# BAB 2
# LANDASAN TEORI

## 2.1 Travelling

*Travelling* adalah aktivitas berpindah dalam satu tempat ketempat lainnya dengan berbagai alasan, seperti bisnis, liburan, dan sebagainya. Aktivitas *travelling* diharapkan bukan hanya dilakukan saat libur atau waktu senggang, melainkan sudah menjadi kegiatan yang termasuk 'penting' dalam jadwal kesibukan harian, keperluannya setara dengan aktivitas kerja sehari-hari. Di dalam buku *travelling* karangan Herajeng Gustiayu, terdapat 3 hal dasar yang penting dari membuat perencanaan *travelling* dan menentukan biaya yang tepat diantaranya adalah transportasi, akomodasi, dan makanan [**6**].

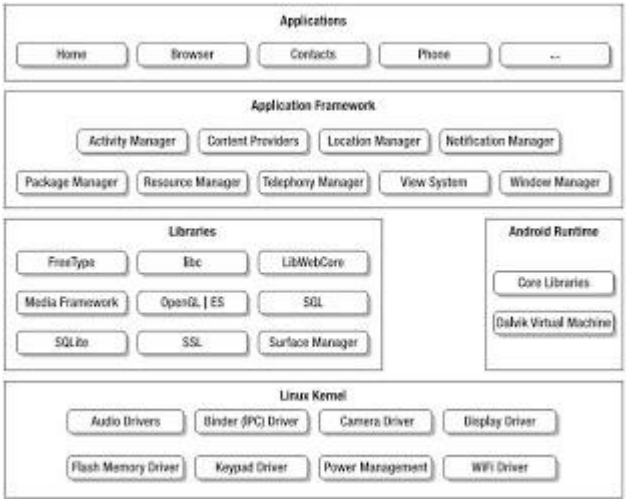
## 2.2 Android

Android merupakan sistem operasi mobile yang tumbuh di tengah sistem operasi lainnya yang berkembang dewasa ini. Sistem operasi lainnya seperti Windows Mobile, i-Phone OS, Symbian dan masih banyak lagi juga menawarkan kekayaan isi dan keoptimalan berjalan di atas perangkat hardware yang ada. Akan tetapi, sistem operasi yang ada ini berjalan dengan memprioritaskan aplikasi inti yang dibangun sendiri tanpa melihat potensi yang cukup besar dari aplikasi pihak ketiga. Oleh karena itu, adanya keterbatasan distribusi aplikasi pihak ketiga untuk platform mereka [**7**].

Android menawarkan sebuah lingkungan yang berbeda untuk pengembang. Setiap aplikasi memiliki tingkatan yang sama. Android tidak membedakan antara aplikasi inti dengan aplikasi pihak ketiga. *Application Programming Interface* (API) yang disediakan menawarkan akses ke *hardware*, maupun data-data ponsel sekalipun, atau data sistem sendiri. Bahkan pengguna dapat menghapus aplikasi inti dan menggantikannya dengan aplikasi pihak ketiga [**7**].

### 2.2.1 Arsitektur Android

Arsitektur Android dapat digambarkan seperti pada Gambar 2.1 Arsitektur Android dan secara garis besar Arsitektur Android dapat dijelaskan sebagai berikut:




**Gambar 2. 1 Arsitektur Android [**8**]**

1. Application dan Widget

*Application* dan *Widgets* ini adalah layer dimana kita berhubungan dengan aplikasi saja, dimana biasanya kita download aplikasi kemudian kita lakukan instalasi dan jalankan aplikasi tersebut. Di layer terdapat aplikasi inti termasuk klien email, program SMS, kalender, peta, browser, kontak, dan lain-lain. Hampir semua aplikasi ditulis menggunakan bahasa pemrograman Java [**7**].

2. Application Frameworks

Android adalah "*Open Development Platform*" yaitu Android menawarkan kepada pengembang atau memberi kemampuan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *resource*, menjalankan *service background*, mengatur alarm, dan menambah status notifikasi, dan sebagainya. Pengembang memiliki akses penuh menuju API *framework* seperti yang dilakukan oleh aplikasi kategori inti. Arsitektur aplikasi dirancang supaya kita dengan mudah dapat menggunakan kembali komponen yang sudah digunakan (*reuse*). Sehingga bisa kita simpulkan *Application Frameworks* ini adalah layer

dimana para pembuat aplikasi melakukan pengembangan/pembuatan aplikasi yang akan dijalankan di sistem operasi Android, karena pada layer inilah aplikasi dapat dirancang dan dibuat, seperti content providers yang berupa sms dan panggilan telepon[14]. Komponen-komponen yang termasuk di dalam *Application Frameworks* adalah sebagai berikut:

*1) Views*
*2) Content Provider*
*3) Resource Manager*
*4) Notification Manager*
*5) Activity Manager*

3. Libraries

*Libraries* ini adalah layer dimana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses libraries untuk menjalankan aplikasinya. Berjalan di atas Kernel, layer ini meliputi berbagai library C/C++ inti seperti Libc SSL, serta:

