BAB 2
TINJAUAN PUSTAKA

2.1 Tracking

Tracking secara harifah memiliki arti mengikuti jalan, atau dalam arti bebasnya adalah suatu kegiatan untuk mengikuti jejak suatu obyek. Pengertian *tracking* atau pemantauan dalam hal ini adalah kegiatan untuk memantau keberadaan anak berdasarkan posisi yang di dapatkan dari *smartphone* [6].

2.2 Aplikasi

Secara istilah pengertian aplikasi adalah suatu program yang siap untuk digunakan yang dibuat untuk melaksanankan suatu fungsi bagi pengguna jasa aplikasi serta penggunaan aplikasi lain yang dapat digunakan oleh suatu sasaran yang akan dituju. Menurut kamus computer eksekutif, aplikasi mempunyai arti yaitu pemecahan masalah yang menggunakan salah satu tehnik pemrosesan data aplikasi yang biasanya berpacu pada sebuah komputansi yang diinginkan atau diharapkan maupun pemrosesan data yang di harapkan.

Pengertian aplikasi menurut Kamus Besar Bahasa Indonesia, "Aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu" [7].

2.3 Assisted – Global Positioning System (A-GPS)

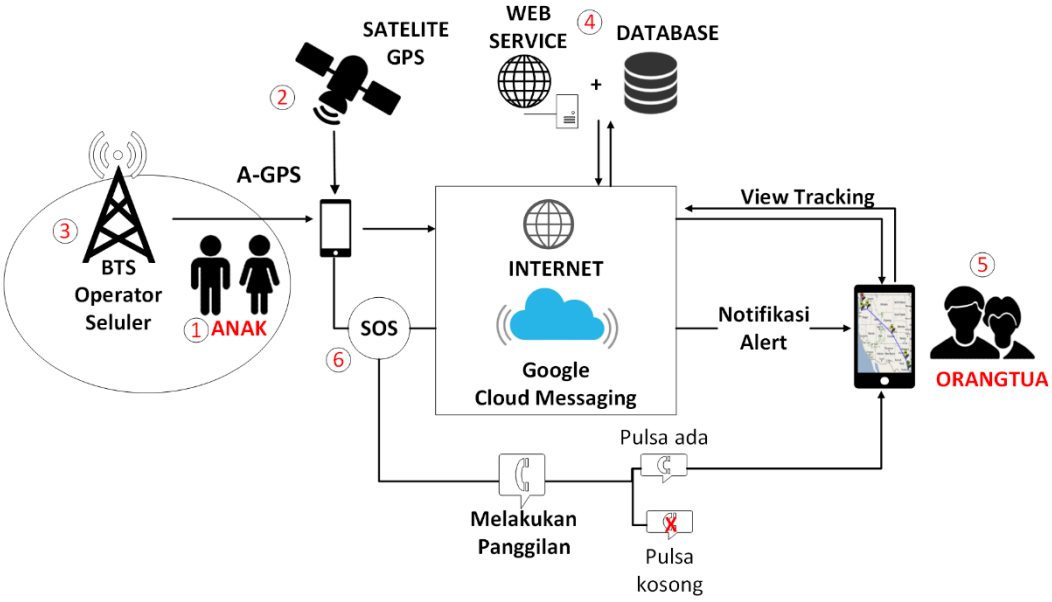
Assisted Global Positioning System (A-GPS) merupakan pengembangan dari sistem GPS biasa sebagai penentu posisi. A-GPS memperbaiki performansi GPS biasa dengan menyediakan informasi lewat kanal komunikasi alternative yang terhubung ke suatu *server* pembantu, dimana A-GPS *receiver* akan mendapatkan beberapa informasi yang biasanya diterima dari satelit melalui *server* tersebut. Inilah yang membedakan antara A-GPS dengan GPS, yaitu penambahan elemen *assistance server* atau juga disebut LBS (*location Base Service*). Dengan demikian, membantu proses penerimaan data untuk menentukan posisi *user* menjadi benar-

benar lebih mudah dan dapat mengurangi waktu serta jumlah informasi yang dibutuhkan dari satelit. Lebih lanjut, karena A-GPS *receiver* didesain untuk bisa mengetahui terlebih dalam penentuan posisinya, maka A-GPS *receiver* dapat meningkatkan jumlah energy yang diterima dari frekuensi-frekuensi tertentu saja. Hal ini meningkatkan sensitivitas dari A-GPS *receiver* dan memungkinkan untuk melakukan penentuan posisi *user* walaupun memperoleh sinyal GPS pada kekuatan sinyal yang lebih rendah [2].



Gambar 2.1 Contoh A-GPS (*assisted Global Positioning System*)

Berikut adalah gambaran Sistem *Tracker* pada aplikasi *Child Tracker*.



Gambar 2.2 Alur *Tracker* aplikasi Child Tracker

2.3.1 Global Positioning System (GPS)

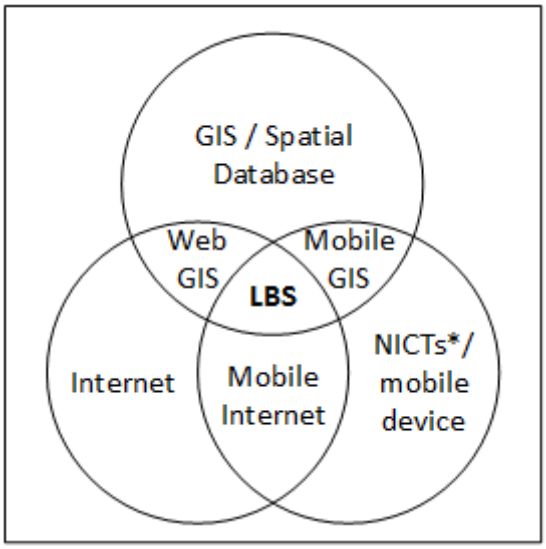
GPS adalah singkatan dari *Global Positioning System*, yang merupakan sistem navigasi dengan menggunakan teknologi satelit yang dapat menerima sinyal dari satelit. Sistem ini menggunakan 24 satelit yang mengirimkan sinyal gelombang mikro ke bumi. Sinyal ini diterima oleh alat penerima (*receiver*) di permukaan, dimana GPS *receiver* ini akan mengumpulkan informasi dari satelit GPS, seperti [8] :

