BAB II
LANDASAN TEORI

2.1 Android

Menurut Nazaruddin Safaat H [1] 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 peranti lunak untuk ponsel/*smartphone*. Kemudian untuk mengembangkan android dibentuklah Open Handset Aliance, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

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

Hingga awal menjelang penghujung tahun 2017 terdapat beberapa versi Android yang tersedia untuk digunakan, diantaranya adalah sebagai berikut:

Tabel 2.1 Rilis Versi Android

Nama Kode	Versi	API level
Oreo	8.0	Api level 26
Nougat	7.1	API level 25
Nougat	7.0	API level 24
Marshmallow	6.0	API level 23
Lollipop	5.1	API level 22
Lollipop	5.0	API level 21
KitKat	4.4 - 4.4.4	API level 19

Nama Kode	Versi	API level
Jelly Bean	4.3.x	API level 18
Jelly Bean	4.2.x	API level 17
Jelly Bean	4.1.x	API level 16
Ice Cream Sandwich	4.0.3 - 4.0.4	API level 15, NDK 8
Ice Cream Sandwich	4.0.1 - 4.0.2	API level 14, NDK 7
Honeycomb	3.2.x	API level 13
Honeycomb	3.1	API level 12, NDK 6
Honeycomb	3.0	API level 11
Gingerbread	2.3.3 - 2.3.7	API level 10
Gingerbread	2.3 - 2.3.2	API level 9, NDK 5
Froyo	2.2.x	API level 8, NDK 4
Eclair	2.1	API level 7, NDK 3
Eclair	2.0.1	API level 6
Eclair	2.0	API level 5
Donut	1.6	API level 4, NDK 2
Cupcake	1.5	API level 3, NDK 1
(Tidak ada nama kode)	1.1	API level 2
(Tidak ada nama kode)	1.0	API level 1

2.2 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 Anda saat membuat aplikasi Android, misalnya:

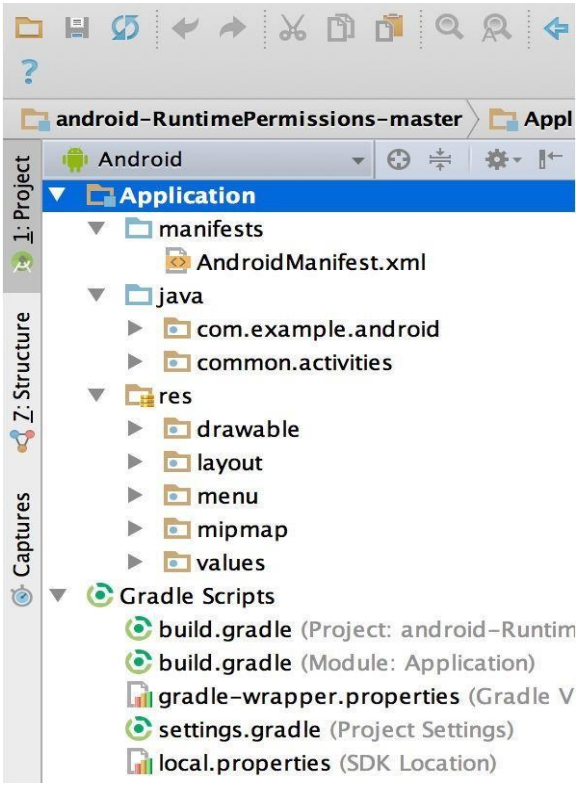
1. Sistem versi berbasis Gradle yang fleksibel.
2. *Emulator* yang cepat dan kaya fitur.

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 pengujian 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*.

Setiap proyek di Android Studio berisi satu atau beberapa modul dengan file kode sumber dan file sumber daya. Jenis-jenis modul mencakup:

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

Secara *default*, Android Studio akan menampilkan file proyek Anda dalam tampilan proyek Android, Tampilan disusun berdasarkan modul untuk memberikan akses cepat ke *file* sumber utama proyek Anda Pada Gambar 2.1.



Gambar 2. 1 File Proyek di Tampilan Android

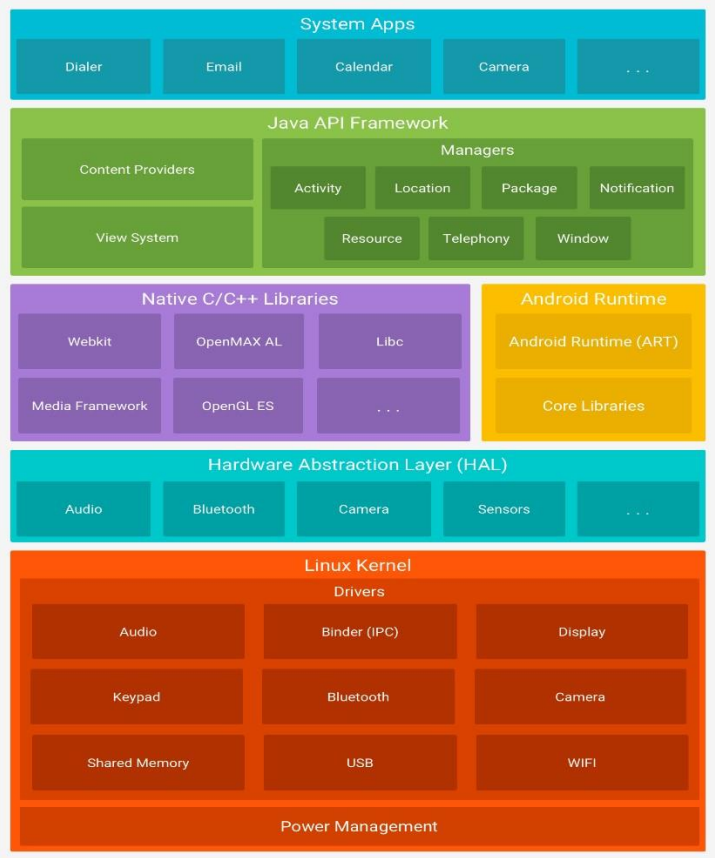
Semua file versi terlihat di bagian atas di bawah Gradle *Scripts* dan masing-masing modul aplikasi berisi folder berikut:

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.

Struktur proyek Android pada *disk* berbeda dari representasi rata ini. Untuk melihat struktur file sebenarnya dari proyek ini, pilih Project dari menu tarik turun Project (struktur ditampilkan sebagai Android) pada Gambar 2.2,[1].

2.2.1 Arsitektur Android

Secara garis besar arsitektur android dari komponen utama yang tersusun seperti pada Gambar 2.2.



Gambar 2. 2 Arsitektur *Android*

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

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].

2. *Hardware Abstaction Layer* (HAL)

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[2].

3. *Android Runtime*

Untuk perangkat 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. Buat rantai aplikasi, misalnya Jack, mengumpulkan sumber Java ke bytecode DEX, yang dapat berjalan pada *platform* Android.

Beberapa fitur utama ART mencakup:

1. Kompilasi mendahului waktu (AOT) dan waktu (JIT).
2. Pengumpulan sampah (GC) yang dioptimalkan.
3. Dukungan debug yang lebih baik, mencangkup 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 Anda 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[2].

4. *Native C/C++ Library*

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.

