BAB 2
LANDASAN TEORI

2.1. Ruang Lingkup Objek Penelitian

Objek penelitian yang sedang diteliti adalah perusahaan F-Travel yang berada di Kabupaten Bandung, yaitu bertempat di Jalan Sarimanah Raya No.67 Sarijadi, Bandung 40151.

2.2. Sejarah Perusahaan

PT. FARISA GLOBAL TOURISTAMA adalah perusahaan yang bergerak di bidang jasa pariwisata. PT. FARISA GLOBAL TOURISTAMA berfokus pada dua bisnis utama yaitu bisnis transportasi darat yang secara konsisten memberikan pelayanan transportasi unggulan yang telah memegang peranan penting dalam pengembangan maupun penyediaan kebutuhan layanan transportasi baik perorangan maupun korporasi dengan mengusung brand F-TRANS dalam kegiatan bisnis nya. Dan bisnis jasa perjalanan wisata yang berkomitmen mengutamakan pelayanan terbaik sesuai dengan kebutuhan dunia pariwisata baik peroangan maupun korporasi dengan mengusung brand TOURISTAMA.

2.2.1. Visi dan Misi F-Trans Bandung

F-TRANS Bandung mempunyai visi dan misi sebagai berikut:

2.2.1.1. Visi F-Trans Bandung

Menjadi perusahaan penyedia jasa transportasi terkemuka yang fokus pada sektor pariwisata dengan dukungan implementasi teknologi terkini.

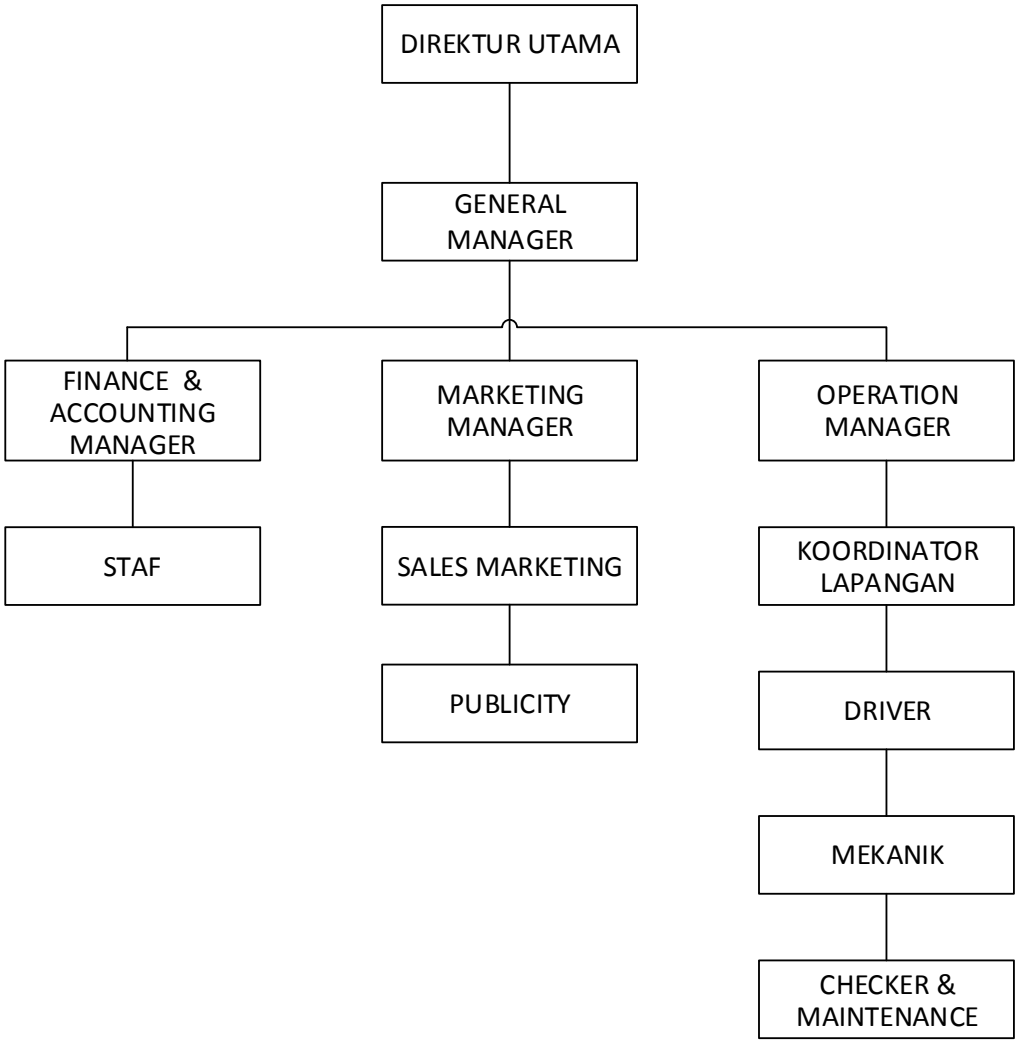
2.2.1.2. Misi F-Trans Bandung

Menyediakan produk dan jasa berkualitas mengikuti trend dan kebutuhan konsumen dengan memberikan pelayanan prima, menjalankan usaha jasa

transportasi dan jasa pariwisata yang selalu menjaga sinergitas dengan seluruh pemangku kepentingan industri pariwisata melalui pola “win-win situation”.

2.2.2. Struktur Organisasi F-Trans Bandung

Berikut ini adalah susunan organisasi yang ada di F-TRANS bandung.



Gambar 2.1 Struktur Organisasi F-Trans Bandung

Berikut adalah rincian fungsi dari masing-masing jabatan:

Tabel 2.1 Fungsi-fungsi dari jabatan

No	Jabatan	Keterangan
1	Direktur Utama	Sebagai pimpinan tertinggi dalam perusahaan juga bertugas mengawasi kerja anak buahnya tetapi tidak terjun langsung ke perusahaan.
2	General Manager	Sebagai wakil dalam mengatasi dan memecahkan masalah yang ada di dalam perusahaan serta mewakili Direktur mengambil dan membuat keputusan jika Direktur sedang tidak ada di tempat.
3	Finance & Accounting Manager	Merencanakan, mengembangkan, dan mengontrol fungsi keuangan dan akuntansi di perusahaan dalam memberikan informasi keuangan secara komprehensif dan tepat waktu untuk membantu perusahaan dalam proses pengambilan keputusan yang mendukung pencapaian target financial perusahaan.
4	Staf	Membantu pimpinan dalam melancarkan tugas-tugas organisasi, dalam memberikan pelayanan untuk seluruh lini dan unsur organisasi.
5	Marketing Manager	Merencanakan, mengontrol dan mengkoordinir proses penjualan dan pemasaran untuk mencapai target penjualan dan mengembangkan pasar secara efektif dan efisien.
8	Sales Marketing	Memiliki tugas dan tanggung jawab melaksanakan kegiatan penjualan melalui telepon terhadap target konsumen serta melengkapi laporan kegiatan untuk setiap hubungan yang dilakukan.
9	Publicity	Bertugas untuk memasarkan armada ke pelanggan dengan mengadakan promo dari perushaan.
10	Operation Manager	Memiliki tugas untuk memastikan organisasi berjalan sebaik mungkin dalam memberikan pelayanan dan memenuhi harapan para pelanggan klien.
11	Koordinator Lapangan	Memiliki tugas untuk mengatur jadwal para driver untuk memenuhi kebutuhan klien dan mengawasi para pekerja di bagian mekanik dan Checker & Maintenance

12	Driver	Bertugas untuk menjalankan armada F-TRANS Bandung sesuai dengan kebutuhan klien.
13	Mekanik	Bertugas untuk mengecek kondisi semua kendaraan dan memperbaiki armada bila ada kerusakan.
14	Checker & Maintenance	Membantu dan melaksanakan tugas yang diberikan oleh operation manager untuk menjaga kelaiakan setiap unit kendaraan serta menyusun jadwal perawartan unit kendaraan dan mengkordinir mekanik dalam dalam pelaksanaan perawatan unit kendaraan dan melaporkan hasil kegiatan perawatan pada kordinator lapangan.

2.3. Pick & Drop

Pick & Drop adalah layanan jemput dan antar pelanggan yang bersifat sekali jalan yang bersifat *one way* / *point to point* dari satu titik ke titik kedua di dalam suatu wilayah yang sudah di tentukan. Layanan ini menyediakan seperti jasa rental mobil tanpa harus membayar harga sewa per jam atau harian .

2.4. Empty run

Empty run adalah suatu istilah yang sering digunakan oleh perusahaan di bidang jasa transportasi yang merupakan layanan jemput pelanggan saja, kendaraan harus dalam keadaan kosong dan sudah selesai melakukan pelayanan untuk layanan *pick & drop*. Kendaraan yang sudah disewa bisa di pesan oleh pelanggan lain.

2.5. Aplikasi

Aplikasi merupakan program yang berisikan perintah-perintah untuk melaksanakan pengolahan data, aplikasi secara umum adalah suatu proses dari cara manual yang ditransformasikan ke komputer dengan membuat sistem atau program agar data diolah lebih berdaya guna secara optimal. Aplikasi (*application*) juga adalah *software* yang dibuat oleh suatu perusahaan komputer untuk mengerjakan tugas-tugas tertentu, misalnya Microsoft Word dan Microsoft Excel [2].

