# BAB 2
# TINJAUAN PUSTAKA

## 2.1 Sistem

Secara umum sistem dapat didefenisikan sebagai sekumpulan objek-objek yang saling berinteraksi serta hubungan antar objek bisa dilihat sebagai satu kesatuan yang dirancang untuk mencapai satu tujuan[4],[5]. Sistem mempunyai karakteristik atau sifat-sifat tertentu, yaitu mempunyai komponen-komponen (*components*), batas (*boundary*), lingkungan luar sistem (*environments*), penghubung (*interface*), masukan (*input*), keluaran (*output*), pengolah (*process*), dan sasaran (*objectives*) atau tujuan (*goal*).

a. Komponen Sistem

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerjasama membentuk suatu kesatuan. Komponen-komponen sistem atau elemen-elemen sistem dapat berupa subsistem atau bagian-bagian dari sistem. Setiap sistem tidak perduli betapapun kecilnya, selalu mengandung komponen-komponen atau subsistem-subsistem. Setiap subsistem mempunyai sifat-sifat dari subsistem untuk menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. Suatu sistem dapat mempunyai suatu sistem yang lebih besar disebut dengan supra sistem, misalnya suatu perusahaan dapat disebut sebagai suatu sistem sedang industri yang merupakan sistem yang lebih besar dapat disebut dengan supra sistem. Kalau dipandang industri sebagai suatu sistem, maka perusahaan dapat disebut sebagai subsistem. Demikian juga bila perusahaan dipandang sebagai suatu sistem, maka sistem akuntansi adalah subsistemnya. Kalau sistem akuntansi dipandang sebagai suatu sistem, maka perusahaan adalah supra sistem dan industri adalah supra dari supra sistem.

b. Batas Sistem

Batas sistem (boundary) merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batas sistem

ini memungkinkan suatu sistem dipasang sebagai suatu kesatuan. Batas suatu sistem menunjukkan ruang lingkup (scope) dari sistem tersebut.

c. Lingkungan Sistem

Lingkungan luar (environment) dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar sistem dapat bersifat menguntungkan dan dapat bersifat merugikan sistem tersebut. Lingkungan luar yang menguntungkan merupakan energi dari sistem dan dengan demikian harus tetap dijaga dan dipelihara. Sedang lingkungan luar yang merugikan harus ditahan dan dikendalikan, kalau tidak maka akan mengganggu kelangsungan hidup dari sistem.

d. Penghubung Sistem

Penghubung (interface) merupakan media penghubung antara satu subsistem dengan subsistem lainnya. Melalui penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem yang lain. Keluaran (output) dari satu subsistem akan menjadi satu masukan (input) bagi subsistem yang lain dan akan melalui penghubung. Dengan penghubung satu subsistem dapat berintegrasi dengan subsistem yang lainnya membentuk satu kesatuan.

e. Masukan Sistem

Masukan (input) adalah energi yang dimasukkan kedalam sistem. Masukan dapat berupa masukan peralatan (maintenence input) dan masukan sinyal (signal input). Maintenance input adalah energi yang diproses agar didapatkan keluaran. Sebagai contoh didalam sistem komputer, program adalah maintenance input yang digunakan untuk mengoperasikan komputernya sedangkan data adalah signal input untuk diolah menjadi informasi.

f. Keluaran Sistem

Suatu sistem dapat mempunyai suatu bagian pengolah yang akan merubah masukan menjadi keluaran. Suatu sistem produksi akan mengolah masukan berupa bahan baku dan bahan-bahan yang lain menjadi keluaran berupa barang jadi. Sistem

akuntansi akan mengolah transaksi menjadi laporan keuangan dan laporan-laporan lain yang dibutuhkan oleh menejemen.

g. Sasaran Sistem

Suatu sistem pasti mempunyai tujuan (goal) atau sasaran (objektif). Kalau sistem tidak mempunyai sasaran, maka operasi sistem tidak akan ada gunanya. Sasaran dari sistem sangat menentukan sekali, masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem

## 2.2 Gamifikasi

Gamifikasi merupakan suatu metode yang bekerja untuk membuat suatu aplikasi atau sistem menjadi lebih menarik, sehinggan menyebabkan pengguna memiliki kecenderungan untuk terus menggunakan sistem tersebut. Metode gamifikasi secara tidak langsung memberikan tantangan bagi pengguna sistem, agar dapat menyelesaikan tugas atau target yang telah ditetapkan oleh pemilik sistem atau pengguna itu sendiri.. Dengan adanya daya tarik pada sistem, pengguna menjadi tidak bosan [6],[7].

Penerapan metode gamifikasi cocok pada sistem yang baru pertama kali diimplementasikan pada suatu organisasi, sehingga pengguna tidak menjadi antipati pada sistem.