Jika Anda mengembangkan aplikasi yang memerlukan kode C atau C++, Anda bisa menggunakan Android NDK untuk mengakses beberapa pustaka *platform* asli langsung dari kode asli[2].

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.

Developer memiliki akses penuh ke API kerangka kerja yang sama dengan yang digunakan oleh aplikasi sistem Android.

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[2].

2.2.2 Komponen Aplikasi Android

Aplikasi android ditulis dalam bahasa pemrograman java. kode java dikompilasi bersama dengan data file yang dibutuhkan oleh aplikasi dimana prosesnya dikemas oleh tools yang dinamakan “apt tools” ke dalam paket android sehingga menghasilkan file dengan ekstensi apk. file apk itulah yang kita sebut dengan aplikasi dan nantinya dapat dipasang pada perangkat mobile [2]. terdapat empat jenis komponen pada aplikasi android yaitu:

1. *Activity*

Activity akan menyajikan *User Interface* (UI) kepada pengguna sehingga pengguna dapat melakukan interaksi. Sebuah aplikasi android bisa jadi hanya memiliki satu *activity*, tetapi umumnya aplikasi memiliki banyak *activity* yang bergantung pada tujuan aplikasi dan desain aplikasi itu sendiri. Untuk berpindah dari satu *activity* ke *activity* lain dapat dilakukan menggunakan sebuah trigger seperti klik tombol pada tampilan aplikasi.

2. *Service*

Service tidak memiliki *Graphic User Interface* (GUI), tetapi *service* berjalan secara *background*. Sebagai contoh dalam memutar musik, *service* mungkin memutar musik atau mengambil data dari jaringan, tetapi setiap *service* harus berada dalam kelas induknya. Apabila sebuah pemutar musik sedang memutar lagu dari list yang ada, aplikasi akan memiliki dua atau lebih *activity* yang memungkinkan pengguna untuk memutar sambil memilih lagu baru. Untuk menjaga musik tetap berjalan sebuah *activity* dapat menjalankan *service*.

3. *Broadcast Receiver*

Broadcast receiver berfungsi menerima dan bereaksi untuk menyampaikan notifikasi. contoh *broadcast* seperti notifikasi zona waktu telah berubah, baterai lemah, gambar berhasil diambil oleh kamera, dan lain-lain. Aplikasi juga dapat meng-inisialisasi *broadcast* misalnya memberikan informasi pada aplikasi lain bahwa ada data yang telah diunduh ke perangkat dan siap untuk digunakan.

4. *Content Provider*

Content provider membuat kumpulan aplikasi data secara spesifik sehingga bisa digunakan oleh aplikasi lain. Data disimpan dalam file sistem seperti database sqlite. *content provider* menyediakan cara untuk mengakses data yang dibutuhkan oleh suatu *activity*.

2.2.3 Android SDK (*Software Development Kit*)

Android SDK (*Software Development Kit*) adalah *tools* API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform* Android. Android SDK disediakan sebagai alat bantu untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Sebagai *platform* aplikasi-netral, Android memberi kesempatan untuk membuat aplikasi yang dibutuhkan [2].

Android SDK versi resmi dapat ditemukan dan diunduh di situs resmi Google. Google akan merilis SDK versi baru yang disesuaikan ketika Android versi terbaru juga. Untuk dapat mengembangkan aplikasi dengan fitur terbaru maka pengembang harus mengunduh Android SDK dengan versi terbaru pula yang disesuaikan dengan target *device* masing-masing.

2.3 *Bahasa Pemrograman Java*

Java merupakan sebuah Bahasa pemrograman berorientasi objek yang dapat berjalan pada *platform* yang berbeda, baik di Windows, Linux, serta system operasi lainnya. Dengan menggunakan Java kita dapat mengembangkan banyak aplikasi yang dapat digunakan pada lingkungan yang berbeda, seprti pada *Desktop, Mobile, Internet* dan lain-lain.

Untuk menginstalasi dan menggunakan Java, Sun Micro *System* selaku pengembang Java menyediakan paket instalasi sesuai dengan kebutuhan kita dalam membangun suatu aplikasi. Berikut ini uraian singkat mengenai paket aplikasi Java yang tersedia.

1. J2ME (Java 2 *Micro Edition*)

   Paket instalasi ini dapat digunakan untuk membangun *software* yang berjalan pada perangkat yang memiliki memori dan sumber daya yang kecil, seperti pada *handphone,* PDA, dan *Smartcard.*

2. J2SE (Java 2 Standard Edition)

   Paket instalasi ini digunakan untuk mengembangkan aplikasi yang berjalan pada lingkungan *workstation,* seperti aplikasi desktop.

3. J2EE (Java 2 *Enterprise Edition*)

   Paket instalasi ini dapat digunakan untuk mengembangkan aplikasi pada lingkungan internet maupun aplikasi skala *enterprise.*

Java juga merupakan bahasa pemrograman resmi yang digunakan untuk pembangunan aplikasi android yang didukung penuh oleh Google. Namun meskipun demikian saat ini java bukanlah satu-satunya Bahasa yang dapat digunakan untuk membangun aplikasi Android seperti

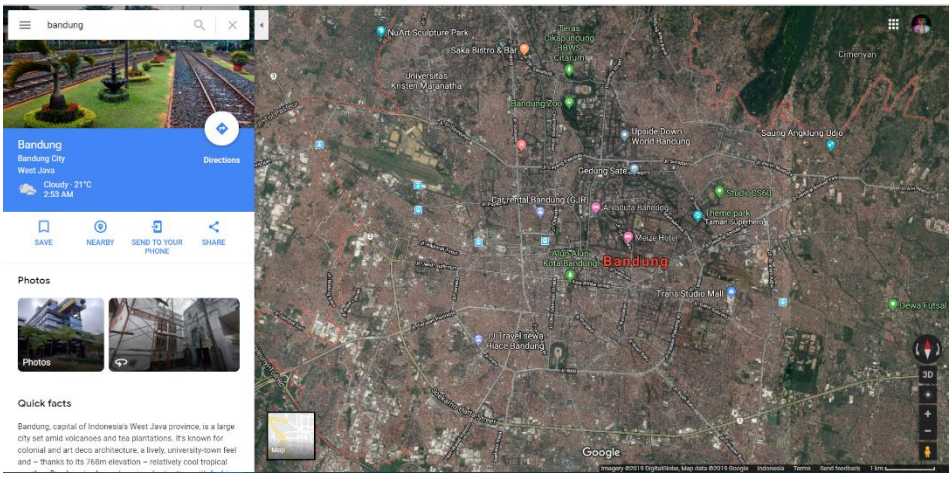
Xamarin dengan menggunakan Bahasa pemrograman C#, Cordova dengan menggunakan bahasa pemrograman web seperti HTML, CSS, dan Javascript dan lain-lain

2.4 *Google Maps*

Google Maps adalah peta online atau membuka peta secara online, dapat dilakukan secara mudah melalui layanan gratis dari Google. Bahkan layanan ini menyediakan API (*Application Programming Interface*) yang memungkinkan developer lain untuk memanfaatkan aplikasi ini di aplikasi buatannya. Tampilan *Google Maps* pun dapat dipilih, berdasarkan foto asli atau peta gambar rute saja.