Dari pengertian di atas dapat disimpulkan bahwa aplikasi merupakan *software* yang ditransformasikan ke komputer yang berisikan perintah-perintah yang

berfungsi untuk melakukan berbagai bentuk pekerjaan atau tugas-tugas tertentu seperti penerapan, penggunaan dan penambahan data. Selain itu aplikasi dapat mengintegrasikan berbagai kemampuan komputer.

2.6. Android

Android adalah sistem operasi berbasis Linux yang dirancang untuk perangkat seluler layar sentuh seperti telepon pintar dan komputer tablet. Android awalnya dikembangkan oleh Android, Inc., dengan dukungan finansial dari Google, yang kemudian membelinya pada tahun 2005. Sistem operasi ini dirilis secara resmi pada tahun 2007, bersamaan dengan didirikannya Open Handset Alliance, konsorsium dari perusahaan-perusahaan perangkat keras, perangkat lunak, dan telekomunikasi yang bertujuan untuk memajukan standar terbuka perangkat seluler. Ponsel Android pertama mulai dijual pada bulan Oktober 2008 [3].

2.6.1. Android Life Cycle

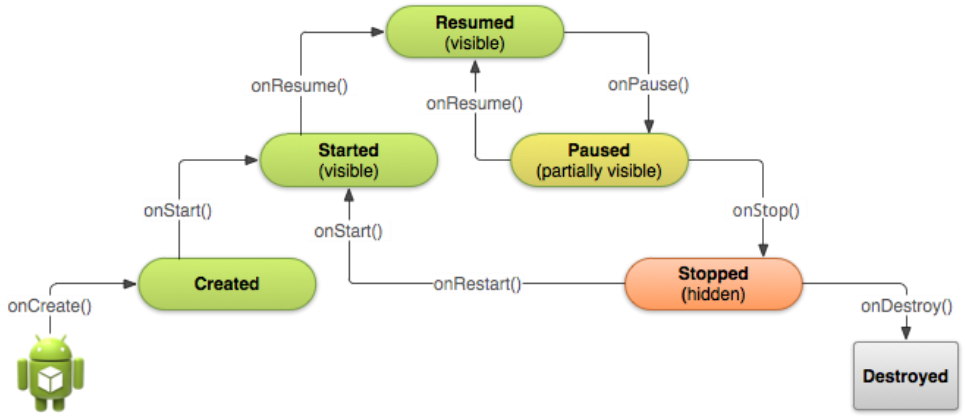
Aplikasi android terdiri dari beberapa fungsi dasar seperti mengedit catatan, memutar file musik, membunyikan alarm, atau membuka kontak telepon. Fungsi-fungsi tersebut dapat diklasifikasikan ke dalam empat komponen android yang berbeda seperti ditunjukkan pada *tabel 2.2 Komponen Aplikasi Android*, klasifikasi tersebut berdasarkan kelas-kelas dasar java yang digunakan [4].

Tabel 2.2 Komponen Aplikasi Android

Functionality	Java Base Class	Examples
Focused thing a user can do	Activity	Edit a note, play a game
Background process	Service	Play music, update weather icon
Receive messages	BroadcastReceiver	Trigger alarm upon event
Store and retrieve data	ContentProvider	Open a phone contact

Setiap aplikasi pasti menggunakan minimal satu dari komponen tersebut, akan tetapi terdapat beberapa komponen yang mengharuskan mencantumkan *specified permission* sebelum digunakan seperti komponen Service, *Broadcast Receiver* dan *Content Provide* [2].

Android memiliki paradigma pemrograman lain tidak seperti paradigma pemrograman biasa di mana aplikasi yang dijalankan pada fungsi main(), sistem android menjalankan kode dalam method Activity dengan menerapkan metode callback tertentu yang sesuai dengan tahap tertentu dari siklus hidup. Setiap aplikasi yang berjalan dalam sistem operasi android memiliki siklus hidup yang berbeda dengan aplikasi desktop atau web, Hal ini dikarenakan aplikasi mobile memiliki tingkat interupsi proses yang lumayan tinggi seperti ketika handling panggilan masuk aplikasi diharuskan menghentikan proses sementara, Penerapan siklus hidup juga berguna untuk memastikan aplikasi tidak menghabiskan sumber daya baterai pengguna [4].



Gambar 2.2 Siklus Hidup Android

Terdapat beberapa state dalam siklus hidup android yang terjadi seperti diilustrasikan pada *Gambar 2.2 Siklus Hidup Android*, akan tetapi hanya beberapa dari state tersebut yang menjadi statis diantaranya:

1. Resumed

Resumed terjadi ketika aplikasi berjalan setelah state *paused* . State ini akan menjalankan perintah program yang ditulis pada method onResume().

2. Paused

Dalam keadaan ini aktivitas yang terjadi dihentikan secara sementara tetapi masih terlihat oleh pengguna karena terdapat proses yang memiliki prioritas lebih

tinggi seperti panggilan telepon. Aplikasi tidak dapat menjalankan perintah apapun ataupun menampilkan apapun dalam state ini.

3. Stopped

Dalam keadaan ini, aplikasi benar-benar tidak ditampilkan dan tidak terlihat oleh pengguna tetapi masih meninggalkan service di background.

State lain seperti *Created* dan *Started* bersifat sementara dan sistem dengan cepat menjalankan state berikutnya dengan memanggil metode *life cycle callback* berikutnya. Artinya, setelah sistem OnCreate() dipanggil, dengan cepat sistem akan memanggil method OnStart(), kemudian diikuti oleh onResume().

2.6.2. Fitur

Android memiliki beberapa fitur utama yang sering digunakan dalam proses pembangunan aplikasi diantaranya adalah:

1. Multi-proses dan App Widgets

Sistem operasi android tidak melarang prosesor menjalankan lebih dari satu aplikasi dalam satu waktu. Sistem operasi android dapat mengatur aplikasi dan thread yang berjalan secara *multitasking.* Keuntungan yang didapat adalah ketika aplikasi berjalan dan berinteraksi dengan pengguna di layer depan sistem operasi, proses dari aplikasi lain dapat berjalan untuk melakukan pembaruan informasi. Sebagai contoh misalnya ketika pengguna memainkan game, proses lain dapat berjalan di belakang aplikasi seperti memeriksa harga saham dan memunculkan peringatan [3].

App Widgets adalah mini aplikasi yang dapat embedded dalam aplikasi seperti home screen. App widgets dapat menjalankan proses request seperti musik streaming atau mendeteksi suhu ruangan secara background.

Multi-proses dapat memberikan manfaat berupa user experience yang lebih banyak, namun penggunaan fitur tersebut dapat menghabiskan banyak energi baterai jika penggunaan tidak benar [4].

2. Touch Gestures dan Multi-touch

Touchscreen adalah *user interface* intuitif yang digunakan banyak *smartphone* di dunia. Dengan fitur ini interaksi dapat dibuat lebih mudah karena cukup dengan menggunakan jari tangan. *Multi-touch* adalah kemampuan yang dapat melakukan tracking lebih dari satu tangan dalam satu waktu, Fitur ini sering digunakan untuk interaksi memperbesar atau memutar objek. Selain itu pengembang dapat membuat interaksi baru dengan memanfaatkan fitur tersebut.

3. Hard dan Soft Keyboard

Salah satu fitur pada perangkat smartphone adalah tombol fisik dan non fisik, tombol fisik digunakan untuk navigasi pendukung dalam pengoperasian android.. Pengembang aplikasi tidak perlu secara manual untuk mengintegrasikan tombol tersebut dalam aplikasi. Tombol non fisik adalah tombol yang dibuat oleh sistem operasi seperti keyboard virtual, dan tombol navigasi aplikasi.

2.6.3. Prinsip Desain

Android memiliki beberapa prinsip desain yang dapat menjadi acuan dalam membuat desain aplikasi android diantaranya adalah :

1. Multiple Assets

Android mendukung jutaan smartphone, tablet dan perangkat lain dalam berbagai ukuran layar dan ukuran, untuk itu *Multiple Assets* sangat disarankan digunakan untuk mengatasi fragmentasi pada android. Seperti ilustrasi pada *Gambar 2.3 Klasifikasi Ukuran Icon*, android menciptakan beberapa klasifikasi ukuran icon yaitu MDPI, HDPI, XHDPI, XXHDPI dan XXXHDPI. MDPI dan HDPI dikhususkan untuk icon yang akan digunakan pada device berukuran smartphone sedangkan untuk XHDPI, XXHDPI dan XXXHDPI digunakan pada device berukuran tablet [5].



Gambar 2.3 Kasifikasi Ukuran Ikon

2. *Touch Feedback*

Touch Feedback dalam android digunakan sebagai respon setiap objek yang ditekan pengguna. Hal ini bertujuan untuk memberi tahu pengguna objek mana yang berinteraksi dengan pengguna [5].