1. Waktu. GPS *receiver* menerima informasi waktu dari jam atom yang mempunyai keakurasian sangat tinggi.
2. Lokasi. GPS memberikan informasi dalam tiga dimensi :
   a. Latitude
   b. Longitude
   c. Elevasi
3. Kecepatan ketika berpindah tempat, GPS dapat menunjukan informasi kecepatan berpindah tersebut.
4. Arah perjalanan. GPS dapat menunjukan arah tujuan.
5. Simpan lokasi. Tempat-tempat yang sudah pernah atau ingin dikunjungi bisa disimpan oleh GPS *receiver.*
6. Komulasi data. GPS *receiver* dapat menyimpan informasi *track*, seperti total perjalanan yang sudah pernah dilakukan, kecepatan rata-rata, kecepatan paling tinggi, kecepatan paling rendah, waktu/jam sampai tujuan dan sebagainya.

2.3.2 Location Base Service (LBS)

Layanan berbasis lokasi adalah layanana informasi yang dapat diakses melalui *mobile device* dengan menggunakan *mobile network*, yang dilengkapi kemampuan untuk memanfaatkan lokasi dari *mobile device* tersebut. LBS memberikan kemungkinan komunikasi dan interaksi dua arah. Oleh karena itu pengguna memberitahu penyedia layanan untuk mendapatkan informasi yang dia butuhkan, dengan referensi posisi pengguna tersebut. Layanan berbasis lokasi dapat digambarkan sebagai suatu layanan yang berada pada pertemua tiga teknologi yaitu

: Geographic Information System, Internet Service, dan Mobile Devices, hal ini dapat dilihat pada gambar LBS adalah pertemuan dari tiga teknologi [9].



Gambar 2.3 LBS sebagai tiga teknologi

Secara garis besar jenis layanan berbasis lokasi juga dapat dibagi menjadi dua, yaitu :

1. *Pull Service* : layanan diberikan berdasarkan permintaan dari pelanggan akan kebutuhan suatu informasi. Jenis layanan ini dapat dianalogikan seperti mengakses suatu web pada jaringan internet.
2. *Push Service* : Layanan ini diberikan langsung oleh *Service Provider* tanpa menunggu permintaan dari pelanggan. Tentu saja informasi yang diberikan tetap berkaitan dengan kebutuhan pelanggan.

2.4 Pengenalan Sistem Operasi Android

Android merupakan sistem operasi yang sekarang sedang terkenal di pasaran *smartphone* saat ini. Berikut adalah pengenalan android :

2.4.1 Sejarah Android

Android adalah sebuah sistem operasi perangkat *mobile* berbasis linux yang mencangkup sistem operasi, *middleware* dan aplikasi. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka.

Awalnya, Google Inc. membeli Android Inc. yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel atau *smartphone*. Kemudian untuk mengembangkan Android, dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan peranti keras, peranti lunak dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia. [10]

Pada saat perilisan perdana Android, 5 November 2007, Android bersama *Open Handset Alliance* menyatakan mendukung pengembangan open source pada perangkat mobile. Di lain pihak, Google merilis kode-kode Android di bawah lisensi Apache, sebuah lisensi perangkat lunak dan open platform perangkat seluler.

2.4.2 Versi android

Adapun versi android yang pernah dirilis adalah sebagai berikut [10]:

1. Android versi 1.1

Pada 9 maret 2009, Google merilis android versi 1.1. Android versi ini dilengkapi dengan pembaruan estetis pada aplikasi, ja, alarm, *voice search* (pencarian suara), pengiriman pesan dengan Gmail, dan pemberitahuan email.

2. Android versi 1.5 (*Cupcake*)

Pada pertengahan mei 2009, Google kembali merilis telepon seluler dengan menggunakan Android dan SDK (*Software Development Kit*). Terdapat banyak pembaharuan yakni mampu merekan dan menonton video denga n modus kamera, meguplоad video ke youtube dan gambar ke picasa langsung dari telepon, dukungan Bluetooth A2DP, kemampuan terhubung otomatis ke headset Bluetooth.

3. Android versi 1.6 (*Donut*)

Dirilis pada September dengan menampilkan proses pencarian yang lebih baik disbanding sebelumnya, penggunaan baterai indicator dan control applet VPN. Fitur lainnya adalah kamera, camcorder, VPN, Gestures, CDMA/EVDO, Text-to-Speech engine, resolusi VWGA.

4. Android versi 2.0/2.1 (*Eclair*)

Pada 3 Desember 2009 kembali diluncurkan ponsel android dengan mengoptimalkan hardware, peningkatan Google Maps 3.1.2, perubahan UI dengan

browser baru dan dukungan HTML5, daftar kontak yang baru, dukungan kamera flash, dan Bluetooth 2.1.

5. Android versi 2.2 (Froyo : Frozen Yoghurt)

Pada bulan mei 2010 Android versi 2.2 diluncurkan. Android yg lebih complex dengan penambahan fitur fitur terbaru seperti *Global Positioning System* (GPS), kompas, kamera, dan *Accelerometer*.

6. Android versi 2.3 (Gingerbread)

Diluncurkan Desember 2010. Hal-hal yang direvisi dari versi sebelumnya adalah kemampuan seperti berikut :

1. *SIP-based VoIP*
2. *Near Field Communication* (*NCF)*
3. *Gyroscope dan sensor*
4. *Multiple cameras support*
5. *Mixable audio effects*
6. *Download manager*

7. Android versi 3.0 (*Honeycomb*)

Dirilis februari 2011 sebagai android 3.0 yang terdapat fitur baru setelah versi sebelumnya.

8. Android versi 4.0 (*Ice Cream Sandwich*)

Dirilis pada 19 Oktober 2011, petinggi Goolge, Gabe Cohen menyatakan bahwa android 4.0 secara teoritis kompatibel dengan perangkat android 2.3x yang diproduksi pada saat itu.