Google Maps adalah layanan gratis yang diberikan oleh Google dan sangat popular. *Google Maps* adalah suatu peta dunia yang dapat kita gunakan untuk melihat suatu daerah. Dengan kata lain, *Google Maps* merupakan suatu peta yang dapat dilihat dengan menggunakan suatu browser. Kita dapat menambahkan fitur *Google Maps* dalam *web* yang telah kita buat atau pada blog kita yang berbayar maupun gratis sekalipun dengan *Google Maps* API. *Google Maps* API adalah suatu *library* yang berbentuk Java Script.

Cara membuat *Google Maps* untuk ditampilkan pada suatu *web* atau blog sangat mudah hanya dengan membutuhkan pengetahuan mengenai HTML serta Java Script, serta koneksi Internet yang sangat stabil. Dengan menggunakan *Google Maps* API, kita dapat menghemat waktu dan biaya untuk membangun aplikasi peta digital yang handal, sehingga kita dapat fokus hanya pada data-data yang akan ditampilkan. Dengan kata lain, kita hanya membuat suatu data sedangkan peta yang akan ditampilkan adalah milik Google sehingga kita tidak dipusingkan dengan mambuat peta suatu lokasi, bahkan dunia, seperti pada Gambar 2.3.



Gambar 2. 3 *Google Maps*

Pada *Google Maps* API terdapat 4 jenis pilihan model peta yang disediakan oleh Google, diantaranya adalah:

1. ROADMAP, untuk menampilkan peta biasa 2 dimensi.
2. SATELLITE, untuk menampilkan foto satelit.
3. TERRAIN, untuk menunjukkan relief fisik permukaan bumi dan menunjukkan seberapa tingginya suatu lokasi, contohnya akan menunjukkan gunung dan sungai.
4. HYBRID, akan menunjukkan foto satelit yang diatasnya tergambar pula apa yang tampil pada ROADMAP (jalan dan nama kota) [3].

2.5 *Location Based Service (LBS)*

Salah satu ciri mobile phone atau tablet pc adalah portabilitas, sehingga tidak mengherankan bahwa beberapa fitur Android sangat menarik seperti layanan yang memungkinkan anda menemukan, mencari serta menvisualisasikan posisi kita ke dalam peta lokasi fisik seperti *Google Maps*. Kita dapat membuat peta berbasis *Google Maps* dan menjadikannya sebagai element dalam layout *User Interface* (UI) yang kita rancang. Kita dapat melakukan akses penuh ke peta *Google Maps*, dan memungkinkan kita untuk mengontrol pengaturan tampilan, mengubah tampilan zoom lokasi, dan memindahkan lokasi tampilan [4].

Location Based Service adalah service yang berfungsi untuk mencari dengan teknologi *Global Positioning Service* (GPS) dan *Google cell-based* location. Map dan layanan berbasis lokasi menggunakan lintang dan bujur untuk menentukan lokasi geografis, namun sebagai user kita membutuhkan alamat atau posisi realtime kita bukan nilai lintang dan bujur. Android menyediakan geocoder yang mendukung *forward* dan *reverse* geocoding. Menggunakan geocoder, anda dapat mengkonversi nilai lintang bujur menjadi alamat dunia nyata atau sebaliknya [4]

Location based service atau layanan berbasis lokasi adalah istilah umum yang digunakan untuk menggambarkan teknologi yang digunakan untuk menemukan lokasi perangkat yang kita gunakan. Dua unsur utama LBS adalah:

1. *Location Manager* (API *Maps*)

   Menyediakan *tools/resource* untuk LBS, *Application Programming Interface* (API) Maps menyediakan fasilitas untuk menampilkan, memanipulasi maps/peta beserta feature-feature lainnya seperti tampilan satelit, street (jalan), maupun gabungannya. Paket ini berada pada com.google.android.maps[3].

2. *Location Provider* (API *Location*)

   Menyediakan teknologi pencarian lokasi yang digunakan oleh *device*/perangkat. API Location berhubungan dengan data GPS dan data lokasi realtime. API *Location* berada pada paket Android yaitu dalam paket android.*location*. Dengan *Location Manager*, kita dapat menentukan lokasi kita saat ini, Track gerakan/perpindahan, serta kedekatan dengan lokasi tertentu dengan mendeteksi perpindahan [3].

2.6 Global Positioning System (GPS)

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

1. Waktu. GPS receiver menerima informasi waktu dari jam atom yang mempunyai keakurasian sangat tinggi.
2. Lokasi. GPS memberikan informasi lokasi dalam tiga dimensi:

a) Latitude
b) Longitude
c) Elevasi

3. Kecepatan. Ketika berpindah tempat, GPS dapat menunjukkan informasi kecepatan berpindah tersebut.
4. Arah perjalanan. GPS dapat menunjukkan arah tujuan.
5. Simpan lokasi. Tempat-tempat yang sudah pernah atau ingin dikunjungi bisa disimpan oleh GPS *receiver*.

Komulasi data. GPS *receiver* dapat menyimpan informasi track, seperti total perjalanan yang sudah pernah dilakukan, kecepatan rata-rata, kecepatan paling tinggi, kecepatan paling rendah, waktu/jam sampai tujuan, dan sebagainya [3].

2.6.1 Akurasi GPS

Posisi yang ditunjukkan oleh suatu GPS mempunyai faktor kesalahan atau juga disebut tingkat akurasi. Sebagai contoh suatu alat GPS menunjukkan titik koordinat dengan tingkat akurasi 5 meter, itu berarti posisi pengguna bisa berada dalam range radius 5 meter dari titik yang ditunjukkan tersebut [3].

Ada beberapa hal yang mempengaruhi tingkat akurasi tersebut, antara lain:

1. Kesalahan Ephemeris. Terjadi jika satelit tidak dapat mentransmisikan posisinya di orbit dengan tepat.
2. Keadaan Ionosphere. Ionosphere berada pada jarak sekitar 43-50 mil di atas permukaan bumi. Satelit yang melewati ionosphere akan menjadi lambat dikarenakan adanya plasma (gas dengan tingkat kepadatan rendah). Walaupun GPS *receiver* berusaha untuk mengkoreksi/memperbaiki faktor keterlambatan yang terjadi tetap saja aktivitas tertentu dari plasma bisa menyebabkan kesalahan perhitungan.
3. Keadaan Troposphere. Troposphere adalah bagian terendah dari atmosfer sampai dengan ketinggian sekitar 11 mil dari permukaan tanah. Variasi pada temperatur, tekanan, dan kelembaban bisa menyebabkan perbedaan kecepatan penerimaan gelombang radio.
4. Kesalahan Waktu. Karena penempatan jam atom pada setiap GPS *receiver* tidak berjalan sebagaimana mestinya. Kesalahan waktu dari GPS *receiver* yang tidak presisi dapat menimbulkan ketidakakurasian.
5. Kesalahan Multipath. Terjadi karena sinyal satelit membentur permukaan keras (seperti bangunan atau tebing) sebelum mencapai GPS *receiver*. Hal tersebut bisa menyebabkan terjadinya delay sehingga perhitungan jarak menjadi tidak akurat.
6. Buruknya Sinyal Satelit. Keadaan langit yang terhalang akan menyebabkan GPS sulit menerima data satelit. Sebuah sinyal satelit yang pada hari tertentu diterima dengan sangat bagus belum tentu pada hari lain bisa diterima dengan kualitas yang sama walaupun user berdiri pada tempat yang sama. Hal tersebut dikarenakan posisi dari satelit yang terus bergerak atau bisa juga disebabkan faktor penghalang lain seperti pohon, gedung bertingkat, dan sebagainya [3].

