BAB 2
TINJAUAN PUSTAKA

2.1 Landasan Teori

Landasan teori menjelaskan beberapa teori-teori dan penjelasan yang berkaitan dengan aplikasi atau media yang akan dibangun. Landasan teori yang digunakan dalam penyusunan aplikasi rekomendasi perjalanan wisata berdasarkan anggaran berbasis *Mobile* Android, meliputi Pengertian Aplikasi, Pengertian *Mobile*, Android, GPS, LBS, CBIR, *Google Maps*, Pengertian Java, JSON, UML, *Web Service*, MySQL dan PHP.

2.1.1 Pariwisata

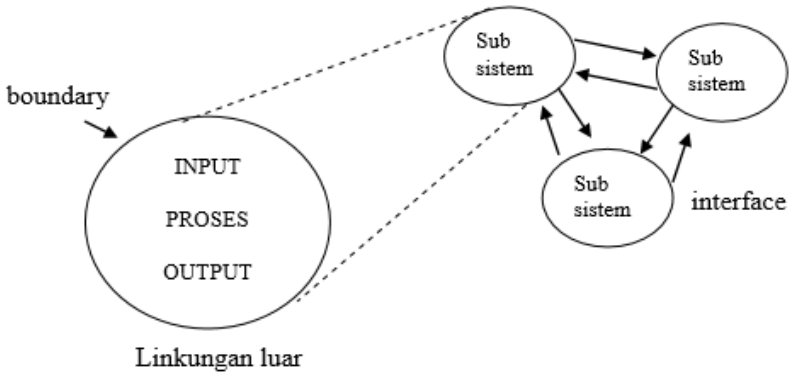
Pariwisata adalah kegiatan dinamis yang melibatkan banyak manusia serta menghidupkan berbagai bidang usaha [2]. Pariwisata secara singkat dapat dirumuskan sebagai kegiatan dalam masyarakat yang berhubungan dengan wisatawan Soekadijo [2000:2]. Pariwisata merupakan bagian yang tidak terpisahkan dari kehidupan manusian terutama menyangkut kegiatan social dan ekonomi. Diawali dari kegiatan yang semula hanya dinikmati oleh segelintir orang-orang yang relative kaya pada awal abad ke-20, kini telah menjadi bagian dari hak azasi manusia. Hal ini menjadi tidak hanya di Negara maju tetapi mulai dirasakan pula di Negara berkembang. Dapat disimpulkan bahwa pariwisata adalah kegiatan yang melakukan perjalanan ke suatu tempat untuk mencari kesenangan. [3]

2.1.2 Sistem

Sistem adalah sekumpulan unsur / elemen yang saling berkaitan dan saling mempengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan. Menurut Ludwig Von Bartalanfy, Sistem merupakan seperangkat unsur yang saling terikat dalam suatu antar relasi diantara unsur-unsur tersebut dengan lingkungan. Sementara menurut Anatol Raporot, Sistem adalah suatu kumpulan kesatuan dan perangkat hubungan satu sama lain [4].

2.1.2.1 Karakteristik Sistem

Suatu sistem memiliki karakteristik yang akan merepresentasikan ciri dari sebuah sistem itu sendiri. Berikut adalah karakteristik dari sebuah sistem :



Sumber : Materi Ilmu Komputer [5]

Gambar 2. 1 Karakteristik Sistem

1. **Komponen Sistem**

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerja sama membentuk suatu kesatuan. Komponen-komponen sistem atau elemen-elemen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem.

2. **Batasan Sistem**

   Batasan merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang suatu kesatuan. Batasan suatu sistem menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

3. **Lingkungan Luar Sistem**

   Lingkungan luar dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar sistem dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut.

4. Penghubung Sistem

Penghubung merupakan media penghubung antara satu subsistem dengan subsistem lainnya. Melalui penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem lainnya.

5. Masukan Sistem

Masukan sistem adalah energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintenance input*) dan masukan sinyal (*signal input*). Maintenance input adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi. Signal input adalah energi yang diproses untuk mendapatkan keluaran.

6. Keluaran Sistem

Keluaran sistem adalah hasil energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan.

7. Pengolah Sistem

Suatu sistem dapat mempunyai suatu bagian pengolah atau sistem itu sendiri sebagai pengolahnya. Pengolah akan mengubah masukan menjadi keluaran.

8. Sasaran Sistem

Suatu sistem mempunyai tujuan atau sasaran Kalau suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak ada gunanya. Sasaran dari sistem sangat menentukan sekali masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuannya [6].

2.1.2.2 Klasifikasi Sistem

Menurut Agus Mulyanto [2009:8] dalam bukunya yang berjudul Sistem Informasi Konsep dan Aplikasi .Sistem dapat di klasifikasikan berbagai sudut pandang, diantaranya adalah sebagai berikut :

1. Sistem Abstrak dan Fisik (*Physical System*)

Sistem Abstrak yaitu sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara Fisik, misalnya sistem agama. Sementara sistem fisik adalah sistem yang keberadaannya dapat dilihat secara fisik, misalnya perusahaan, komputer.

2. Sistem Alamiah dan Buatan (*Human Made System*)

Sistem alamiah (*natural system*) yaitu sistem yang terbentuk melalui proses alami, misalnya sistem tatasurya, pencernaan. Sementara sistem buatan manusia adalah sistem yang dirancang dan dibangun oleh manusia yang melibatkan interaksi dengan mesin misalnya produksi di pabrik.

3. Sistem Tertentu dan Tidak Tertentu (*Probabilistic System*)

Sistem tertentu (*deterministic system*) yaitu sistem yang cara beroperasinya sudah dapat diprediksi, interaksi-interaksi didalamnya dapat dideteksi dengan pasti dan outputnya dapat diramalkan, misalnya pengolahan data. Sementara sistem tak tentu adalah sistem yang outputnya tidak dapat diprediksi dengan pasti karena mengandung unsur probabilitas.

4. **Sistem Tertutup dan Terbuka (*Open System*)**

Sistem tertutup yaitu sistem yang tidak berhubungan dengan dunia luar dan tidak terpengaruh dengan lingkungan luarnya (bekerja secara otomatis). Sebenarnya sistem tertutup tidak ada yang ada adalah relatif tertutup. Sementara sistem terbuka adalah sistem yang mempunyai hubungan dengan dunia luar dan terpengaruh dengan lingkungan luarnya. Sistem ini menerima masukan dan menghasilkan ouput untuk subsistem yang lain [7].

2.2 Informasi

Informasi adalah data yang diolah menjadi suatu bentuk yang berarti bagi penerimanya dan bermanfaat dalam pengambilan keputusan pada saat sekarang atau yang akan datang. Informasi juga merupakan fakta-fakta atau data yang

telah diproses sedemikian rupa atau mengalami proses transformasi data sehingga berubah bentuk menjadi informasi [8] .

Menurut McLeod [2010:35], informasi merupakan data yang telah diproses atau memiliki arti. Adapun karakteristik penting yang harus dimiliki oleh informasi, seperti: relevansi, akurat, ketepatan waktu, dan kelengkapan. Sementara menurut Jogianto, Informasi ialah sebuah data yang telah diolah menjadi bentuk yang lebih berarti dan berguna bagi sih penerimanya untuk mengambil suatu keputusan masa kini maupun masa yang akan datang.

Informasi dibagi menjadi beberapa jenis berdasarkan fungsi, format penyajian, lokasi peristiwa dan bidang kehidupan sebagai berikut :

1. **Informasi berdasarkan fungsi**

   Jenis informasi yang satu ini ialah informasi yang berdasarkan materi dan kegunaan informasi. Informasi jenis ini antara lain yaitu suatu informasi yang menambah suatu pengetahuan dan informasi yang mengajari sih pembaca (Informasi edukatif). informasi yang menambah suatu pengetahuan, misalnya, dalam peristiwa-peristiwa bencana alam, pembangunan daerah, kegiatan selebritis, dan lain sebagainya. Informasi edukatif contohnya yaitu tips menjadi analis yang handal, tips mebersihkan karbo motor, tips berbicara di depan umum, cara jitu menjadi programmer komputer yang handal, dan lain sebagainya.

2. **Informasi berdasarkan format penyajian**

   Jenis informasi yang satu ini ialah sebuah informasi yang berdasarkan bentuk penyajian informasi. Informasi jenis ini, antara lain yaitu berupa foto, karikatur, lukisan abstrak, dan tulisan teks.

3. **Informasi berdasarkan lokasi peristiwa**

   Jenis informasi yang satu ini ialah sebuah informasi yang berdasarkan lokasi peristiwa berlangsung, yaitu sebuah informasi dari dalam negeri dan informasi dari luar negeri.

4. **Informasi berdasarkan bidang kehidupan**

Jenis informasi yang satu ini ialah sebuah informasi yang berdasarkan bidang-bidang kehidupan yang ada, misalnya pada pendidikan, olahraga, musik, sastra, budaya, dan iptek. [9]

2.3 Kualitas Sistem

Kualitas sistem informasi merupakan karakteristik dari informasi yang melekat mengenai sistem itu sendiri [DeLone dan McLean, 1992]. Pada penelitian yang dilakukan oleh istianingsih dan utami pada tahun 2009 hasil estimasinya menunjukkan bahwa kualitas sistem terbukti secara signifikan berpengaruh positif terhadap kepuasan pengguna, dan pada penelitian yang dilakukan Adrianty pada tahun 2012 juga menyatakan bahwa kualitas sistem terbukti secara signifikan berpengaruh positif terhadap kepuasan pengguna. Begitu pula pada hasil penelitian yang dilakukan oleh Iranto dan Januarti pada tahun 2012 bahwa kualitas sistem berpengaruh positif terhadap kepuasan pengguna [10]. Untuk mengetahui suatu sistem itu berkualitas atau tidak, dinilai dari indikator atau dimensi-dimensi, diantaranya :

1. **Kemudahan untuk digunakan (*ease of use*)**

Sistem informasi yang dapat dikatakan sebagai sistem yang berkualitas jika dirancang untuk memenuhi kemudahan dalam penggunaan sistem informasi tersebut. Perhatian dapat diukur berdasarkan pengguna dalam menggunakan sistem informasi tersebut yang hanya memerlukan sedikit waktu untuk mempelajari sistem informasi, hal ini dikarenakan sistem informasi tersebut sederhana, mudah dipahami, dan mudah pengoperasiannya.

2. **Kecepatan akses (*response time*)**

Kecepatan akses merupakan salah satu indikator kualitas sistem informasi. Jika sistem informasi memiliki kecepatan akses yang optimal maka layak untuk dikatakan bahwa sistem informasi yang diterapkan memilik kualitas yang baik. Kecepatan akses akan meningkatkan kepuasan

pengguna dalam menggunakan sistem informasi. Response time juga dapat dilihat dari kecepatan pengguna dalam menelusur akan informasi yang dibutuhkan.