1) *Libraries* media untuk pemutaran media audio dan video
2) *Libraries* untuk manajemen tampilan
3) *Libraries Graphics* mencakup SGL dan OpenGL untuk grafis 2D dan 3D
4) *Libraries* SQLite untuk dukungan database
5) *Libraries* SSL dan WebKit terintegrasi dengan web browser dan security
6) *Libraries* LiveWebcore mencakup modern web browser dengan engine embedded web view
7) *Libraries* 3D yang mencakup implementasi OpenGL ES1.0 API's

4. Android Run Time

Layer yang membuat aplikasi Android dapat dijalankan dimana dalam prosesnya menggunakan Implementasi Linux. *Dalvik Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android. Di dalam Android *Run Time* dibagi menjadi dua bagian yaitu:

1) Core Libraries

   Aplikasi Android dibangun dalam bahasa Java, sementara Dalvik sebagai virtual mesinnya bukan Virtual Machine Java, sehingga diperlukan

sebuah *libraries* yang berfungsi untuk menterjemahkan bahasa Java/C yang ditangani oleh Core Libraries

2) Dalvik Virtual Machine

Virtual mesin berbasis register yang dioptimalkan untuk menjalankan fungsi-fungsi secara efisien, dimana merupakan pengembangan yang mampu membuat Linux Kernel untuk melakukan threading dan manajemen tingkat rendah.

5. Linux Kernel

Linux Kernel adalah layer dimana inti dari sistem operasi Android itu berada. Berisi file-file sistem yang mengatur sistem *processing*, *memory*, *resource*, *drivers*, dan sistem-sistem operasi Android lainnya. Linux Kernel yang digunakan Android adalah Linux Kernel *release* 2.6.

### 2.2.2 Android Life Cycle

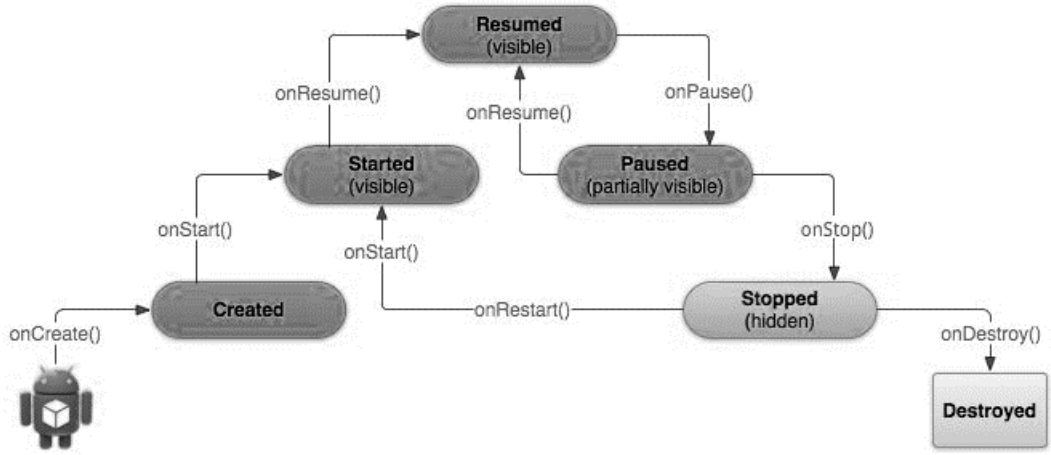
Apikasi android terdiri dari beberapa fungsi dasar seperti mengedit catatan, memutar file musik, membunyikan alarm, atau membuka kontak telepon. Fungsi-fungsi tersebut dapat diklasifikasikan ke dalam empat komponen android yang berbeda seperti ditunjukan pada tabel, klasifikasi tersebut berdasarkan kelas-kelas dasar java yang digunakan [**9**].

**Tabel 2. 1 Komponen Aplikasi Android**

| Functionality | Java Base Class | Example |
|---|---|---|
| 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 retrive 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, BroadcastReceiver, ContentProvider* [**9**].

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 dekstop 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 [**10**].




**Gambar 2. 2 Siklus Hidup Android [**10**]**

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() [**10**].

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 dalan state ini [**10**].

3. Stopped

Dalam keadaan ini, aplikasi benar-benar tidak ditampilkan dan tidak terlihat oleh pengguna tetapi masih meninggalkan service di background [**10**].

State lain seperti Create and 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() [**9**]

### 2.2.3 Fitur

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

1. Multi-proses dan App Widget

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 [**10**].

*App Widget* adalah mini aplikasi yang dapat *embedded* dalam aplikasi seperti home screen. App widget dapat menjalankan proses request seperti musik *streaming* atau mendeteksi suhu ruangan secara *background* [**10**].

*Multi-proses* dapat memberikan manfaat berupa *user experience* yang lebih banyak, namun pengguna fitur tersebut dapat menghabiskan banyak energi baterai jika penggunaan tidak benar[6].

2. Touch Gestures dan Multi-touch

*Touchscreen* adalah *user interface* intuitif yang digunakan banyak *smartphine* di dunia. Dengan fitur ini interaksi dapat dibuat lebih mudah karena cukup dengan menggunakan jari tangan. *Multi-touch* adalah kemampuan yang digunakan untuk interaksi memperbesar atau memutar objek. Selain itu pengembang dapat membuat interaksi baru dengan memanfaatkan fitur tersebut [**9**].