9. Android versi 4.1 (*Jelly Bean*)

Google mengumumkan Android 4.1 dalam konferensi Google I/O pada tanggal 27 juni 2012, jelly bean adalah pembaharuan penting yang bertujuan meningkatkan fungsi dan kinerja antarmuka pengguna (UI).

10. Android versi 4.4 (*Kitkat*)

Google mengumumkan Android 4.4 Kitkat dinamai dengan izin dari Nestle dan Herskey pada 3 september 2013 dengan penambahan fitur UI yang lebih menarik.

11. Android versi 5.0 (*Lollipop*)

Versi terbaru saat ini dengan tampilan UI yang lebih menarik, dengan meningkatkan daya hidup baterai 30% lebih tahan lama.

2.5 Perangkat Lunak Pendukung

Berikut adalah perangkat lunak pendukung dalam penunjang pembangunan aplikasi yang akan di bangun :

2.5.1 Android Studio

Android studio adalah IDE (*Integrated Development Environment*) resmi untuk pengembangan aplikasi Android dan bersifat *open source* atau gratis. Peluncuran Android Studio ini diumumkan oleh Google pada 16 mei 2013 pada event Google I/O Conference untuk tahun 2013. Sejak saat itu, Android Studio mengantikan Eclipse sebagai IDE resmi untuk mengembangkan aplikasi Android [11].



Gambar 2.4 Android Studio

Android studio sendiri dikembangkan berdasarkan IntelliJ IDEA yang mirip dengan Eclipse disertai dengan ADT plugin (Android Development Tools). Android studio memiliki fitur :

a. Projek berbasis pada Gradle Build
b. Refactory dan pembenahan bug yang cepat
c. Tools baru yang bernama “Lint” dikalim dapat memonitor kecepatan, kegunaan, serta kompetibelitas aplikasi dengan cepat.

d. Mendukung Proguard And App-signing untuk keamanan.
e. Memiliki GUI aplikasi android lebih mudah
f. Didukung oleh Google Cloud Platfrom untuk setiap aplikasi yang dikembangkan.

2.5.2 PHP

PHP (*Hypertext Preprocessor*) yang digunakan sebagai bahasa *siscript server-side* dalam pengembangan *web* yang disipkan pada dokumen HTML. Penggunaan PHP memungkinkan *web* dapat dibuat dinamis sehingga *maintenance* situs web tersebut menjadi lebih mudah dan efisien [12].

Contoh script PHP :

```
<html>
<head>
<title> Contoh Halaman PHP </title>
</head>
<body>
<p>
<?php
print ("Selamat datang di halaman depan!");
?>
</body>
</html>
```

Tag php yang di proses untuk di tampilkan :

```
<?php
print ("Selamat datang di halamn depan!");
?>
```

Maka yang akan muncul pada halaman web ialah :

```
“selamat datang di halaman depan!”
```

2.5.3 MySQL

MySQL merupakan database yang paling digemari dikalangan Programmer Web, dengan alasan bahwa program ini merupakan database yang sangat kuat dan

cukup stabil untuk digunakan sebagai media penyimpanan data. Sebagai sebuah database server yang mampu untuk memanajemen database dengan baik, MySQL terhitung merupakan database yang paling digemari dan paling banyak digunakan dibandingkan database lainnya [12].

2.5.3.1. *Create Read Update Delete* (CRUD)

Pada mySQL terdapat operasi dasar yang lebih dikenal dengan operasi CRUD yaitu :

1. *Create*

   Create adalah operasi penambahan data baru ke dalam tabel. Terdapat 2 *Query* untuk menambah data, yang pertama adalah penambahan yang tidak menspesifikasikan nama kolom yang ditambahkan dan hanya memberikan isi dari tabelnya. *Query*nya adalah sebagai berikut :

```
INSERT INTO table_name
VALUES (value1,value2,value3,…);
```

   Untuk yang kedua adalah menginputkan data dengan mencantumkan nama kolom yang akan diisikan dengan isi di dalamnya. Querynya adalah sebagai berikut:

```
INSERT INTO table_name
(column1,column2,column3,...)
VALUES (value1,value2,value3,...);
```

2. *Read*

   Read adalah operasi untuk menampilkan semua atau sebagian data yang berada di dalam database. Querynya adalah sebagai berikut:

```
SELECT column_name,column_name FROM table_name;
```

3. *Update*

   Update adalah operasi untuk mengubah data yang ada di dalam database. *Query*nya adalah sebagai berikut:

```
UPDATE table_name SET column1=value1,column2=value2,...
 WHERE some_column=some_value;
```

4. *Delete*

   Delete adalah operasi untuk menghapus data yang ada di dalam table. *Query*nya adalah sebagai berikut:

```
DELETE FROM table_name
WHERE some_column=some_value;
```

2.5.3.2. Perintah SQL

Menurut bunafit Nguroho dan Indah Indriyana, SQL (*Structured Query Language*) merupakan bahasa *query* standart yang digunakan untuk mengakses basis data relasional. Perintah SQL dapat dikelompokan menjadi 5 yaitu [13]:

1. *Data Definition Language* (DDL)

DDL adalah perintah-perintah SQL yang digunakan untuk menjelaskan objek basis data. Objek basis data terdiri dari *database, table, index,* dan *view*. Perintah SQL yang termasuk DDL adalah :

a. *CREATE* : membuat atau menciptakan objek basis data.
b. *ALTER* : mengubah atau menciptakan objek basis data.
c. *DROP* : menghapus atau menciptakan objek basis data

2. *Data Manipulation Language* (DML)

DML adalah perintah-perintah SQL yang digunakan untuk mengoperasikan isi basis data. Perintah-perintah yang termasuk DML adalah :

a. *INSERT* *:* menambah data ke basis data.
b. *SELECT* : mengambil data ke basis data.
c. *UPDATE* : memodifikasi data ke basis data.
d. *DELETE* : menghapus data ke basis data.

3. *Security*

Security adalah perintah-perintah SQL yang digunakan untuk menjamin

keamanan data. Perintah SQL yang termasuk security antara lain :

a. GRANT : memberi hak akses pada user.
b. REVOKE : mencabut hak akses pada user.