2.7 *Latitude Longtitude*

Latitude dan longitude adalah sebuah koordinat yang ada di bumi dan digunakan untuk menentukan kedudukan kita di atas bumi.

2.7.1 Latitude

Latitude adalah garis yang melintang dari kutub utara dan kutub selatan. Titik 0 adalah sudut ekuator, tanda + menunjukan arah ke atas menuju kutub utara, sedangkan tanda minus di koordinat Latitude menuju ke kutub selatan. Titik yang dipakai dari 0 ke 90 derajat arah kutub utara, dan 0 ke -90 derajat ke kutub selatan [4].

2.7.2 *Longtitude*

Longitude adalah garis lintang. Angka dari sudut bundar bumi horisontal. Titik diawali dari 0 ke 180 derajat, dan - ke arah sebaliknya. Titik 0 dimulai dari garis negara Inggris. Mengarah ke Indonesia akan menjadi angka positif. Kebalikannya koordinat Longitude minus adalah arah kebalikan [4].

2.8 *Representational State Transfer* (REST)

REST *(REpresentational State Transfer)* merupakan standar arsitektur komunikasi berbasis web yang sering diterapkan dalam pengembangan layanan berbasis web. Umumnya menggunakan HTTP *(Hypertext Transfer Protocol)* sebagai protocol untuk komunikasi *data.* REST pertama kali diperkenalkan oleh Roy Fielding pada tahun 2000 [5].

Pada arsitektur REST, REST *server* menyediakan *resources* (sumber daya/data) dan REST *client* mengakses dan menampilkan *resource* tersebut untuk penggunaan selanjutnya. Setiap *resource* diidentifikasi oleh URIs (*Universal Resource Identifiers*) atau global ID. *Resource* tersebut direpresentasikan dalam bentuk format teks, JSON atau XML. Pada umumnya formatnya menggunakan JSON dan XML.

Keuntungan REST antara lain:

1. Bahasa dan *platform agnostic*.
2. Lebih sederhana/simpel untuk dikembangkan daripada SOAP.
3. Mudah dipelajari, tidak bergantung pada *tools*.

4. Ringkas, tidak membutuhkan *layer* pertukaran pesan (*messaging*) tambahan.
5. Secara desain dan filosofi lebih dekat dengan web.

Kelamamahan REST antara lain:

1. Mengasumsi model *point-to-point* komunikasi tidak dapat digunakan untuk lingkungan komputasi terdistribusi di mana pesan akan melalui satu atau lebih perantara.
2. Kurangnya dukungan standar untuk keamanan, kebijakan, keandalan pesan, dll, sehingga layanan yang mempunyai persyaratan lebih canggih lebih sulit untuk dikembangkan ("dipecahkan sendiri").
3. Berkaitan dengan model *transport* HTTP *(Hypertext Transfer Protocol)*.

Berikut metode HTTP *(Hypertext Transfer Protocol)* yang umum digunakan dalam arsitektur berbasis REST:

1. *GET*, menyediakan hanya akses baca pada *resource*.
2. *PUT*, digunakan untuk menciptakan *resource* baru.
3. *DELETE*, digunakan untuk menghapus *resource*.
4. *POST*, digunakan untuk memperbarui *resource* yang ada atau membuat *resource* baru.
5. *OPTIONS*, digunakan untuk mendapatkan operasi yang disupport pada *resource*.

Web service yang berbasis arsitektur REST kemudian dikenal sebagai RESTful *web services*. Layanan web ini menggunakan metode HTTP *(Hypertext Transfer Protocol)* untuk menerapkan konsep arsitektur REST.

Cara Kerja RESTful web services yaitu:

Sebuah *client* mengirimkan sebuah *data* atau *request* melalui HTTP *Request* dan kemudian *server* merespon melalui HTTP *Response*. Komponen dari HTTP *request*:

a. *Verb*, HTTP *method* yang digunakan misalnya *GET, POST, DELETE, PUT* dll.

b. *Uniform Resource Identifier* (URI) untuk mengidentifikasikan lokasi *resource* pada *server*.
c. HTTP *Version*, menunjukkan versi dari HTTP yang digunakan, contoh HTTP v1.1.
d. *Request Header*, berisi metadata untuk HTTP *Request*. Contoh, *type client/browser*, format yang didukung oleh *client*, format dari *body* pesan, seting *cache* dll.
e. *Request Body*, konten dari *data*.

Sedangkan komponen dari http *response* yaitu:

1. Status/*Response Code*, mengindikasikan status *server* terhadap *resource* yang *direquest*. misal: 404, artinya *resource* tidak ditemukan dan 200 *response OK*.
2. HTTP *Version*, menunjukkan versi dari HTTP yang digunakan, contoh HTTP v1.1.
3. *Response Header*, berisi *metadata* untuk HTTP Response. Contoh, *type server*, panjang *content*, tipe *content*, waktu *response*, dll

   Response Body, konten dari data yang diberikan.

2.9 *Java Script Object Notation* (JSON)

JavaScript Object Notation (JSON) 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 pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data [6].

JSON terbuat dari dua struktur:

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*),

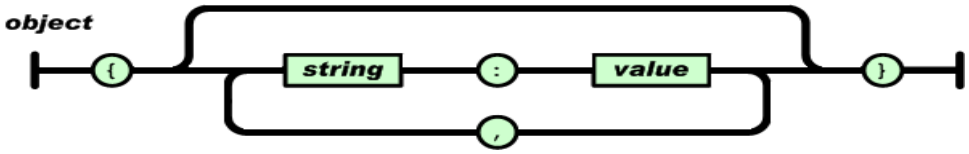
kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.

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*).

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 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) berikut pada Gambar 2.4.

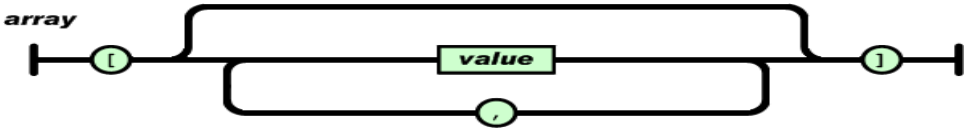


Gambar 2. 4 JSON Object

Sumber Gambar: http://www.json.org/json-id.html

2. Larik

Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma) berikut pada Gambar 2.5.