Gambar 2.4 *Touch Feedback*

3. *Pattern Gesture*

Touch gesture berguna untuk meningkatkan experience pengguna dalam menggunakan aplikasi. Terdapat beberapa gesture yang didukung oleh android diataranya adalah [5]:

a. Touch

Touch adalah gesture default sebagai event untuk item tertentu yang paling sering digunakan dalam perangkat android, hampir seluruh kontrol utama aplikasi ini menggunakan touch.

b. Long Press

Biasanya digunakan untuk seleksi data , dengan gesture ini dimungkinkan untuk memilih satu atau lebih item dalam sebuah tampilan dan menjalankan suatu fungsionalitas tertentu.

c. Swipe Or Drag

Swipe adalah menyentuh sebuah titik pada layar dan menggerakkan jari yang tetap tersentuh pada layar ke titik lain pada layar. Swipe biasanya dilakukan untuk kebutuhan berpindah layar atau tergantung dari aplikasi. Swipe dapat dilakukan dari dan ke arah mana saja.

d. Double Touch

Pada Smartphone dan tablet android, melakukan dua kali tapping secara berturut-turut pada satu obyek, fungsinya berbeda dengan double klik mouse komputer. Pada android, teknik ini biasanya dipakai untuk melakukan zoom in dan zoom out atau memperbesar / memperkecil sebuah obyek gambar.

e. Pinch Open

Teknik lain yang biasa digunakan adalah dengan menggunakan dua jari, di mana kedua jari tersebut menyentuh dua titik pada layar yang terpisah di mana ujung dari dua jari tersebut tidak bersentuhan, kemudian kedua jari tersebut sambil tetap menyentuh layar bergerak saling mendekati. Teknik ini digunakan untuk membuka aplikasi tertentu.

f. Pinch Close

Teknik Pinch Close adalah kebalikan dari Pinch Open di mana spread dilakukan dengan berawal dari dua jari yang bersentuhan pada ujungnya dan ditempelkan sebuah titik yang sama pada layar, kemudian kedua jari tersebut bergerak memisahkan atau menjauhi satu sama lain Gerakan ini untuk menutup layar.

2.7. Android Studio

Android Studio merupakan IDE yang digunakan para developer Android di seluruh dunia untuk mengembangkan aplikasi Android dengan mengusung kualitas dan performa aplikasi. Android Studio adalah IDE Android resmi yang dibangun oleh Google. Android Studio memiliki beberapa fitur unggulan yang dapat memudahkan developer Android, diantaranya [6].

1. ***Instant Run***, untuk setiap developer yang menyukai proses build yang cepat, developer Android bisa melakukan build ulang untuk setiap perubahan yang

telah dibuat dengan cepat, layaknya rendering html pada web browser. Dengan akselerasi swapping VM ke swapping aplikasi, fitur Instan Run dapat mengoptimalkan waktu pengembangan aplikasi Android.

2. ***Android Emulator***, emulator Android dapat berjalan 3x lebih cepat dibanding emulator Android sebelumnya. Peningkatan ini tidak terlepas dari peningkatan ADB yang sekarang dapat mentransfer aplikasi dan datanya 10x lebih cepat ke emulator dibanding kompilasi langsung ke device (Smartphone). Layaknya Smartphone, emulator Android juga tersedia Google Play Service sehingga developer Android dapat menguji fungsionalitas API secara langsung.
3. ***Cloud Test Lab Integration,*** tulis sekali dan jalankan dimana saja. Selain emulator Android, developer Android bisa meningkatkan pengembangan aplikasi dimana saja dengan menggunakan Cloud Test Lab yang terdiri dari berbagai device Android.
4. ***App Indexing Code Generation & Test,*** keberadaan aplikasi yang telah developer Android kembangkan sangatlah penting, dengan fitur ini, developer Android dapat mempromosikan aplikasinya dengan cara menambahkan URL yang dibangkitkan secara otomatis melalui fitur App Indexing. Dengan menambahkan URL yang indexable, developer bisa menguji keberadaan aplikasinya secara langsung melalui Android Studio minimum versi 2.0.
5. ***GPU Debuger Preview,*** untuk para developer Android khususnya yang bergelut di bidang pengembangan game. Android Studio menyediakan fitur debugger GPU, sehingga developer Android dapat melihat setiap frame dan state GL. Jika ada permasalahan rendering GL, developer Android bisa menangkap dan menganalisa stream GPU yang ada pada device Android.

2.8. Android SDK (Software Development Kit)

Sesuai dengan namanya merupakan kumpulan tools-tools yang di perlukan untuk mengembangkan aplikasi Android. Seperti pada java yang dapat dijalankan pada multiplatform, SDK Android pun tersedia untuk beberapa platform seperti Windows, Mac OS dan Linux [4].

Adapun aplikasi SDK yang didalamnya terdapat beberapa folder sebagai berikut :

1. Tools

Berisikan tools-tools yang diperlukan dalam membangun aplikasi Android seperti emulator, Android Virtual Device (AVD), SDK Manager, ddms dan banyak lagi untuk memudahkan membangun aplikasi Android. Semua tools ini dapat di update mengikuti setiap perkembangan platfom Android.

2. Platforms

Direktori ini ditujukkan untuk meletakkan versi platform-platform Android yang telah kita download dari SDK Manager. Seperti Android 1.5, 1.6, 2.0 dan seterusnya.

3. Add-ons

Direktori ini nantinya berisikan tools tambahan untuk Android SDK, yang fungsinya untuk menambahkan library yang akan digunakan untuk pengembangan aplikasi. Seperti jika ingin mengembangkan aplikasi menggunakan map dari google maka harus memerlukan platform Android yang sudah terdapat library map. Semua ini dapat di download melalui SDK Manager.

Selain folder-folder diatas, akan terdapat file tambahan jika kita mendownload beberapa platfom dan tool dalam folder Android SDK tersebut. Seperti *Docs, platform tools, dan samples*.

2.9. Android Development Tools

Merupakan plugin Eclipse yang disediakan oleh google untuk membantu dan memudahkan Developer mengembangkan aplikasi Android. Dengan adanya ADT membuat project android, mendesain tampilan aplikasi, menambahkan komponen-komponen yang berasal dari Android Framework, debug aplikasi menggunakan tools-tools pada SDK Android hingga membuat file installer menjadi lebih mudah berkat plugin ini. Untuk proses penginstallan ADT dapat dilakukan dengan 2 cara Pertama secara Offline dengan mendownload file ADT secara manual pada situs

Android dan kedua Secara online dengan menggunakan eclipse maupun Android studio [4].

2.10. Web Services

W3C mendefinisikan web service sebagai sebuah software aplikasi yang dapat teridentifikasi oleh URI dan memiliki interface yang didefinisikan, dideskripsikan, dan dimengerti oleh XML atau JSON dan juga mendukung interaksi langsung dengan software aplikasi yang lain dengan menggunakan message berbasis XML atau JSON melalui protokol internet Web service adalah sebuah software aplikasi yang tidak terpengaruh oleh platform, menyediakan method-method yang dapat diakses oleh network. Web Service juga akan menggunakan XML untuk pertukaran data, khususnya pada dua entities bisnis yang berbeda. Beberapa karakteristik dari web service adalah [7] :

1. *Message-based*
2. *Standards-based*
3. *Programming language independent*
4. *Platform-neutral*

Beberapa key standard di dalam web service adalah: JSON, XML, SOAP, WSDL and UDDI.

2.11. JSON

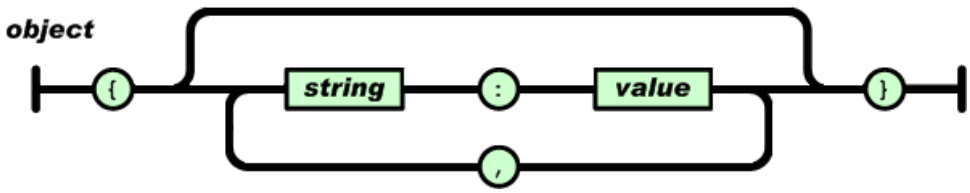
JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (generate) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemrograman JavaScript, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran data. JSON terbuat dari dua struktur [8].

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.
2. Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemrograman modern mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemrograman yang juga berdasarkan pada struktur data ini. JSON menggunakan bentuk sebagai berikut:

1. Objek

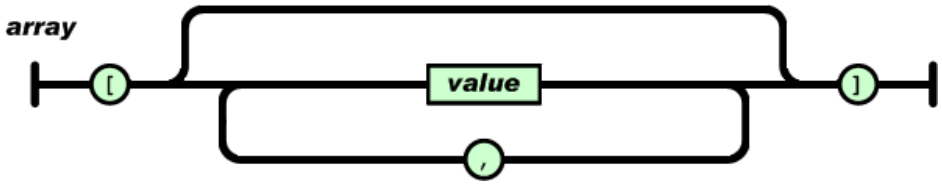
Objek adalah sepasang nama / nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma). Objek biasanya digunakan untuk menyimpan data tunggal dalam bentuk JSON.



Gambar 2.5 Objek JSON

2. Larik

Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma). Larik dalam JSON dapat digunakan sebagai value dari JSON object hal ini dapat berguna jika JSON menyimpan data bertingkat.



Gambar 2.6 Array JSON

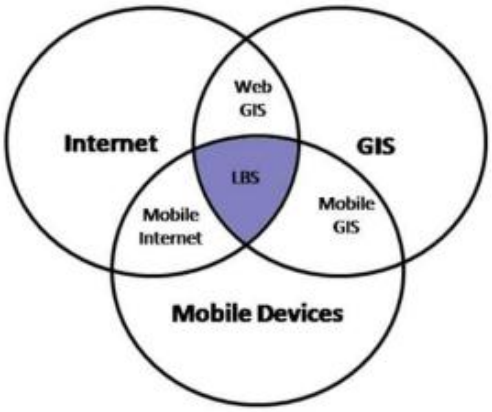
Bentuk data JSON objek dan larik dapat saling dikombinasikan untuk mendukung struktur data yang lebih kompleks. JSON mendukung beberapa tipe data untuk menjadi value seperti Angka, String, Boolean dan nilai NULL.