3. Hard dan Soft Keyboard

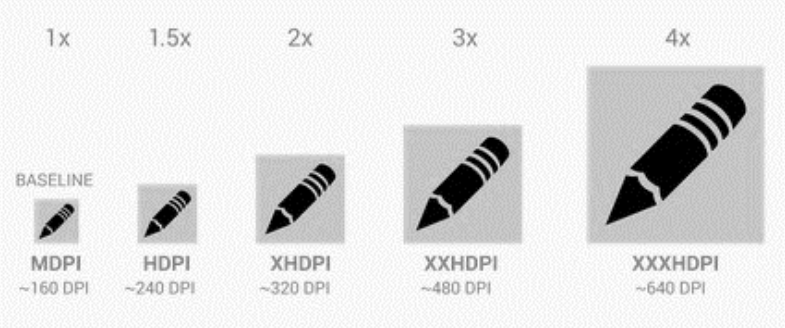
Salah satu fitur pada perangkat *smartphone* adalah tombol fisik dan non fisik, tombol fisik digunakan untuk navigasi pendukung dalam pengoprasian android. Pengembang aplikasi tidak perlu secara manual untuk mengintegrasikan tombol tersebut dalam aplikasi. Tombol non fisik adalah tombol yang diuat oleh sistem operasi seperti keyboard virtual, dan tombol navigasi aplikasi [**9**].

### 2.2.4 Prinsip Desain

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

1. Multiple Assets

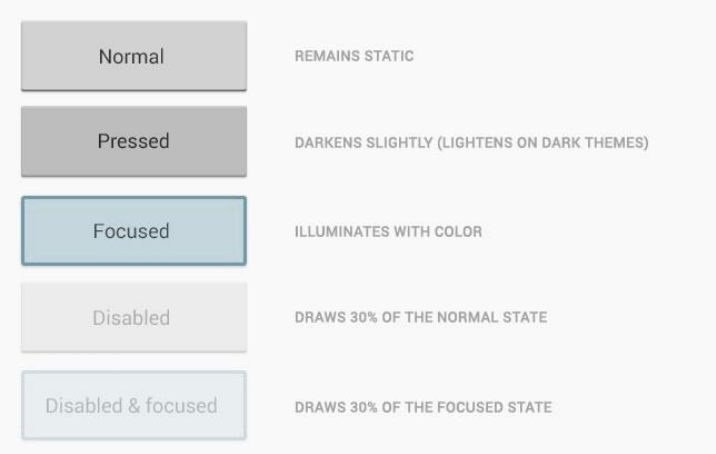
Android mendukung jutaan *smartphone,* tablet dan perangkat lain dalam berbagai ukuran layar dan ukuran, untuk itu *Multiple Assets* sangat disarankan digunakan untuk mengatasi fragmentasi dapa android. Seperti ilustrasi pada Gambar 2.3 Klasifikasi Ukuran Ikon, android menciptakan beberapa klasifikasi 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, XXHDP dan XXXHDPI digunakan pada *device* berukuran tablet[8].




**Gambar 2. 3 Klasifikasi Ukuran Ikon**

2. Touch Feedback

*Touch Feedback* dalam android digunakan sebagai repon setiap objek yang ditekan pengguna. Hal ini bertujuan untuk memberi tahu pengguna objek mana yang berinteraksi dengan pengguna [**11**].




**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 diantaranya adalah [**11**]:

1) 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*.

2) 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 [**11**].

3) Swipe or Drag

*Swipe* adalah menyentuh sebuah titik pada layar dan menggerakkan jari yang tetap tersentuh pada layar ke titik lain pada layar. *Swipe* dapat dilakukan dari dan ke arah mana saja [**11**].

4) Double Touch

Pada *smartphone* dan tablet android, melakukan dua kali tapping secara berturut-turut pada satu objek, fungsinya berbeda dengan double klik mouse komputer. Pada android, teknik ini biasanya dipakai untuk melakukan *zoom in* atau memperbesar dan *zoom out* atau memperkecil sebuah objek gambar [**11**].

5) 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 terseut tidak bersentuhan, kemudian kedua jari tersebut sambil teteap menyentuh layar bergerak saling mendekati. Teknik ini digunakan untuk membuka aplikasi tertentu [**11**].

6) Pinch Close

Teknik *Pinch Close* adalah kebalikan dari *Pinch Open* di mana *spread* dilakukan dengan berawal dua jari 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 [**11**].

## 2.3 User-Generated Content

*User-generated Content* (UGC) dikenal dengan istilah *consumer-generated media* (CGM) atau *user-created content* (UCC) adalah merujuk pada berbagai tipe konten materi atau media yang tersedia untuk umum dan diproduksi oleh *end-user* atau para non-media profesional. Kemajuan teknologi web memungkinkan konten

website tidak lagi dimonopoli oleh profesional media, melainkan dapat dibuat oleh para penggunanya. UGC merupakan salah satu ciri dominan Web 2.0. contoh penerapan pada UGC adalah YouTube hampir semua konten yang dimiliki situs tersebut dibuat dan diupload oleh penggunanya [**12**].