3. Keandalan sistem (*reliability*)

Keandalan sistem informasi adalah ketahanan sistem informasi dari kerusakan dan kesalahan. Keandalan sistem informasi ini juga dapat dilihat dari sistem informasi dalam melayani kebutuhan pengguna tanpa adanya masalah yang dapat mengganggu kenyamanan pengguna dalam menggunakan sistem tersebut.

4. Fleksibilitas sistem (*flexibility*)

Fleksibilitas yang dimaksud adalah kemampuan sistem informasi dalam melakukan perubahan-perubahan yang terkait dengan memenuhi kebutuhan pengguna. Pengguna akan merasa lebih puas menggunakan sistem informasi jika sistem tersebut fleksibel dalam memenuhi kebutuhan pengguna.

5. Keamanan sistem (*security*)

Keamanan sistem dapat dilihat melalui program yang tidak dapat diubahubah oleh pengguna yang tidak bertanggung jawab dan juga program tidak dapat terhapus jika terdapat kesalahan dari pengguna. [11]

2.4 Kualitas Informasi

Kualitas informasi menunjukkan kualitas produk yang dihasilkan oleh aplikasi sistem informasinya dan informasinya akan mempunyai pengaruh pada pemakaiannya dan pada sistemnya. Kualitas informasi yang bermanfaat adalah relevan, akurat, tepat waktu, conciseness, jelas, dapat dikuantifikasi, dan konsisten. Semakin baik kualitas informasi, akan semakin tepat pula keputusan yang diambil [12]. Kualitas dari suatu informasi (*quality of information*) tergantung dari 3 hal, sebagai berikut :

1. Akurat

Akurat berarti informasi harus bebas dari kesalahan-kesalahan dan tidak biasa atau menyesatkan. Akurat juga berarti informasi harus jelas mencerminkan maksudnya. Informasi harus akurat karena dari sumber informasi sampai ke penerima informasi kemungkinan banyak terjadi gangguan (*noise*) yang dapat merubah atau merusak informasi tersebut.

2. Tepat pada waktunya

Tepat berarti informasi yang datang pada penerima tidak boleh terlambat. Informasi yang sudah usang tidak akan mempunyai nilai lagi. Karena informasi merupakan landasan didalam pengambilan keputusan. Bila pengambilan keputusan terlambat, maka dapat berakibat fatal untuk organisasi. Dewasa ini mahalnya nilai informasi disebabkan harus cepatnya informasi tersebut didapat, sehingga diperlukan teknologi - teknologi mutakhir untuk mendapatkan, mengolah dan mengirimkannya.

3. Relevan

Relevan berarti informasi tersebut mempunyai manfaat untuk pemakainya. Relevansi informasi untuk tiap-tiap orang satu dengan yang lainnya berbeda. Misalnya informasi mengenai sebab-musabab kerusakan mesin produksi kepada akuntan perusahaan adalah kurang relevan dan akan lebih relevan bila ditujukan kepada ahli teknik perusahaan. Sebaliknya informasi mengenai harga pokok produksi untuk ahli teknik merupakan informasi yang kurang relevan, tetapi relevan untuk akuntan [13]

2.5 Android

Android adalah sebuah sistem operasi untuk perangkat mobile berbasis linux yang mencakup 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 piranti lunak untuk pon sel/smartphone. Kemudian untuk mengem-bangkan Android, dibentuklah Open Handset

Alliance, konsorsium dari 34 perusahaan piranti keras, piranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia [14] .

Sistem Operasi Android memiliki beberapa komponen utama yang disebut dengan Arsitektur Platform Android. Berikut adalah diagram komponen-komponen utama dari platform Android :



Sumber Gambar :

https://developer.android.com/guide/platform/index.html?hl=id

Gambar 2. 2 Arsitektur Platform Android

Berikut adalah penjelasan dari setiap komponen utama arsitektur pada platform android pada Gambar 2.6 :

1. **Linux Kernel**

   Fondasi platform Android adalah kernel Linux. Sebagai contoh, *Android Runtime (ART)* bergantung pada kernel Linux untuk fungsionalitas dasar seperti threading dan manajemen memori tingkat

rendah. Menggunakan kernel Linux memungkinkan Android untuk memanfaatkan fitur keamanan inti dan memungkinkan produsen perangkat untuk mengembangkan driver perangkat keras untuk kernel yang cukup dikenal.

2. Hardware Abstraction Layer

Hardware Abstraction Layer (HAL) menyediakan antarmuka standar yang mengekspos kemampuan perangkat keras di perangkat ke kerangka kerja *Java API* yang lebih tinggi. *HAL* terdiri atas beberapa modul pustaka, masing-masing mengimplementasikan antarmuka untuk komponen perangkat keras tertentu, seperti modul kamera atau bluetooth. Bila *API* kerangka kerja melakukan panggilan untuk mengakses perangkat keras, sistem Android memuat modul pustaka untuk komponen perangkat keras tersebut.

3. Android Runtime

Perangkat Android yang menjalankan Android versi 5.0 (*API* level 21) atau yang lebih tinggi, setiap aplikasi menjalankan proses masing-masing dengan tahap *Android Runtime (ART)*. *ART* ditulis guna menjalankan beberapa mesin virtual pada perangkat bermemori rendah dengan mengeksekusi file *DEX*, format *bytecode* yang didesain khusus untuk Android yang dioptimalkan untuk footprint memori minimal. Beberapa fitur utama *ART* mencakup:

1. Kompilasi mendahului waktu (*AOT*) dan tepat waktu (*JIT*)
2. Pengumpulan sampah (*GC*) yang dioptimalkan
3. Dukungan *debug* yang lebih baik, mencakup profiler sampling terpisah, pengecualian diagnostik mendetail dan laporan kerusakan dan kemampuan untuk mengatur titik pantau guna memantau bidang tertentu.

Sebelum ke Android versi 5.0 (*API* level 21), *Dalvik* adalah waktu proses Android. Jika aplikasi berjalan baik pada *ART*, semestinya berfungsi baik juga pada *Dalvik*, tetapi mungkin tidak sebaliknya.

Android juga menyertakan serangkaian pustaka waktu proses inti yang menyediakan sebagian besar fungsionalitas bahasa pemrograman Java, termasuk beberapa fitur bahasa *Java 8*, yang digunakan kerangka kerja *Java API*.

4. Native C/C++ Libraries

Banyak komponen dan layanan sistem Android inti seperti *ART* dan *HAL* dibuat dari kode asli yang memerlukan pustaka asli yang tertulis dalam C dan C++. Platform Android memungkinkan kerangka kerja *Java API* mengekspos fungsionalitas beberapa pustaka asli pada aplikasi. Misalnya, Anda bisa mengakses *OpenGL ES* melalui kerangka kerja *Java OpenGL API* Android guna menambahkan dukungan untuk menggambar dan memanipulasi grafik 2D dan 3D pada aplikasi Anda.

5. Java API Framework

Keseluruhan rangkaian fitur pada Android OS tersedia untuk Anda melalui *API* yang ditulis dalam bahasa *Java*. *API* ini membentuk elemen dasar yang Anda perlukan untuk membuat aplikasi Android dengan menyederhanakan penggunaan kembali inti, komponen dan layanan sistem modular, yang menyertakan berikut ini:

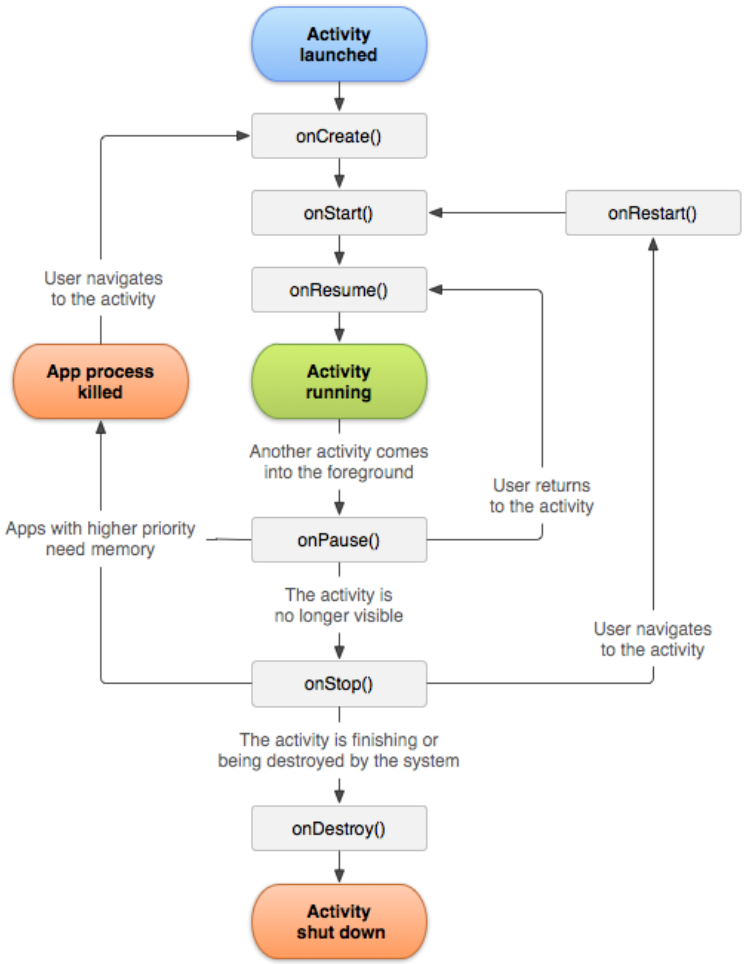
1. Tampilan Sistem yang kaya dan luas bisa Anda gunakan untuk membuat UI aplikasi, termasuk daftar, kisi, kotak teks, tombol, dan bahkan browser web yang dapat disematkan
2. Pengelola Sumber Daya, memberikan akses ke sumber daya bukan kode seperti string yang dilokalkan, grafik, dan file layout
3. Pengelola Notifikasi yang mengaktifkan semua aplikasi guna menampilkan lansiran khusus pada bilah status
4. Pengelola Aktivitas yang mengelola daur hidup aplikasi dan memberikan back-stack navigasi yang umum
5. Penyedia Materi yang memungkinkan aplikasi mengakses data dari aplikasi lainnya, seperti aplikasi Kontak, atau untuk berbagi data milik sendiri.