4. *Integrity*

Integrity adalah perintah-perintah SQL yang digunakan untuk menjaga kesatuan data. Perintah SQL yang termasuk integrity adalah:

RECOVER TABLE : memperbaiki tabel pada basis data.

5. *Auxiliary*

Auxiliary adalah perintah-perintah pelengkap, antara lain :

a. UNLOAD
b. RENAME

2.5.4 Java Development Kit (JDK)

Java Development Kit (JDK) adalah sekumpulan perangkat lunak yang dapat kamu gunakan untuk mengembangkan perangkat lunak yang berbasis Java, sedangkan JRE adalah sebuah implementasi dari Java Virtual Machine yang benar-benar digunakan untuk menjalankan program java. Baisanya, setiap JDK berisi satu atau lebih JRE dan berbagai alat pengembangan lain seperti sumber compiler java, bundling, debuggers, development libraries dan lain sebagainya [11].

2.5.5 Software Development Kit (SDK)

Software Development Kit (SDK) adalah suatu kit atau library dari bahasa perograman untuk pengembangan atau pembangunan suatu perangkat lunak dan biasanya SDK terdiri dari kumpulan *tools* yang dibutuhkan. Misalnya bahasa pemrograman java, mempunyai SDK yang berisi suatu library yang dapat digunakan untuk membuat suatu aplikasi berbasis java [11].

2.5.6 Web Service

Web service adalah sekumpulan *application logic* beserta objek-objek dan metode-metode yang dimilikinya, yang terletak di suatu server yang terhubung ke

internet sehingga dapat diakses menggunakan *protocol* HTTP dan SOAP (*Simple Object Access Protocl*). Dalam penggunaanya, *web service* dapat digunakan hanya untuk memeriksa data *user* yang login ke sebuah website ataupun untuk digunakan pada transaksi perbankan online yang rumit.

Tujuan dari teknologi ini adalah untuk memudahkan beberapa aplikasi komponenya untuk saling terhubung dengan aplikasi lain dalam sebuah organisasi maupun diluar organisasi menggunakan standar yang tidak terikat *platform* (platform-*neutral*) dan tidak terikat akan bahasa pemrograman yang digunakan. Hal tersebut dapat terjadi karena XML standar yang didukung oleh banyak perusahaan besar didunia, yang digunakan untuk bertukar data. Selain daripada itu, penggunaan SOAP menjadi metode-metode dari objek-objek yang ada dalam sebuah *web service* dapat di akses dari aplikasi lain seperti halnya aplikasi tersebut mengakses metode *local* [14].

2.5.7 Java Script Object Nonation (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 Pemprograman JavaScript, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman 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] [15].

JSON terbuat dari dua struktur:

a) 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*.

b) 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 pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini.

Contoh JSON pada halaman website :

```
<html>
<head>
<title>JSON testing - adiputra</title>
<script type='text/javascript'>
    var JSON_adi = {nama:'adiputra'};
    document.write (JSON_adi.nama);
</script>
</head>
<body>
</body>
</html>
```

JSON_adi adalah nama variable pada javascript yang menunjukan bahwa variable tersebut berbentuk Object JSON karena terdapat kurung kurawal *{} *. Di dalam kurung kurawal terdapat atribut * nama * dengan value * adiputra *. * nama * adalah atribut, sedangkan * adiputra * adalah value dari atribut * nama * dan keduanya dipisahkan dengan titik dua * : *. Lalu untuk memanggil valuenya kita dapat panggil dengan code.

```
document.write (JSON_adi.nama);
```

code tambahan ke 2 dengan menggunakan if.

```
<html>
<head>
<title>JSON testing - adiputra</title>
<script type='text/javascript'>
    var JSON_adi = {
                          nama:'adiputra',
                          umur:21,
                          status_nikah:false,
                          hobi:['coding','main game','ngetik']
                      };

    var nikah;

    if(JSON_adi.status_nikah){
        nikah = "sudah nikah";
    }else{
        nikah = "belum nikah";
    }

   document.write (JSON_adi.nama +" berumur "+JSON_adi.umur+" suka
"+JSON_adi.hobi[0]+ " dan "+nikah);

</script>
</head>
<body>

</body>
</html>
```

2.5.8 Google Cloud Messaging API

Google Cloud Messaging (GCM) untuk android adalah layanan yang membantu pengembang mengirim data dari server untuk aplikasi android mereka ke perangkat android lainnya. Ini bisa menjadi pesan singkat untuk memberitahu aplikasi android bahwa data baru yang akan diambil dari server (misalnya, film diunggah oleh seorang teman), atau bisa juga pesan yang berisi sampai dengan 4KB data payload (sehingga aplikasi seperti *instant messaging* dapat menerima pesan langsung) [16] .

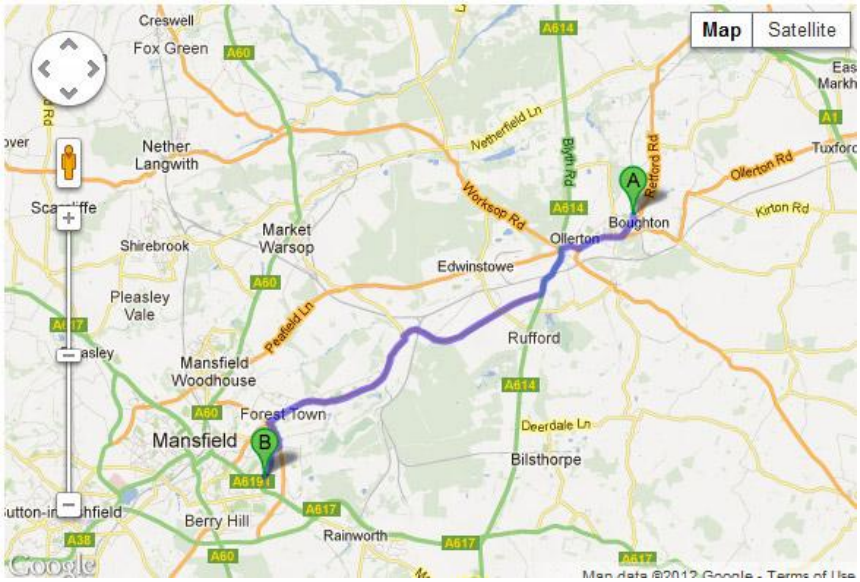
Layanan GCM menangani semua aspek antrian pesan dan pengiriman ke aplikasi target, dan berjalan pada perangkat target. GCM memungkinkan aplikasi android untuk mengirim pesan kepada server untuk membroadcast sebuah

notifikasi kepada seluruh *client* yang ada. Hanya membutuhkan sebuah *account* Gmail maka akan langsung dapat menggunakan fasilitas GCM ini.