Konsep UGC pada dasarnya telah banyak merubah cara berinteraksi pengguna dengan internet begitu juga dalam media periklanan. Bagi media periklanan jejaring sosial dengan konsep UGC memiliki potensi besar menyediakan market yang lebih terarah dan terpusat bagi mereka [**9**].

## 2.4 JSON

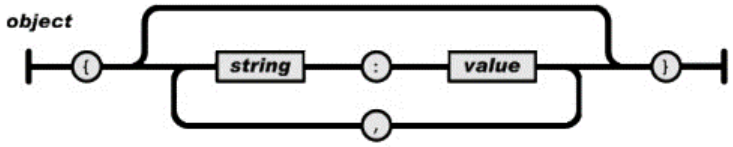
JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemrograman JavaScript, Standar ECMA-262 Edisi ke 3 – Desember 1999. 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#, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran data. JSON terbuat dari dua struktur:

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), daftar berkunci (*keyed list*), atau associative array[10].
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*) [**13**].

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemrograman modern mendukung sruktur 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

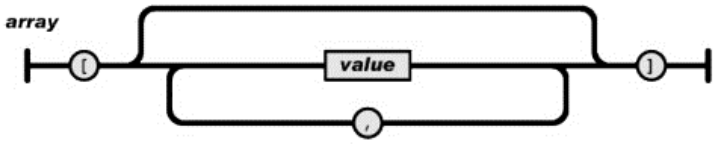
Objek adalah sepasang nama / nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma). Objek biasanya digunakan untuk menyimpan data tunggal dalam bentuk JSON [**13**].




**Gambar 2. 5 Objek JSON [**13**]**

2. Larik

Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [ 9kurung 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 [**13**].




**Gambar 2. 6 Array JSON [**13**]**

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, Bollean dan Nilai NULL [**13**].

## 2.5 MySQL

*Structur Query Language* disingkat SQL awalnya digunakan untuk mengambil atau meminta (*query*) informasi dari *database*, proses pembuatan tabel, pengguna, memasukan data, membeuat *stored procedure*, *trigger*, fungsi, pengaturan keamanan hak akses juga dapat dikerjakan SQL. SQL tidak hanya mengambil informasi dari tabel-tabel *database* SQL dapat digunakan dalam

membuat elemen dalam *database*, memasukan, mengubah dan menghapus data dari *database*. Dengan memanfaatkan bahasa web seperti HTML, dan PHP, SQL merupakan tool penting yang tidak dapat ditinggalkan untuk pengembangan aplikasi database berbasis web. Query dasar yang sering digunakan adalah create, read, update dan delete.

Create digunakan untuk menginputkan data kesebuah tabel di database, contoh querynya sebagai berikut:

```
INSERT INTO anggota(nama_anggota, umur) VALUES('Asep', 21)
```

Read digunakan untuk menampilakan data sesuai dengan ketentuan tertentu, contoh querynya sebagai berikut :

```
SELECT * FROM anggota
```

Update digunakan untuk mengubah data sesuai dengan apa yang diinginkan, contoh querynya adalah sebagai berikut:

```
UPDATE anggota SET umur = '21' WHERE id_anggota = 1
```

Delete digunakan untuk menghapus data yang ingin dihilangkan, contoh querynya sebagai berikut:

```
DELETE FROM anggota WHERE id_anggota =5
```

## 2.6 Object Oriented Analysis Desain

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**].

### 2.6.1 Unified Modeling Language(UML)

*Unified Modeling Language* (UML) adalah termasuk ke dalam rumpun jenis pemodelan notasi grafis yang didukung oleh model-model tunggal. Pemodelan ini berguna untuk membantu dalam menjelaskan data rancang perangkat lunak yang dibangun dengan *object-oriented* (OO). UML merupakan standar terbuka yang dikelola *Open Management Group* (OMG) yang berada dibawah naungan perusahaan-perusahaan konsorium terbuka. UML merupakan suatu bahasa pemodelan yang terdiri banyak model diantaranya adalah [**14**]:

1. Use Case Diagram

Diagram *use case* merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem secara keseluran yang akan dibuat. Diagram *use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Dengan pengertian yang cepat, diagram *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Yang ditekankan pada diagram ini adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". *Use case* menjelaskan secara sederhana fungsi sistem dari sudut pandang user. Adapun komponen-komponen dalam use case diagram anataranya [**14**]:

1) Actor

Aktor adalah segala hal diluar sistem yang akan menggunakan sistem tersebut untuk melakukan sesuatu. Bisa merupakan manusia, sistem, atau *device* yang memiliki peranan dalam keberhasilan operasi dari system.