1. Points

Point *adalah elemen utama dari seluruh sistem yang di gamifikasi. Dengan adanya poin ini, kita dapat memonitor kegiatan apa saja yang dilakukan oleh pemakai sistem kita [4].* Point *juga dapat ditampilkan kepada pemakai sistem sehingga pemakai dapat melihat kegiatan yang belum perlu dia lakukan. Secara umum point terbagi menjadi 5 jenis:*

a. Experience point.

Experience point *(E*XP*) merupakan point yang paling utama. Setiap kegiatan yang dilakukan pengguna sistem akan mendapatkan* XP. XP *dapat menjadi tolak ukur pengguna mana yang sering berinteraksi dengan sistem.*

b. Redeemable point.

Redeemable point *(*RP*) berfungsi sebagai alat tukar dalam sistem.* RP *secara*

*umum akan membangun ekonomi virtual dalam sistem. Biasanya* RP *lebih sering dinamai* gold, silver, cash, *dan lain-lain.*

2. *Leaderboard*

Leaderboard *merupakan media untuk menampilkan urutan terbaik dari semua aspek interaksi pemain. Biasanya semua pemain diurutkan dari yang memiliki nilai terbesar hingga yang paling kecil [4].* Leaderboard *juga menunjukkan status sosial dalam permainan. Pemain di peringkat atas biasanya akan merasa puas dengan pencapaiannya, sedangkan pemain di peringkat bawah akan berusaha untuk mengejar nilai peringkat atas.* Leaderboard *ada sebagai media interaksi sosial yang mengarah kepada kompetisi.*

3. Badges

*Setiap orang suka mengumpulkan berbagai macam koleksi. Keinginan untuk memiliki sesuatu pasti ada dalam diri setiap orang. Biasanya apabila koleksi yang dimilikinya belum lengkap, orang akan selalu berusaha dengan keras untuk melengkapinya [4].*

4. Challenge & Quest.

*Terkadang beberapa pengguna sistem tidak tahu apa tujuan dari menggunakan sebuah sistem.* Challenge & Quest *memberikan arahan apa yang harus dilakukan pengguna sistem / pemain. Dengan adanya* challange & quest *ini, pengguna sistem akan terus tetap menggunakan sistem dan tujuan fundamental dari sistem ini tercapai [4].*

5. Onboarding

Onboarding *merupakan langkah yang diambil untuk memperkenalkan sistem kepada pengguna baru. Saat-saat pertama untuk pengguna baru dalam menggunakan sebuah sistem atau memainkan sebuah permainan adalah saat yang sangat penting untuk meyakinkan pengguna baru tersebut bahwa sistem yang ditawarkan menarik dan manfaat penggunaan untuk jangka panjang, sehingga pengguna baru tersebut memiliki keinginan yang kuat untuk terus menggunakan sistem tersebut [4].*

6. Social Engagement Loop

Sebuah motivasi akan membawa pengguna untuk melakukan sebuah

kegiatan, kemudian pengguna menyelesaikan kegiatan tersebut, dan pengguna akan dihadapkan dengan hasil kerjanya yang tampak atau hadiah yang diberikan, kemudian akan membangkitkan kembali motivasi untuk mengulangnya.

### *2.3 Unified Modelling Language (UML)*

*Unified Modeling Language* (*UML*) merupakan sistem arsitektur yang bekerja dalam *OOAD* (*Object-Oriented Analysis/Design*) dengan satu bahasa yang konsisten untuk menentukan, visualisasi, mengkontruksi, dan mendokumentasikan *artifact* (sepotong informasi yang digunakan atau dihasilkan dalam suatu proses[8],[9]. Tujuan *UML* diantaranya adalah :