2.12. GPS

Global Positioning System (GPS) merupakan sistem navigasi yang menggunakan satelit dalam penggunaannya. GPS terdiri dari tiga bagian yaitu sistem kontrol, satelit dan pengguna. Sistem kontrol adalah bagian yang mengontrol pergerakan satelit – satelit yang ada dan saling berinteraksi satu sama lain kemudian pengguna adalah alat navigasi yang digunakan seperti perangkat mobile yang kini sudah memiliki fitur GPS di dalamnya. GPS biasanya digunakan untuk menunjukan suatu lokasi yang berada di permukaan bumi dengan tingkat akurasi yang cukup baik yaitu kurang dari 10 meter selama tidak ada benda padat yang dapat menghambat sinyal untuk mendapatkan lokasi pengguna [9].

2.13. Location Based Service

Location Based Service (LBS) atau layanan berbasis lokasi adalah layanan informasi yang dapat diakses melalui *mobile device* dengan menggunakan mobile network, yang dilengkapi dengan kemampuan untuk memanfaatkan lokasi dari *mobile device* tersebut. LBS memungkinkan untuk melakukan komunikasi dan interaksi dua arah. LBS ini dapat digambarkan sebagai suatu layanan yang berada pada pertemuan tiga teknologi yaitu *Geographic Information System* (GIS), *Internet Service* dan *mobile device*, seperti pada *Gambar 2.7 Teknologi Location Based Service* [9].



Gambar 2.7 Teknologi Location Based Service

2.13.1. Metode Penentu Lokasi

LBS merupakan layanan yang mampu bereaksi aktif pada suatu perubahan posisi sehingga dapat mendeteksi letak objek dan memberikan layanan sesuai dengan letak objek yang telah diketahui tersebut. Pada teknologi LBS berbasis jaringan seluler, penentuan posisi pada sebuah peralatan komunikasi ditentukan berdasarkan posisi relatif terhadap lokasi BTS (*Base Transceiver Station*). Dalam menentukan posisi dari sebuah perangkat mobile yang sedang aktif, secara umum terdapat tiga tingkat metode yang digunakan saat ini, di antaranya adalah sebagai berikut [9].

1. Metode *Basic Positioning* berbasis pada *Cell Identification* (Cell ID)

Penentuan posisi didasarkan pada daerah geografis yang tercakup oleh sebuah Cell berhubungan dengan daerah cakupan dari sinyal radio. Ketika sebuah handphone terhubung secara aktif dengan sebuah base station, berarti handphone tersebut diasumsikan berada dalam cell dari base stadion tersebut. Untuk mengukur jarak dan arah handset dari base station tidak dapat diketahi dengan pasti, oleh karena itu untuk lebih meningkatkan lagi akurasi hasil pencarian pada metode ini seringkali dikombinasikan dengan metode lain seperti:

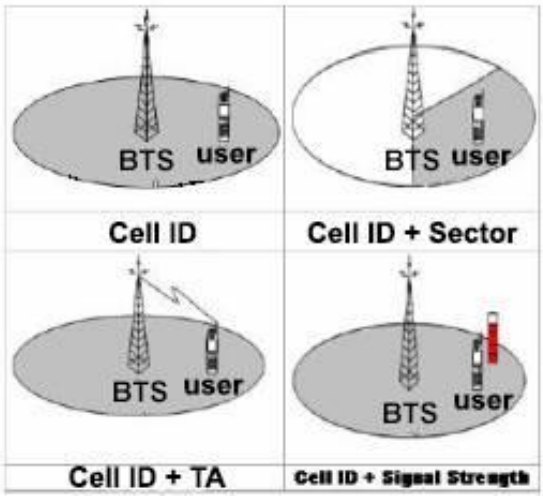
a. *Timing Advance* (TA)

Dengan menggunakan TA ini metode Cell ID akan ditambahkan sebuah fungsionalitas untuk menghitung *Round Trip* Time (RTT), yaitu waktu transmisi sebuah frame (dari base station ke handphone) dan waktu sebuah *frame* (dari

handphone ke station). Dengan tambahan ini jarak antara handphone dan base station dapat ditentukan dengan keakuratan 50 m.

b. Network Measurement Report (NMR)

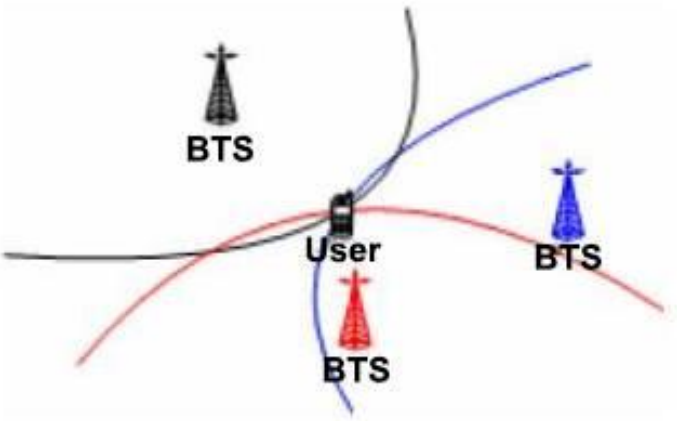
Berdasarkan pada besar kecilnya sinyal (Receiver Signal Strength) yang diterima handphone yang ada di suatu "sector cell", maka posisi itu dapat ditentukan dengan lebih akurat.



Gambar 2.8 Network Measurement Report

2. Metode *Enhanced Positioning*

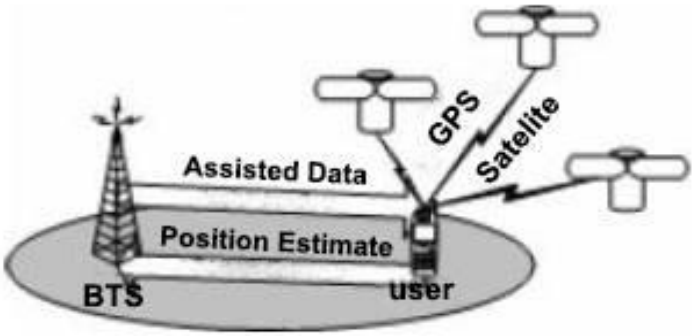
Metode ini menggunakan pendekatan *Observe Time Difference* (OTD) dengan jaringan GSM yang sering digunakan seperti Enhanced-OTD (EOTD). E-OTD adalah metode pencarian posisi yang berdasarkan pada waktu. Untuk menentukan posisi relative, sebuah handphone harus aktif terhadap tiga base station dan perlu ditentukan terlebih dahulu jarak handphone terhadap masing- masing base station berdasarkan waktu yang ditempuh oleh sebuah sinyal dari handphone ke masing – masing *base station.* Untuk menentukan posisi dari handphone yang sedang aktif digunakan rumus matematika untuk triangulasi. Dengan metode ini akurasinya akan meningkat hingga ketilitian kurang dari 50 m.



Gambar 2.9 *Enhahed Positioning*

3. Metode *Advance Positioning*

Metode ini menggunakan teknologi *Assisted-Global Positioning System* (AGPS). A-GPS ini merupakan metode yang berbasis pada waktu. Pada metode ini, akan dilakukan pengukuran waktu tiba dari sebuah sinyal yang dikirim dari tiga buah satelit GPS. Hal ini berarti handset harus memiliki fasilitas untuk mengakses GPS. A-GPS juga menghasilkan akurasi secara vertical dan estimasi jarak yang baik. Akurasinya pun kurang dari 10 m.



Gambar 2.10 *Advance Positioning*

2.13.2. Komponen Location Based Service

Dalam menggunakan *Location Based Service* atau layanan berbasis lokasi terdapat beberapa elemen yang diperlukan, antara lain [9].

1. *Mobile Devices*

   Sebuah alat yang digunakan untuk meminta informasi yang dibutuhkan seperti perangkat *mobile smartphone* yang memiliki fasilitas navigasi.

2. *Communication Network*

   Jaringan selular yang mengirimkan data pengguna dan permintaan layanan.

3. *Positioning Component*

   Untuk dapat mengolah layanan harus menentukan lokasi pengguna. Posisi pengguna ini dapat diperoleh menggunakan jaringan komunikasi atau dengan menggunakan Global Positioning System (GPS)

4. *Service and Application Provider*

   Penyedia layanan pengguna selular yang bertanggung jawab untuk memproses suatu layanan.

5. *Data and Content Provider*

   Penyedia layanan informasi data yang dapat diminta oleh pengguna.

2.14. Latitude (Garis Lintang)

Latitude adalah garis yang melintang di antara kutub utara dan kutub selatan yang menghubungkan sisi bagian bumi dari timur dan barat. Garis lintang memiliki posisi membentangi bumi seperti halnya garis khatulistiwa tetapi dengan kondisi nilai tertentu. Latitude dibedakan menjadi dua wilayah yaitu lintang utara dan selatan dimana nilai koordinat di bagian utara selalu positif dan koordinat selatan selalu negative [9].

2.15. Longitude (Garis Bujur)

Longitude adalah garis bujur yang menghubungkan antara sisi utara dan sisi selatan permukaan bumi. Garis bujur digunakan untuk mengukur sisi barat dan timur koordinat suatu titik di belahan bumi. Longitude dibedakan menjadi dua wilayah yaitu bujur timur dan bujur barat dimana nilai koordinat yang berada di timur selalu negatif dan nilai koordinat dibagian barat selalu positif [9].