6. System Apps

Android dilengkapi dengan serangkaian aplikasi inti untuk email, perpesanan SMS, kalender, menjelajahi internet, kontak, dll. Aplikasi yang disertakan bersama platform tidak memiliki status khusus pada aplikasi yang ingin dipasang pengguna. Jadi, aplikasi pihak ketiga dapat menjadi browser web utama, pengolah pesan SMS atau bahkan keyboard utama (beberapa pengecualian berlaku, seperti aplikasi Settings sistem).

Aplikasi sistem berfungsi sebagai aplikasi untuk pengguna dan memberikan kemampuan kunci yang dapat diakses oleh developer dari aplikasi mereka sendiri. Misalnya, jika aplikasi Anda ingin mengirimkan pesan SMS, Anda tidak perlu membangun fungsionalitas tersebut sendiri-sebagai gantinya Anda bisa menjalankan aplikasi SMS mana saja yang telah dipasang guna mengirimkan pesan kepada penerima yang Anda tetapkan [15]

Di dalam Sistem operasi Android terdapat Siklus hidup dimana pada siklus ini untuk menavigasi antara tahap *Activity Life-Cycle,* Android itu sendiri dengan menyediakan 6 method inti *callback* yaitu *onCreate(), onStart(), onResume(), onPause(), onStop(), and onDestroy()*. Dalam siklus hidup *method callback*, kita dapat mendeklarasikan cara perilaku *activity* saat pengguna meninggalkan dan memasuki kembali ke sebuah *activity* itu. Disitulah peranan *method callback* dalam sebuah aplikasi pada platform Android . Berikut adalah Activity Life-Cycle pada platform Android:



Sumber : https://developer.android.com/guide/components/activities/activity-lifecycle.html#alc

Gambar 2. 3 Siklus Hidup Android

Keenam method callback tersebut memiliki pernananya masing-masing dan dijelaskan secara rinci sebagai berikut :

1. **OnCreate()**

Method ini adalah method utama dari setiap *Activity. Method* ini akan dipanggil pertama kali ketika menjalankan sebuah sistem. Pengembang harus mengimplementasikan metode *onCreate()* untuk menjalankan logika memulai aplikasi dasar yang hanya boleh terjadi satu kali selama hidup aktivitas. Misalnya, implementasi *onCreate()* Anda harus mendefinisikan antarmuka pengguna dan mungkin membuat instance beberapa variabel dalam cakupan-kelas.

2. **OnStart()**

Method ini dipanggil ketika method *onCreate()* telah dipanggil. *onStart()* dipanggil ketika terlihat oleh user. Method ini selesai dengan cepat dan dilanjutkan dengan method setelahnya yaitu *onResume().*

3. **OnResume()**

Method ini dipanggil ketika method *onStart()* selesai dipanggil. Method ini adalah keadaan dimana pengguna berinteraksi dengan aplikasi. Aplikasi akan tetap dalam keadaan ini sampai terjadi suatu statement dari aplikasi semisal menerima panggilan telepon atau mematikan layar *smartphone.*

4. **OnPause()**

Method ini dipanggil ketika pengguna meninggalkan *activity (*meskipun tidak selalu berarti activity dihancurkan). *Method* ini berguna untuk menghentikan sementara operasi yang sedang berjalan semisal menjeda pemutaran musik dan lain-lain.

5. **OnStop()**

Method ini dipanggil ketika *activity* tidak terlihat lagi oleh pengguna, dengan kata lain *activity* berhenti dijalankan. Hal ini dapat terjadi semisal ada aktivitas baru dijalankan meliputi seluruh layar. Sistem juga dapat menghubungi method ini ketika *activity* selesai berjalan, dan akan segera dihentikan.

6. OnDestroy()

Method ini adalah method callback ketika *activity* telah selesai dijalankan dan kemudian memanggil *method finish()* atau karena sistem untuk sementara menghancurkan proses yang berisi *activity* tersebut untuk menghemat ruang memori.

2.6 Android Studio

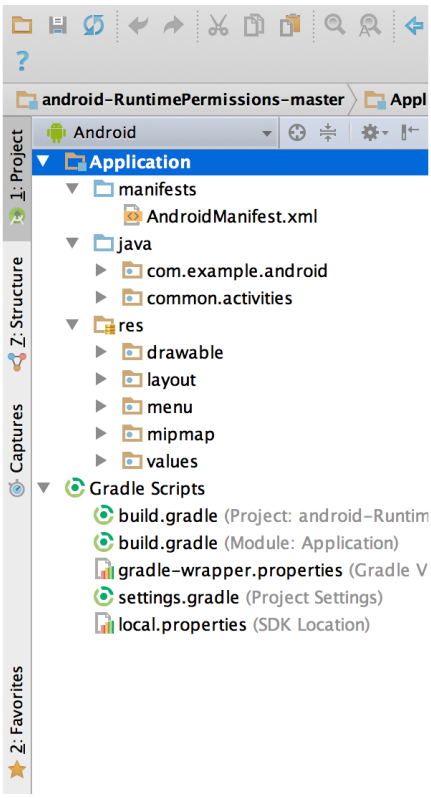
Android Studio adalah Lingkungan Pengembangan Terpadu - *Integrated Development Environment (IDE)* untuk pengembangan aplikasi Android berdasarkan IntelliJ IDEA . Selain merupakan editor kode IntelliJ dan alat pengembang yang berdaya guna, Android Studio menawarkan fitur lebih banyak untuk meningkatkan produktivitas saat membuat aplikasi Android, misalnya:

1. Sistem pembuatan berbasis *Gradle* yang fleksibel
2. Emulator yang cepat dan kaya fitur
3. Lingkungan yang menyatu untuk pengembangan bagi semua perangkat Android
4. *Instant Run* untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat *APK* baru
5. Template kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh
6. Alat penguji dan kerangka kerja yang ekstensif
7. Alat Lint untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain
8. Dukungan C++ dan NDK
9. Dukungan bawaan untuk *Google Cloud Platform*, mempermudah pengintegrasian *Google Cloud Messaging* dan *App Engine* [16]

Android Studio mempunyai struktur project yang berisi satu atau beberapa modul dengan file kode sumber dan file sumber daya. Jenis-jenis modul mencakup:

1. Modul aplikasi Android
2. Modul perpustakaan
3. Modul *Google App Engine*

Pada tampilan IDE Android studio, struktur project dapat terlihat pada bagian kiri IDE seperti Gambar berikut :



Sumber : https://developer.android.com/studio/intro/index.html?hl=id

Gambar 2. 4 Struktur Project Android Studio

Secara default, Android Studio menampilkan file proyek Anda dalam tampilan proyek Android seperti yang ditunjukkan dalam Gambar 2.4. Tampilan ini diatur menurut modul untuk memberi akses cepat ke file sumber kunci project. Semua file versi terlihat di bagian atas di bawah Gradle Scripts dan masing-masing modul aplikasi berisi folder berikut:

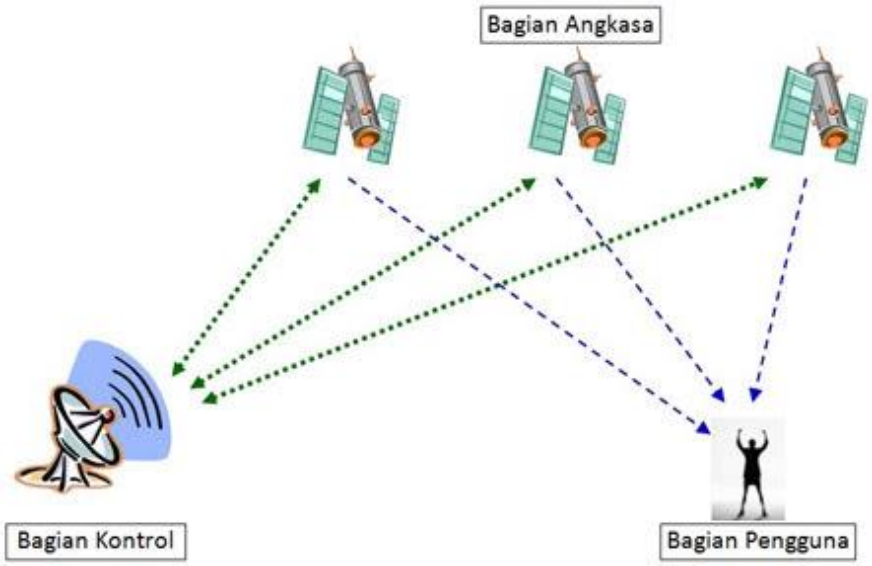
1. manifests : Berisi file *AndroidManifest.xml*.
2. java : Berisi file kode sumber *Java*, termasuk kode pengujian *JUnit*.

3. res : Berisi semua sumber daya bukan kode, seperti tata letak *XML*, string UI, dan gambar *bitmap*. [16]

2.7 GPS

Global Positioning System atau yang biasa disingkat dengan *GPS* adalah alat navigasi elektronik yang menerima informasi dari 4 - 12 satelit sehingga *GPS* bisa memperhitungkan posisi dimana kita berada di Bumi. Satelit *GPS* tidak mentransmisikan informasi posisi kita, yang ditransmisikan satelit adalah posisi satelit dan jarak penerima *GPS* kita dari satelit [17] . *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 [18].

Setiap daerah di atas permukaan bumi ini minimal terjangkau oleh 3-4 satelit. Pada prakteknya, setiap *GPS* terbaru bisa menerima sampai dengan 12 chanel satelit sekaligus. Kondisi langit yang cerah dan bebas dari halangan membuat *GPS* dapat dengan mudah menangkap sinyal yang dikirimkan oleh satelit. Semakin banyak satelit yang diterima oleh *GPS*, maka akurasi yang diberikan juga akan semakin tinggi [19] . Pada Gambar 2.9 berikut dijelaskan bagaimana cara kerja *GPS*.



Sumber : nusantride.com

Gambar 2. 5 Cara Kerja GPS

1. GPS Control Segment

Control segment *GPS* terdiri dari lima stasiun yang berada di pangkalan Falcon Air Force, Colorado Springs, Ascension Island, Hawaii, Diego Garcia dan Kwajalein. Kelima stasiun ini adalah mata dan telinga bagi *GPS*. Sinyal-sinyal dari satelit diterima oleh bagian kontrol, kemudian dikoreksi, dan dikirimkan kembali ke satelit. Data koreksi lokasi yang tepat dari satelit ini disebut data ephemeris, yang kemudian nantinya dikirimkan ke alat navigasi yang kita miliki.

