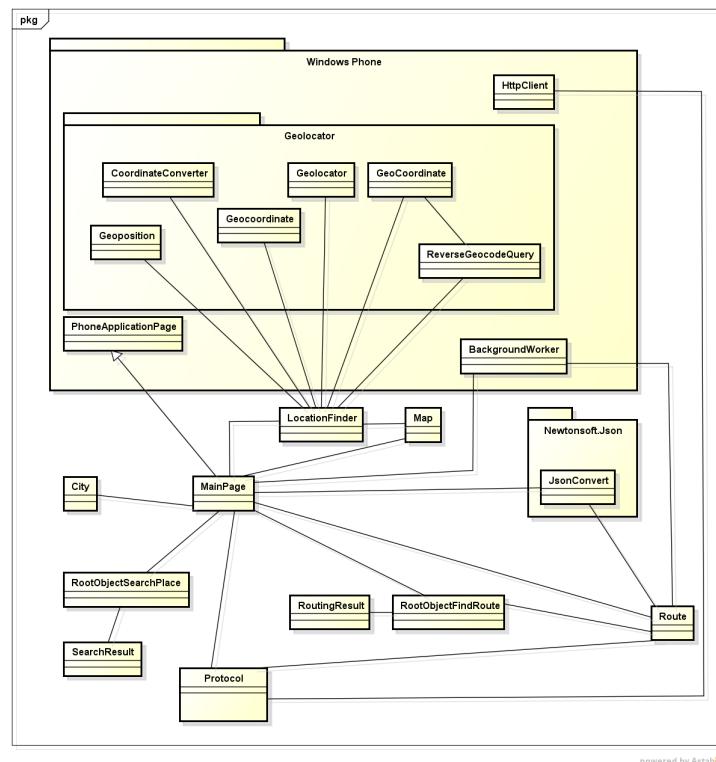


powered by Astah

Gambar 4.3: Diagram *sequence* Mendapatkan Rute

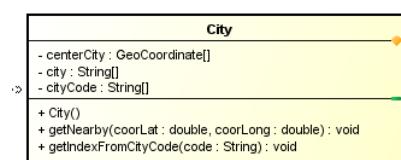
1374 4.2 Perancangan Kelas

1375 Pada sub bab ini akan dibahas mengenai deskripsi kelas secara rinci pada aplikasi Pencari
 1376 Rute Kendaraan Umum untuk Windows Phone. Untuk lebih jelas mengenai kelas yang ada
 1377 pada aplikasi ini, penulis menyajikan gambar diagram kelas yang dapat dilihat pada gambar
 1378 ??.

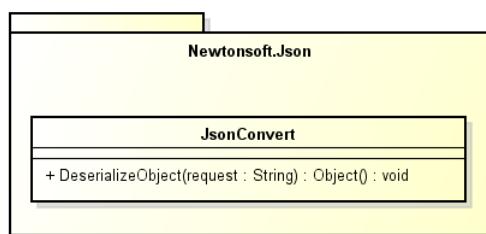


Gambar 4.4: Diagram Kelas

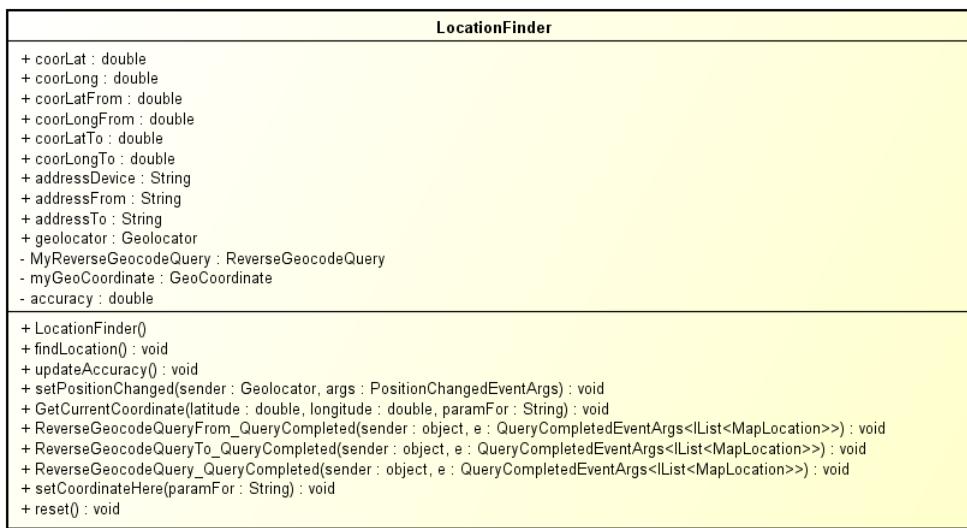
1379 Berikut Penjelasan kelas diagram secara rinci dengan setiap kelas disertai atribut dan
 1380 *method*.