2.16. Google Maps

Google Maps adalah sebuah layanan gratis dari Google tentang berbagai macam teknologi pemetaan. Google Maps menampilkan peta dunia secara digital sehingga lebih memudahkan ketika ingin mengetahui suatu lokasi tertentu. Google Maps

menyediakan beberapa keunggulan seperti peta jalan, menunjukan rute untuk suatu perjalanan yang diinginkan baik itu melalui jalur mobil sampai pejalan kaki [9].

2.17. Google Maps API

Google Maps API adalah sebuah layanan yang memungkinkan Google Maps agar dapat diintegrasikan ke dalam aplikasi yang memang membutuhkan fitur dari Google Maps ini. Google Maps API adalah sebuah library Javascript. Dengan menggunakan Google Maps API ini dapat memudahkan pengembang dalam membuat aplikasi yang membutuhkan teknologi peta di dalamnya dan tidak perlu memikirkan cara membuatnya secara manual karena dengan mengunakan Google Maps API ini sudah semua yang dibutuhkan terkait teknologi peta digital sudah cukup lengkap dan dapat dimanfaatkan untuk aplikasi yang membutuhkannya [7].

Saat ini Google Maps API telah mengalami perkembangan, hingga saat ini Google Maps API telah mencapai versi 3. Namun, Google Maps API versi 3 ini masih hanya bisa digunakan oleh pemrograman berbasis web dengan menggunakan Javascript. Untuk penggunaan Google Maps API pada perangkat mobile seperti misalnya android Google Maps API yang bisa digunakan saat ini telah mencapai versi 2. Perkembangan pada Google Maps Versi 2 ini adalah dengan memiliki fitur–fitur yang lengkap dari versi sebelumnya salah satunya adalah dapat menampilkan peta dengan sudut yang di inginkan dan berupa 3D view. Selain itu pada Google Maps API terdapat beberapa pilihan model peta yang ditampilkan seperti Roadmap, Satellite, Terrain dan Hybrid.

2.18. Google Maps Directions API

Google Maps Directions API adalah layanan yang menghitung arah antar lokasi menggunakan permintaan HTTP.

Anda bisa mencari arah untuk beberapa moda transportasi, termasuk angkutan umum, mengemudi, berjalan atau bersepeda. Arah bisa menetapkan tempat asal, tujuan dan titik jalan baik sebagai string teks (misalnya "Chicago, IL" atau "Darwin, NT, Australia") atau sebagai koordinat garis lintang/garis bujur. Directions API bisa mengembalikan arah multi-bagian menggunakan serangkaian titik jalan.

Layanan ini biasanya didesain untuk menghitung arah alamat statis (sudah diketahui sebelumnya) untuk penempatan konten aplikasi pada peta; layanan ini tidak didesain untuk merespons input pengguna secara real-time, misalnya. Untuk perhitungan arah dinamis (misalnya, dalam elemen antarmuka pengguna), lihat dokumentasi untuk Google Maps JavaScript API Layanan Directions [10].

a. Permintaan Arah

Permintaan Google Maps Directions API mengambil bentuk berikut: `https://maps.googleapis.com/maps/api/directions/output?parameters`

dalam hal ini, output bisa berupa salah satu nilai berikut:

1. Json (disarankan) menunjukkan output dalam JavaScript Object Notation (JSON)
2. xml menunjukkan output berupa XML

Untuk mengakses Google Maps Directions API melalui HTTP, gunakan:

http://maps.googleapis.com/maps/api/directions/output?parameters

HTTP disarankan untuk aplikasi yang berisi data pengguna sensitif, seperti lokasi pengguna, dalam permintaan.

URL Google Maps Directions API dibatasi sekitar 2000 karakter, setelah Pengkodean URL. Karena sebagian URL Google Maps Directions API bisa melibatkan banyak lokasi sepanjang lintasan, berhati-hatilah dengan batas ini saat membangun URL Anda.

b. Parameter Permintaan

Beberapa parameter tertentu diperlukan sementara yang lainnya bersifat opsional. Sebagaimana standar dalam URL, semua parameter dipisah menggunakan karakter ampersand (&). Daftar parameter dan kemungkinan nilainya disebutkan di bawah ini.

c. Parameter yang diperlukan

1. origin — Alamat, nilai garis lintang/garis bujur tekstual, atau ID tempat asal yang ingin Anda hitung arahnya.
2. Jika Anda meneruskan sebuah alamat sebagai string, layanan Directions akan melakukan geocode atas string itu dan mengubahnya menjadi koordinat garis

lintang/garis bujur untuk menghitung arah. Koordinat ini mungkin berbeda dengan yang dikembalikan oleh Google Maps, Geocoding API, misalnya pintu masuk bangunan dan bukan pusatnya.

3. Jika Anda meneruskan koordinat, itu akan digunakan tanpa diubah untuk menghitung arah. Pastikan tidak ada spasi di antara nilai garis lintang dan garis bujur.
4. ID Tempat harus diawali dengan place_id:. ID tempat hanya bisa ditetapkan jika permintaan menyertakan kunci API atau ID klien Google Maps API for Work. Anda bisa mendapatkan ID tempat dari Google Maps Geocoding API dan Google Places API (termasuk Place Autocomplete). Untuk contoh menggunakan ID tempat dari Place Autocomplete, lihat Place Autocomplete dan Arah. Untuk informasi selengkapnya tentang ID tempat, lihat ikhtisar ID tempat.
5. destination — Alamat, nilai garis lintang/garis bujur tekstual, atau ID tempat tujuan yang ingin Anda hitung arahnya. Opsi untuk parameter destination sama dengan opsi untuk parameter origin yang dijelaskan di atas.
6. key — Kunci API aplikasi Anda. Kunci ini mengidentifikasi aplikasi Anda untuk keperluan manajemen kuota. Pelajari cara mendapatkan kunci.

d. Mode Perjalanan

Saat menghitung arah, Anda bisa menetapkan mode transportasi yang akan digunakan. Secara default, arah dihitung sebagai arah driving. Mode perjalanan berikut ini telah didukung:

1. Driving (default) menunjukkan arah mengemudi standar yang menggunakan jaringan jalan raya.
2. Walking meminta arah berjalan melalui jalur pejalan kaki & trotoar (bila tersedia).
3. Bicycling meminta arah bersepeda melalui lintasan sepeda & jalan yang disukai (bila tersedia).
4. Transit meminta arah melalui rute angkutan umum (bila tersedia). Jika mengatur mode ke transit, Anda juga bisa menetapkan departure_time atau

arrival_time. Jika tidak ada waktu yang ditetapkan, default departure_time adalah sekarang (yaitu, waktu keberangkatan default adalah waktu saat ini). Anda juga bisa menyertakan transit_mode dan/atau transit_routing_preference.

e. Titik jalan

Saat menghitung rute menggunakan Google Maps Directions API, Anda juga bisa menetapkan titik jalan untuk arah mengemudi, berjalan atau bersepeda (titik jalan tidak tersedia untuk arah angkutan umum). Titik jalan memungkinkan Anda menghitung rute melalui lokasi tambahan, di mana rute yang dikembalikan menyertakan persinggahan di setiap titik jalan yang diberikan.

Titik jalan ditetapkan dalam parameter waypoints dan terdiri dari satu atau beberapa alamat atau lokasi yang dipisah dengan karakter pipa (|).

Misalnya, URL berikut membuat permintaan Directions untuk rute antara Boston, MA dan Concord, MA dengan persinggahan di Charlestown dan Lexington, dalam urutan itu:

```
https://maps.googleapis.com/maps/api/directions/json?origin=Boston,MA&destination=Concord,MA&waypoints=Charlestown,MA|Lexington,MA&key=YOUR_API_KEY
```

Untuk setiap titik jalan dalam permintaan, respons arah menyertakan entri tambahan dalam larik legs guna memberikan detail yang sesuai untuk segmen perjalanan itu.

Jika ingin memengaruhi rute menggunakan titik jalan tanpa menambahkan persinggahan, tambahkan awalanvia: pada titik jalan tersebut. Titik jalan yang diberi awalan via: tidak akan menambahkan entri ke larik legs, namun akan mengarahkan perjalanan melalui titik jalan yang diberikan. URL berikut memodifikasi permintaan sebelumnya yang perjalanannya diarahkan melewati Lexington tanpa berhenti:

```
https://maps.googleapis.com/maps/api/directions/json?origin=Boston,MA&destination=Concord,MA&waypoints=Charlestown,MA|via:Lexington,MA&key=YOUR_API_KEY
```

Awalan via: sangat efektif saat membuat rute sebagai respons terhadap pengguna yang menyeret titik jalan pada peta. Hal itu memungkinkan pengguna melihat bagaimana tampilan rute akhir secara real-time dan membantu memastikan titik-titik jalan tersebut ditempatkan di lokasi yang bisa diakses oleh Google Maps Directions API.

f. Optimalkan titik jalan Anda

Secara default, layanan Directions menghitung rute melalui titik jalan yang diberikan dalam urutan tertentu. Atau, Anda bisa meneruskan optimize:true sebagai argumen pertama dalam parameter waypoints untuk memungkinkan layanan Directions mengoptimalkan rute yang disediakan dengan menata ulang titik jalan dalam urutan yang lebih efisien. (Optimalisasi ini merupakan aplikasi dari Travelling Salesman Problem.)