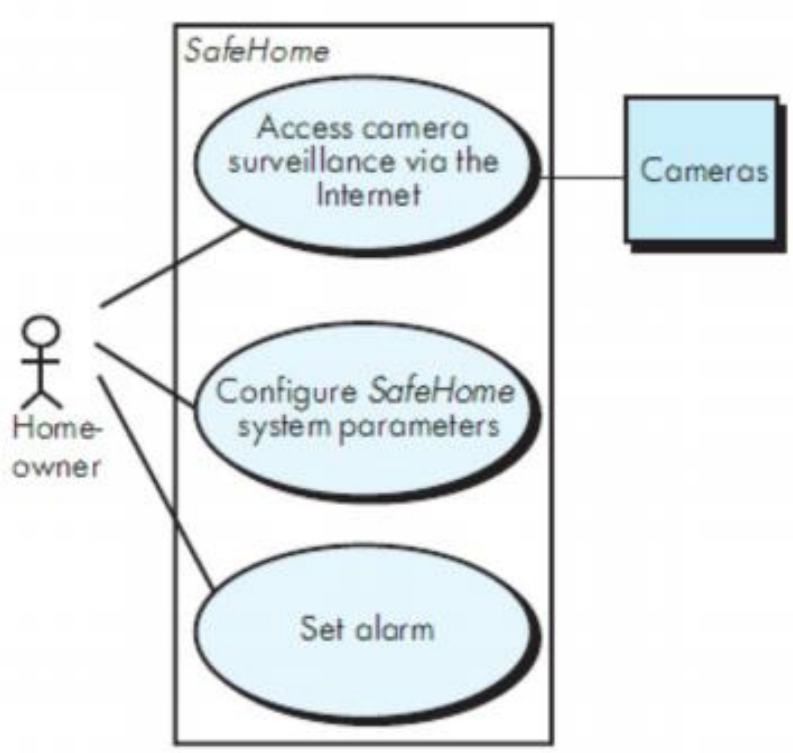
1. Memberikan model yang siap pakai, pemodelan *visual* yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
2. Memberikan contoh bahasa pemodelan yang bebas dari berbagai Bahasa pemrograman dan proses rekayasa.
3. Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan

*UML* terdiri dari beberapa diagram pemodelan berikut ini.

1. Use Case Diagram

*Use Case Diagram* merupakan model diagram *UML (Unified Modeling Language)* yang digunakan untuk menggambarkan kesepakatan fungsional yang diharapkan dari sebuah system. Diagram ini menjelaskan manfaat suatu sistem jika dilihat menurut pandangan orang yang berada diluar sistem atau kelas dan bagaimana sistem tersebut berinteraksi dengan dunia luar.

Berikut contoh untuk penggunaan *Use Case Diagram* dalam pembangunan sistem ditunjukan pada Gambar 2.1.

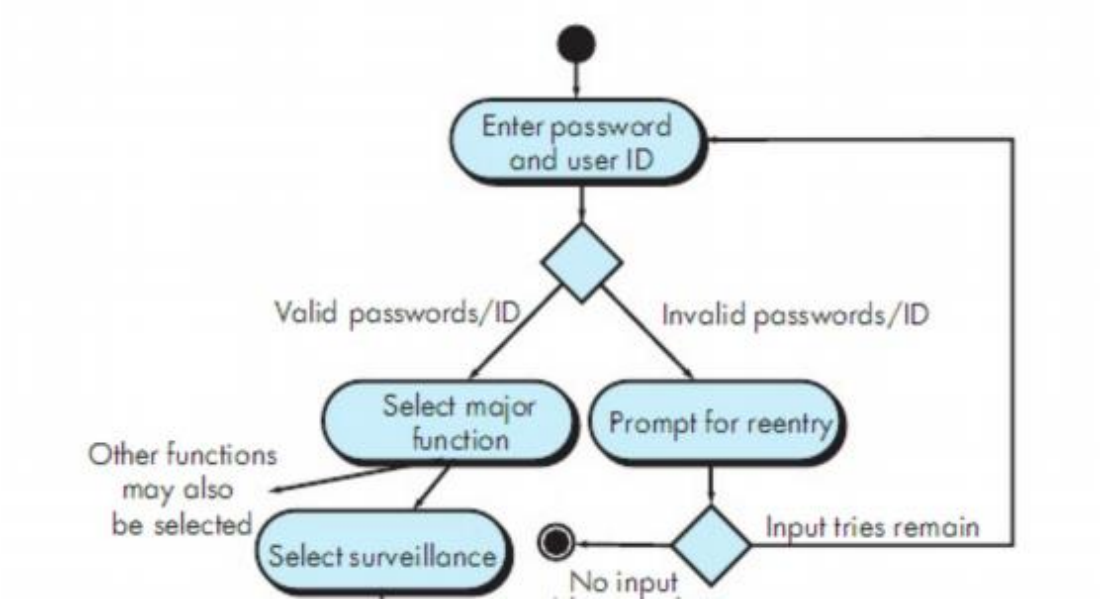



Gambar 2 1 *Use case Diagram*

2. Activity Diagram

*Activity Diagram* memodelkan alur proses dan hubungan antar proses dalam suatu sistem informasi dan juga berisikan tentang scenario yang ada dalam sistem tersebut[8].