2. GPS Space Segment

Space Segment adalah terdiri dari sebuah jaringan satelit yang tediri dari beberapa satelit yang berada pada orbit lingkaran yang terdekat dengan tinggi nominal sekitar 20.183 km di atas permukaan bumi. Sinyal yang dipancarkan oleh seluruh satelit tersebut dapat menembus awan, plastik dan kaca, namun tidak bisa menembus benda padat seperti tembok dan rapatnya pepohonan. Terdapat 2 jenis gelombang yang hingga saat ini digunakan sebagai alat navigasi berbasis satelit. Masing-masingnya adalah gelombang L1 dan L2, dimana L1 berjalan pada frequensi 1575.42 MHz yang bisa

digunakan oleh masyarakat umum, dan L2 berjalan pada frequensi 1227.6 Mhz dimana jenis ini hanya untuk kebutuhan militer saja.

3. GPS User Segment

User segment terdiri dari antenna dan prosesor receiver yang menyediakan positioning, kecepatan dan ketepatan waktu ke pengguna. Bagian ini menerima data dari satelit-satelit melalui sinyal radio yang dikirimkan setelah mengalami koreksi oleh stasiun pengendali (*GPS Control Segment*).

Untuk receiver *GPS* sendiri bermacam-macam jenisnya, sehingga para pengguna bisa menentukan mana yang lebih disukai. Pada saat pertama kali *GPS* ini dikenal, *GPS* tersemat pada sebuah alat khusus, namun saat ini fitur *GPS* sudah dapat ditemui di beberapa perangkat lain seperti smartphone hingga jam tangan.



Sumber : www.garmin.com

Gambar 2. 6 Macam-macam Receiver GPS

Adapun jenis-jenis dari *GPS* itu sendiri dapat secara rinci dijelaskan sebagai berikut :

1. GPS Tracker

GPS Tracker / Tracking adalah salah satu jenis *GPS* yang dipasang di mobil atau kendaraan yang difungsikan untuk mengirim data posisi kendaraan tersebut ke ponsel si pemilik kendaraan. Alat ini sangat berguna bagi pemilik usaha rental mobil atau armada kendaraan lainnya.

2. Marine GPS

GPS jenis ini telah dirancang khusus untuk digunakan di laut dan telah dilengkapi dengan plotting fungsi. *GPS* ini juga dilengkapi dengan database laut khusus dan bantuan navigasi seperti sinyal suara, pelampung, dan lain-lain. Salah satu contoh Marine *GPS* yaitu Fishfinder Garmin 350C.

3. Portable GPS

Portable *GPS* merupakan *GPS* yang mudah dibongkar pasang. Kita dapat memasangkan portable *GPS* ini pada dasbor mobil atau sepeda, bahkan bisa digunakan bagi pejalan kaki. Ketika portable *GPS* tidak sedang digunakan, kita dapat melepaskannya dari dasbor mobil untuk mengurangi resiko terhadap pencurian *GPS* mobil. yang marak terjadi saat ini.

4. Watch GPS

GPS ini biasanya digunakan untuk olahraga seperti latihan marathon, berlari, berjalan atau mendaki. *GPS* jenis ini sangat mudah disesuaikan dengan kebutuhan, selain itu kita juga dapat membaca layarnya ketika sedang beraktivitas. *GPS* berbentuk jam tangan ini juga bisa memindahkan data ke komputer atau laptop. [20]

5. Pocket GPS

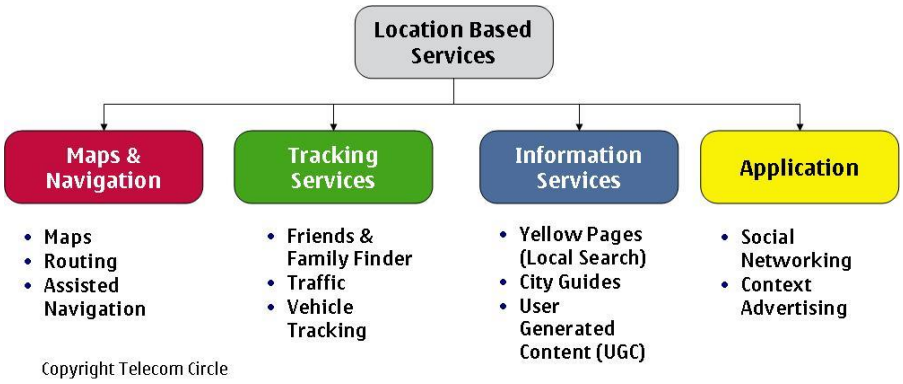
Pocket *GPS* merupakan *GPS* yang dirancang khusus untuk dapat masuk di dalam saku celana atau jaket. Beratnya hanya sekitar 5ons saja,dan dengan layar yang berukuran 3,5 inci. Berikut ini merupakan gambar dari pocket GPS.

2.8 Location Based Service

Location Based Service merupakan layanan informasi yang dapat diakses menggunakan mobiled evices, yang dilengkapi kemampuan untuk mengetahui keberadaan lokasi dari si pengguna perangkat dan kemampuan memberikan

informasi mengenai layanan yang tersedia berdasarkan lokasi mereka pada saat itu. Menurut Schiller J, *Location Based Service* dapat didefinisikan sebagai "layanan yang mengintegrasikan lokasi perangkat mobile atau posisi dengan informasi lain sehingga dapat memberikan nilai tambah bagi pengguna. "Virrantaus mendefinisikan layanan *LBS* sebagai berikut: “Informasi yang dapat diakses dengan perangkat mobile melalui jaringan layanan mobile dan memanfaatkan kemampuan untuk memanfaatkan lokasi perangkat mobile. “Open Geospatial Consortium mendefinisikan layanan LBS sebagai berikut: "Sebuah layanan nirkabel-IP yang menggunakan informasi geografis untuk melayani pengguna ponsel, terdapat layanan aplikasi yang memanfaatkan posisi terminal mobile” [21].

Layanan berbasis lokasi dapat diklasifikasikan ke dalam empat jenis yaitu *Maps and Navigation, Tracking Services, Information Services dan Application* seperti yang digambarkan dalam gambar berikut :



Sumber : http://www.telecomcircle.com/2009/06/introduction-to-lbs/

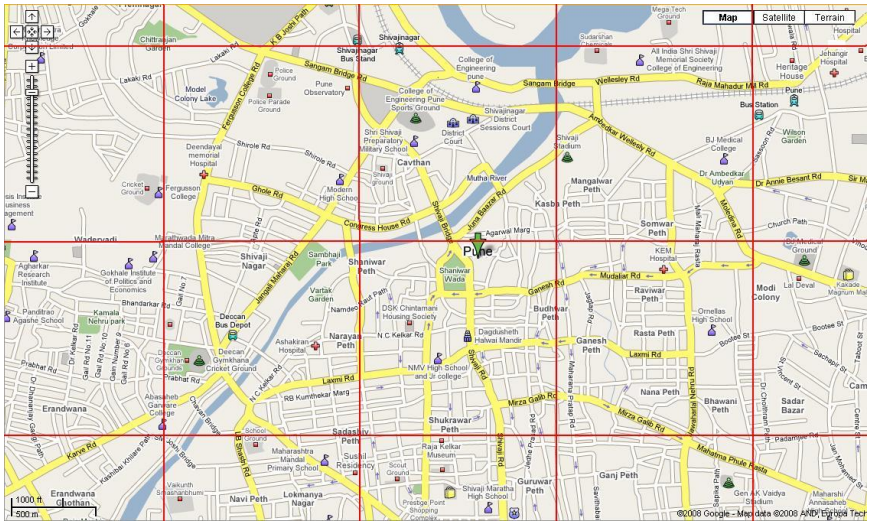
Gambar 2. 7 Klasifikasi LBS

2.9 Google Maps

Google Map Service adalah sebuah jasa peta global virtual gratis dan online yang disediakan oleh perusahaan *Google*. *Google Maps* yang dapat ditemukan di website resmi *developer Google*. *Google Maps* menawarkan peta yang dapat diseret dan gambar satelit untuk seluruh dunia. *Google Maps* juga menawarkan pencarian suatu tempat dan rute perjalanan. *Google Maps API*

adalah sebuah layanan (service) yang diberikan oleh Google kepada para pengguna untuk memanfaatkan *Google Maps* dalam mengembangkan aplikasi. *Google Maps API* menyediakan beberapa fitur untuk memanipulasi peta, dan menambah konten melalui berbagai jenis services yang dimiliki, serta mengijinkan [22] .

Google Maps dibuat dengan menggunakan kombinasi dari gambar peta, database, serta obyek obyek interaktif yang dibuat dengan bahasa pemrograman HTML, Javascript dan AJAX, serta beberapa bahasa pemrograman lainnya. Gambar-gambar yang muncul pada peta merupakan hasil komunikasi dengan database pada web server Google untuk menampilkan gabungan dari potonganpotongan gambar yang diminta. Keseluruhan citra yang ada diintegrasikan ke dalam database pada Google Server, yang nantinya akan dapat dipanggil sesuai kebutuhan permintaan. Bagian- bagian gambar map merupakan gabungan dari potongan gambar-gambar bertipe PNG yang disebut tile yang berukuran 256 x 256 pixel seperti gambar berikut [23].



Sumber : ankitjain.info

Gambar 2. 8 Google *Maps Tile*

2.10 API (*Application Programming Interface*)

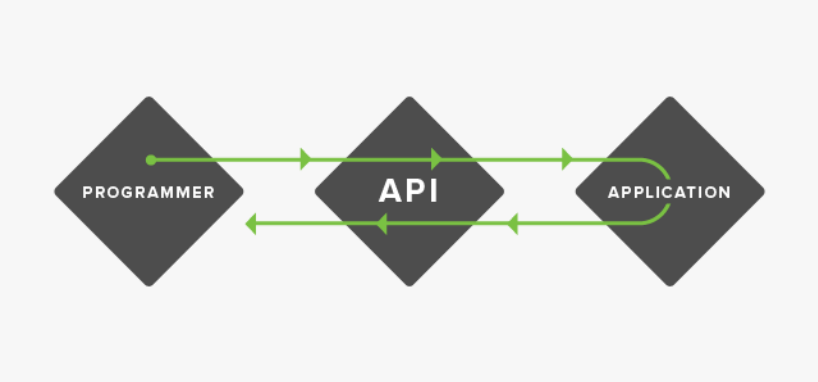
API adalah antarmuka yang digunakan untuk mengakses aplikasi atau layanan dari sebuah program. *API* memungkinkan pengembang untuk memakai fungsi yang sudah ada dari aplikasi lain sehingga tidak perlu membuat ulang dari awal. Pada konteks web, *API* merupakan pemanggilan fungsi lewat *Hyper Text Transfer Protocol (HTTP)* dan mendapatkan respon berupa *Extensible Markup Language (XML)* atau JavaScript Object Notation (JSON). Pemanggilan fungsi ke suatu situs tertentu akan menghasilkan respon yang berbeda kepada pengguna untuk membangun aplikasi enterprise di dalam websitenya [24].