Jika Anda memerintahkan layanan Directions untuk mengoptimalkan urutan titik jalan, maka urutannya akan dikembalikan dalam bidang waypoint_order di dalam objek routes. Bidang waypoint_order mengembalikan nilai yang berbasis nol.

Contoh berikut menghitung rute perjalanan darat dari Adelaide, Australia Selatan ke setiap wilayah penghasil wine utama Australia Selatan dengan menggunakan optimalisasi rute.

```
https://maps.googleapis.com/maps/api/directions/json?origin=
Adelaide,SA&destination=Adelaide,SA&waypoints=optimize:true|
Barossa+Valley,SA|Clare,SA|Connawarra,SA|McLaren+Vale,SA&key
=YOUR_API_KEY
```

Pemeriksaan rute yang dihitung akan menunjukkan rute tersebut dihitung menggunakan urutan titik jalan berikut:

```
"waypoint_order": [ 1, 0, 2, 3 ]
```

g. Pembatasan

Arah bisa dihitung dengan mengikuti pembatasan tertentu. Pembatasan ditunjukkan melalui penggunaan parameter avoid, dan sebuah argumen ke

parameter itu yang menunjukkan pembatasan untuk avoid. Pembatasan berikut ini telah didukung:

1. avoid=tolls
2. avoid=highways
3. avoid=ferries

Bisa saja meminta rute yang menghindari kombinasi tol, jalan raya dan feri dengan meneruskan kedua pembatasan itu ke parameter avoid. Misalnya: avoid=tolls|highways|ferries.

h. Sistem Satuan

Hasil arah berisi text dalam bidang distance yang bisa ditampilkan kepada pengguna untuk menunjukkan jarak "langkah" tertentu pada rute tersebut. Secara default, teks ini menggunakan sistem satuan pada negara atau wilayah asal.

1. Misalnya, rute dari "Chicago, IL" ke "Toronto, ONT" akan menampilkan hasil dalam mil, sedangkan rute terbalik akan menampilkan hasil dalam kilometer. Anda bisa mengesampingkan sistem satuan ini dengan menetapkannya secara eksplisit dalam parameter units permintaan, dengan meneruskan salah satu dari nilai berikut: metric menetapkan penggunaan sistem metrik. Jarak tekstual dikembalikan menggunakan kilometer dan meter.
2. Imperial menetapkan penggunaan sistem Imperial (Inggris). Jarak tekstual dikembalikan menggunakan mil dan kaki.

i. Mencondongkan Wilayah

Anda juga bisa mengatur layanan Directions untuk mengembalikan hasil yang dicondongkan ke wilayah tertentu dengan menggunakan parameter region. Parameter ini mengambil argumen ccTLD (top-level domain kode negara) yang menetapkan kecondongan wilayah. Umumnya kode ccTLD identik dengan kode ISO 3166-1, dengan beberapa pengecualian. Misalnya, ccTLD Kerajaan Inggris adalah "uk" (.co.uk) sementara kode ISO 3166-1 untuknya adalah "gb" (secara teknis untuk entitas "Kerajaan Bersatu Britania Raya dan Irlandia Utara").

Anda bisa memanfaatkan domain apa saja tempat aplikasi Google Maps utama menjalankan arah mengemudi.

Misalnya, permintaan arah dari "Toledo" ke "Madrid" akan mengembalikan hasil bila region diatur ke es karena "Toledo" ditafsirkan sebagai kota Spanyol:

```
https://maps.googleapis.com/maps/api/directions/json?origin=To
ledo&destination=Madrid&region=es&key=YOUR_API_KEY
```

```
{
  "status": "OK",
  "routes": [ {
    "summary": "AP-41",
    "legs": [ {

...

    } ],
    "copyrights": "Map data ©2010 Europa Technologies, Tele
Atlas",
    "warnings": [ ],
    "waypoint_order": [ ]
  } ]
}
```

Arah untuk "Toledo" ke "Madrid" yang dikirim tanpa parameter region tidak akan mengembalikan hasil, karena "Toledo" ditafsirkan sebagai kota di Ohio:

```
https://maps.googleapis.com/maps/api/directions/json?origin=To
ledo&destination=Madrid&key=YOUR_API_KEY
```

```
{
  "status": "ZERO_RESULTS",

  "routes": [ ]

}
```

j. Respons Arah

Respons arah dikembalikan dalam format yang ditunjukkan oleh flag output dalam path permintaan URL.

k. Respons Contoh

Contoh permintaan HTTP ditampilkan di bawah ini, yang menghitung rute dari Chicago, IL ke Los Angeles, CA melewati dua titik jalan di Joplin, MO dan Oklahoma City, OK.

https://maps.googleapis.com/maps/api/directions/json?origin=Chicago,IL&destination=Los+Angeles,CA&waypoints=Joplin,MO|Oklahoma+City,OK&key=YOUR_API_KEY

Contoh di atas meminta output JSON. Bisa juga meminta output XML. Klik tab di bawah ini untuk melihat contoh respons JSON dan XML.

Karena hasil arah bisa sangat detail, maka elemen berulang dalam respons telah dihilangkan agar lebih jelas.

JSON


```
{
  "status": "OK",
  "geocoded_waypoints" : [
     {
        "geocoder_status" : "OK",
              "place_id" : "ChIJ7cv00DwsDogRAMDACa2m4K8",
        "types" : [ "locality", "political" ]
     },
     {
        "geocoder_status" : "OK",
        "place_id" : "ChIJ69Pk6jdlyIcRDqM1KDY3Fpg",
        "types" : [ "locality", "political" ]
     },
     {
        "geocoder_status" : "OK",
        "place_id" : "ChIJgdL4flSKrYcRnTpP0XQSojM",
        "types" : [ "locality", "political" ]
     },
     {
```

```
            "geocoder_status" : "OK",
            "place_id" : "ChIJE9on3F3HwoAR9AhGJW_fL-I",
            "types" : [ "locality", "political" ]
         }
      ],
      "routes": [ {
        "summary": "I-40 W",
        "legs": [ {
          "steps": [ {
            "travel_mode": "DRIVING",
            "start_location": {
              "lat": 41.8507300,           "lng": -87.6512600
            },
            "end_location": {
              "lat": 41.8525800,           "lng": -87.6514100
            },
            "polyline": {
              "points": "a~l~Fjk~uOwHJy@P"
            },
            "duration": {
                    "value": 19,
              "text": "1 min"
            },
            "html_instructions": "Head
    \u003cb\u003enorth\u003c/b\u003e on \u003cb\u003eS Morgan
    St\u003c/b\u003e toward \u003cb\u003eW Cermak
    Rd\u003c/b\u003e",
            "distance": {
              "value": 207,
              "text": "0.1 mi"
            }
          },
          ...
          ... additional steps of this leg     ...
        ... additional legs of this route
          "duration": {
            "value": 74384,
```

```
               "text": "20 hours 40 mins"
            },
            "distance": {
              "value": 2137146,
              "text": "1,328 mi"
            },
            "start_location": {
              "lat": 35.4675602,         "lng": -97.5164276
            },
            "end_location": {
              "lat": 34.0522342,
              "lng": -118.2436849
            },
            "start_address": "Oklahoma City, OK, USA",
            "end_address": "Los Angeles, CA, USA"
          } ],
          "copyrights": "Map data ©2010 Google, Sanborn",
                "overview_polyline": {       "points":
      "a~l~Fjk~uOnzh@vlbBtc~@tsE`vnApw{A`dw@~w\\|tNtqf@l{Yd_Fblh
@rxo                                                          @b}
@xxSfytAblk@xxaBeJxlcBb~t@zbh@jc|Bx}C`rv@rw|@rlhA~dVzeo@vrS
nc}Axf]fjz@xfFbw~@dz{A~d{A|zOxbrBbdUvpo@`cFp~xBc`Hk@nurDznmFfw
Mbwz@bbl@lq~@loPpxq@bw_@v|{CbtY~jGqeMb{iF|n\\~mbDzeVh_Wr|Efc\\
x`Ij{kE}mAb~uF{cNd}xBjp]fulBiwJpgg@|kHntyArpb@bijCk_Kv~eGyqTj_
|@`uV`k|DcsNdwxAott@r}q@_gc@nu`CnvHx`k@dse@j|p@zpiAp|gEicy@`om
FvaErfo@igQxnlApqGze~AsyRzrjAb__@ftyB}pIlo_BflmA~yQftNboWzoAlz
p@mz`@|}_@fda@jakEitAn{fB_a]lexClshBtmqAdmY_hLxiZd~XtaBndgC"
          },
          "warnings": [ ],
          "waypoint_order": [ 0, 1 ],
          "bounds": {
            "southwest": {
              "lat": 34.0523600,
              "lng": -118.2435600
            },
            "northeast": {
              "lat": 41.8781100,         "lng": -87.6297900
```

```
            }
          }
        } ]
      }
```

Biasanya, hanya dihasilkan satu entri dalam larik routes untuk pencarian arah, meskipun layanan Directions bisa mengembalikan beberapa rute jika Anda meneruskan alternatives=true.

2.19. Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam p-code (*bytecode*) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM). Java merupakan Bahasa pemrograman yang bersifat umum/non-spesifik (*general purpose*), dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin [11].

Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa platform sistem operasi yang berbeda, java dikenal pula dengan slogannya, "*Tulis sekali, jalankan di mana pun*". Saat ini java Merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis web.