2.5.9 Google Direction API

Google Direction API adalah layanan dari google yang memudahkan kita (developer) untuk mencari rute dan navigasi dari satu tempat tertentu. Kita tinggal memasukkan latitude dan longitude posisi berangkat dan juga latitude longitude posisi tujuan [17].

Keungulan dari API ini adalah mudah digunakan, kita hanya tinggal melakukan HTTP *request* untuk memanggil Google *Direction* API. Selain mudah, juga menyediakan banyak moda transportasi, setiap moda transportasi bisa saja memiliki rute tersendiri dan waktu tempuh tersendiri.



Gambar 2.5 Google *Direction* API

Adapun fasilitas yang bisa digunakan secara gratis tetapi mempunyai batasan sebagai berikut :

a. 2500 *request* dalam 24 jam
b. Mode transit dihitung 4 *request*
c. Bisa menggunakan sampai 8 waypoint dari setiap *request*

Jika aplikasi yang dibangun besar dan 2500 tidaklah cukup , bisa menggunakan google maps API for *business*.

2.5.10 Google Geocoding API

Geocoding adalah proses konversi alamat ke dalam kordinat geografis seperti didapatkan sebuah alamat (1600 Amphiteatre Mountain View, CA) dan di konversikan menjadi kordinat latitude 37,423021 longitude – 122,083739, yang dapat digunakan untuk menempatkan penanda pada peta atau posisi peta.

Adapun *reverse geocoding* adalah kebalikan dari *geocoding* yaitu konversi kordinat menjadi sebuah alamat yang memungkinkan dapat menemukan sebuah alamat [18].



Gambar 2.6 Contoh Google Geocoding dan Reverse Geocoding API

2.5.11 Google Maps API

Google *maps* merupakan layanan dari google yang mempermudah pengunanya untuk melakukan kemampuan pemetaan untuk aplikasi yang dibuat. Sedangkan google maps API memungkinkan pengembangan untuk mengintegrasikan Google Maps ke dalam situs web. Dengan menggunakan Google Maps Api memungkinkan untuk menanamkan situs Google Maps ke dalam situs eksternal, di mana situs data tertentu dapat dilakukan *overlay* [19].



Gambar 2.7 Google *maps*

Meskipun pada awalnya hanya javascript API, API Maps sejak diperluas untuk menyertakan sebuah API untuk Adobe Flash aplikasi, layanan untuk mengambil gambar peta status, dan layanan web untuk melakukan geocoding, menghasilkan petunjuk arah mengemudi, dan mendapatkan profil elevasi.

Kelas kunci dalam perpustakaan Maps adalah MapView, sebuah subclass dari ViewGroup dalam standar perpustakaan Android. Sebuah MapView menampilkan peta dengan data yang diperoleh dari layanan Google Maps. Bila MapView memiliki focus, dapat menangkap tombol yang ditekan dan gerakan sentuh untuk pan dan zoom peta secara otomatis, termasuk penanganan permintaan jaringan untuk ubin peta tambah. Ini juga menyediakan semua elemen UI yang diperlukan bagi pengguna untuk mengendalikan peta. Aplikasi tersebut juga dapat menggunakan metode MapView kelas untuk mengontrol MapView secara terprogram dan menarik sejumlah jenis tampilan di atas peta.

Secara umum, kelas MapView menyediakan pembungkus di Google Maps API yang memungkinkan aplikasi tersebut memanipulasi data Google Maps melalui metode kelas, dan itu memungkinkan dikerjakan dengan data Maps seperti jenis lain Views.

2.6 Pemrograman Berorientasi Objek

Metodologi berorientasi objek adalah suatu strategi pembangunan peralngkan lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya . Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientas objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas [20].

Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek.

Pada saat ini, metode berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekata terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan pada saat ini beragam dengan *platform* yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi tersebut.

Keuntungan dalam menggunakan metodologi berorientasi objek adalah sebagai berikut :

1. Meningkatkan produktivitas

   Karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut (*reusable*).

2. Kecepatan pengembangan

   Karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengodean.

3. Kemudahan pemeliharaan

   Karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.

4. Adanya konsistensi

   Karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.

5. Meningkatkan kualitas perangkat lunak

   Karena pendekatan pengembangan lebih dekat dengan dunai nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

2.6.1 Konsep Dasar Berorientasi Objek

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan sistem (sistem perangkat lunak, sistem informasi, atau sistem lainnya) [20]. Pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespondensi dengan objek-objek dunia nyata.

Ada banyak cara untuk mengabstraksikan dan memodelkan objek-objek tersebut, mulai dan abstraksi objek, kelas, hubungan antar kelas sampai abstraksi sistem. Saat mengabstraksikan dan memodelkan objek, data dan proses-proses yang dipunyai oleh objek akan dienkapsulasi (dihubungkus) menjadi suatu kesatuan.

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancanggan, pemrograman, dan pengujian perangkat lunak. Ada berbagai teknik yang dapat digunakan pada masing-masing tahap tersebut, dengan aturan dan alat bantu pemodelan tertentu.

Sistem berorientasi objek merupakan sebuah sistem yang dibangun dengan berdasarkan metode berorientasi objek adalah sebuah sistem yang komponennya dibungkus (dienkapsulasi) menjadi kelompok daata dan fungsi. Setiap komponen

dalam sistem tersebut dapat mewarisi atribut dan sifat dan komponen lainnya, dan dapat berinteraksi satu sama lain.