Dengan API, panggilan-panggilan yang bolak-balik antaraplikasi diatur melalui web service. Web service adalah kumpulan standar teknis dan protokol, termasuk XML (Extensible Markup Language), bahasa umum yang digunakan oleh aplikasi-aplikasi tersebut selama berkomunikasi di internet. API sendiri merupakan sekumpulan kode software yang ditulis sebagai serangkaian pesan XML. Setiap pesan XML berhubungan dengan fungsi spesifik dari aplikasi yang akan diajak berkomunikasi.

Pengembang aplikasi pihak ketiga menggunakan pesan-pesan XML yang berhubungan dengan fungsi-fungsi spesifik dari layanan web yang akan diajak berkomunikasi. Pengembang bebas memilih fungsi khusus apa saja yang akan diajak berkomunikasi, dan ditampilkan pada aplikasi rancangannya.

API dan web service sepenuhnya bekerja di belakang layar. Para peselancar web dan pengguna software sama sekali tidak melihatnya. Keduanya bekerja diam-diam, menyediakan jalan bagi beberapa aplikasi untuk bekerja sama untuk memberikan informasi atau fungsionalitas yang dibutuhkan oleh pengguna aplikasi.

Dengan demikian, API adalah standar komunikasi yang dibuka oleh perusahaan software, agar dapat dimanfaatkan oleh pengembang pihak ketiga untuk mendesain aplikasi yang memanfaatkan layanan mereka dengan mudah.



Sumber : sproutsocial.com

Gambar 2. 9 Ilustrasi API

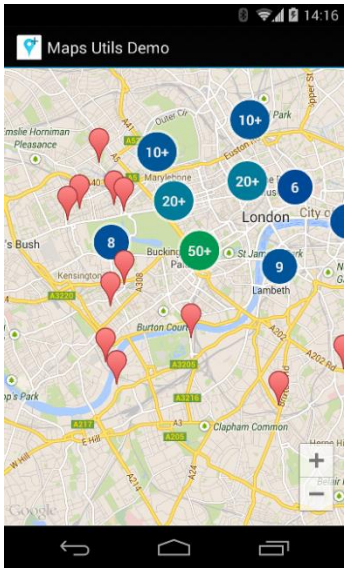
2.11 Google Maps Android API

Google Maps Android API adalah layanan untuk menampilkan peta di aplikasi android. Pengembang dapat menambahkan peta ke aplikasi berdasarkan data di *Google Maps API* secara otomatis menangani akses ke server Google Maps, mengunduh data, menampilkan peta, dan merespons gerakan peta. Anda juga bisa menggunakan panggilan *API* untuk menambahkan marker, poligon, dan overlay ke peta dasar, serta mengubah tampilan area peta tertentu ke pengguna. Semua objek ini memberikan informasi tambahan tentang lokasi peta, dan memungkinkan interaksi pengguna dengan peta. *API* memungkinkan Anda menambahkan semua gambar ini ke peta seperti :

1. Ikon yang dikaitkan dengan posisi tertentu pada peta (Marker).
2. Rangkaian segmen garis (Polyline).
3. Segmen yang disertakan (Poligon).
4. Gambar bitmap yang dikaitkan dengan posisi tertentu pada peta (Overlay Bumi).
5. Rangkaian gambar yang ditampilkan di bagian atas petak peta dasar (Overlay Petak) [26].

Untuk dapat menikmati layanan Google Maps Android API ini, terdapat langkah-langkah yang harus dilakukan yaitu :

1. Install Android Studio sebagai tools *IDE* pengembangan beserta *SDK* (*Software Development Kit*) pada Android Studio.
2. Install Google Play Services *SDK* ke dalam paket *SDK*.
3. Membuat *Activity Google Maps* di dalam *project* android.
4. Dapatkan Kunci *Google Maps API* menggunakan tautan yang terdapat pada file google_maps_api.xml pada *package value* di direktori *project.*
5. Setelah membuat kunci *Google Maps*, tempelkan kode tersebut di dalam file google_maps_api.xml.



Sumber : developer.google.com

Gambar 2. 10 Google Maps di Android

2.12 Google Maps Direction API

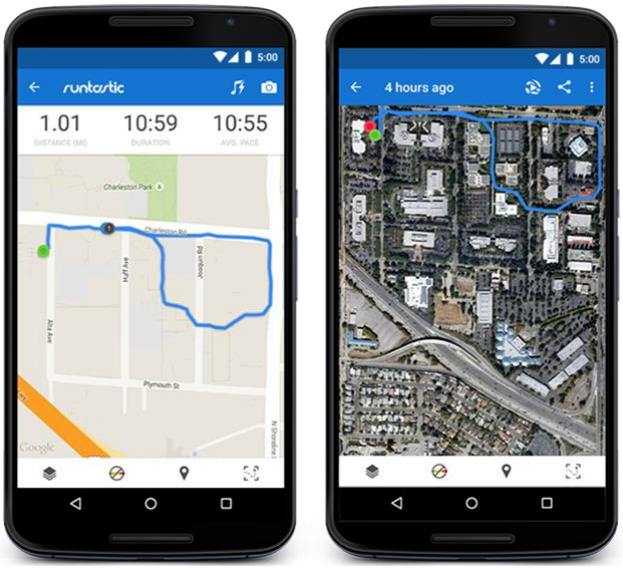
Google Maps Directions API adalah layanan yang menghitung arah antar lokasi menggunakan permintaan *HTTP*. Dengan *Google Maps Direction API* pengembang dapat membangun fitur untuk mencari arah untuk beberapa moda transportasi, termasuk angkutan umum, mengemudi, berjalan atau bersepeda. *Google Maps Directions API* bisa mengembalikan arah multi-bagian menggunakan serangkaian titik jalan [25].

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) [26].

Untuk menggunakan *Google Maps Direction API* ini, pengembang dapat memanggil URL :

```
http://maps.googleapis.com/maps/api/directions/outputFormat?parameters
```

Adapun contoh penggunaan *Google Maps Direction API* pada aplikasi smartphone Android dapat dilihat pada Gambar 2.15 berikut :



Sumber : developers.google.com/maps

Gambar 2. 11 Penggunaan Google Maps Direction API Pada Android

2.13 Google Place API

Google Place adalah layanan *developers.google.com/map* untuk menemukan suatu informasi detail tentang suatu lokasi, yang masih bisa diperinci untuk mencari informasi detail dengan berbagai kategori, mencari suatu *event*, mencari suatu image yang terkait dengan suatu lokasi, dan fasilitas autocomplete pada suatu pencarian.

Request dalam *Google Places* antara lain *Place Searches* (mencari tempat), *Place Details* (mengembalikan hasil pencarian dengan lebih detail), dan *Places Photos* (image yang terkait dengan informasi yang dicari).

Berikut kode untuk *detailplaces.php Google Place API :*

```
<?php
 $keyword =isset($_GET['keyword']) ? urlencode($_GET['keyword']) : '';
 $lat = isset($_GET['lat']) ? $_GET['lat'] : '';
 $lng = isset($_GET['lng']) ? $_GET['lng'] : '';
 $pagetoken = isset($_GET['pagetoken']) ? $_GET['pagetoken'] : '';
 if (($lat == "" || $lng == "") && $pagetoken == "")
 { die("Tentukan dulu kooordinat lokasi di <a href='places.php'>link berikut ini</a>");
 }
?>
<html>
<head>
<title>Pencarian detail lokasi</title>
</head>
<body>
<h2>Pencarian detail lokasi</h2>
<form name='Tanya' method='GET'>
Apa yang anda cari? (restaurant, bank, sekolah, dsb):
<input id="keyword" name='keyword' type='text' value='<?php echo $keyword ?>'>
<input type="hidden" name="lat" value="<?php echo $lat ?>">
<input type="hidden" name="lng" value="<?php echo $lng ?>">
<input type="submit" name="cari" value="Search">
</form>
<?php
  $url = "";
  $api_key="=== MASUKKAN API KEY GOOGLE DI SINI ===";
  if ($pagetoken != "")  // jika bukan halaman pertama
  {
$url="https://maps.googleapis.com/maps/api/place/textsearch/json?pagetoken=$pagetoken&sensor=false&key=$
api_key&";
  }
  else if ($pagetoken == "" && $keyword != "")  // halaman pertama
  {
$url="https://maps.googleapis.com/maps/api/place/textsearch/json?query=$keyword&sensor=false&key=$api_k
ey&location=$lat,$lng&radius=100";
  }
  if ($url != "")   // jika ada url yg akan dicari
  {  $result_json = json_decode(file_get_contents($url));
    echo '<div id="HasilPencarian">';
if ($result_json->status == 'OK')
{ for ($i=0; $i<sizeof($result_json->results); $i++)
  { $name = $result_json->results[$i]->name;
$address = $result_json->results[$i]->formatted_address;
    echo "$name, $address<br/>";
  }
  echo "<p><a href='javascript:back()'>Previous page</a>";
  if (isset($result_json->next_page_token))
  { $next_page = $result_json->next_page_token;
    echo "<p><a href='?pagetoken=$next_page'>Next page</a>";
  }
}
else
{ echo $result_json->status;
}
echo '</div>';
  }
?>
</body>
</html>
```

2.14 Java

Java adalah salah satu bahasa pemrograman yang paling populer. *Java* dapat digunakan untuk berbagai hal, termasuk pengembangan perangkat lunak, aplikasi mobile, dan pengembangan sistem yang besar. Inilah yang membuat bahasa pemrograman *Java* sangat terkenal di lingkungan pengembang perangkat lunak [27].

Seperti bahasa pemrograman lain, bahasa *Java* memiliki struktur sendiri, aturan sintaks, dan paradigma pemrograman. paradigma pemrograman bahasa Java didasarkan pada konsep *OOP*. Bahasa *Java* merupakan turunan bahasa C , sehingga aturan sintaks yang terlihat akan seperti bahasa C. Misalnya, blok kode yang modular dalam metode dan dibatasi oleh karakter '({' dan '})', dan variabel dideklarasikan sebelum digunakan. Secara struktural, bahasa *Java* diatur dengan *package*. Di Dalam package ada *class*, dan dalam *class* ada *method*, variabel, konstanta, dan banyak lagi [28].