2) Use Case

Mengidentifikasi fitur kunci dari sistem. Tanpa fitur ini, sistem tidak akan memenuhi permintaan *user/actor*. Setiap *use case* mengekspresikan *goal* dari sistem yang harus dicapai. Diberi nama sesuai dengan *goal*-nya dan digambarkan dengan *elips* dengan nama di dalamnya. Fokus tetap pada *goal* bukan bagaimana mengimplementasikannya walaupun *use case* berimplikasi pada prosesnya nanti. Setiap *use case* biasanya memiliki *trigger*/pemicu yang menyebabkan *use case.*

3) System

Menyatakan batasan sistem dalam relasi dengan *actor-actor* yang menggunakannya (di luar sistem) dan fitur-fitur yang harus disediakan (dalam sistem). Digambarkan dengan segi empat yang membatasi semua *use case* dalam sistem terhadap pihak mana sistem akan berinteraksi. Sistem disertai label yang menyebutkan nama dari sistem, tapi umumnya tidak digambarkan karena tidak terlalu memberi arti tambahan pada diagram.

4) Association

Mengidentifikasikan interaksi antara setiap *actor* tertentu dengan setiap *use case* tertentu. Digambarkan sebagai garis antara *actor* terhadap *use case* yang bersangkutan. Asosiasi bisa berarah (garis dengan anak panah) jika komunikasi satu arah, namun umumnya terjadi kedua arah (tanpa anak panah) karena selalu diperlukan demikian.

5) Dependency

Dependensi <<include>>

1. Mengidentifikasi hubungan antar dua *use case* di mana yang satu memanggil yang lain.
2. Jika pada beberapa *use case* terdapat bagian yang memiliki aktivitas yang sama maka bagian aktivitas tersebut biasanya dijadikan *use case* tersendiri dengan relasi dependensi setiap *use case* semula ke *use case* yang baru ini sehingga memudahkan pemeliharaan.
3. Digambarkan dengan garis putus-putus bermata panah dengan notasi <<include>> pada garis.
4. Arah mata panah sesuai dengan arah pemanggilan

Dependensi <<extend>>

1. Jika pemanggilan memerlukan adanya kondisi tertentu maka berlaku dependensi <<extend>>.
2. Digambarkan serupa dengan dependensi <<include>> kecuali arah panah berlawanan.

3. Note: konsep "*extend*" ini berbeda dengan "extend" dalam Java.

6) Generalization

Mendefinisikan relasi antara dua *actor* atau dua *use case* yang mana salah satunya meng-*inherit* dan menambahkan atau *override* sifat dari yang lainnya. Penggambaran menggunakan garis bermata panah kosong dari yang meng-*inherit* mengarah ke yang di-*inherit.*

2. 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 [**14**].

3. Sequence Diagram

*Sequence diagram* menjelaskan secara detail urutan proses yang dilakukan dalam system untuk mencapai tujuan dari *use case*, interaksi yang terjadi antar *class*, operasi apa saja yang terlibat, urutan antar operasi dan informasi yang diperlukan oleh masing-masing operasi [**14**].

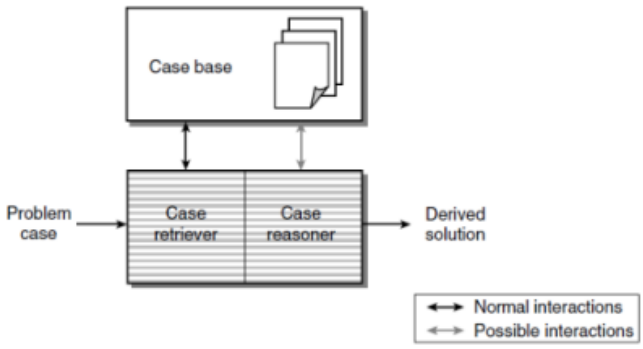
## 2.7 Sistem Rekomendasi

Konsep sistem rekomendasi telah digunakan secara luas oleh hampir semua area bisnis dimana seorang konsumen memerlukan informasi untuk membuat suatu keputusan [**15**]. Sistem rekomendasi pariwisata atau perjalanan menggunakan konsep ini untuk menolong para wisatawan untuk memutuskan tujuan perjalanan mereka, akomodasi dan aktivitas di tempat tujuan.

Sistem rekomendasi merupakan model aplikasi dari hasil observasi terhadap keadaan dan keinginan pelanggan. Oleh karena itu sistem rekomendasi memerlukan model rekomendasi yang tepat agar yang direkomendasikan sesuai dengan keinginan pelanggan, serta mempermudah pelanggan mengambil keputusan yang tepat dalam menentukan produk yang akan digunakannnya [**16**].

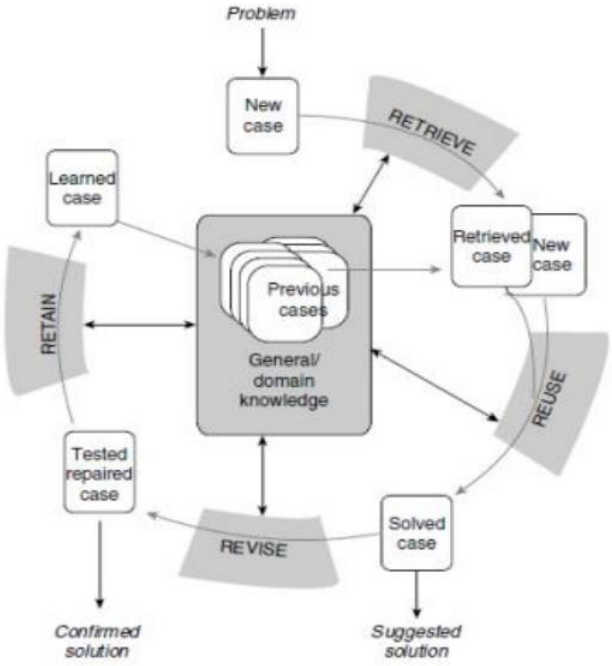
## 2.8 Case Base Reasoning (CBR)