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek:

1. Kelas (*Class*)

   Kelas adalah sekumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statis dan himpunan objek yang sama yang mungkin lahir atau diciptakan dan kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (metode/operasi), hubungan (*relationship*) dan arti. Suatu kelas dapat diturunkan dan kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru. Secara tekniks kelas adalah sebuah struktur dalam pembuatan perangkat lunak. Kelas merupakan bentuk struktur pada kode program yang menggunakan metodologi berorientasi objek.

2. Objek (*object*)

   Objek adalah abstraksi dari sesuatu yang mewakili dunia nyata benda, manusia, satua organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstra. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.

   Secara teknis, sebuah kelas saat program dieksekusi makan akan dibuat sebuah objek. Objek dilihat darisegi teknis adalah elemen pada saat *runtime* yang akan diciptakan, dimanipulasi, dan dihancurkan saat eksekusi sehinga sebuah objek hanya ada saat sebuah program dieksekusi. Jika masih dalam bentuk kode, disebut sebagai kelas jadi pada saat *runtime* (saat sebuah program dieksekusi), yang kita punya adalah objek, di dalam teks program yang kita lihat hanyalah kelas.

3. Metode (*method*)

   Operasi atau metode pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari

satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek.

Metode atau operasi dapat berasal dari *event*, aktifitas atau aksi keadaan, fungsi, atau kelakuan dunnia nyata. Contoh metode atau operasi misalnya *Read, Write, Move, Copy,* dan sebagainya.

4. Atribut (*attribute*)

   Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat beruoa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya.

5. Abstraksi (*abstraction*)

   Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

6. Enkapsulasi (*encapsulation*)

   Pembungkusa atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerja-nya.

7. Pewarisan (*inheritance*)

   Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dan dirinya.

8. Antarmuka (*interface*)

   Antarmuka sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah *interface* dapat diimplementasikan oleh kelas lain.

9. *Reusability*

   Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

10. Generalisasi dan Spesialisasi

    Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, dan kereta.

11. Komunikasi Antarobjek

    Komunikasi antarobjek dilakukan lewat pesan (*message*) yang dikirim dari satu objek ke objek lainnya.

12. Polimorfisme (*polymorphism*)

    Kemampuan suatu objek digunakan di banyak tujuan yang berbeda dengan nama yang sehingga menghemat baris program.

13. *Package*

    Package adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang berbeda

2.6.2 Unified Model Language (UML)

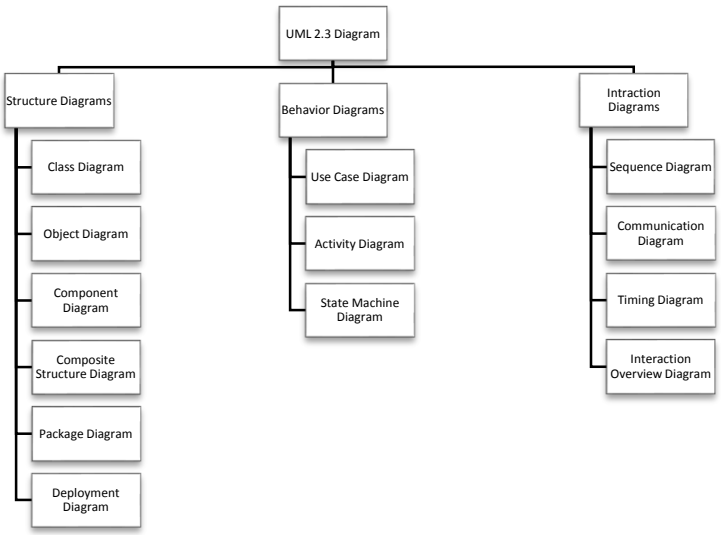
Pada perkembangan teknologi perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang di berbagai negara dapat mengerti pemodelan perangkat lunak. Seperti yang kita ketahui bahwa menyatukan banyak kepala untuk menceritakan sebuah ide dengan tujuan untuk memahami hal yang tidaklah mudah, oleh karena itu diperlukan sebuah bahasa pemodelan perangkat lunak yang dapat dimengerti oleh banyak orang. Banyak orang yang telah membuat bahasa pemodelan pembangunan perangkat lunak sesuai dengan teknologi pemrograman yang berkembang pada saat itu, misalnya yang sempat berkembang dan digunakan banyak pihak adalah *Data Flow Diagram* (DFD) untuk memodelkan perangkat.

lunak yang menggunakan pemrograman prosedural atau struktural, kemudian juga ada *State Transition Diagram* (STD) yang digunakan untuk memodelkan sistem *real time* (waktu nyata). Pada perkembangan teknil pemrograman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan

untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modeling Language* (UML). UML muncul karena adanya kebutuhan pemodelan voisual untu menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan digaram dan teks-teks pendukung.UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek. Seperti yang kita ketahui di dunia sistem informasi yang tidak dapat dibakukan, semua tergantung kebutuhan, lingkunan dan konteksnya. Begitu juga dengan perkembangan penggunaan UML bergantung pada level abstraksi penggunaannya. Jadi, belum tentu pandangan yang perbeda dalam penggunaan UML adalah suatu yang salah, tapi perlu ditelaah dimanakah UML digunakan dan hal apa yang ingin digambarkan. Secara analogi jika dengan bahasa yang digunakan sehari-hari, belum tentu penyampaian bahasa dengan puisi adalahhal yang salah. Sistem informasi bukanlah ilmu pasti, maka jika ada banyak perbedaan dan interpretasi di dalam bidang sistem informasi merupakan hal yang sangat wajar [20].