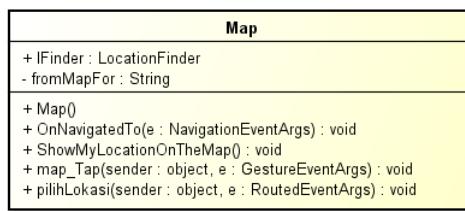
Gambar 4.5: Kelas City



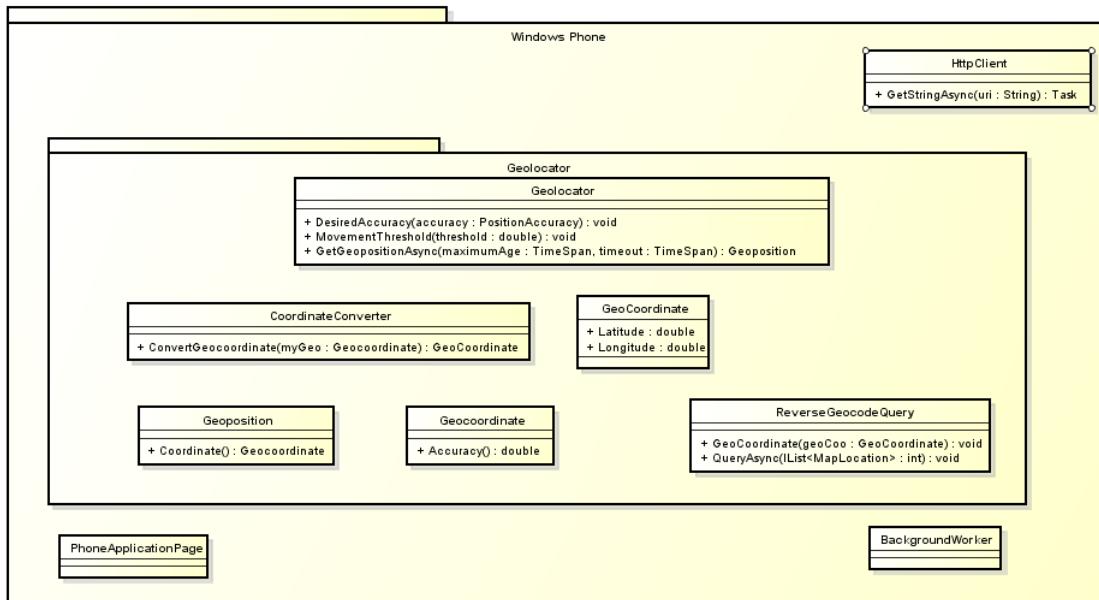
Gambar 4.6: Kelas JsonConvert



Gambar 4.7: Kelas LocationFinder



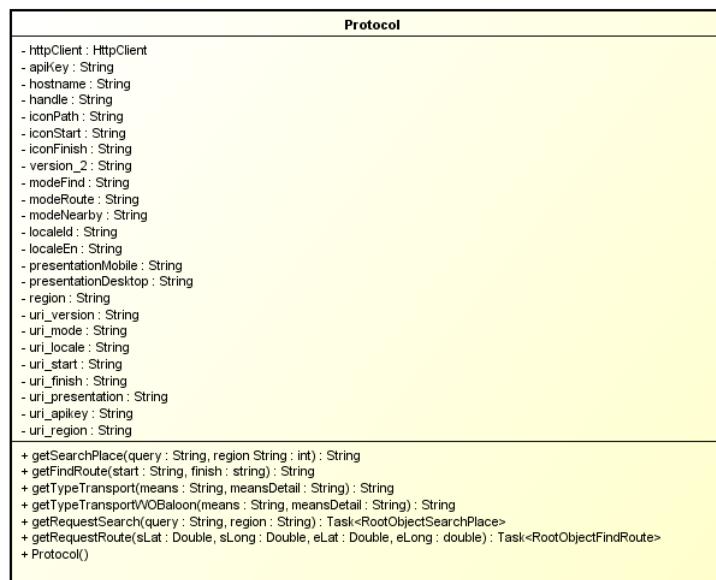
Gambar 4.8: Kelas Map



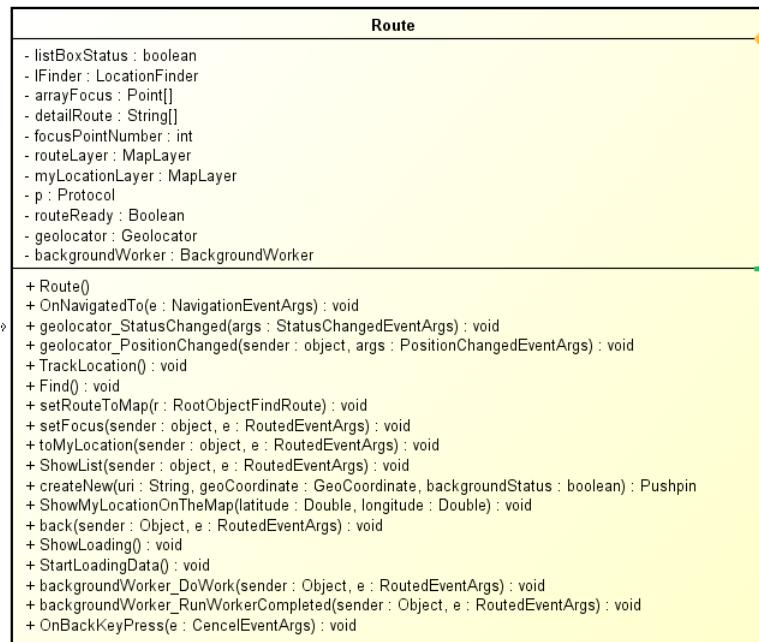
Gambar 4.9: Package WindowsPhone

1381 4.2.1 Kelas PhoneApplicationPage

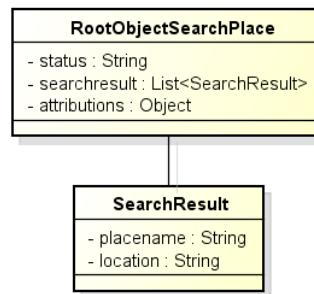
1382 *PhoneApplicationPage* merupakan kelas bawaan Windows Phone yang menangani in-
 1383 terksi pengguna dengan aplikasi dan siklus hidup aplikasi.



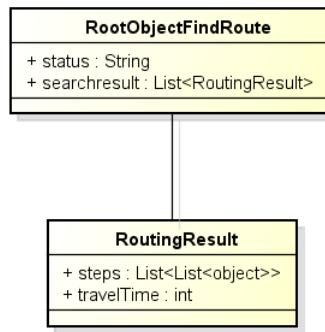
Gambar 4.10: Kelas Protocol



Gambar 4.11: Kelas Route



Gambar 4.12: Kelas SearchPlace



Gambar 4.13: Kelas FindRoute

1384 4.2.2 Kelas MainPage

1385 *MainPage* merupakan kelas turunan dari kelas *PhoneApplicationPage* yang menangani
1386 interaksi langsung antara halaman aplikasi dengan pengguna. Pada kelas ini akan ditaruh
1387 kontrol yang diperlukan. Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

- 1388** 1. protocol bertipe Protocol untuk mendapatkan URL yang digunakan dalam permintaan
1389 ke Kiri API.
- 1390** 2. IFinder bertipe LocationFinder digunakan untuk menampung semua informasi meng-
1391 enai lokasi.
- 1392** 3. httpClient bertipe HttpClient merupakan objek yang akan mengurus permintaan dan
1393 kembalian dari Kiri API.
- 1394** 4. city bertipe kumpulan String untuk menampung kota yang didukung oleh layanan
1395 Kiri.
- 1396** 5. myCity bertipe String untuk menampung kode kota sesuai Kode Penerbangan IATA.
- 1397** 6. backgroundWorker bertipe BackgroundWorker untuk mengurus pencarian lokasi di
1398 belakang layar.

1399 Berikut adalah penjelasan beberapa *method* yang dimiliki kelas ini:

- 1400** 1. Konstruktor MainPage digunakan untuk memuat komponen yang ada di halaman
1401 MainPage dan mendapatkan kota terdekat dari lokasi perangkat.
- 1402** 2. *method* OnNavigatedTo digunakan untuk mendapatkan lokasi dari peta dan menda-
1403 patkan objek LocationFinder. *Method* ini secara otomatis dipanggil jika pengguna
1404 kembali ke kelas MainPage. *method* ini memiliki parameter NavigationEventArgs.
- 1405** 3. *method* OnNavigatedFrom digunakan untuk mengirimkan *state* objek LocationFinder
1406 saat berpindah kelas. *Method* ini secara otomatis dipanggil jika pengguna berpindah
1407 dari kelas MainPage ke kelas lain. *method* ini memiliki parameter NavigationEventArgs.
- 1408** 4. *method* Application_Deactivated digunakan untuk menyimpan *state* objek saat me-
1409 ninggalkan aplikasi. *method* ini memiliki parameter object dan DeactivatedEventArgs.