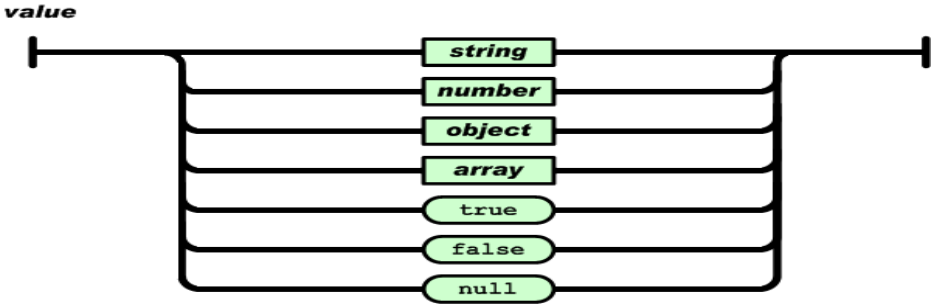


Gambar 2. 5 JSON Array

Sumber Gambar: http://www.json.org/json-id.html

3. Nilai

Nilai (*value*) dapat berupa sebuah *string* dalam tanda kutip ganda, atau angka, atau *true* atau *false* atau *null*, atau sebuah objek atau sebuah larik. Strukturstruktur tersebut dapat disusun bertingkat berikut pada Gambar 2.6.

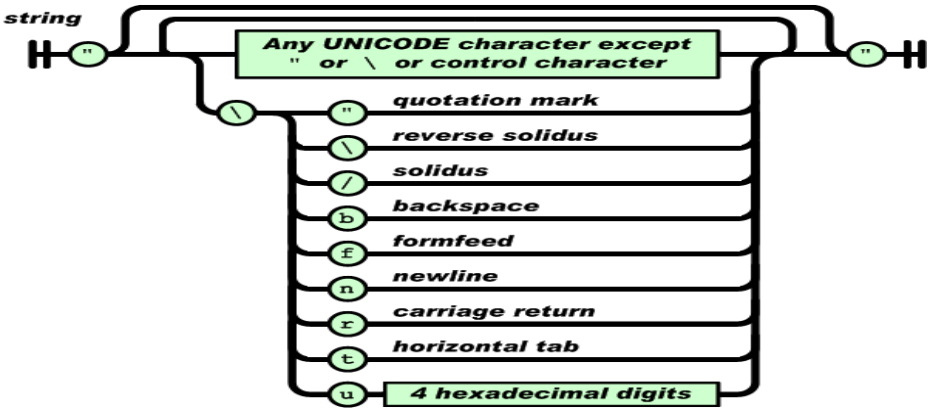


Gambar 2. 6 JSON *Value*

Sumber Gambar: http://www.json.org/json-id.html

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 berikut pada Gambar 2.7.

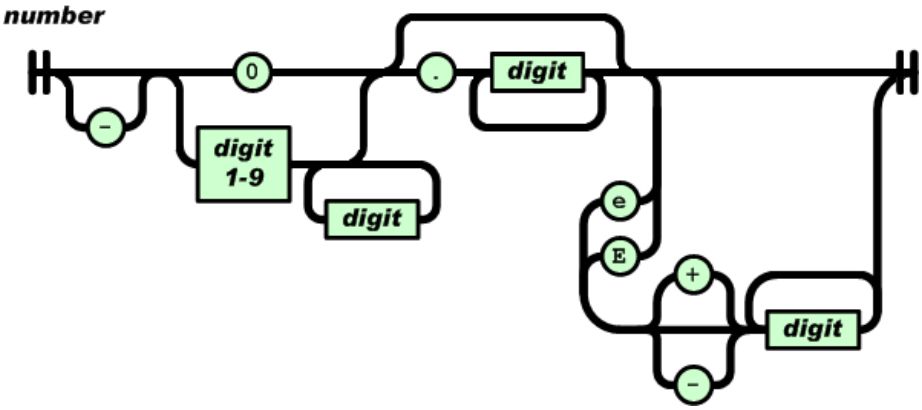


Gambar 2. 7 JSON *String*

Sumber Gambar: http://www.json.org/json-id.html

5. Angka

Angka adalah sangat mirip dengan angka di C atau Java, kecuali format oktal dan heksadesimal tidak digunakan berikut pada Gambar 2.8.



Gambar 2. 8 JSON Angka

Sumber Gambar: http://www.json.org/json-id.html

Spasi kosong (*whitespace*) dapat disisipkan di antara pasangan tanda-tanda tersebut, kecuali beberapa detil *encoding* yang secara lengkap dipaparkan oleh bahasa pemprograman yang bersangkutan.

2.10 MySQL

MySQL merupakan *software RDBMS (Relational Database Management System)* atau server *database* yang dapat mengelola *database* dengan sangat cepat, dapat menampung data dalam jumlah sangat besar, dapat diakses oleh banyak user *multi-user),* dan dapat melakukan suatu proses secara sinkron atau bersama *(multi-threaded)* [7].

Saat ini, MySQL banyak digunakan di berbagai kalangan untuk melakukan penyimpanan dan pengolahan data, mulai dari kalangan akademis sampai ke industry, baik industri kecil, menengah, maupun besar.

Lisensi MySQL terbagi menjadi dua. Anda dapat menggunakan MySQL sebagai produk *opn source* di bawah GNU *(General Public License)* secara gratis atau dapat membeli lisensi dari versi komersialnya. MySQL versi komersial tentu memiliki nilai lebih atau kemampuan-kemampuan yang tidak disematkan pada versi gratis. Pada kenyataannya, untuk keperluan industri menengah kebawah, versi gratis masih dapat digunakan dengan baik.

2.11 *Structured Query Language* (SQL)

SQL adalah pendekatan dari *Structural Query Language*, yang merupakan Bahasa atau kumpulan perintah standar yang digunakan untuk berkomunikasi dengan *database.* Perintah dalam SQL diklasifikasikan menjadi lima bagian besar [7], yaitu:

1. *Data Definition Language* (DDL)

   Data Definition Language (DDL) adalah sekumpulan perintah SQL yang berkaitan dengan pembuatan, perubahan dan penghapusan *database* prosedur/fungsi, *trigger*, dan sebagainya.

   Perintah SQL yang termasuk kategori DDL adalah:

   1. CREATE DATABASE, digunakan untuk membuat *database* dan objekobjek di dalam *database.*
   2. CREATE TABLE, digunakan untuk membuat tabel di dalam database.
   3. CREATE INDEX, digunakan untuk membuat index.
   4. CREATE VIEW, digunakan untuk membuat view.
   5. DROP DATABASE, digunakan untuk menghapus basis data.
   6. DROP TABLE, digunakan untuk menghapus tabel.
   7. DROP INDEX, digunakan untuk menghapus index.
   8. DROP VIEW, digunakan untuk menghapus view.
   9. ALTER TABLE, digunakan untuk mengubah struktur suatu tabel.

2. *Data Manipulation Language* (DML)

Data Manipulation Language (DML) adalah kumpulan perintah SQL yang berkaitan dengan data atau isi dari suatu tabel. Dengan perintah-perintah di dalam DML, kita dapat memanipulasi (menambah, mengubah, dan menghapus) data yang terdapat pada suatu tabel secara mudah. Perintah-perintah yang termasuk dalam DML adalah:

1. *INSERT*, berfungsi untuk menambah atau memasukan data baru ke dalam tabel.
2. *UPDATE*, berfungsi untuk mengubah data dalam tabel dengan nilai baru.
3. *DELETE*, berfungsi untuk menghapus data dari suatu tabel.