*Case-Based Reasoning* (CBR) adalah metode penyelesaian masalah dengan menggunakan solusi masalah-masalah sebelumnya yang serupa [**2**]. CBR sendiri adalah metode yang umum digunakan manusia dalam menyelesaiakan masalah sehari-hari. Layaknya metode penyelesaian masalah lainnya, computer dapat meniru CBR.




**Gambar 2. 7 Komponen Case Based Reasoning**

Komponen penyelesaian masalah CBR memiliki dua bagian utama : *case retriever* dan *case reasoner*. *Case retriever* bertugas untuk mencari masalah-masalah yang mirip dengan masalah yang dimasukkan pengguna di dalam basis pengetahuan (*case base*), sementara *case reasoner* bertugas untuk mencari solusi masalah pengguna dengan melihat atau menyesuaikan solusi masalah yang ditemukan *case retriever*.




**Gambar 2. 8 Siklus Metode *Case Based Reasoning***

Dalam eksekusinya, ada empat tahapan dalam proses CBR [**2**]:

1. *Retrieve*: mengambil kasus-kasus lama dari case base yang mirip dengan kasus yang dihadapi.
2. *Reuse*: menggunakan solusi kasus-kasus lama hasil retrieve tersebut untuk menyelesaikan kasus yang baru tersebut.
3. *Revise*: jika diperlukan, mengadaptasi solusi kasus lama agar sesuai dengan kondisi masalah baru.
4. *Retain*: menyimpan solusi hasil revise yang telah divalidasi ke dalam basis data, agar dapat digunakan untuk menyelesaikan masalah serupa di masa depan.

Tidak semua bagian *case retriever* dan *case reasoner* harus diotomatis. Ada banyak program aplikasi CBR yang hanya melakukan *case retriever* dan memberikan hasil *case retriever* ke pengguna. Pengguna sendiri yang selanjutnya menafsirkan data tersebut dan mengambil keputusan. Dalam hal ini proses *case retrieve* dikerjakan oleh komputer, namun *case reasoning*-nya diserahkan pada pengguna [**17**].

## 2.9 Algoritma Nearest Neighbor Retrieval

Algoritma *Nearest Neighbor Retrieval* (*k-nearest neighbor* atau k-NN) sebuah algoritma untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. Kasus khusus di mana klasifikasi diprediksikan berdasarkan data pembelajaran yang paling dekat (dengan kata lain, k = 1) disebut algoritma nearest neighbor [**18**].

Algoritma *nearest neighbor* berdasarkan pada proses pembelajaran menggunakan analogi / *learning by* analogi. Training sampelnya dideskripsikan dalam bentuk atribut numerik n-dimensi. Tiap sampel mewakili sebuah titik pada ruang n- dimensi. Dengan cara ini, semua training sampel disimpan pada pola ruang n-dimensi. Ketika diberikan "*unknown*" sampel, *k- nearest neighbor classifier* mencari pola ruang K training sampel yang paling dekat "*unknown*" sampel tersebut. K training sampel ini adalah *k nearest neighbor* dari *unknown* sampel. *Unknown* sampel ditetapkan dengan class yang paling umum *diantara k nearest neighbor*s-nya. Ketika k = 1, *unknown* sampel ditetapkan dengan class dari training sampel yang paling dekat dengan pola ruangnya.

Algoritma *nearest neighbor retrieval* menyimpan semua training sampel dan tidak. Membangun *classifier* sampai sampel baru (*unlabeled*) perlu diklasifikasikan, sehingga algoritma *nearest neighbor retrieval* sering disebut dengan *instance-based* atau *lazy learners*.

Rumus untuk menghitung bobot kemiripan (*similarity*) dengan *nearest neighbor Retrieval* adalah

$$\text{Similarity (problem,case)} = \frac{(S1*W1)+(S2*W2)+(S3*W3)+\cdots+(Sn*Wn)}{W1+W2+W3+\cdots+Wn} \qquad (1)$$

Keterangan:

S = *similarity* (nilai kemiripan)

W = *weight* (bobot yang diberikan)

## 2.10 Rank Order Centroid (ROC)

Teknik pembobotan ROC adalah teknik memberikan bobot pada setiap kriteria sesuai dengan ranking yang dinilai berdasarkan tingkat prioritas kriterianya [**19**]. Biasanya dibentuk dengan pernyataan "Kriteria 1 lebih penting dari kriteria 2, yang lebih *penting* dari kriteria 3" dan seterusnya hingga kriteria ke n, jika ditulis menjadi :

Cr1 >= Cr2 >= Cr3 … >= Crn

Untuk memberikan bobotnya, diberikan aturan yang sama yaitu :