- 1411 5. *method* Application_Activated digunakan untuk mengambil *state* objek saat kembali
1412 ke aplikasi. *method* ini memiliki parameter object dan ActivatedEventArgs.
- 1413 6. *method* startRoute digunakan untuk mendapatkan masukan pengguna. Jika masukan
1414 didapat dari peta atau lokasi perangkat berarti sudah dalam kordinat, namun jika
1415 masukan didapat dari *query* maka akan dicari lokasi yang terkait sesuai *query* tersebut.
1416 Lokasi terkait yang didapatkan akan dikembalikan ke pengguna untuk dipilih. Setelah
1417 pengguna lokasi dalam bentuk kordinat didapatkan maka kelas akan diarahkan ke kelas
1418 Route. *method* ini memiliki parameter objek tombol dan event.
- 1419 7. *method* changeMapFrom digunakan untuk berpindah ke halaman mapFrom. *method*
1420 ini memiliki parameter objek tombol dan event.
- 1421 8. *method* changeMapTo digunakan untuk berpindah ke halaman mapTo. *method* ini
1422 memiliki parameter objek tombol dan event.
- 1423 9. *method* getHereFrom digunakan untuk mendapatkan kordinat perangkat lalu menyimpan
1424 nilainya di kordinat asal di kelas LocationFinder dan menulisakan "Here" pada
1425 TextBox lokasi asal. *method* ini memiliki parameter objek tombol dan event.
- 1426 10. *method* getHereTo digunakan untuk mendapatkan kordinat perangkat lalu menyimpan
1427 nilainya di kordinat tujuan di kelas LocationFinder dan menulisakan "Here" pada
1428 TextBox lokasi tujuan. *method* ini memiliki parameter objek tombol dan event.
- 1429 11. *method* getListItem digunakan untuk membuat listBox lalu menampilkan ke pengguna.
1430 *method* ini memiliki parameter RootObjectSearchPlace dan string yang menunjukkan
1431 list yang ditampilkan untuk lokasi asal dan lokasi tujuan.
- 1432 12. *method* ListBoxSelectedPlace digunakan untuk mendapatkan tempat asal yang dipilih
1433 pengguna. *method* ini memiliki parameter objek dan event SelectionChangedEventArgs.
1434 gs.
- 1435 13. *method* searchCoordinatePlace digunakan untuk mencari kordinat dari tempat pilihan
1436 pengguna. *method* ini memiliki parameter ListBox dan tempat yang dipilih dalam
1437 bentuk string.
- 1438 14. *method* findRoute digunakan untuk berpindah ke kelas Route jika lokasi asal dan lokasi
1439 tujuan sudah ditentukan.
- 1440 15. *method* changeCity digunakan untuk mengubah kota tujuan dari pencarian. *method*
1441 ini memiliki parameter objek dan event SelectionChangedEventArgs.
- 1442 16. *method* showCity digunakan untuk mencari kota yang paling dekat dengan lokasi per-
1443 angkat.
- 1444 17. *method* ShowSplash digunakan untuk menampilkan tampilan awal untuk proses inisi-
1445 alisasi aplikasi.
- 1446 18. *method* StartLoadingData digunakan untuk memanggil BackgroundWorker. Backgro-
1447 undWorker digunakan untuk melakukan aksi di belakang layar.

- 1448 19. *method* backgroundWorker_DoWork digunakan untuk melakukan pemanggilan aksi
1449 di belakang layar. *method* ini memiliki parameter objek dan DoWorkEventArgs.
- 1450 20. *method* backgroundWorker_RunWorkerCompleted digunakan untuk melakukan pe-
1451 manggilan saat BackgroundWorker selesai melakukn tugasnya. *method* ini memiliki
1452 parameter objek dan RunWorkerCompletedEventArgs.

1453 **4.2.3 Kelas *City***

1454 *City* merupakan kelas yang menyimpan kota-kota yang mendukung pencarian rute ken-
1455 daraam umum dengan bantuan Kiri. Berikut adalah penjelasan atribut-atribut yang dimiliki
1456 kelas ini:

- 1457 1. centerCity bertipe *array of GeoCoordinate* untuk menyimpan kordinat pusat dari kota.
- 1458 2. city bertipe *array of String* untuk menyimpan nama kota.
- 1459 3. cityCode bertipe *array of String* untuk menyimpan kode kota dalam huruf kecil sesuai
1460 aturan IATA Airport Code.

1461 Berikut adalah penjelasan beberapa *method* yang dimiliki kelas ini:

- 1462 1. Konstruktor City digunakan untuk untuk inisialisasi nilai atribut.
- 1463 2. *method* getNearby digunakan untuk mencari kota terdekat dengan lokasi perangkat.
1464 *method* ini mengembalikan interger yang merupakan index kota pada atribut city.
1465 *method* ini memiliki 2 buah parameter yaitu *latitude* dan *longitude* yang bertipe double.
- 1466 3. *method* getIndexFromCityCode digunakan untuk mencari index pada *array* sesuai kode
1467 kota. *method* ini memiliki parameter bertipe String yang merupakan kode kota.

1468 **4.2.4 Kelas *BackgroundWorker***

1469 *BackgroundWorker* merupakan kelas yang dipakai untuk mengeksekusi operasi pada
1470 *thread* terpisah. Berikut adalah penjelasan *event* yang dimiliki kelas ini dan dipakai untuk
1471 perancangan aplikasi:

- 1472 1. *Event* DoWork
- 1473 2. *Event* RunWorkerCompleted

1474 Berikut adalah penjelasan beberapa *method* yang dimiliki kelas ini:

- 1475 1. *method* RunWorkerAsync digunakan untuk memulai operasi di belakang layar.

1476 **4.2.5 Kelas *Geocoordinate***

1477 *Geocoordinate* merupakan kelas bawaan dari Windows Phone yang akan dimanfaatkan
1478 untuk membaca *latitude* dan *longitude*.

1479 **4.2.6 Kelas *Geolocator***

1480 *Geolocator* merupakan kelas bawaan Windows Phone untuk mengkases lokasi. Dengan
1481 bantuan kelas ini maka dapat mengetahui status lokasi dari perangkat dan menemukan lokasi
1482 secara akurat.

1483 **4.2.7 Kelas *Geoposition***

1484 *Geoposition* merupakan kelas yang menampung lokasi sesuai kembalian *Geolocator*.

1485 **4.2.8 Kelas *LocationFinder***

1486 *LocationFinder* merupakan kelas yang akan menampung lokasi dan pencarian lokasi.
1487 Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

- 1488 1. coorLat bertipe Double untuk menampung kordinat latitude pengguna.
- 1489 2. coorLong bertipe Double untuk menampung kordinat longitude pengguna.
- 1490 3. coorLatFrom bertipe Double untuk menampung kordinat latitude lokasi asal yang
1491 diinginkan pengguna.
- 1492 4. coorLongFrom bertipe Double untuk menampung kordinat longitude lokasi asal yang
1493 diinginkan pengguna.
- 1494 5. coorLatTo bertipe Double untuk menampung kordinat latitude lokasi tujuan yang
1495 diinginkan pengguna.
- 1496 6. coorLongTo bertipe Double untuk menampung kordinat longitude lokasi tujuan yang
1497 diinginkan pengguna.
- 1498 7. addressDevice bertipe String untuk menyimpan alamat perangkat berada.
- 1499 8. addressDeviceFrom bertipe String untuk menyimpan alamat berdasarkan lokasi lokasi
1500 asal yang diinginkan pengguna.
- 1501 9. addressDeviceTo bertipe String untuk menyimpan alamat berdasarkan lokasi tujuan
1502 yang diinginkan pengguna.
- 1503 10. geolocator bertipe Geolocator untuk menampung pengaturan mendapatkan lokasi.
- 1504 11. myReverseGeocodeQuery bertipe ReverseGeocodeQuery untuk konversi dari alamat
1505 ke lokasi dan sebaliknya.
- 1506 12. myCoordinate bertipe GeoCoordinate untuk menampung kordinat geografis.
- 1507 13. accuracy bertipe double untuk menampung akurasi perangkat mendapatkan lokasi.

1508 Berikut adalah penjelasan beberapa *method* yang dimiliki kelas ini:

- 1509 1. Konstruktor LocationFinder berfungsi mengatur atribut geolocator dan mencari lokasi
1510 perangkat.