2.6.2.1. Diagram UML

Pada UML terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar berikut



Gambar 2.8 Contoh Diagram UML [20]

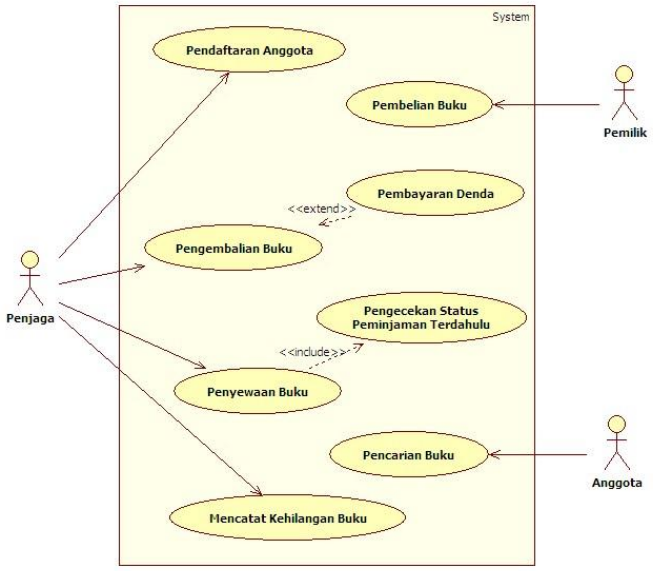
Berikut ini penjelasan singkat dari pembagian kategori tersebut :

1. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
3. *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antarsubsistem pada suatu sistem

2.6.2.2. *Use Case* Diagram

Use Case atau diagram *Use Case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Syarat penamaan pada *Use Case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *Use Case* yaitu pendefinisian apa yang disebut aktor dan *Use Case.*



Gambar 2.9 Contoh *Use case Diagram* [20]

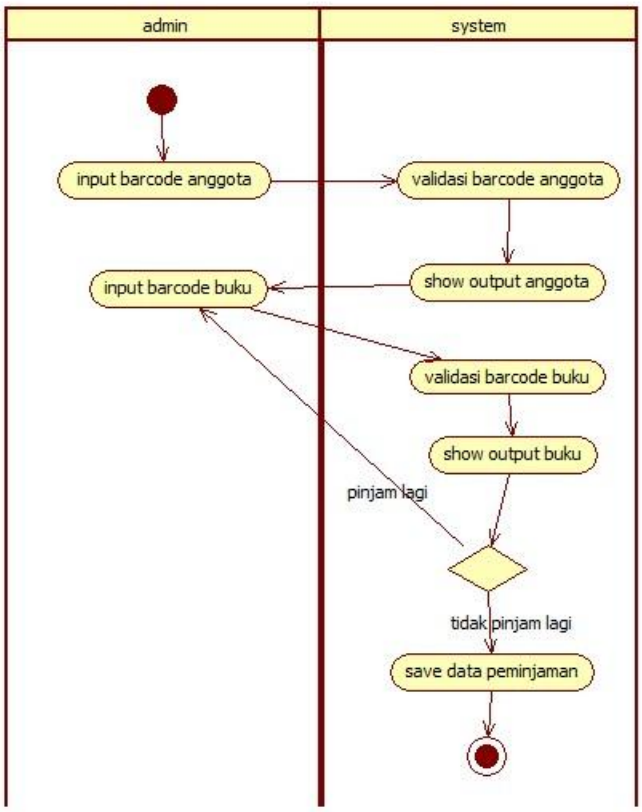
2.6.2.3. *Activity* Diagam

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut :

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan
2. Urutan atau pengelompokan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan

Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.



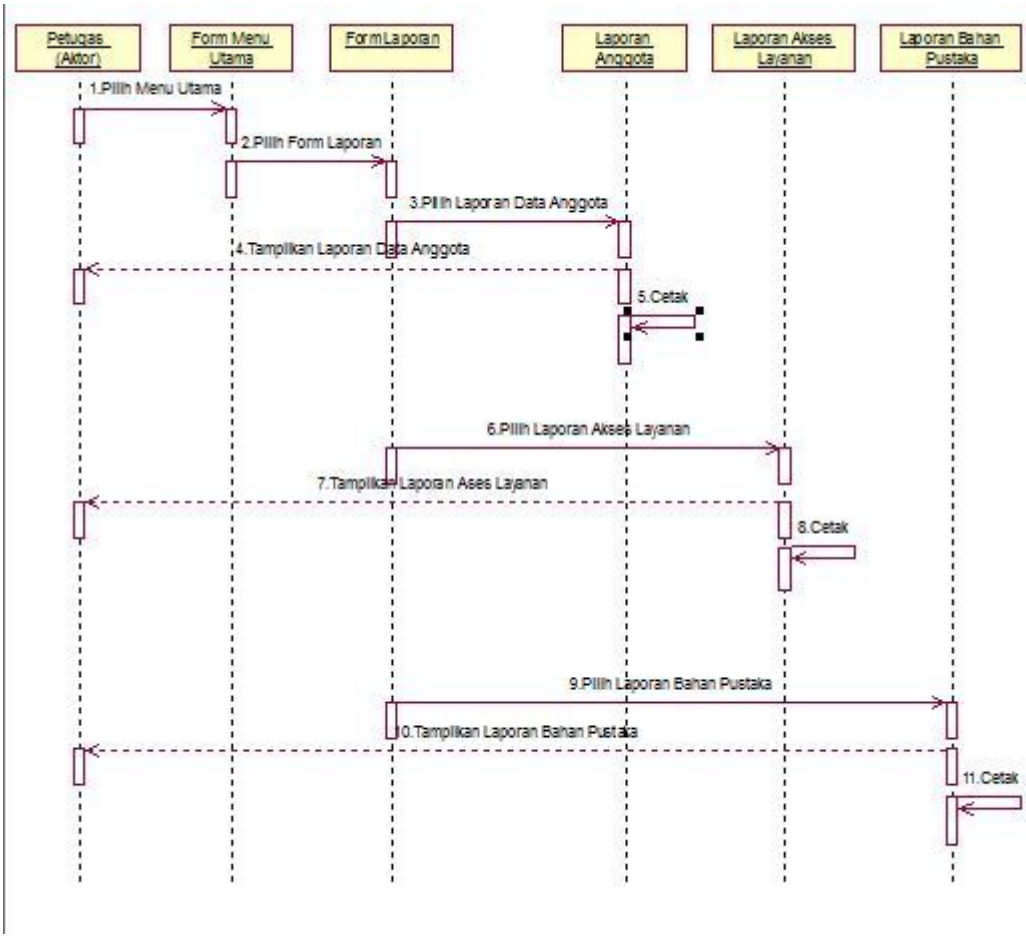
Gambar 2.10 Contoh *Activity Diagram* [20]

2.6.2.4. *Sequence* Diagram

Diagram sequence menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sequence maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram sequence yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sequence sehingga semakin banyak *use case* yang didefinisikan maka diagram sequence yang harus dibuat juga semakin banyak.