2.20. OOAD (Object Oriented Analysis and Design)

Konsep OOAD mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaitu analisis berorientasi objek (OOA) dan desain berorientasi objek (OOD). OOA adalah metode analisis yang memeriksa requirement

(syarat/keperluan) yang harus dipenuhi sebuah sistem dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup sistem. Sedangkan OOD adalah metode untuk mengarahkan arsitektur *software* yang didasarkan pada manipulasi objek-objek sistem atau subsistem [14].

2.21. OOP (Object Oriented Programming)

Secara spesifik, pengertian berorientasi objek berarti bahwa mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya. Hal ini yang membedakan dengan pemograman konvensional dimana struktur data dan perilaku hanya berhubungan secara terpisah. Terdapat beberapa cara untuk menentukan karakteristik dalam pendekatan berorientasi objek, tetapi secara umum mencakup empat hal, yaitu identifikasi, klasifikasi, *polymorphism* (polimorfisme) dan *inheritance* (pewarisan) [15].

2.21.1 Konsep Dasar Berorientasi Object

Pengembangan sistem berorientasi objek berbeda dari pengembangan konvensional yang memandang perangkat lunak sebagai fungsi dan data yang terisolasi. Program perangkat lunak adalah sekumpulan mekanisme yang melakukan aksi-aksi tertentu pada data tertentu. Pada pendekatan konvensional, kebanyaan berfokus pada fungsi. Namun juga terdapat pendekatan yang fokus pada data terutama pada kubu basis data dan pemodelan informasi. Sementara itu, pandangan berorientasi objek berpusat pada objek yang mengkombinasikan data dan fungsionalitas. Berikut ini adalah beberapa kriteria-kriteria penting pendekatan objek yang harus dipahami tentang metodologi berorientasi objek [15]:

a. *Seamlessness*

Konsep berorientasi objek dapat diterapkan di seluruh siklus hidup pengembang sistem (*System Developmnent life cycle*) dari analisis, perancangan, sampai implementasi.

b. Kelas

Pendekatan berorientasi objek berdasar pada pentingnya kelas. Kelas adalah elemen perangkat lunak yang mendeskripsikan tipe data abstrak (ADT – *Abstract*

data type) dimana implementasinya dapat secara parsial atau total. Implementasi parsial bila di kelas terdapat operas/metode abstrak.

c. Assertion

Fitur tipe data abstrak harus secara formal menspesifikasikan properti kelas Assertion berperan pada prakondisi (*precondition*), poskondisi (*postcondition*), invarian kelas. *Assertion* mendeskripsikan efek fitur-fitur di objek independen terhadap implementasi fitur-fitur itu.

d. Kelas Merupakan Cara Pemodulan

Pendekatan berorientasi objek merupakan teknik pembentukan arsitektur perangkat lunak. Dampak utama penerapannya adalah pada struktur modul dari sistem perangkat lunak. Peran utamanya di sistem berorientasi objek dimainkan oleh kelas. Pada pendekatan murni berorientasi objek maka kelas merupakan satu-satunya pemodulan.

e. Kelas merupakan *type*

Kelas cukup ampuh untuk menghilangkan kebutuhan mekanisme typing yang lain. Setiap type harus didasarkan pada kelas.

f. *Exception handling*

Sistem perangkat lunak seharusnya mampu menangani kondisi abnormal secara terstruktur dengan baik.

g. *Genericity*

Agar penerapan typing menjadi praktis adalah kemampuan untuk mendefinisikan *type-parameterized* classes atau disebut generik.

h. Pewarisan

Pendekatan berorientasi objek harus memungkinkan mendefinisikan kelas yang merupakan turunan kelas yang lain. Pewarisan dapat berupa pewarisan tunggal (*single inheritance*) dan pewarisan jamak (*multiple inheritance*).

i. *Redefinition*

Kita mungkin perlu mengubah implementasi atau properti lain dari fitur-fitur yang diwariskan. Redefinisi dapat mempengaruhi fitur, penanda (tipe argumen dan hasil) dan spesifikasi.

j. *Polymorphism*

Dimungkinkan untuk entitas ditautkan ke objek-objek beragam tipe. *Polymorphism* dikendalikan oleh pewarisan.

k. *Dynamic binding*

Dynamic binding mempunyai pengaruh utama pada aplikasi berorientasi objek. Mekanisme ini memungkinkan pengembang menulis panggilan secara sederhana untuk beberapa panggilan yang dapat dimungkinkan, bergantung pada situasi saat sebenarnnya panggilan itu dieksekusi.

l. *Run-time type interogation*

Perangkat lunak memerlukan mekanisme untuk mengakses objek dengan cara yang aman tanpa melanggar konstrain *static typing*. Mekanisme ini harus dirancang agar tidak meniadakan manfaat-manfaat dari *dynamic binding* dan *polymorphism*.

2.22. UML (Unified Modelling Language)

Unified Modelling Language (UML) adalah bahasa grafis untuk mendokumentasi, menspesifikasikan, dan membangun sistem perangkat lunak. UML berorientasi objek, menerapkan banyak level abstraksi, tidak bergantung proses pengembangan, tidak bergantung bahasa dan teknologi, pemaduan beberapa notasi di beragam metodologi, usaha bersama dari banyak pihak, didukung oleh kakas-kakas yang diintegrasikan lewat XML (XMI). Standar UML dikelola oleh OMG (*Object Management Group*) [15].

UML adalah bahasa pemodelan untuk menspesifikasikan, memvisualisasikan, membangun dan mendokumentasikan artifak-artifak dari sistem. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya yaitu : *Grady Booch* OOD (*Object Oriented Design*), James Rumbaugh OMT (*Object Modeling Technique*) dan Ivar Jacobson OOSE (*Object Oriented Software Engineering*).

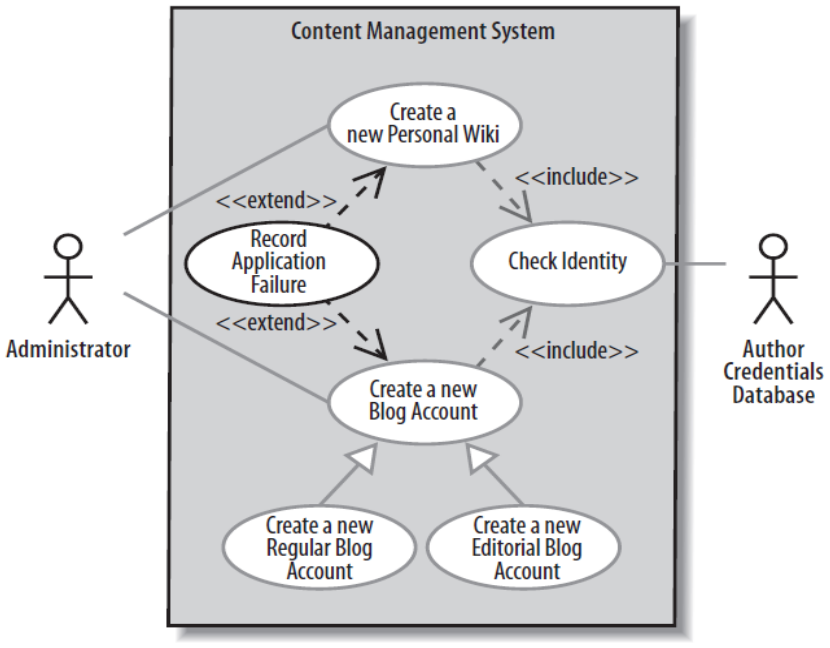
2.22.1. *Use Case* Diagram

Use case Diagram merupakan salah satu diagram untuk memodelkan aspek perilakau sistem. Masing-masing diagram *use case* menunjukkan sekumpulan *use case*, aktor, dan hubungannya. Diagram *use case* adalah penting untuk

memvisualisasikan, menspesifikasikan, dan mendokumentasikan kebutuhan perilaku sistem. Diagram-diagram *use case* merupakan pusat pemodelan perilaku sistem, subsistem, dan kelas. Diagram *use case* digunakan untuk mendeskripsikan apa yang seharusnya dilakukan oleh sistem. Ada empat hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan use case [15]:

1. Sistem yaitu sesuatu yang hendak kita bangun.
2. Relasi adalah relasi antara aktor dengan *use case*.
3. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
4. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

   Berikut Contoh *Use Case* Diagram dapat dilihat pada Gambar 2.11.