- 1511 2. *method* findLocation berfungsi inisialisasi GPS lalu mendapat kordinat dan menampungnya di atribut.
- 1512 3. *method* updateAccuracy berfungsi untuk mengubah nilai akurasi dari perangkat.
- 1514 4. *method* setPositionChanged berfungsi mengubah atribut coorLat, atribut coorLong, dan akurasi jika terdapat perubahan lokasi. *method* ini memiliki parameter Geolocator dan PositionChangedEventArgs yang akan menjalankan *method* jika terdapat perubahan yang diberitahukan melalui kelas Geolocator.
- 1518 5. *method* GetCurrentCoordinate berfungsi mengubah posisi saat ini, posisi lokasi asal, dan lokasi tujuan. *method* ini memiliki tiga buah parameter *latitude* bertipe Double, *longitude* bertipe Double, dan paramFor bertipe String. Parameter *latitude* dan *longitude* merupakan lokasi sedangkan parameter paramFor digunakan sebagai tujuan perubahan lokasi.
- 1523 6. *method* ReverseGeocodeQueryFrom_QueryCompleted berfungsi untuk mencari alamat lokasi asal. *method* ini memiliki parameter objek dan QueryCompletedEventArgs<IList<MapLocation>>.
- 1526 7. *method* ReverseGeocodeQueryTo_QueryCompleted berfungsi untuk mencari alamat lokasi tujuan. *method* ini memiliki parameter objek dan QueryCompletedEventArgs<IList<MapLocation>>.
- 1529 8. *method* ReverseGeocodeQuery_QueryCompleted berfungsi untuk mencari alamat lokasi perangkat. *method* ini memiliki parameter objek dan QueryCompletedEventArgs<IList<MapLocation>>.
- 1532 9. *method* setCoordinateHere berfungsi untuk menyimpan kordinat dan alamat perangkat ke kordinat dan alamat lokasi asal dan lokasi tujuan. *method* ini memiliki parameter paramFor bertipe String yang akan digunakan sebagai masukan disimpannya lokasi perangkat.
- 1536 10. *method* reset berfungsi untuk memasang kembali lokasi asal dan lokasi tujuan.

1537 **4.2.9 Kelas Map**

1538 *Map* merupakan kelas yang akan mendapatkan titik yang ditunjuk pengguna pada peta
1539 lalu menerjemahkannya dalam bentuk titik kordinat. Berikut adalah penjelasan atribut-
1540 atribut yang dimiliki kelas ini:

- 1541 1. IFinder bertipe LocationFinder digunakan untuk menampung semua informasi mengenai lokasi.
- 1543 2. fromMapFor bertipe string digunakan sebagai indikator lokasi asal atau lokasi tujuan yang didapatkan dari map.

1545 Berikut adalah penjelasan beberapa *method* yang dimiliki kelas ini:

- 1546 1. Konstruktor Map untuk inisialisasi dan penambahan *event* mengetuk pada peta.

- 1547 2. *method* OnNavigatedTo berfungsi untuk mendapatkan masukan lokasi asal atau lokasi
1548 tujuan yang akan ditentukan dari map untuk kemudian ditampung di Objek Location-
1549 Finder. *method* ini memiliki sebuah parameter NavigationEventArgs.
- 1550 3. *method* OnNavigatedFrom digunakan untuk mengirimkan *state* objek LocationFinder
1551 saat berpindah kelas. *method* ini memiliki parameter NavigationEventArgs.
- 1552 4. *method* ShowMyLocationOnTheMap digunakan untuk memberitahu dan menandai
1553 lokasi perangkat.
- 1554 5. *method* map_Tap berfungsi untuk menandai lokasi yang ditunjuk pengguna lalu me-
1555 nerjemahkan lokasi yang ditunjuk pengguna pada peta dan mengirimnya ke kelas Lo-
1556 cationFinder.
- 1557 6. *method* pilihLokasi berfungsi berpindah ke kelas MainPage dan memberitahu kelas
1558 MainPage bahwa lokasi sudah dipilih. *method* ini memiliki parameter objek tombol
1559 dan event.

1560 **4.2.10 Kelas *HttpClient***

1561 *HttpClient* merupakan kelas bawaan Windows Phone untuk mengatur pengiriman dan
1562 kembalian menggunakan protokol HTTP. Berikut adalah penjelasan *method* kelas *HttpClient*
1563 yang dipakai untuk perancangan aplikasi ini:

- 1564 1. *method* GetStringAsync membutuhkan parameter alamat bertipe string dan mengem-
1565 balikan kembalian dari Kiri dalam bentuk Task<string>.

1566 **4.2.11 Kelas *JsonConvert***

1567 *JsonConvert* merupakan kelas yang menyediakan *method* untuk mengkonversi berbagai
1568 jenis komponen *common language runtime* dan *JSON*. Kelas ini merupakan bagian *namespa-*
1569 ce *Newtonsoft*. Berikut adalah penjelasan *method* yang dipakai untuk perancangan aplikasi:

- 1570 1. *method* DeserializeObject berfungsi untuk konversi dari bentuk string menjadi objek.
1571 *method* ini memiliki satu parameter bertipe string lalu mengembalikan string tersebut
1572 dalam bentuk objek.

1573 **4.2.12 Kelas *Protocol***

1574 *Protocol* merupakan kelas untuk menampung semua alamat dalam pengiriman menggu-
1575 nakan protokol HTTP. Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

- 1576 1. uri_version bertipe string digunakan untuk menyimpan nama dari parameter uri.
- 1577 2. uri_mode bertipe string digunakan untuk menyimpan nama dari parameter mode.
- 1578 3. uri_locale bertipe string digunakan untuk menyimpan nama dari parameter locale.
- 1579 4. uri_start bertipe string digunakan untuk menyimpan nama dari parameter start.
- 1580 5. uri_finish bertipe string digunakan untuk menyimpan nama dari parameter finish.

- 1581 6. uri_presentation bertipe string digunakan untuk menyimpan nama dari parameter
1582 presentation.
- 1583 7. uri_apikey bertipe string digunakan untuk menyimpan nama dari parameter apikey.
- 1584 8. uri_region bertipe string digunakan untuk menyimpan nama dari parameter region.
- 1585 9. uri_query bertipe string digunakan untuk menyimpan nama dari parameter query.
- 1586 10. apiKey bertipe string digunakan untuk menyimpan nilai kunci API untuk mengirim
1587 permintaan ke Kiri.
- 1588 11. hostname bertipe string digunakan untuk digunakan untuk menyimpan alamat host
1589 dari Kiri.
- 1590 12. handle bertipe string digunakan untuk menyimpan alamat host ditambah "handle.php".
- 1591 13. iconPath bertipe string digunakan untuk menyimpan lokasi gambar yang dibutuhkan.
- 1592 14. iconStart bertipe string digunakan untuk menyimpan lokasi gambar awal perjalanan
1593 dari lokasi awal.
- 1594 15. iconFinish bertipe string digunakan untuk menyimpan lokasi gambar akhir perjalanan
1595 ke lokasi tujuan.
- 1596 16. version_2 bertipe string digunakan untuk menyimpan nilai versi dari API yang di-
1597 gunaikan (saat pembuatan penelitian ini versi Kiri API yang digunakan adalah versi
1598 2).
- 1599 17. modeFind bertipe string yang digunakan untuk menyimpan nilai "searchplace" yang
1600 merupakan mode mencari lokasi terkait pada Kiri API.
- 1601 18. modeRoute bertipe string yang digunakan untuk menyimpan nilai "findroute" yang
1602 merupakan mode mencari rute pada Kiri API.
- 1603 19. modeNearby bertipe string yang digunakan untuk menyimpan nilai "nearbytransport"
1604 " yang merupakan mode mencari lokasi terdekat pada Kiri API.
- 1605 20. localeId bertipe string yang digunakan untuk menyimpan nilai bahasa jika kembalian
1606 yang diinginkan ingin berbahasa Indonesia.
- 1607 21. localeEn bertipe string yang digunakan untuk menyimpan nilai bahasa jika kembalian
1608 yang diinginkan ingin berbahasa Inggris.
- 1609 22. presentationMobile bertipe string yang digunakan untuk menyimpan nilai penyajian
1610 untuk perangkat *mobile*.
- 1611 23. presentationDesktop bertipe string yang digunakan untuk menyimpan nilai penyajian
1612 untuk perangkat *desktop*.