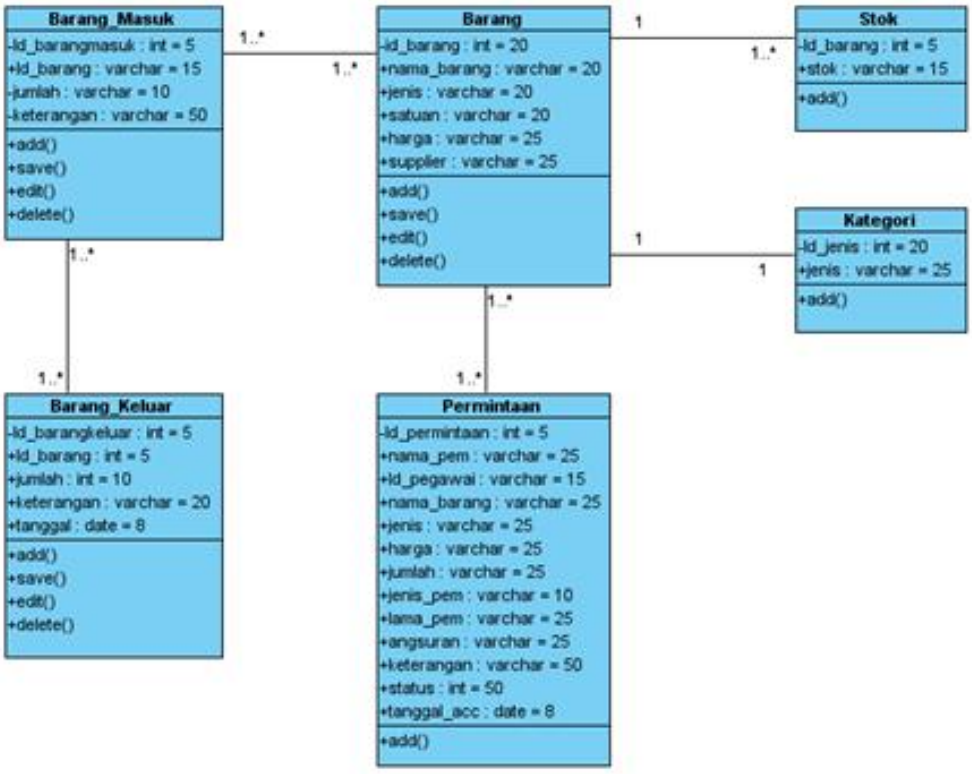
Penomoran pesan berdasarkan urutan interaksi pesan. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu.



Gambar 2.11. Contoh *Sequence Diagram* [20]

2.6.2.5. *Class* Diagram

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.



Gambar 2.12. Contoh *Class Diagram* [20]

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem. Susunan struktur kelas yang baik pada diagram kelas ebaiknya memiliki jenis-jenis kelas berikut :

1) Kelas Main

   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.

2) Kelas yang menangani tampilan sistem

   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.

3) Kelas yang diambil dari pendefinisian *use case*

   Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*.

4) Kelas yang diambil dari pendefinisian data

   Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Jenis-jenis kelas diatas juga dapat digabungkan satu sama lain sesuai dengan pertimbangan yang dianggap baik asalkan fungsi-fungsi yang sebaiknya ada pada struktur kelas tetap ada. Susunan kelas juga dapat ditambahkan kelas utilitas seperti koneksi ke basis data, membaca *file* teks, dan lain sebagainya sesuai kebutuhan.

Dalam mendefinisikan metode yang ada di dalam kelas perlu memperhatikan apa yang disebut dengan *cohesion* dan coupling. *Cohesion* adalah ukuran seberapa dekat keterkaitan instruksi di dalam sebuah metode terkait satu sama lain sedangkan *coupling* adalah ukuran seberapa dekat keterkaitan instruksi antara metode yang satu dengan metode yang lain dalam sebuah kelas. Sebagai aturan secara umum maka sebuah metode yang dibuat harus memiliki kadar *choesion* yang kuat dan kadar *coupling* yang lemah.

Dalam class diagram terdapat beberapa relasi (hubungan antar class) yaitu:

1. *Generalization* dan *Inheritence*

   Diperlukan untuk memperlihatkan hubungan pewarisan (inheritance) antar unsur dalam diagram kelas. Pewarisan memungkinkan suatu kelas mewarisi semua atribut, operasi, relasi, dari kelas yang berada dalam hirarki pewarisannya.

2. *Associations*

   Hubungan statis antar class. Umumnya menggambarkan class yang memiliki atribut berupa class lain, atau class yang harus mengetahui ekstensi class lain. Dalam notasi UML kita mengenal asosiasi 2 arah (bidirectional) dan 1 arah (undirectional).

3. *Aggregation*

   Hubungan antar-class dimana class yang satu (part class) adalah bagian dari class lainnya (whole class).

4. *Composition*

   Aggregation dengan ikatan yang lebih kuat. Di dalam composite aggregation, siklus hidup part class sangat bergantung pada whole class sehingga bila objek instance dari whole class dihapus maka objek instance dari part calss juga akan terhapus.

5. *Depedency*

   Hubungan antar-class dimana sebuah class memiliki ketergantungan pada class lainnya tetapi tidak sebaliknya.

6. *Realization*

   Hubungan antar-class dimana sebuah class memiliki keharusan untuk mengikuti aturan yang ditetapkan class lainnya. Biasanya realization digunakan untk menspesifikasikan hubungan antara sebuah interface dengan class yang mengimplementasikan interface tersebut.