W1 >= W2 >= W3 … >= Wn,

Dimana W1 merupakan bobot untuk kriteria Cr1. Secara umum pembobotan ROC dapat dirumuskan seperti persamaan (2) sebagai berikut :

$$Wk = 1/k \sum_{n=1}^{k} \left(\frac{1}{i}\right) \qquad (2)$$

Keterangan:

Wk = Bobot atribut n

k = banyaknya atribut

## 2.11 Jenis Liburan

Sebelum melakukan kegiatan travelling, sangat penting untuk menentukan jenis tujuan wisata. Menentukan tujuan wisata seharusnya sesuai dengan minat dan hobi. Setidaknya, ada 4 jenis tujuan wisata yang bisa dijadikan pilihan sesuai minat dan hobi [**20**]. Berikut ini adalah 4 jenis tujuan wisata :

a. Warisan Budaya atau Heritage

Di tempat seperti ini akan banyak bangunan tua dengan sejarahnya masing-masing. Gaya hidup penduduk lokalnya juga masih asli dan menjunjung tinggi budaya lokal. Wilayah sekitar peninggalan sejarah dilindungi baik oleh negara maupun dun ia internasional. Pilihan makanan dan transportasinya masih banyak yang lokal. Kalau tertarik dengan sejarah atau berniat kabur dari ingar bingar kota dan kantor yang super sibuk, biasanya heritage menjadi pilihan [**20**].

b. Lingkungan Alam

Bagi pecinta alam, tidaklah harus berpetualang di hutan atau naik gunung. Jika harus membawa keluarga atau tidak memiliki persiapan yang memadai berpetualang di alam bebas, masih bisa menikmati taman kota atau alun-alun, kebun buah, cagar alam atau kebun binatang [**20**].

c. Kota Besar Metropolitan

Cocok bagi yang tujuannya belanja, menikmati gemerlap malam, melihat-lihat gaya dan busana penduduk lokal dari pendatang dari berbagai negara, termasuk mempelajari tata kota dan kehidupan perkotaannya. Sistem transportasi lokal, tempat belanja, dan kehidupan malamnya sangat mudah didapat di penjuru kotanya [**20**].

d. Wisata Religi atau Agama

Mengunjungi tempat tempat ibadah atau tempat yang memiliki latar belakang keagamaan, baik dari sisi kegiatan atau sejarahnya. Wisata religi tidak harus ke tujuan utama ibadah, misalnya ke Arab Saudi atau ke Israel. Bisa juga ke banyak tempat yang terkait dengan sejarah penyebaran agama [**20**].

## 2.12 CodeIgniter

CodeIgniter merupakan salah satu dari sekian banyak *framework* PHP yang sudah ada. CodeIgniter dikembangkan itu sendiri di kembangkan oles Rick Ellis. CodeIgniter adalah aplikasi *open source* (Gratis) berupa *framework* dengan model MVC (*Model*, *View*, *Controller*) yang di fungsikan untuk membuat sebuah *website* dinamis bebasis PHP. CodeIgniter dirilis pertama kali pada 28-02-2006. Versi stabil yang akhir ialah V 2.1.3.




**Gambar 2. 9 CodeIgniter [**21**]**

*Framework* secara sederhana dapat diartikan kumpulan dari fungsi-fungsi/prosedur-prosedur dan *class-class* untuk tujuan tertentu yang sudah siap digunakan sehingga bisa lebih mempermudah dan mempercepat pekerjaan seorang *programer*, tanpa harus membuat fungsi atau *class* dari awal.

Model MVC merupakan konsep yang cukup populer dalam pembangunan aplikasi web. MVC (*Model, View, Controller*) itu memisahkan antara logika pembuatan kode dengan pembuatan *template* website/tampilan dari web. Jika kita menggunakan *Model-View-Controller* (MVC) menjadikan pembuatan sebuah *website* akan menjadi lebih terstruktur, lebih singkat atau menyingkat koding dalam pengkodingan dan lebih sederhana. Secara sederhana konsep MVC terdiri dari 3 bagian yaitu bagian pertama yaitu *Model*, lalu *View* dan yang terakhir adalah bagian *Controller*. Di dalam sebuah web yang dinamis paling tidak terdiri dari 3 hal utama yang menyusunya, yaitu basis data, logika aplikasi & cara menampilkan halaman web. 3 hal itu direpresentasikan menggunakan MVC yaitu:

1. *Model* untuk basis data biasanya berhubungan langsung ke-database untuk memanipulasi data (*insert, update, delete, search*), menangani validasi dari *controller*, tetapi tidak *controller* itu tidak berhubungan langsung dengan bagian *view.*
2. *Controller* untuk logika aplikasi yang merupakan bagian pengatur dari bagian *model* & bagian *view, controller* memiliki fungsi sebagai penerima

*request* & data dari pengguna (*user*) kemudian menentukan apa yang akan diproses oleh aplikasi.