1613 Berikut adalah penjelasan beberapa *method* yang dimiliki kelas ini:

- 1614 1. *method* getTypeTransport merupakan *method* yang akan mengembalikan alamat dari
1615 gambar transportasi dengan bingkai. *method* ini memiliki 2 parameter yaitu means
1616 sebagai tipe transportasi dan meansDetail sebagai nama kendaraan.
- 1617 2. *method* getTypeTransportWOBaloon merupakan *method* yang akan mengembalikan
1618 alamat dari gambar transportasi tanpa bingkai tambahan. *method* ini memiliki 2 par-
1619 meter yaitu means sebagai tipe transportasi dan meansDetail sebagai nama kendaraan.
- 1620 3. getSearchPlace merupakan *method* yang akan mengembalikan URI pencarian lokasi
1621 sesuai paramater. Parameter yang dimaksud adalah kata kunci masukan pengguna.
- 1622 4. *method* getFindRoute merupakan *method* yang akan mengembalikan URI pencarian
1623 rute sesuai parameter. Parameter yang dimaksud adalah kordinat lokasi asal dan
1624 kordinat lokasi tujuan yang bertipe string.
- 1625 5. *method* getRequestSearch digunakan untuk mendapatkan lokasi terkait sesuai masuk-
1626 an pengguna. *method* ini akan mengembalikan Task<RootObjectSearchPlace> karena
1627 menggunakan operasi *asynchronous*. *method* ini memiliki parameter kata kunci ma-
1628 sukan pengguna dan kota yang masing-masing parameter bertipe String.
- 1629 6. *method* getRequestRoute digunakan untuk mendapatkan rute sesuai lokasi asal dan
1630 lokasi tujuan. *method* ini akan mengembalikan Task<RootObjectFindRoute> karena
1631 menggunakan operasi *asynchronous*. *method* ini memiliki parameter *latitude* lokasi
1632 asal, *longitude* lokasi asal, *latitude* lokasi tujuan, dan *longitude* lokasi tujuan yang
1633 masing-masing bertipe double.

1634 4.2.13 Kelas *RootObjectSearchPlace*

1635 *RootObjectSearchPlace* merupakan kelas untuk menampung objek hasil pencarian lokasi.
1636 Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

- 1637 1. status bertipe *string* digunakan untuk menampung hasil kembalian status dari Kiri.
- 1638 2. searchresult bertipe *list* dan menampung banyak objek *SearchResult*.
- 1639 3. attributions bertipe objek untuk menampung attributions.

1640 4.2.14 Kelas *SearchResult*

1641 *SearchResult* merupakan kelas untuk menampung nama tempat dan kordinat dari nama
1642 tempat tersebut. Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

- 1643 1. placename bertipe *string* digunakan untuk menampung nama tempat.
- 1644 2. location bertipe *string* digunakan untuk menampung nama tempat.

1645 4.2.15 Kelas *RootObjectFindRoute*

1646 *RootObjectFindRoute* merupakan kelas untuk menampung hasil pencarian rute. Berikut
1647 adalah penjelasan atribut-atribut yang dimiliki kelas ini:

- 1648 1. status

1649 2. routingresults

1650 **4.2.16 Kelas *RoutingResult***

1651 *RoutingResult* merupakan kelas untuk menampung langkah menuju tempat tujuan dan
1652 waktu yang dibutuhkan. Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

1653 1. steps

1654 2. traveltime

1655 **4.2.17 Kelas *Route***

1656 *Route* merupakan kelas untuk pencarian rute dan menampilkannya kepada pengguna.
1657 Berikut adalah penjelasan atribut-atribut yang dimiliki kelas ini:

1658 1. listBoxStatus bertipe boolean digunakan untuk menentukan nilai status rute dalam
1659 bentuk daftar sedang tertutup atau terbuka.

1660 2. IFinder bertipe LocationFinder digunakan untuk menampung semua informasi meng-
1661 enai lokasi.

1662 3. arrayFocus bertipe *array of Point* digunakan untuk menampung titik fokus perubahan
1663 jenis transportasi yang digunakan pengguna.

1664 4. detailRoute bertipe *array of String* digunakan untuk menampung keterangan yang
1665 dibutuhkan pengguna dari Kiri API.

1666 5. focusPointNumber bertipe *integer* digunakan untuk menentukan *index of* dari atribut
1667 arrayFocus dan detailRoute.

1668 6. routeLayer bertipe MapLayer digunakan untuk menampung *Polyline* dan *Pushpin*

1669 7. myLocationLayer bertipe MapLayer digunakan untuk menampung titik dari lokasi
1670 perangakt berada.

1671 8. p bertipe Protocol digunakan untuk menampung permintaan data dengan Kiri API.

1672 9. geolocator bertipe Geolocator untuk menangani perubahan lokasi yang terjadi.

1673 10. newTimer bertipe DispatcherTimer digunakan untuk membuat perngatur waktu.

1674 11. timeOut bertipe boolean digunakan untuk menentukan nilai status apakah waktu un-
1675 tuk satu proses sudah mencapai batas.

1676 12. backgroundWorker bertipe BackgroundWorker digunakan untuk menangain proses di
1677 belakang layar.

1678 Berikut adalah penjelasan beberapa *method* yang dimiliki kelas ini:

1679 1. Konstruktor Route digunakan untuk memuat komponen yang ada di halaman Route,
1680 inisialisasi atribu, dan pemanggilan backgroundWorker.

- 1681 2. *method* OnNavigatedTo digunakan untuk mendapatkan objek LocationFinder dan me-
1682 manggil *method* TrackLocation. *method* ini memiliki parameter NavigationEventArgs.
- 1683 3. *method* OnNavigatedFrom digunakan untuk mengirimkan *state* objek LocationFinder
1684 saat berpindah kelas. *method* ini memiliki parameter NavigationEventArgs.
- 1685 4. *method* Application_Deactivated digunakan untuk menyimpan *state* objek saat me-
1686 ninggalkan aplikasi. *method* ini memiliki parameter object dan DeactivatedEventArgs.
- 1687 5. *method* TrackLocation digunakan untuk inisialisasi GeoLocator dan memulai pe-
1688 lacakan lokasi terus menerus.
- 1689 6. *method* geolocator_StatusChanged digunakan untuk mengetahui status GeoLocator.
- 1690 7. *method* geolocator_PositionChanged digunakan untuk mengetahui perubahan lokasi
1691 yang terjadi dan menyimpannya di kelas LocationFinder.
- 1692 8. *method* Find digunakan untuk mencari rute yang dicari pengguna.
- 1693 9. *method* setRouteToMap digunakan untuk menggambar rute dan lokasi pada *layer* pe-
1694 ta. *method* ini memiliki parameter RootObjectFindRoute yang merupakan objek un-
1695 tuk pencarian rute.
- 1696 10. *method* setFocus digunakan untuk mengarahkan pusat pandangan ke titik lokasi per-
1697 gantian jenis transportasi. *method* ini memiliki parameter objek dan RoutedEventArgs.
- 1698 11. *method* toMyLocation digunakan untuk mengarahkan pusat pandangan ke titik lokasi
1700 perangakat berada. *method* ini memiliki parameter objek dan RoutedEventArgs.
- 1701 12. *method* ShowList digunakan untuk membuka dan menutup rute dalam bentuk daftar.
1702 *method* ini memiliki parameter objek dan RoutedEventArgs.
- 1703 13. *method* createNew digunakan untuk membuat objek Pushpins. *method* ini memiliki
1704 parameter uri bertipe String, transport bertipe String yang menandakan jenis trans-
1705 portasi, dan geoCoordinate bertipe GeoCoordinate sebagai lokasi dari Pushpins.
- 1706 14. *method* drawMyLocationOnTheMap digunakan untuk membuat penanda lokasi di la-
1707 pisan myLocationLayer pada peta. *method* ini memiliki parameter latitude bertipe
1708 double dan longitude bertipe double.
- 1709 15. *method* back digunakan untuk konfirmasi ke pengguna jika pengguna ingin mening-
1710 galkan aplikasi. *method* ini memiliki dua buah parameter sender bertipe objek dan e
1711 bertipe RoutedEventArgs.
- 1712 16. *method* ShowLoading digunakan untuk memunculkan *popup* menunggu.
- 1713 17. *method* StartLoadingData digunakan untuk pemanggilan BackgroundWorker.
- 1714 18. *method* backgroundWorker_DoWork digunakan untuk eksekusi *method* dengan Bac-
1715 kgroundWorker. *method* ini memiliki dua buah parameter sender bertipe objek dan e
1716 bertipe DoWorkEventArgs.