Adapun sintak dalam bahasa pemrograman Java sebagai berikut :

```
public class string
{
public static void main(String[]args)
{
Scanner MyScanner=new Scanner(System.in);
String[]buah=new String[100];
String temp;
int banyak;
System.out.print("masukkan berapa banyak buah
yang mau dimasukkan =");
banyak=MyScanner.nextInt();
for(int i=0;i<banyak;i++)
{
```

2.15 PHP

PHP adalah singkatan dari PHP: *Hypertext PreProcessors*, *PHP* saat ini adalah bahasa pemrograman interpreter yang paling banyak digunakan saat ini dikarenakan bersifat *open source* dan juga paling banyak didukung oleh banyak *web server*. PHP (Hypertext Preprocessor), merupakan bahasa pemrograman pada sisi server yang memperbolehkan programmer menyisipkan perintah – perintah perangkat lunak web server (Apache, IIS, atau apapun) akan dieksekusi sebelum perintah itu dikirim oleh halaman ke browser yang me-request-nya

Sebagai bahasa pemrograman untuk tujuan umum, kode PHP diproses oleh aplikasi penerjemah dalam modus baris - baris perintah modus dan melakukan operasi yang diinginkan sesuai sistem operasi untuk menghasilkan keluaran program dichannel output standar. Hal ini juga dapat berfungsi sebagai aplikasi grafis. PHP tersedia sebagai prosesor untuk server web yang paling modern dan sebagai penerjemah mandiri pada sebagian besar system operasi dan komputer platform. [29]

Adapun contoh sintak dalam bahasa pemrograman PHP sebagai

```
<?php
      require_once "DbConnection.php";
      $link = connect();

      if (isset($_GET['limit'])) {
            $limit = $_GET['limit'];
      }else{
            $limit=10;
      }

      $sql = ("SELECT id_wisata, nama_wisata, alamat,
      rating_wisata, url_photo_wisata FROM wisata
      LIMIT $limit");

      $result = mysqli_query($link, $sql);
      while ($row = mysqli_fetch_assoc($result)) {
            $array[] = $row;
      }

      header('Content-Type:Application/json');
      echo json_encode($array);

      ?>
```

berikut :

2.16 JSON (Javascript Object Notation)

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

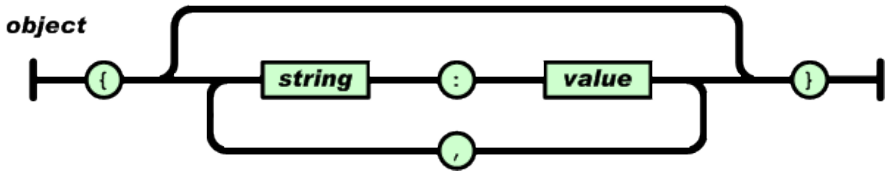
pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data. JSON terbuat dari dua struktur:

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*) [30].

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. 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).

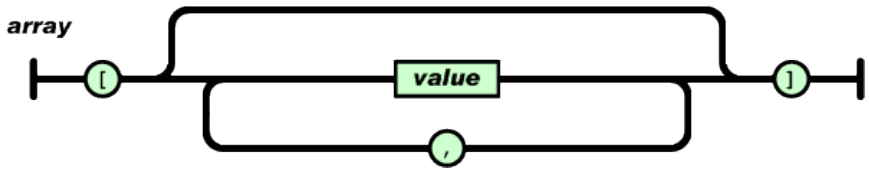


Sumber : http://www.json.org/json-id.html [31].

Gambar 2. 12 Alur Bentuk Objek

2. Array

Array adalah kumpulan nilai yang terurutkan. Larik dimulai dengan `[` (kurung kotak buka) dan diakhiri dengan `]` (kurung kotak tutup). Setiap nilai dipisahkan oleh `,` (koma).

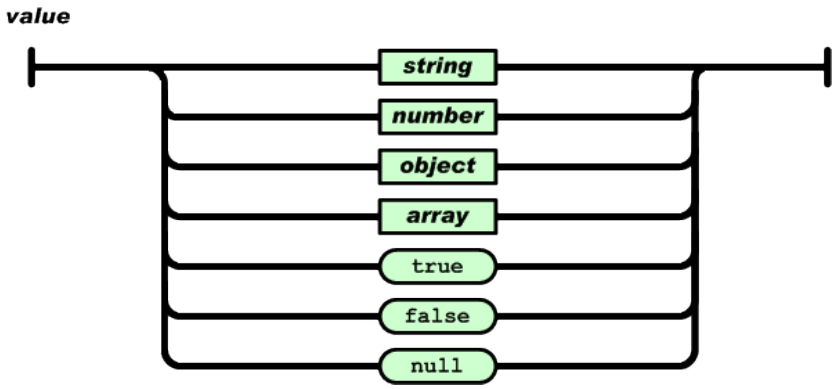


Sumber : http://www.json.org/json-id.html [32].

Gambar 2. 13 Alur Bentuk Array

3. Value

Value dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau **`true`** *atau* ***`false`*** atau ***`null`***, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat disusun bertingkat.

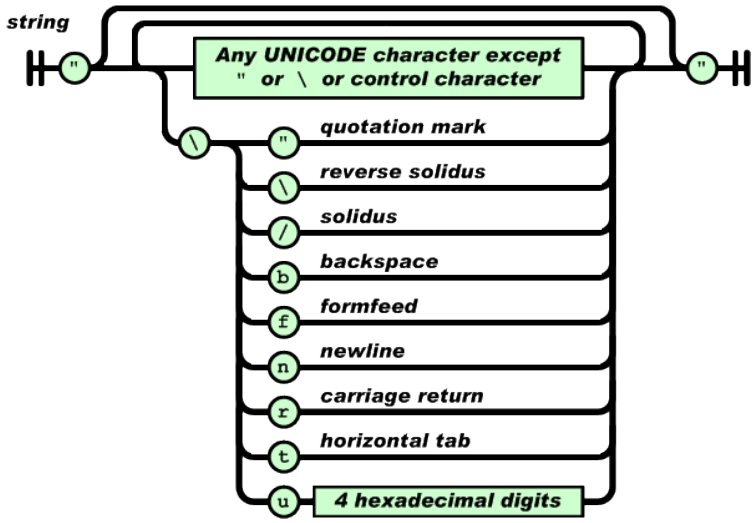


Sumber : http://www.json.org/json-id.html [33].

Gambar 2. 14 Alur Bentuk Value

4. String

String adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan *backslash escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.

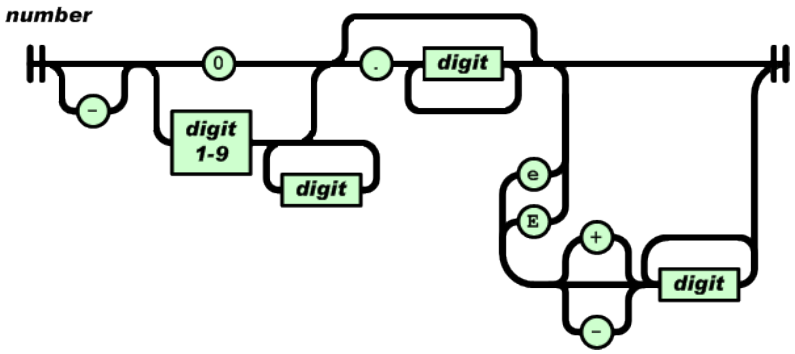


Sumber : http://www.json.org/json-id.html [34].

Gambar 2. 15 Alur Bentuk String

5. Number

Number (angka) sangat mirip dengan angka di C atau Java, kecuali format oktal dan heksadesimal tidak digunakan.



Sumber : http://www.json.org/json-id.html [35].

Gambar 2. 16 Alur Bentuk Number

2.17 Bootstrap

Bootstrap User Interface adalah sebuah *library* desain untuk membangun aplikasi web yang responsif. Boostrap adalah *framework* yang populer pada aplikasi web. Bootstrap diciptakan oleh dua orang programmer yang bekerja pada sebuah perusahaan sosial media Twitter. Sejak diluncurkan pada tahun 2011, bootsrap telah menjadi sebuah proyek yang hanya berbasis

CSS menjadi sebuah framewoek yang lengkap karena berisi css, javascript icon dan lain-lain.

Bootstrap memiliki slogan yaitu “Sleek, intuitive, and powerful front-end framework for faster and easier web development”. Slogan tersebut berarti kita dapat mendesain sebuah website dengan lebih rapi, cepat dan mudah. Selain itu Bootstrap juga responsive terhadap banyak platform, artinya tampilan halaman website yang menggunakan Bootstrap ini akan tampak tetap rapi, baik versi mobile maupun desktop. [36]

2.18 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (*Database Management System*) atau DBMS yang multithread, multi-user, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB (perusahaan asal Finlandia) membuat MySQL sebagai perangkat lunak gratis dibawah lisensi *GNU General Public License (GPL)*, tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan *GPL*. *Relational Database Management System (RDBMS)*. MySQL merupakan *Relational Database Management System (RDBMS)* yang didistribusikan secara gratis dibawah lisensi *GPL (General Public License)* dimana setiap orang bebas untuk menggunakan MySQL, namun tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu *SQL (Structured Query Language)* [37].

Dalam penggunaan database MySQL, setiap perintah yang diketikkan disebut ***query***. Perintah MySQL dapat dikategorikan menjadi 3 sub perintah, yaitu *DDL (Data Definition Language)*, *DML (Data Manipulation Language)*, dan *DCL (Data Control Language)*.

1. DDL (Data Definition Language)

Perintah dalam *SQL* yang pertama adalah perintah *DDL*. *DDL* sendiri mrupakan kependekan dari apa yang dikenal dengan nama Data Definition Language. Apabila diartikan secara harafiah, Data Definition Language berarti merupakan sebuah bahasa *SQL* yang digunakan untuk mendefinisikan suatu data. Secara teoritis, *DDL* dapat berarti sebuah perintah yang berhubungan dengan pendefinisian dari suatu struktur database. Terdapat beberapa perintah *DDL* pada MySQL sebagai berikut :

1. *CREATE* berfungsi untuk membuat database baru, tabel baru, view baru dan kolom.
2. *ALTER* berfungsi untuk mengubah struktur tabel. Seperti mengganti nama tabel, menambah kolom, mengubah kolom, menghapus kolom maupun memberikan atribut pada kolom.
3. *DROP* berfungsi untuk menghapus database dan tabel.
4. *TRUNCATE* berfungsi untuk Menghapus semua catatan dari tabel.
5. *COMMENT* berfungsi untuk Menambahkan komentar pada data.
6. *RENAME* berfungsi untuk mengubah nama obyek.

Adapun contoh sintaks dari perintah DDL pada MySQL sebagai berikut :

```
CREATE DATABASE NILAI;
ALTER TABLE Mahasiswa ADD (NoTelp char(8));
DROP INDEX nama_index    ;
TRUNCATE TABLE table_barang;
RENAME table_barang to table_barang_gudang;
```

2. DML (*Data Manipulation Language*)

Data Manipulation Language (DML) ialah perintah yang digunakan untuk mengelola/memanipulasi data dalam database. Terdapat beberapa perintah *DML* pada MySQL sebagai berikut :

1. *SELECT* berfungsi untuk mengambil/menampilkan data dari database.