3. *View* merupakan bagian yang menangani proses *presentation logic*. Pada web bagian ini biasanya berupa *file template* HTML, yang diatur *controller*. Sedangkan *view* berfungsi sebagai penerima dan merepresentasikan data kepada pengguna (*user*). Nah pada bagian ini tidak memiliki hak akses langsung di bagian *model.*

## 2.13 Bootstrap

Bootstrap merupakan sebuah *framework* CSS dari twitter, yang menyediakan kumpulan komponen-komponen antarmuka dasar pada web yang telah dirancang sedemikian rupa untuk digunakan bersama-sama. Selain komponen antarmuka, Bootstrap juga menyediakan sarana untuk membangun *layout* halaman dengan mudah dan rapi, serta modifikasi pada tampilan dasar HTML untuk membuat seluruh halaman web yang dikembangkan senada dengan komponen-komponen lainnya.




**Gambar 2. 10 Bootstrap [**22**]**

Lisensi yang digunakan oleh Bootstrap yaitu lisensi Apache 2.0, sebuah lisensi yang sangat terbuka sehingga kita dapat dengan bebas dan mudah menggunakan Bootstrap, tanpa perlu khawatir akan ancaman legal dari Twitter atau pihak lainnya.

## 2.14 Metode Pengujian

Metode pengujian adalah cara atau teknik untuk menguji perangkat lunak, mempunyai mekanisme untuk menentukan data uji yang dapat menguji perangkat lunak secara lengkap dan mempunyai kemungkinan tinggi untuk menemukan kesalahan.

Perangkat lunak dapat diuji dengan dua cara, yaitu:

1. Pengujian dengan menggunakan data uji untuk menguji semua elemen program (data internal, loop, logika, keputusan dan jalur). Data uji dibangkitkan dengan mengetahui struktur internal (kode sumber) dari perangkat lunak.
2. Pengujian dilakukan dengan mengeksekusi data uji dan mengecek apakah fungsional perangkat lunak bekerja dengan baik. Data uji dibangkitkan dari spesifikasi perangkat lunak.

### 2.14.1 White-Box Testing

Pengujian white box (glass box) adalah pengujian yang didasarkan pada pengecekan terhadap detil perancangan, menggunakan struktur kontrol dari desain program secara procedural untuk membagi pengujian ke dalam beberapa kasus pengujian. Penentuan kasus uji disesuaikan dengan struktur system, pengetahuan mengenai program digunakan untuk mengidentifikasikan kasus uji tambahan.

Tujuan penggunaan white box untuk menguji semua statement program. Penggunaan metode pengujian white box dilakukan untuk:

1. Memberikan jaminan bahwa semua jalur independen suatu modul digunakan minimal satu kali.
2. Menggunakan semua keputusan logis untuk semua kondisi true atau false.
3. Mengeksekusi semua perulangan pada batasan nilai dan operasional pada setiap kondisi.
4. Menggunakan struktur data internal untuk menjamin validitas jalur keputusan.

### 2.14.2 Black-Box Testing

Pengujian black box merupakan pendekatan komplementer dari teknik white box, karena pengujian black box diharapkan mampu mengungkap kelas kesalahan yang lebih luas dibandingkan teknik white box. Pengujian black box berfokus pada pengujian persyaratan fungsional perangkat lunak, untuk mendapatkan serangkaian kondisi input yang sesuai dengan persyaratan fungsional suatu program.

Pengujian black box adalah pengujian aspek fundamental sistem tanpa memperhatikan struktur logika internal perangkat lunak. Metode ini digunakan untuk mengetahui apakah perangkat lunak berfungsi dengan benar. Pengujian black box merupakan metode perancangan data uji yang didasarkan pada spesifikasi perangkat lunak. Data uji dibangkitkan, dieksekusi pada perangkat lunak dan kemudian keluaran dari perangkat lunak dicek apakah telah sesuai dengan yang diharapkan.

Pengujian black box berusaha menemukan kesalahan dalam kategori :

1. fungsi-fungsi yang tidak benar atau hilang
2. kesalahan interface
3. kesalahan dalam struktur data atau akses database eksternal
4. kesalahan kinerja
5. inisialisasi dan kesalahan terminasi.

Berbeda dengan pengujian white box, pengujian black box cenderung diaplikasikan selama tahap akhir pengujian. Pengujian black box harus dapat menjawab pertanyaan sebagai berikut:

1. Bagaimana validitas fungsional diuji
2. Kelas input apa yang akan membuat kasus pengujian menjadi lebih baik
3. Apakah system akan sangat sensitive terhadap harga input tertentu
4. Bagaimana batasan dari suatu data diisolasi
5. Kecepatan data apa dan volume data apa yang akan ditoleransi oleh system
6. Apa pengaruh kombinasi tertentu dari data terhadap system operasi.

## 2.15 Skala Likert

Skala Likert ditemukan dan dikembangkan oleh Rensis Likert pada tahun 1932 di mana skala Likert merupakan metode skala bipolar karena dapat digunakan untuk mengukur tanggapan positif ataupun negatif. Riduwan [**23**] menyatakan bahwa skala Likert digunakan untuk mengukur sikap, pendapat dan persepsi seseorang atau sekelompik orang tentang fenomena sosial. Pada skala Likert, variable yang akan diukur dijabarkan menjadi indikator variable. Setiap jawaban dihubungkan dengan bentuk pernyataan atau dukungan sikap yang diungkapkan dengan kata-kata misalnya untuk pernyataan positif [**23**].