3. *Data Control Language* (DCL)

Data Control Language (DCL) adalah salah satu komponen SQL yang berfungsi untuk mengontrol hak akses *user*. Perintah yang termasuk ke dalam DCL adalah GRANT dan REVOKE. Berikut penjelasannya.

1. GRANT, Perintah GRANT digunakan untuk memberikan hak akses (privilege) kepada user tertentu. Melalui perintah hak akses semacam ini, seseorang user memiliki keterbatasan dalam menggunakan database sehingga data akan aman dari pihak-pihak yang tidak berkepentingan. Untuk melihat daftar hak akses yang dimiliki oleh seorang user, gunakan perintah SHOW GRANTS. Untuk mengeksekusi perintah GRANT, anda perlu memiliki hak akses GRANT OPTION yang sebelumnya diberikan oleh administrator kepada anda.
2. REVOKE, Perintah REVOKE merupakan kebalikan dari perintah GRANT, yang berfungsi untuk mencabut salah satu atau beberapa hak akses dari user tertentu di dalam database. Sama seperti perintah GRANT, untuk menjalankan perintah REVOKE anda perlu memiliki hak akses GRANT OPTION.

4. *Transaction control Language* (TCL)

Selain pengolahan hak akses user, dalam SQL kita juga dapat mengontrol transaksi data yang telah dilakukan. Perintah-perintah yang berkaitan dengan pengontrolan transaksi data digolongkan ke dalam *Transactional Control Language* (TCL). Perintah yang termasuk ke dalam TCL adalah COMMIT dan ROLLBACK. Berikut penjelasannya.

1. COMMIT, perintah COMMIT berfungsi untuk menyimpan perubahanperubahan yang dilakukan terhadap *database* (melalui perintah INSERT, UPDATE, atau DELETE) secara permanent.
2. ROLLBACK, perintah ROLLBACK merupakan kebalikan dari perintah COMMIT, yang berfungsi untuk membatalkan transaksi atau perubahanperubahan yang telah dilakukan ke dalam *database* (melalui perintanh INSERT, UPDATE, atau DELETE). Dengan melakukan pembatalan transaksi, data di dalam *database* akan kembali ke keadaan awal (keadaan sebelum dilakukan perubahan). Sehingga dengan kata lain, perubahan yang dilakukan tidak akan pernah disimpan ke dalam *database.*

2.12 M3U8

M3U (Moving Picture Experts Group Audio Layer 3 Uniform Resource Locator, URL MP3) adalah format file komputer yang menyimpan playlist multimedia. File ini pada awalnya dijalankan di Winamp, meskipun sekarang didukung oleh banyak aplikasi, termasuk VLC media player, XMMS, foobar2000, juk, RealPlayer, Windows Media Player, iTunes, QuickTime Player, Yahoo! Music mesin, JetAudio, RokuLabs SOUNDBRIDGE, Spider Player , dan PlayStation Portable. Sebuah file * M3u adalah playlist metafile yang merujuk file .mp3 dan menyediakan metadata tambahan untuk item dalam daftar putar. Setiap lagu dan lokasinya memiliki jalur sendiri. Lokasi dapat berupa nama path lokal absolut atau relatif (misalnya, "C: \ My Music \ Bento.mp3" atau "Bento.mp3") atau bisa juga menjdi URL. File ini

juga dapat mencakup komentar, diawali oleh karakter "#" . Dalam ekstensi M3U, karakter "#" juga memperkenalkan acuan extensi file M3U.Salah satu penggunaan umum dari format file M3U adalah menciptakan sebuah file playlist yang berisi entri tunggal untuk aliran di Internet. File ini dibuat untuk mempermudah akses streaming dan dapat digunakan untuk hal-hal seperti men-download dari situs web atau email, atau untuk mendengarkan radio internet. Versi unicode format M3U format M3U8, dapat mencakup UTF-8 karakter unicode. [8].

2.13 *Personal Home Page* (PHP)

PHP Pertama kali ditemukan pada 1995 oleh seorang *Software Developer* bernama Rasmus Lerdrof. Ide awal PHP adalah ketika itu Rasmus ingin mengetahui jumlah pengunjung yang membaca resume onlinenya. Script yang dikembangkan baru dapat melakukan dua pekerjaan, yakni merekam informasi *visitor*, dan menampilkan jumlah pengunjung dari suatu *website*. Dan sampai sekarang kedua tugas tersebut masih tetap populer digunakan oleh dunia *web* saat ini. Kemudian, dari situ banyak orang di milis mendiskusikan script buatan Rasmus Lerdrof, hingga akhirnya rasmus mulai membuat sebuah *tool/script*, bernama *Personal Home Page* (PHP) [9].

Kebutuhan PHP sebagai *tool* yang serba guna membuat Lerdorf melanjutkan untuk mengembangkan PHP hingga menjadi suatu bahasa tersendiri yang mungkin dapat mengkonversikan data yang di masukkan melalui *Form* HTML menjadi suatu variabel, yang dapat dimanfaatkan oleh sistem lainnya. Untuk merealisasikannya, akhirnya Lerdrof mencoba mengembangkan PHP menggunakan bahasa C ketimbang menggunakan Pe*rl. Tahun 1997, PHP versi 2.0 di rilis, dengan *nama Personal Home Page Form Interpreter* (PHP-FI). PHP Semakin popular, dan semakin diminati oleh programmer web dunia.

Pengembangan demi pengembangan terus berlanjut, ratusan fungsi ditambahkan sebagai fitur dari bahasa PHP, dan di awal tahun

1999, netcraft mencatat, ditemukan 1.000.000 situs di dunia telah menggunakan PHP. Ini membuktikan bahwa PHP merupakan bahasa yang paling populer digunakan oleh dunia *web development*. Hal ini mengagetkan para developernya termasuk Rasmus sendiri, dan tentunya sangat diluar dugaan sang pembuatnya. Kemudian Zeev Suraski dan Andi Gutsman selaku *core developer* (programmer inti) mencoba untuk menulis ulang PHP Parser, dan diintegrasikan dengan menggunakan Zend scripting engine, dan mengubah jalan alur operasi PHP. Dan semua fitur baru tersebut di rilis dalam PHP 4.

13 Juli 2004, evolusi PHP, PHP telah mengalami banyak sekali perbaikan di segala sisi, dan wajar jika netcraft mengumumkan PHP sebagai bahasa web populer di dunia, karena tercatat 19 juta domain telah menggunakan PHP sebagai *server side* scriptingnya. PHP saat ini telah Mendukung XML dan Web Services, Mendukung SQLite. Tercatat lebih dari 19 juta domain telah menggunakan PHP sebagai *server* scriptingnya. Benarbenar PHP sangat mengejutkan.

Yang menjadikan PHP berbeda dengan HTML adalah proses dari PHP itu sendiri. HTML merupakan bahasa statis yang apabila kita ingin merubah konten/isinya maka yang harus dilakukan pertama kali nya adalah, membuka filenya terlebih dahulu, kemudian menambahkan isi kedalam file tersebut. Beda hal nya dengan PHP. Bagi anda yang pernah menggunakan CMS seperti wordpress atau joomla yang dibangun dengan PHP tentunya, ketika akan menambahkan konten kedalam website, anda tinggal masuk kedalam halaman admin, kemudian pilih *new artikel* untuk membuat halaman/*content* baru. Artinya hal ini, seorang user tidak berhubungan langsung dengan scriptnya. Sehingga seorang pemula sekalipun dapat menggunakan aplikasi seperti itu.