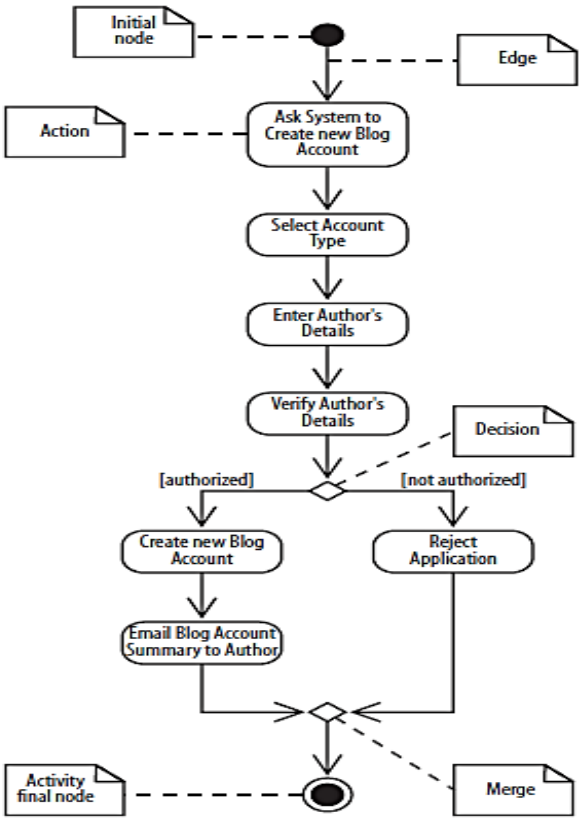
Gambar 2.11 Contoh Use Case Diagram

2.22.2. *Activity* Diagram

Diagram aktivitas adalah diagram *flowchart* yang diperluas yang menunjukan aliran kendali satu aktivitas ke aktivitas lain di sistem. Diagram aktivitas ini digunakan untuk memodelkan aspek dinamis sistem. Diagram aktivitas mendeskripsikan aksi-aksi dan hasilnya. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut [15]:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokkan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak.

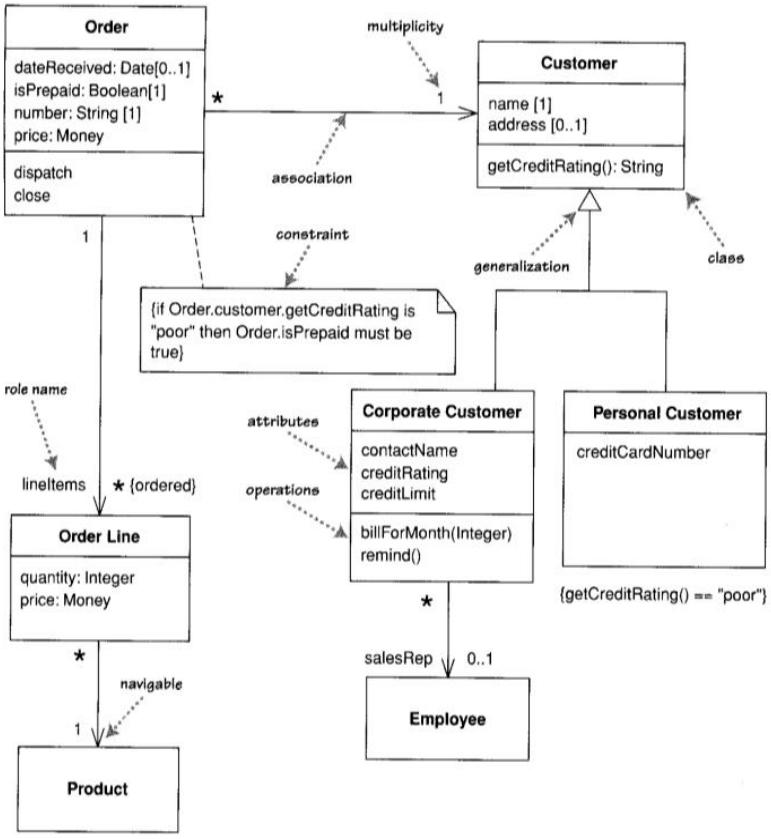
Berikut Contoh *Activity* Diagram dapat dilihat pada Gambar 2.12.



Gambar 2.12 Contoh Activity Diagram

2.22.3. *Class* Diagram

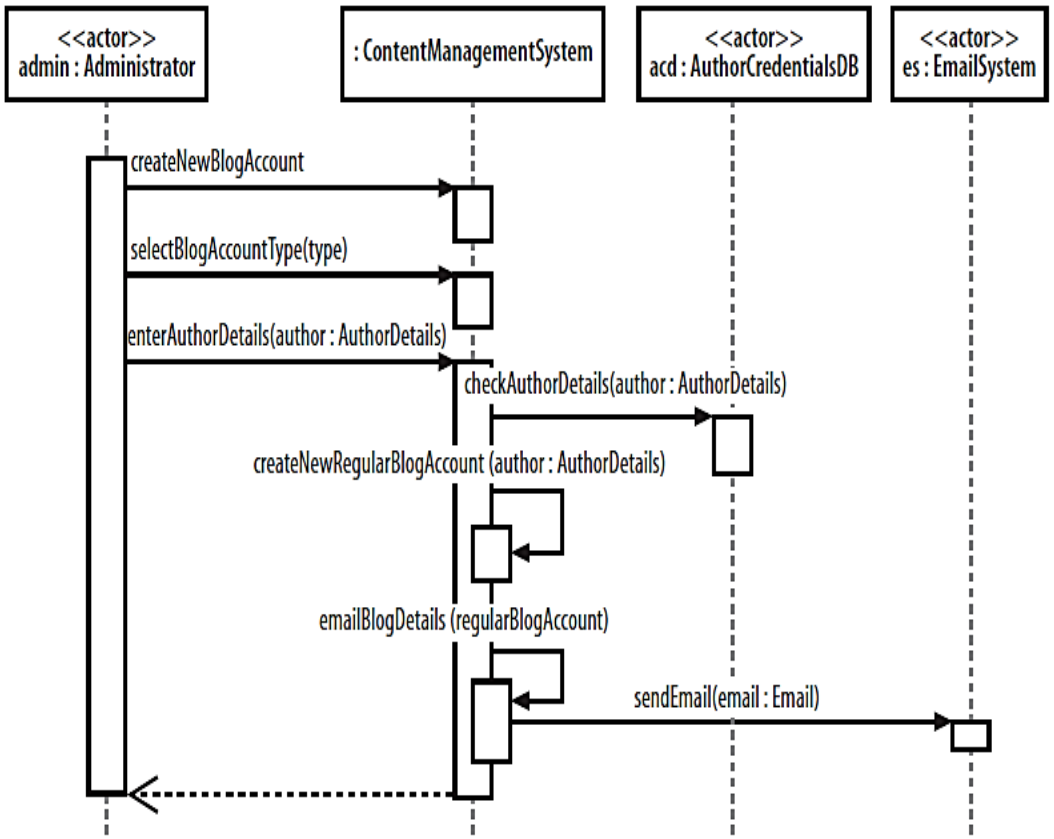
Class diagram merupakan diagram yang selalu ada di pemodelan system berorientasi objek. *Class* diagram menunjukkan hubungan antar class dalam system yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai satu tujuan. Kelas pada kelas diagram terdiri dari 3 bagian utama yaitu nama kelas, isi *property* dari kelas beserta metode yang ada pada kelas tersebut. Kelas juga memiliki jenis-jenis hubungan seperti asosiatif, dependensi, agregasi, komposisi, spesifikasi dan generalisasi. Hubungan ini digunakan untuk menggambarkan bagaimana hubungan dan interaksi yang terjadi antar kelas. Masing-masing komponen penyusun kelas memiliki hak akses seperti *public, private* dan *protected* [15]. Berikut contoh *Class* Diagram dapat dilihat pada Gambar 2.13.



Gambar 2.13 Contoh Class Diagram

2.22.4. *Sequence* Diagram

Sequence Diagram menunjukan interaksi yang terjadi antar objek. Diagram ini merupakan pandangan dinamis terhadap sistem. Diagram ini menekankan pada sisi basis keberurutan waktu dari pesan-pesan yang terjadi. Diagram sekuen menunjukan objek sebagai garis vertikal dan tiap kejadian sebagai panah horisontal dari objek pengirim ke objek penerima. Waktu berlalu dari atas ke bawah dengan lama waktu tidak relevan. Diagram ini hanya menunjukan barisan kejadian, bukan perwaktuan nyata. Kecuali untuk sistem waktu nyata yang mengharuskan konstrain barisan terjadi [15]. Berikut Contoh sequence diagram dapat dilihat pada gambar 2.14.



Gambar 2.14 Contoh Sequence Diagram

2.23. Pengertian Metode TSP

Traveling Salesman Problem (TSP) adalah suatu permasalahan dimana seorang sales harus melalui semua kota yang ditunjuk dengan jarak yang paling pendek dan setiap kota hanya boleh dilalui satu kali. Penyelesaian dalam TSP adalah jalur yang dilalui oleh salesman sesuai dengan batasan diatas. Penyelesaian terbaik adalah jalur dengan jarak terpendek.

Berikut adalah aturan-aturan yang mengidentifikasikan bahwa permasalahan tersebut adalah TSP:

1. Perjalanan dimulai dan diakhiri di kota yang sama sebagai kota asal sales. Seluruh kota harus dikunjungi tanpa satupun kota yang terlewatkan.
2. Salesman tidak boleh kembali ke kota asal sebelum seluruh kota terkunjungi. Tujuan penyelesaian permasalahan ini adalah mencari nilai optimum dengan meminimumkan jarak total rute yang dikunjungi dengan mengatur urutan kota.
3. Permasalahan TSP merupakan permasalahan yang memang mudah untuk diselesaikan dengan algoritma Brute Force, tetapi hal itu hanya dapat dilakukan dengan jumlah kota atau simpul yang tidak banyak. Kompleksitas algoritma untuk permasalahan TSP dengan algoritma Brute Force adalah O(n!) dengan catatan n adalah jumlah kota atau simpul dan setiap kota atau simpul terhubung dengan semua kota atau simpul lainnya. Dengan jumlah sebanyak 20 kota, maka banyak sirkuit Hamilton yang mungkin adalah sebanyak 6 x 1016.