- 1717 19. *method* backgroundWorker_RunWorkerCompleted digunakan untuk menutup *popup*
 1718 menunggu jika semua *method* sudah selesai dijalankan. *method* ini memiliki dua buah
 1719 parameter sender bertipe objek dan e bertipe RunWorkerCompletedEventArgs.
- 1720 20. *method* OnBackKeyPress digunakan untuk konfirmasi ke pengguna jika pengguna ingin
 1721 meninggalkan aplikasi dengan menekan tombol "back". *method* ini memiliki parameter
 1722 e bertipe CancelEventArgs.

1723 4.3 Perancangan Antar Muka

1724 Pada sub bab ini akan dibahas mengenai antarmuka pada aplikasi Pencari Rute Kendaraan
 1725 Umum untuk Windows Phone. Antarmuka berfungsi sebagai jembatan yang menghubungkan
 1726 antara aplikasi dengan pengguna. Berikut ini akan dijelaskan mengenai rancangan
 1727 antarmuka aplikasi Pencari Rute Kendaraan Umum untuk Windows Phone.

1728 4.3.1 Antarmuka Kelas *MainPage*



Gambar 4.14: Antarmuka *MainPage*

1729 Antarmuka Kelas Map pada gambar ?? merupakan tampilan awal saat aplikasi
 1730 dijalankan. Antarmuka Kelas Map memiliki dua buah masukan, lima buah tombol, dan
 1731 satu menu daftar. Berikut adalah detailnya.

1732 Dua buah masukan yaitu.

- 1733 • Masukan lokasi asal
 1734 Merupakan masukan lokasi asal mula pengguna ingin melakukan perjalanan.
- 1735 • Masukan lokasi tujuan
 1736 Merupakan masukan lokasi tujuan berhentinya perjalanan.

1737 Lima buah tombol yaitu.

- 1738 • Tombol map untuk lokasi asal
 1739 Jika tombol ditekan maka akan berpindah ke kelas map untuk memilih lokasi asal
 1740 di peta. Jika di kelas Map pengguna memilih lokasi maka pada masukan lokasi asal
 1741 terdapat tulisan "Maps".

- 1742 • Tombol here untuk lokasi asal
1743 Jika tombol ditekan maka lokasi asal adalah lokasi perangkat saat tombol ditekan dan
1744 masukan lokasi asal menjadi "here".
 - 1745 • Tombol map untuk lokasi tujuan
1746 Jika tombol ditekan maka akan berpindah ke kelas map untuk memilih lokasi tujuan
1747 di peta. Jika di kelas Map pengguna memilih lokasi maka pada masukan lokasi tujuan
1748 terdapat tulisan "Maps".
 - 1749 • Tombol here untuk lokasi tujuan
1750 Jika tombol ditekan maka lokasi tujuan adalah lokasi perangkat saat tombol ditekan
1751 dan masukan lokasi tujuan menjadi "here".
 - 1752 • Tombol find
1753 Jika tombol ditekan maka akan menampilkan daftar tempat asal dan tempat tujuan
1754 lalu mengarahkan ke Kelas Route.
- 1755 Satu buah daftar yaitu.
- 1756 • Daftar kota yang tersedia
1757 Merupakan daftar kota yang tersedia (kota yang rute angkutan umumnya dapat ditemukan dengan aplikasi ini). Disaat aplikasi dijalankan maka daftar akan menunjuk ke
1758 kota terdekat tempat perangkat berada.
1759

Pilih Tempat

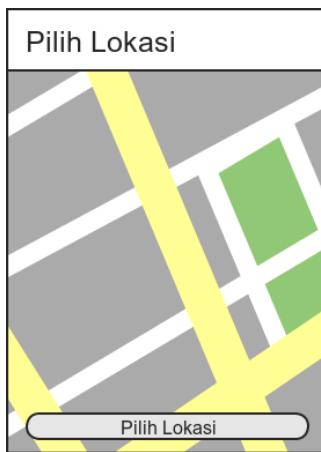
Kota 1
Kota 2
Kota 3
Kota 4
Kota 5

Gambar 4.15: Antarmuka Daftar Tempat

1760 Antarmuka Daftar Tempat pada gambar ?? merupakan daftar yang akan dimunculkan
1761 jika pengguna memasukan kata kunci pada pencarian tempat asal dan tempat tujuan.
1762 Daftar akan muncul jika didapat kembalian hasil pencarian lebih dari satu.

1763 4.3.2 Antarmuka Kelas Map

1764 Antarmuka Kelas Map pada gambar ?? merupakan antarmuka untuk menunjuk lokasi
1765 pada peta. Terdapat satu buah tombol yang akan dimunculkan jika pengguna sudah memilih
1766 lokasi. Jika tombol ditekan maka kordinat lokasi akan disimpan dan dikirim pada kelas
1767 MainPage dan halaman akan diarahkan ke kelas MainPage.



Gambar 4.16: Antarmuka *Map*

¹⁷⁶⁸ 4.3.3 Antarmuka Kelas *Route*



Gambar 4.17: Antarmuka *Route*

¹⁷⁶⁹ Antarmuka Kelas Route pada gambar ?? merupakan antarmuka untuk melihat rute
¹⁷⁷⁰ dari lokasi asal ke lokasi tujuan dalam bentuk daftar maupun peta. Terdapat empat buah
¹⁷⁷¹ tombol pada antarmuka Kelas Route. Berikut tombol yang terdapat pada Kelas Route.

- ¹⁷⁷² • Tombol prev
¹⁷⁷³ Jika tombol ditekan maka akan menunjuk titik sebelumnya pada rute peta.
- ¹⁷⁷⁴ • Tombol next
¹⁷⁷⁵ Jika tombol ditekan maka akan menunjuk titik setelahnya pada rute peta.
- ¹⁷⁷⁶ • Tombol here
¹⁷⁷⁷ Jika tombol ditekan maka akan menunjuk lokasi perangkat berada pada peta.
- ¹⁷⁷⁸ • Tombol Show List
¹⁷⁷⁹ Jika tombol ditekan maka akan menunjuk atau menyembunyikan daftar rute.