Keunggulan PHP antara lain:

a. Gratis

Apa yang membuat PHP begitu berkembang sangat pesat hingga jutaan domain menggunakan PHP, begitu populernya PHP? Jawabannya adalah karena PHP itu gratis.

b. *Cross Platform*

Artinya dapat di gunakan di berbagai sistem operasi, mulai dari linux, windows, mac os dan os yang lain.

c. Mendukung banyak *database*

PHP telah mendukung banyak *database*, ini mengapa banyak *developer web* menggunakan PHP Adabas D Adabas D, dBase dBase, Empress Empress, FilePro (*read-only*) FilePro (*read-only*) Hyperwave, IBM DB2, Informix, Ingres, InterBase, FrontBase mSQL, Direct MS-SQL, MySQL MySQL, ODBC, Oracle (OCI7 and OCI8), Ovrimos, PostgreSQL SQLite, Solid, Sybase, Velocis, Unix dbm.

d. On The Fly

PHP sudah mendukung *on the fly*, artinya dengan php anda dapat membuat *document text*, Word, Excel, PDF, menciptakan *image* dan *flash*, juga menciptakan file-file seperti zip, XML, dan banyak lagi.

2.14 Formulla Haversine

Metode Haversine digunakan untuk menghitung jarak antara titik di permukaan bumi menggunakan garis lintang (longitude) dan garis bujur (lattitude) sebagai variabel inputan. Haversine formula adalah persamaan penting pada navigasi, memberikan jarak lingkaran besar antara dua titik pada permukaan bola (bumi) berdasarkan bujur dan lintang. Dengan mengasumsikan bahwa bumi berbentuk bulat sempurna dengan jari-jari R 6.367, 45 km, dan lokasi dari 2 titik di koordinant bola (lintang dan bujur) masing-masing adalah lon1, lat1, dan lon2, lat2, maka rumus Haversine dapat ditulis dengan persamaan sebagai berikut [10].

Rumus Haversine

x = (lon2-lon1) * cos ((lat1+lat2)/2);

y= (lat2-lat1); d= sqrt(x*x+y*y)*R

Keterangan: x = Longitude (Lintang)

y= Lattitude (Bujur)

d= Jarak

R= Radius

Bumi =6371 km 1

derajat= 0.0174532925 radian

2.15 *Unified Modeling Language* (UML)

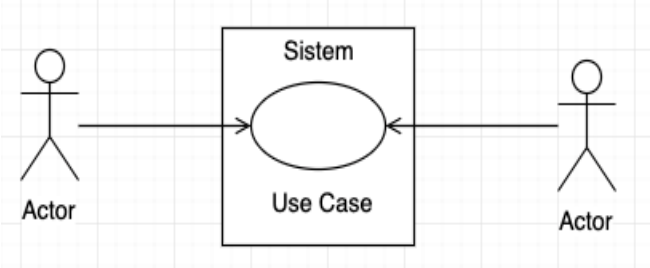
Unified Modeling Language (UML) adalah Bahasa permodelan standar pada rekayasa perangkat lunak. Dengan menggunakan UML akan berdampak kepada peningkatan produktifitas dan kualitas serta pengurangan biaya dan waktu. Kerumitan arsitektur dalam pengembangan perangkat lunak bisa diatasi dengan menggambarkan cetak biru sistem tersebut [11]

2.15.1 Diagram UML

UML menyediakan 4 macam diagram untuk memodelkan aplikasi perangkat lunak berorientasi objek [11], yaitu:

1. Use Case Diagram

Use case diagram digunakan untuk menangkap aspek dinamis dari sistem. Secara lebih spesifik, use case diagram digunakan untuk mengumpulkan kebutuhan dari sebuah sistem baik karena pengaruh internal maupun eksternal. Gambar 2.10 menunjukan *Use Case Diagram* dalam UML.

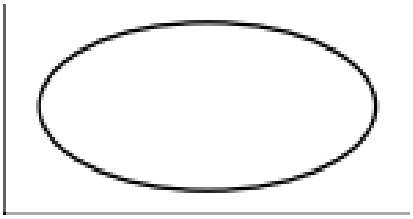


Gambar 2. 9 Use Case Diagram [9]

Berikut ini adalah bagian dari sebuah use case diagram:

a. Use Case

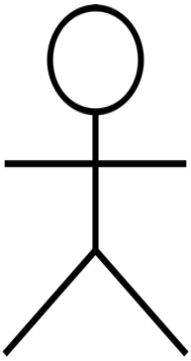
Use case adalah deskripsi fungsi dari sebuah system dari prespektif pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah system dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Gambar 2.10 menunjukan bentuk *Use Case* dalam UML.



Gambar 2. 10 Use Case [10]

b. *Actors*

Actors adalah *abstraction* dari orang dan sistem yang lain yang mengaktifkan fungsi dari target sistem. Orang atau sistem bisa muncukl dalam beberapa peran. Gambar 2.11 menunjukan bentuk *actors* dalam UML.



Gambar 2. 11 Actors [10]

c. *Relationship*

Relationship adalah hubungan antara *use cases* dengan *actors*. *Relationship* dalam *use case* diagram meliputi:

a. Asosiasi antara *actor* dan *use case*.

Hubungan antara *actor* dan *use case* yang terjadi karena adanya interaksi antara kedua belah pihak. Asosiasi tipe ini menggunakan garis lurus dari actor menuju *use case* baik dengan menggunakan mata panah terbuka ataupun tidak.

b. Asosiasi antara 2 *use case*

Hubungan antara *use case* yang satu dan *use case* lainnya yang terjadi karena adanya interaksi antara kedua belah pihak. Asosiasi tipe ini menggunakan garis putus-putus/garis lurus dengan mata panah terbuka di ujungnya.

c. Generalisasi antara 2 *actor*

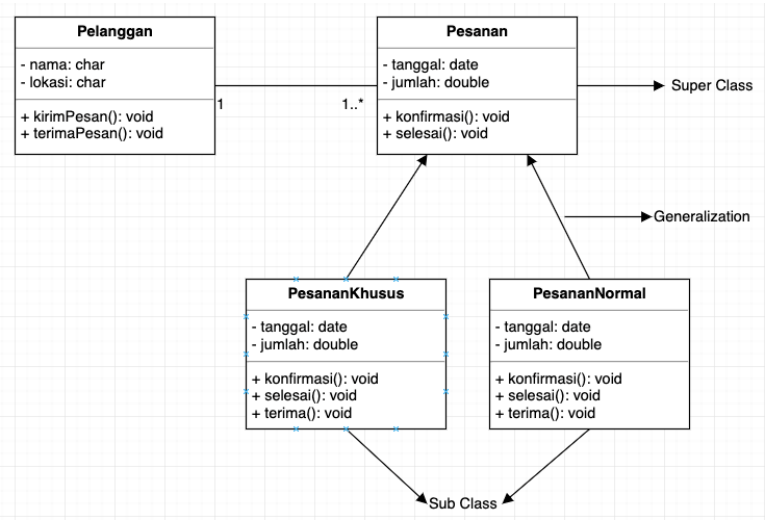
Hubungan *inheritance* (pewarisan) yang melibatkan *actor* yang satu (*the child*) dengan *actor* lainnya (*the parent*). Generalisasi tipe ini menggunakan garis lurus dengan mata panah tertutup di ujungnya.

d. Generalisasi antara 2 *use case*.