2. *INSERT* berfungsi untuk memasukkan data ke dalam tabel.
3. *UPDATE* berfungsi untuk memperbarui data dalam tabel.
4. *DELETE* berfungsi untuk menghapus data dari tabel.
5. *CALL* berfungsi untuk memanggil subprogram PL / SQL atau Java
6. *EXPLAIN PLAN* berfungsi untuk menjelaskan jalur akses ke data
7. *LOCK TABLE* berfungsi untuk mengunci tabel.

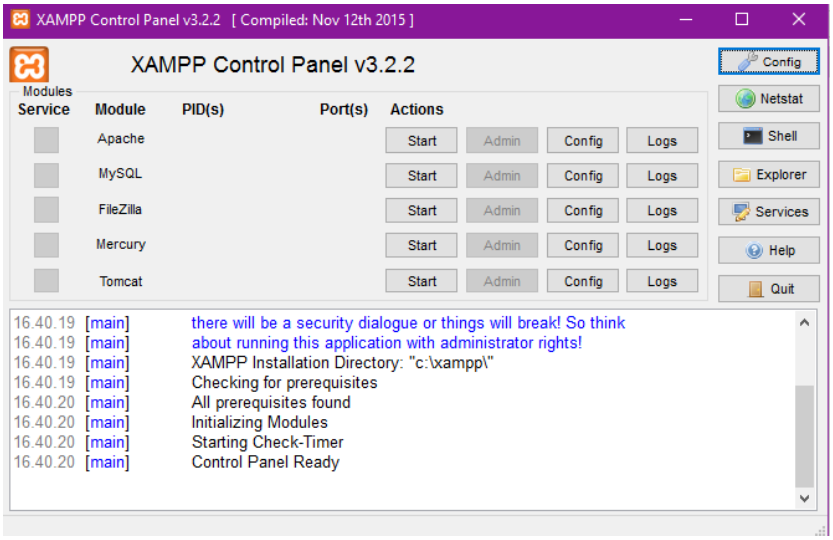
Adapun contoh sintaks dari perintah DML pada MySQL sebagai berikut :

```
INSERT INTO mahasiswa VALUES("08052926",
"Frenky","70");
SELECT nama_mahasiswa FROM mahasiswa;
DELETE FROM mahasiswa;
UPDATE mahasiswa SET nama_mahasiswa='ujang'
WHERE nim='08052926';
```

2.19 XAMPP

XAMPP adalah sebuah *software* web server apache yang didalamnya sudah tersedia database server MySQL dan dapat mendukung pemrograman PHP. XAMPP merupakan software yang mudah digunakan, gratis dan mendukung instalasi di Linux dan Windows. Keuntungan lainnya adalah cuma menginstal satu kali sudah tersedia Apache Web Server, MySQL Database Server, PHP Support (PHP 4 dan PHP 5) dan beberapa module lainnya [38].

Fungsi XAMPP sendiri adalah sebagai server yang berdiri sendiri (localhost), yang terdiri beberapa program antara lain : Apache HTTP Server, MySQL database, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan Perl. Nama XAMPP sendiri merupakan singkatan dari X (empat sistem operasi apapun), Apache, MySQL, PHP dan Perl. Program ini tersedia dalam *GNU (General Public License)* dan bebas, merupakan web server yang mudah untuk digunakan yang dapat menampilkan halaman web yang dinamis.



Gambar 2. 17 Aplikasi XAMPP

2.20 Web Server

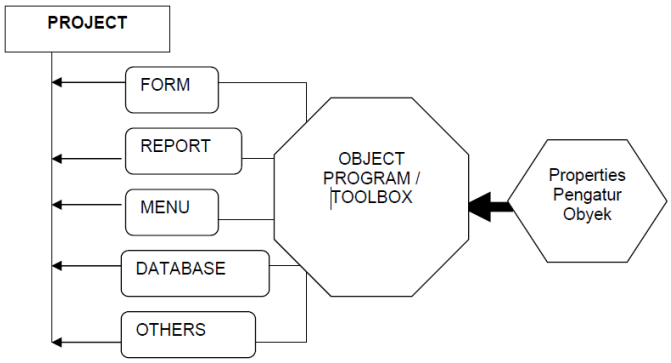
Web Server adalah sebuah perangkat lunak yang bertugas menerima permintaan client melalui port HTTP maupun HTTPS dan merubahnya ke dalam format HTML. Terdapat beberapa format selain HTML yaitu PHP atau ASP, tetapi format – format tersebut hanyalah berfungsi untuk menghubungkan HTML dengan database.

Web server yang umum diguanakan adalah *Apache*. Tugas utama dari *Web server* adalah menerjemahkan permintaan ke dalam respon yang cocok untuk keadaan pada saat itu, ketika klien membuka komunikasi dengan *Apache*, *Apache* mengirimkan permintaan untuk sumber daya. *Apache* menyediakan sumber daya yang baik atau memberikan respon alternatif untuk menjelaskan mengapa permintaan tidak dapat terpenuhi. Dalam banyak kasus, sumber daya adalah *Hypertext Markup Language (HTML)* halaman *web* yang berada pada *disk* lokal, tetapi ini hanya pilihan sederhana. Hal ini dapat berupa *file* gambar, hasil dari sebuah *script* yang menghasilkan *output* HTML, applet Java yang diunduh dan dijalankan oleh klien, dan seterusnya, *Apache* menggunakan HTTP untuk berbicara dengan klien. Permintaan / tanggapan protokol ini, yang berarti bahwa ia mendefinisikan bagaimana membuat permintaan klien dan bagaimana *server* menanggapi mereka. Setiap komunikasi HTTP dimulai dengan

permintaan dan berakhir dengan jawaban. *Executable Apache* mengambil nama dari protokol, dan pada sistem *Unix* umumnya disebut *httpd*, kependekan *daemon* HTTP. [39]

2.21 OOP (*Object Oriented Programming*)

Pemrograman berorientasi objek atau *object-oriented programming* merupakan suatu pendekatan pemrograman yang menggunakan *object* dan *class*. Saat ini konsep OOP sudah semakin berkembang. Hampir semua *programmer* maupun pengembang aplikasi menerapkan konsep OOP. OOP bukanlah sekedar cara penulisan sintaks program yang berbeda, namun lebih dari itu, OOP merupakan cara pandang dalam menganalisa sistem dan permasalahan pemrograman. Dalam OOP, setiap bagian dari program adalah *object*. Sebuah *object* mewakili suatu bagian program yang akan diselesaikan. [40]. Gambaran tentang pemrograman ini seperti dibawah ini :



Sumber :E-journal Infokam (2013) [41]

Gambar 2. 18 Model Pemrograman OOP

2.22 OOAD (Object Oriented Analysis and Design)

Analisis dan desain berorientasi objek adalah cara baru dalam memikirkan satu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Tujuan dari analisis berorientasi objek adalah untuk mengembangkan model yang menggambarkan perangkat lunak komputer

karena bekerja untuk memenuhi seperangkat persyaratan yang ditentukan *user* [37].

Keuntungan menggunakan OOAD adalah memudahkan dalam memecah sistem yang kompleks menjadi lebih kecil, modul aplikasi lebih mudah dikelola, mudah merangkai modul kembali bersama-sama untuk membentuk suatu sistem informasi, menghemat waktu, meningkatkan interaksi antara pengguna dan analis. Tools yang digunakan untuk pengembangan sistem dengan pendekataan berorientasi objek ini adalah dengan menggunakan diagram *UML*.

2.23 UML (Unified Modelling Language)

Unified Modeling Language (UML) adalah bahasa yang digunakan untuk menspesifikasikan, memvisualisasikan, dan mendokumentasikan artifak dari sebuah sistem berorientasi objek yang dalam tahap pengembangan. Merupakan penggabungan dari Booch, OMT, dan notasi Objectory, serta ide-ide terbaik dari sejumlah metodologi lainnya. Dengan menyatukan notasi-notasi yang digunakan oleh metode-metode tersebut dalam pengembangan sistem berorientasi objek, UnifiedModelingLanguage menyediakan standar defacto dalam analisis sistem berorientasi objek dan desain yang didasarkan dari pengalaman pengguna [42].

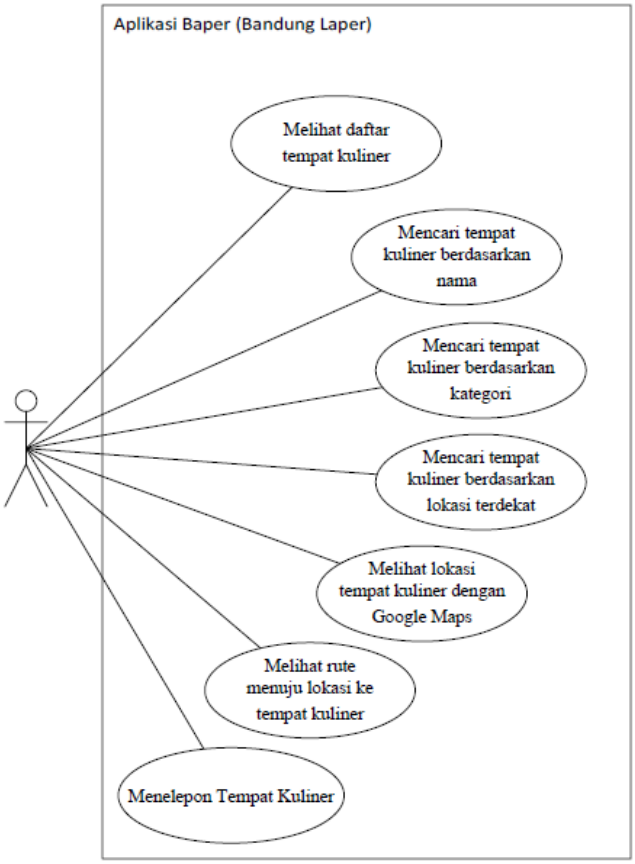
Terdapat diagram-diagram UML yang paling sering digunakan dalam memodelkan sebuah sistem berorientasi objek diantaranya :

1. **Use Case Diagram**

   Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

Use case diagram dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem [43].

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain [44]. Berikut adalah contoh *Use Case Diagram* :



Gambar 2. 19 Contoh *Use Case* Diagram

2. *Class* Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain [45].