¹⁷⁸⁰ Antarmuka rute dalam bentuk daftar pada gambar ?? merupakan antarmuka untuk
¹⁷⁸¹ melihat rute secara lebih jelas dengan keterangan tahap demi tahap disertai jarak dan waktu
¹⁷⁸² perjalanan. Antarmuka daftar dapat dilihat atau disembunyikan sesuai keinginan pengguna
¹⁷⁸³ namun saat kelas Rute dibuka antarmuka daftar rute akan disembunyikan.



Gambar 4.18: Antarmuka Rute dalam bentuk Daftar

1784

BAB 5

1785

IMPLEMENTASI DAN PENGUJIAN APLIKASI

1786 Pada bab 5 akan dibahas implementasi dan pengujian aplikasi pencari rute kendaraan umum
1787 untuk Windows Phone.

1788 **5.1 Implementasi**

1789 Pada sub bab ini akan dijelaskan mengenai lingkungan yang digunakan untuk membangun
1790 aplikasi Pencari Rute Kendaraan Umum untuk Windows Phone. Pada lingkungan yang akan
1791 dibahas juga penulis membangun aplikasi sesuai rancangan yang telah dibahas pada bab 4
1792 dan mengujinya.

1793 **5.1.1 Perangkat Keras untuk Implementasi**

1794 Dalam membangun aplikasi ini perangkat keras yang digunakan adalah sebagai berikut:

1795 1. Komputer

- 1796 (a) Processor: intel Core i7-2620M CPU 2,7 GHz
- 1797 (b) RAM: 4 GB
- 1798 (c) Hardisk: 640 GB
- 1799 (d) VGA: Intel HD 3000

1800 2. Perangkat Bergerak

- 1801 (a) Processor: 1,2 GHz
- 1802 (b) RAM: 1 GB
- 1803 (c) ROM: 8 GB
- 1804 (d) Layar: 720 x 1280 pixel, 4,7 inch
- 1805 (e) GPS
- 1806 (f) Sensor: kompas, *accelerometer*

1807 **5.1.2 Perangkat Lunak untuk Implementasi**

1808 Dalam membangun aplikasi ini perangkat lunak yang digunakan adalah sebagai berikut:

1809 1. Komputer

- 1810 (a) Sistem Operasi Windows 8.1

1811 (b) IDE Visual Studio Express 2012

1812 (c) Bahasa Pemrograman C#

1813 (d) Library .Net Framework 4.5

1814 2. Perangkat Bergerak

1815 (a) Sistem Operasi Windows Phone 8.1

1816 5.1.3 Hasil Implementasi

1817 Hasil implementasi dari perangkat lunak ini terbagi dalam tiga bagian, yaitu:

1818 1. Kode Program

1819 Kode Program pada perangkat lunak ditulis dengan menggunakan bahasa c#. Bahasa
1820 C# dipilih berdasarkan analisa pada bab 3 dan kemampuan penulis.

1821 2. Hasil kompilasi program

1822 Hasil dari kompilasi program berupa file Kiri_Debug_AnyCPU.xap. File ini dapat
1823 dipasang pada perangkat dengan sistem operasi Windows Phone versi 8 atau lebih
1824 tinggi.

1825 3. Antarmuka Aplikasi

1826 Berikut merupakan hasil implementasi antarmuka aplikasi Pencari Rute Kendaraan
1827 Umum untuk Windows phone.



Gambar 5.1: Gambar antarmuka kelas MainPage



Gambar 5.2: Gambar antarmuka Splash di kelas MainPage



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



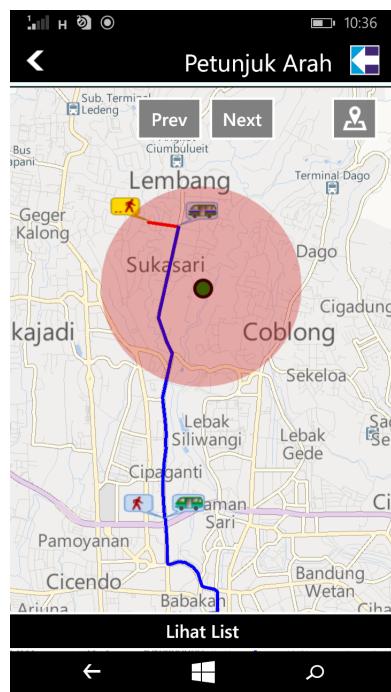
Gambar 5.4: Gambar antarmuka kelas Map



Gambar 5.5: Gambar antarmuka menunggu di kelas Route

1828 5.2 Pengujian

1829 Pada bagian ini akan dibahas mengenai hasil pengujian yang telah dilakukan terhadap
 1830 aplikasi yang telah dibangun penulis. Pengujian yang dilakukan terdiri dari dua bagian
 1831 yaitu pengujian fungsional dan pengujian eksperimental. Pengujian fungsional bertujuan
 1832 untuk memastikan semua fungsi aplikasi berjalan sesuai harapan. Sementara pengujian
 1833 eksperimental bertujuan untuk mengetahui keberhasilan proses kerja dari aplikasi.



Gambar 5.6: Gambar antarmuka kelas Route



Gambar 5.7: Gambar antarmuka *list* di kelas Route

1834 5.2.1 Lingkungan Pengujian

1835 Dalam proses pengujian perangkat lunak penulis menggunakan sistem operasi Windows
1836 Phone 8.1 dengan spesifikasi perangkat keras sebagai berikut.

1837 1. Processor : 1.2 Ghz Quad Core

1838 2. RAM : 1 GB

1839 3. Layar : 1280 x 720 pixels, 4,7 inch

1840 4. GPS : A-GPS, GLONASS, Beidou

1841 **5.2.2 Pengujian Fungsional**

1842 Pengujian fungsional dilakukan untuk menguji kesesuaian reaksi yang terjadi dengan
1843 reaksi yang diharapkan. Untuk menguji kesesuaian reaksi yang terjadi dengan reaksi yang
1844 diharapkan penulis menguji 5 orang pengguna dengan perintah sebagai berikut.

1845 1. Carilah rute dari sini ke BIP!

1846 2. Carilah rute dari BIP ke Ciwalk!

1847 3. Carilah rute dari sini ke ITB pintu jalan Ganesa!

1848 Dari tiga perintah berikut didapat hasil sebagai berikut.

1849 Penulis juga mencoba melakukan pengujian fungsionalitas untuk menguji aksi dan reaksi
1850 dari aplikasi. Hasil pengujian tersebut ditunjukan pada tabel ??.

1851 Pada pengujian keadaan aplikasi dari keadaan *running* ke *dormant* tidak terjadi.
1852 Pada saat aplikasi dalam keadaan *running* dan pengguna menekan tombol "windows" seha-
1853 rusnya keadaan aplikasi berpindah dari keadaan *running* ke *dormant* tetapi keadaan aplikasi
1854 malah *terminate*. Hal tersebut dikarenakan aplikasi yang penulis kembangkan masih dalam
1855 mode debug.

1856 **5.2.3 Pengujian Eksperimental**

1857 Pengujian eksperimental dilakukan untuk mengetahui keberhasilan aplikasi yang dibawa-
1858 gun penulis. Berikut pengujian eksperimental yang dilakukan penulis.