Hubungan *inheritance* (pewarisan) yang melibatkan *use case* yang satu (*the child*) dengan use case lainnya (*the parent*). Generalisasi tipe ini menggunakan garis lurus dengan mata panah tertutup di ujungnya.

2. *Class Diagram*

Class diagram adalah diagram statis. Ini mewakili pandangan statis dari suatu aplikasi. Class diagram tidak hanya digunakan untuk memvisualisasikan, menggambarkan, dan mendokumentasikan berbagai aspek sistem tetapi juga membangun kode eksekusi dari aplikasi perangkat lunak. Gambar 2.12 menunjukan *Class Diagram* dalam UML.



Gambar 2. 12 Class Diagram [10]

Class diagram mempunyai 3 relasi dalam penggunaanya, yaitu:

a. *Assosiation*

Assosiation adalah sebuah hubungan yang menunjukkan adanya interaksi antar *class*. Hubungan ini dapat ditunjukkan dengan garis dengan mata panah terbuka di ujungnya yang mengindikasikan adanya aliran pesan dalam satu arah.

b. *Generalization*

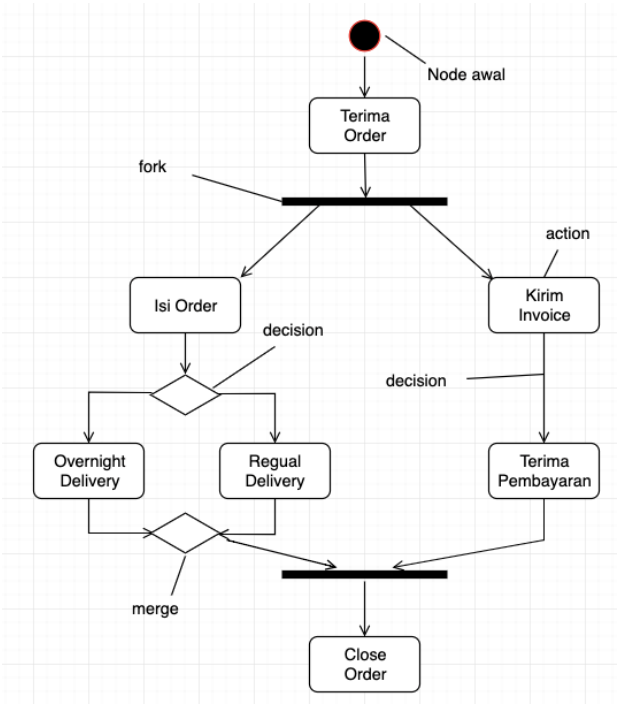
Generalization adalah sebuah hubungan antar *class* yang bersifat dari khusus ke umum

c. *Constraint*

Constraint adalah sebuah hubungan yang digunakan dalam sistem untuk memberi batasan pada sistem sehingga didapat aspek yang tidak fungsional.

3. *Activity Diagram*

Activity diagram adalah bagian penting dari UML yang menggambarkan aspek dinamis dari sistem. Logika procedural, proses bisnis dan aliran kerja suatu bisnis bisa dengan mudah dideskripsikan dalam *activity diagram*. Gambar 2.14 menunjukan *Activity Diagram* dalam UML.



Gambar 2. 13 Activity Diagram [10]

Berikut ini merupakan komponen dalam activity diagram, yaitu:

a. *Activity node*

Activity node menggambarkan bentuk notasi dari beberapa proses yang beroperasi dalam kontrol dan nilai data

b. *Activity edge*

Activity edge menggambarkan bentuk *edge* yang menghubungkan aliran aksi secara langsung dimana menghubungkan *input* dan *output* dari aksi tersebut

c. *Initial state*

Bentuk lingkaran berisi penuh melambangkan awal dari suatu proses.

d. *Decision*

Bentuk wajib dengan suatu *flow* yang masuk beserta dua atau lebih *activity node* yang keluar. *Activity node* yang keluar ditandai untuk mengindikasikan beberapa kondisi.

e. *Fork*

Satu bar hitam dengan satu *activity node* yang masuk beserta dua atau lebih *activity node* yang keluar.

f. *Join*

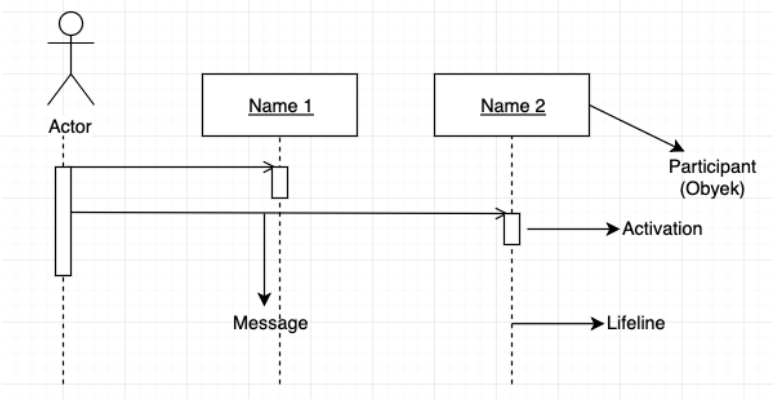
Satu bar hitam dengan dua atau lebih *activity node* yang masuk beserta satu *activity node* yang keluar, tercatat pada akhir dari proses secara bersamaan. Semua *actions* yang menuju *join* harus lengkap sebelum proses dapat berlanjut.

g. *Final state*

Bentuk lingkaran berisi penuh yang berada di dalam lingkaran kosong, menunjukkan akhir dari suatu proses.

4. *Sequence Diagram*

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah *scenario*. Diagram ini menunjukan sejumlah contoh obyek dan *message* (pesan) yang diletakan diantara obyek-obyek ini di dalam use case. Gambar 2.15 menunjukan *Sequence Diagram* dalam UML.



Gambar 2. 14 *Sequence Diagram* [10]

Berikut ini merupakan komponen dalam *sequence diagram*:

1. *Activations*

   Activations menjelaskan tentang eksekusi dari fungsi yang dimiliki oleh suatu objek.

2. *Actor*

   Actor menjelaskan tentang peran yang melakukan serangkaian aksi dalam suatu proses.

3. *Collaboration boundary*

   Collaboration boundary menjelaskan tentang tempat untuk lingkungan percobaan dan digunakan untuk memonitor objek.

4. *Parallel vertical lines*

   Parallel vertical lines menjelaskan tentang suatu garis proses yang menunjuk pada suatu *state*.

5. *Processes*

   Processes menjelaskan tentang tindakan/aksi yang dilakukan oleh aktor dalam suatu waktu.

6. *Window*

   Window menjelaskan tentang halaman yang sedang ditampilkan dalam suatu proses.

7. *Loop*

   Loop menjelaskan tentang model logika yang berpotensi untuk diulang beberapa kali.