Class memiliki tiga area pokok :

1. Nama memiliki fungsi merepresentasikan identitas dari sebuah *class.*
2. Atribut memiliki fungsi merepresentasikan atribut – atribut yang ada di dalam suatu *class*
3. Metoda memiliki fungsi merepresentasikan metode-metode yang ada pada suatu *class.*

Atribut dan metoda dapat memiliki salah satu sifat berikut :

1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
3. *Public*, dapat dipanggil oleh siapa saja

Setiap *class* pasti memiliki hubungan dengan *class* lain. Terdapat 4 cara menghubungkan satu *class* dengan *class* lain tergantung hubungan *class* – *class* tersebut. Hubungan tersebut direpresentasikan dengan sebuah panah diantara *class* tersebut [46]. Adapun 4 hubungan antar *class* pada *class diagram* sebagai berikut :

1. Asosiasi

   Asosiasi adalah hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.

2. Agregasi

   Agregasi adalah hubungan yang menyatakan bagian “terdiri atas”.

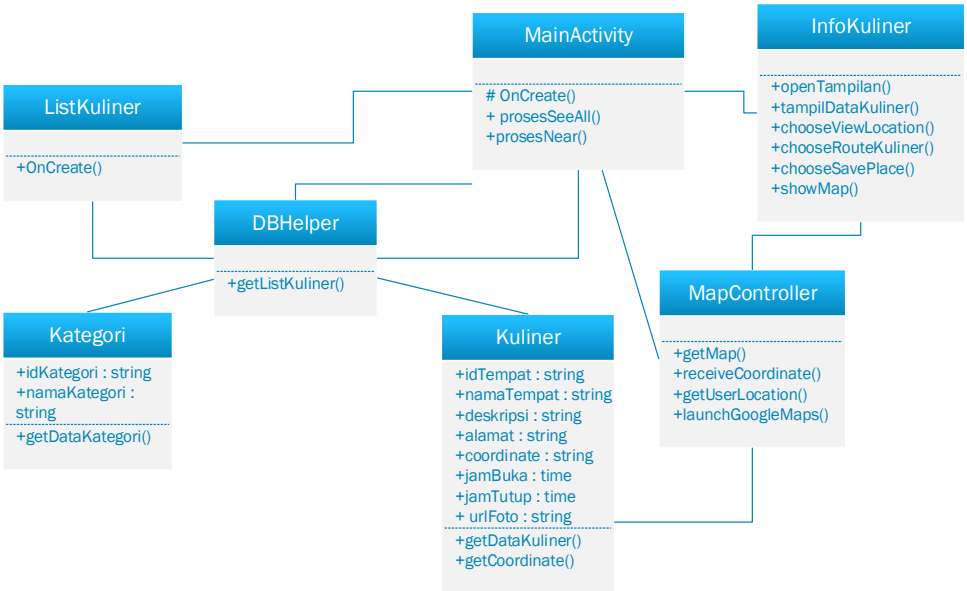
3. Pewarisan

   Pewarisan adalah hubungan hirarki antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.

4. Hubungan dinamis

   Hubungan dinamis adalah rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.

Berikut adalah contoh dari *class* diagram pada sebuah pemodelan berbasis objek :



Gambar 2. 20 Contoh *Class* Diagram

3. *Activity* Diagram

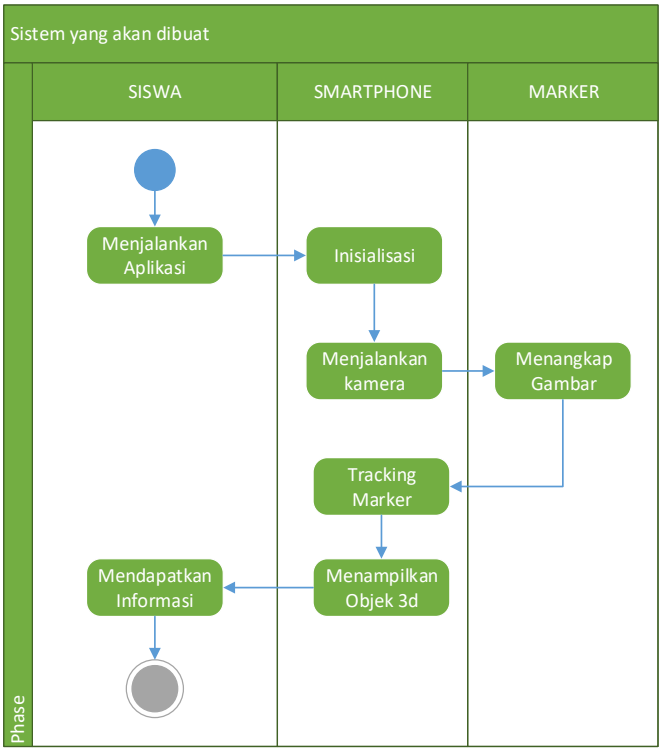
Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision*

yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi [47].

Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum[48].

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu [39]. Berikut adalah contoh dari *activity diagram :*



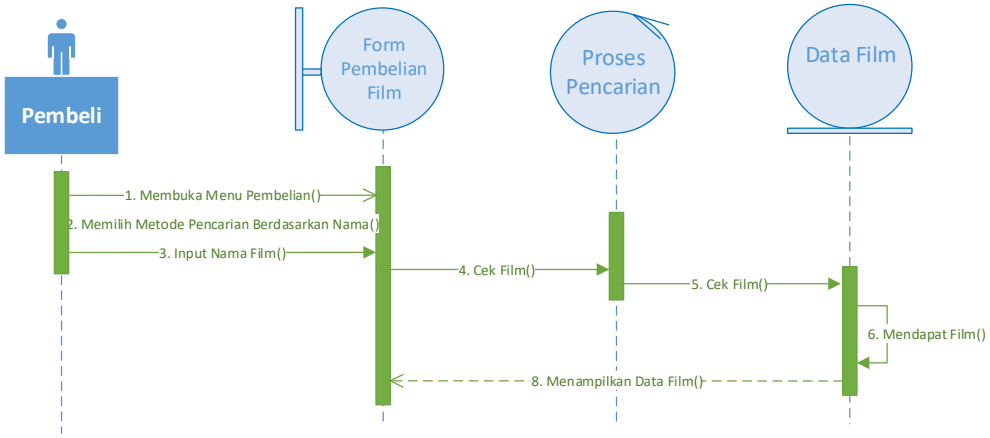
Gambar 2. 21 Contoh *Activity* Diagram

4. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait) [49].

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message. Untuk objek-objek yang memiliki sifat

khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary, controller* dan *persistent entity* [38]. Berikut adalah contoh dari *Sequence Diagram :*



Gambar 2. 22 Contoh *Sequence* Diagram

2.24 Kuesioner (Skala Likert)

Skala Likert adalah sebuah tipe skala psikometri yang menggunakan angket dan menggunakan skala yang lebih luas dalam penelitian survei. Metode *rating* yang dijumlahkan (*summated rating*) popular juga dengan nama penskalaan model Likert. Metode Likert merupakan metode penskalaan pernyataan sikap yang menggunakan distribusi respons sebagai dasar penentuan nilai skalanya. Dalam pendekatan ini tidak diperlukan adanya kelompok panel penilai (*Judging Group*) dikarenakan nilai skala setiap pernyataan tidak akan ditentukan oleh derajat favorabelnya masing-masing, akan tetapi ditentukan oleh distribusi respons setuju atau tidak setuju dari sekelompok responden yang bertindak sebagai kelompok uji coba. [50]

Kelompok uji coba ini hendaknya memiliki karakteristik yang semirip mungkin dengan karakteristik individu yang hendak diungkap sikapnya oleh skala yang sedang disusun. Di samping itu, agar hasil analisis dalam penskalaannya lebih cermat dan stabil. Responden yang digunakan sebagai kelompok uji coba harus berjumlah banyak sehingga distribusi skor mereka lebih bervariasi. Banyaknya responden dalam uji coba ini menurut saran Gable adalah

sekitar 6 sampai 10 kali lipat banyaknya pernyataan yang akan dianalisis. [51]

Dalam skala Likert terdapat lima kategori yang digunakan. Kelima kategori tersebut masuk ke dalam pernyataan positif dan negatif seperti tabel 2.1 berikut :

Tabel 2. 1 Lima Kategori Skala Liker

No	Pernyataan Positif	Pernyataan Negatif
1	Sangat Setuju	Sangat Setuju
2	Tidak Setuju	Setuju
3	Ragu-ragu	Ragu-ragu
4	Setuju	Tidak Setuju
5	Sangat Setuju	Sangat Tidak Setuju

2.25 Pengujian *Alpha*

Pengujian *alpha* dilakukan terhadap aplikasi untuk memastikan bahwa aplikasi dapat berjalan dengan benar sesuai dengan kebutuhan dan tujuan yang diharapkan [52]. Adapun 2 metode dalam pengujian *alpha* sebagai berikut :

1. White box

Merupakan metode perancangan *test case* yang menggunakan struktur kontrol dari perancangan prosedural untuk mendapatkan *test case*. Dengan menggunakan metode *white box*, analis sistem akan dapat memperoleh keuntungan uji kasus sebagai berikut :

1. Menjamin seluruh *independent path* di dalam modul yang dikerjakan sekurang-kurangnya sekali
2. Mengerjakan seluruh keputusan logikal
3. Mengerjakan seluruh *loop* yang sesuai dengan batasannya
4. Mengerjakan seluruh struktur data internal yang menjamin validitas

2. Black box

Metode uji coba *blackbox* memfokuskan pada keperluan fungsional dari *software*. Karna itu uji coba *blackbox* memungkinkan pengembang *software* untuk membuat himpunan kondisi *input* yang akan melatih seluruh syarat-syarat fungsional suatu program. Uji coba *blackbox* bukan merupakan

alternatif dari uji coba *whitebox*, tetapi merupakan pendekatan yang melengkapi untuk menemukan kesalahan lainnya, selain menggunakan metode *whitebox*.

2.26 Pengujian Beta

Pengujian Beta dilakukan secara langsung terhadap pengguna dengan menggunakan kuesioner mengenai kepuasan pengguna atas sistem yang telah dibangun. Pengguna akan menjawab setiap pertanyaan mengenai kualitas dari sebuah sistem yang dibangun. Hasil jawaban pengguna tersebut kemudian menjadi dasar pengembang aplikasi apakah sistem yang dibangun dapat diterima atau tidak hingga menjadi bahan evaluasi untuk pengembangan lebih lanjut [53].

Tes beta merupakan tahap kedua dari pengujian perangkat lunak di mana pengguna mencoba aplikasi. Awalnya, tes *alpha* berarti tahap pertama pengujian dalam proses pengembangan perangkat lunak. Tahap pertama meliputi unit testing, pengujian komponen, dan pengujian sistem. Pengujian beta dapat dianggap "pengujian pra-rilis artinya sebelum aplikasi tersebut dirilis maka harus dipastikan dari sisi pengguna bahwa perangkat lunak tersebut terbebas dari cacat atau kegagalan [54].