1859 Pengujian 1

1860 • Tempat : Rumah(Jalan Kejaksaan menuju Unpar)

1861 • Waktu : Pukul 9 pada tanggal 7 April 2014

1862 • Pengalaman :

1863 Pada pengujian 1 penulis memasukan alamat di dalam rumah untuk menuju Unpar.
1864 Saat penulis memasukan kata Unpar muncul daftar tempat sesuai kata kunci "Un-
1865 par", lalu penulis memilih "Universitas Katolik Parahyangan". Saat rute ditampilkan
1866 ada ketidaktepatan lokasi yang ditunjukan untuk lokasi asal(rumah) sedangkan un-
1867 tuk lokasi tujuan(Unpar) ditunjukan dengan tepat. Ketidaktepatan tersebut kira-kira
1868 seratus meter dan lokasi ditandai lingkaran merah dengan ukuran besar. Hal tersebut
1869 dikarenakan tempat tertutup yang membuat GPS tidak menunjukan lokasi yang
1870 akurat. Tapi saat pengujian penulis berusaha mengikuti titik naik angkutan umum.
1871 Penulis mulai dengan mengikuti petunjuk berjalan kaki menuju Jalan Tamblong. Saat
1872 keluar rumah lingkaran merah disekitar lokasi perangkat mulai mengecil yang menan-
1873 dakan akurasi GPS semakin baik. Angkutan kota yang penulis naiki pertama kali
1874 adalah angkutan kota kalapa. Sambil berada di angkot penulis mengamati rute yang
1875 ditunjukan aplikasi sudah sesuai dengan rute angkutan kota dan penunjuk lokasi me-
1876 nunjukan pergerakan dengan akurat. Angkutan kota yang penulis naiki berhenti di

1877 stasiun lalu penulis turun dan menuju Jalan Kebon Jukut. Di jalan kebon jukut pe-
1878 nulis naik angkutan kota Ciumbuleuit - St. Hall. Selama perjalanan petunjuk rute
1879 dan *polyline* sudah tepat sesuai rute angkutan kota. Angkutan kota ke dua langsung
1880 mengantarkan penulis menuju Jalan Ciumbuleuit untuk selanjutnya penulis berjalan
1881 menuju Unpar. Ketika sampai di Unpar, kordinat lokasi pada peta juga menunjukkan
1882 titik di Unpar dan memakan waktu 65 menit. Untuk perjalanan pulang penulis me-
1883 mulai dengan mencari rute dari Unpar menuju ke rumah di jalan Kejaksaan dengan
1884 mencari di peta. Perjalanan penulis mulai dengan naik angkot Ciumbuleuit - St. Hall
1885 (lurus). Angkot Ciumbuleuit - St. Hall (lurus) penulis naiki dari jalan Ciumbuleuit
1886 sampai jalan Cihampelas. Penulis turun di jalan Cihampelas dan naik angkot Kalapa
1887 - Ledeng di jalan Cihampelas. Namun di jalan Wastukencana rute pada peta menun-
1888 jukkan rute menyusuri jalan Wastukencana lalu ke jalan Aceh, namun angkot malah
1889 berbelok ke jalan Riau lalu ke jalan Purnawarman lalu ke jalan Aceh. Rute yang
1890 benar yaitu rute angkot dan bukan pada peta. Dari jalan Aceh angkot menuju jalan
1891 Sumatera lalu ke jalan Tamblong. Perjalannan penulis pun berakhir setelah sampai di
1892 rumah.

1893 **5.2.4 Perbandingan Aplikasi Pencari Rute di Android dengan Aplikasi
1894 Pencari Rute di Windows Phone**

1895 Aplikasi Pencari rute yang penulis buat dengan yang ada di Android memiliki beberapa
1896 perbedaan. Berikut perbedaan antara aplikasi pencari rute di Android dengan Windows
1897 Phone.

Pengguna	Perintah	Aksi Pengguna	Reaksi yang Diharapkan
1	1	Menekan tombol "Here" dan menulis "BIP" pada TextBox	sesuai
	2	Menulis "BIP" pada TextBox dan menulis "Ciwalk" pada TextBox	sesuai
	3	Menekan tombol "Here" dan menulis "jl Ganesa" pada TextBox	tidak sesuai
2	1	Menekan tombol "Here" dan menulis "BIP" pada TextBox	sesuai
	2	Menulis "BIP" pada TextBox dan menulis "Ciwalk" pada TextBox	sesuai
	3	Menekan tombol "Here" dan menunjuk pada peta	sesuai
3	1	Menekan tombol "Here" dan menulis "BIP" pada TextBox	sesuai
	2	Menulis "BIP" pada TextBox dan menulis "Ciwalk" pada TextBox	sesuai
	3	Menekan tombol "Here" dan menulis "ITB Ganeca" pada TextBox lalu menyadari tombol map dan menggunakannya	sesuai
4	1	Menekan tombol "Here" dan menulis "BIP" pada TextBox	sesuai
	2	Menulis "BIP" pada TextBox dan menulis "Ciwalk" pada TextBox	sesuai
	3	Menekan tombol "Here" dan menulis "Institut Teknologi Bandung" pada TextBox	tidak sesuai
5	1	Menekan tombol "Here" dan menulis "BIP" pada TextBox	sesuai
	2	Menulis "BIP" pada TextBox dan menulis "Ciwalk" pada TextBox	sesuai
	3	Menekan tombol "Here" dan menulis "jalan ganesa" pada TextBox	tidak sesuai

Tabel 5.1: Tabel Hasil pengujian fungsionalitas terhadap pengguna

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

Tabel 5.2: Tabel Hasil pengujian fungsionalitas

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

Tabel 5.3: Tabel perbedaan

1898

BAB 6

1899

KESIMPULAN DAN SARAN

1900 Bab ini berisi kesimpulan dari pembangunan aplikasi beserta saran untuk pengembangan
1901 selanjutnya.

1902 6.1 Kesimpulan

1903 Dari hasil penelitian penulis terhadap perancangan aplikasi pencari rute kendaraan
1904 umum didapat kesimpulan sebagai berikut.

- 1905 1. Pengguna dapat memasukan lokasi berdasarkan peta, lokasi perangkat, dan dengan
1906 kata kunci.
- 1907 2. Penggunaan Kiri API sudah dapat digunakan untuk mencari rute angkutan umum.
- 1908 3. Akses internet mempengaruhi performansi mendapatkan rute.

1909 6.2 Saran

1910 Berdasarkan hasil kesimpulan yang telah dipaparkan, penulis memberi saran sebagai
1911 berikut.

- 1912 1. Memanfaatkan tombol "enter" untuk memanggil *method startRoute*. Masukan dari
1913 pengguna karena setelah pengguna menentukan tujuan tombol "Find" terhalang *key-*
1914 *board* maka pengguna harus menekan tombol "back" terlebih dahulu untuk menekan
1915 tombol "Find".
- 1916 2. Memanfaatkan pencarian lokasi pada peta agar pengguna dapat memilih lokasi dengan
1917 lebih mudah.
- 1918 3. Menambahkan *gesture* dan animasi yang akan meningkatkan interaksi pengguna ter-
1919 hadap aplikasi.
- 1920 4. Mengubah beberapa tampilan yang menjadi ciri khas Windows Phone seperti meng-
1921 gunakan *application bar*.

BIBLIOGRAFI

- 1923 [1] "Windows phone silverlight development." <http://msdn.microsoft.com/library/windows/apps/ff402535.aspx>, 2014. Accessed: 2014-8-17.
- 1924
- 1925 [2] P. Manning, *Pro Windows Phone App Development*. Apress, 2013.
- 1926 [3] A. Whitechapel and S. McKenna, *Windows Phone 8 Development Internals*. Microsoft,
- 1927 2013.
- 1928 [4] T. Bray, "The JavaScript Object Notation (JSON) Data Interchange Format." RFC 7159
- 1929 (Proposed Standard), Mar. 2014.
- 1930 [5] "Kiri api v2 documentation." https://bitbucket.org/projectkiri/kiri_api/wiki/KIRI%20API%20v2%20Documentation, 2014. Accessed: 2014-8-17.
- 1931