Berikut contoh dari penerapan *activity diagram* ditunjukan pada Gambar 2.2:

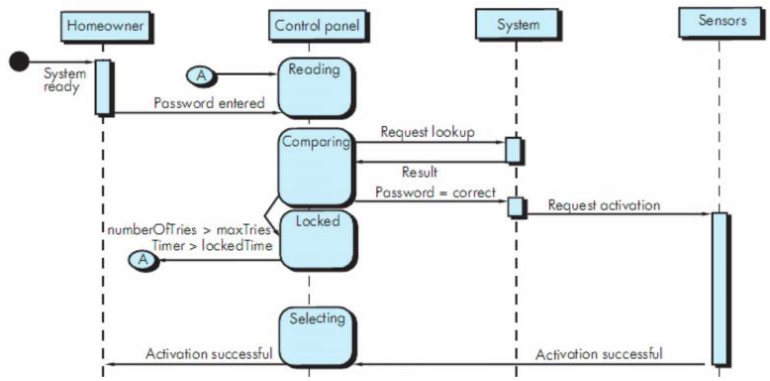



Gambar 2 2 *Activity Diagram*

3. Sequence Diagram

*Sequence Diagram* adalah diagram yang menjelaskan interaksi antar objek yang disusun berdasarkan waktu proses berlangsung. Diagram ini digunakan untuk menggambarkan tahap demi tahap yang harus dilakukan oleh pengguna sistem untuk menghasilkan sesuatu dari *use case diagram* yang sudah dibuat[8].

Berikut contoh dari penerapan *sequence diagram* di tunjukan pada Gambar 2.3:

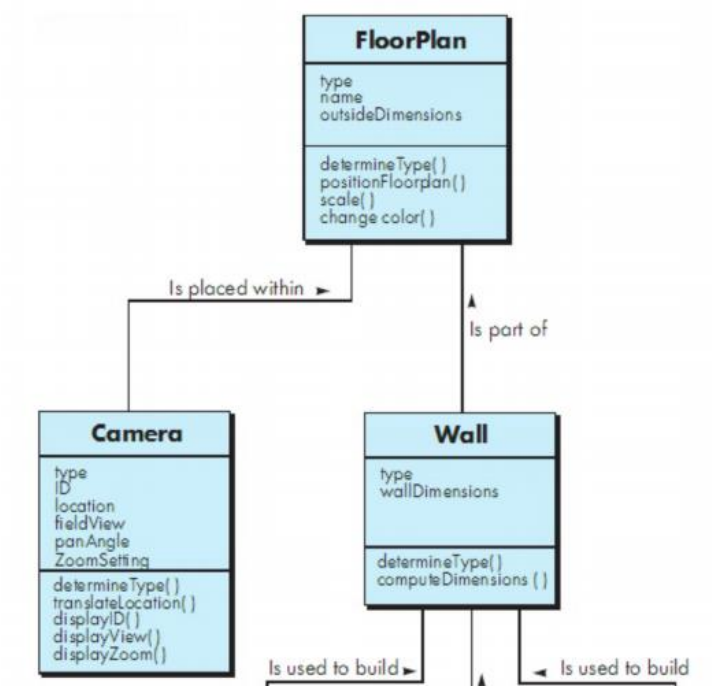



Gambar 2 3 *Sequence Diagram*

4. Class Diagram

*Class Diagram* adalah diagram yang menunjukkan *class-class* yang ada dari sebuah sistem dan saling berhubungan secara logika (Nugroho, 2015). Diagram ini menggambarkan struktur statis dari sebuah sistem.

Berikut contoh dari penerapan *Class diagram* ditunjukan pada Gambar 2.4:




Gambar 2 4 *Class Diagram*

## 2.4 Android

Android adalah sebuah system operasi untuk perangkat mobile berbasis linux yang mencakup sistem operasi, *middleware* dan applikasi. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri. Android diluncurkan untuk umum pada tahun 2008. Android sangat berkembang pesat di industry karena dua asopek utama yaitu bersifat opensource dan model arsitekturnya. Sebagai sebuah proyek yang bersifat opensource, memungkinkan android untuk sepenuhnya dipahami dan dianalisis mengenai fitur, penyelesaian pada bug program hingga hardware[10].

Aplikasi Android terdiri dari komponen yang bebas digabungkan, diikat menggunakan suatu *project manifest* yang meng kan masing-masing komponen dan bagaimana komponen-komponen itu saling berhubungan[10]. Ada enam komponen dalam Android untuk membangun suatu aplikasi yaitu:

1. Activities

   *Activity* merupakan bagian yang paling penting dalam sebuah aplikasi,karena *activity* menyajikan tampilan visual program yang sedang dijalankan oleh *user*. Setiap *activity* dideklarasikan dalam sebuah *class* yang bertugas untuk menampilkan antarmuka yang terdiri dari *views* dan *respons* terhadap *event*. Setiap aplikasi mempunyai satu buah *activity* atau lebih. Biasanya pasti aka nada *activity* yang pertama kali tampil ketika aplikasi sedang dijalankan. Perpindahan antara *activity* dengan *activity* lainnya ditentukan oleh posisinya oada *activity stack*. Bila suatu *activity* baru dimulai *activity* yang sebelumnya digunakan maka akan dipindahkan ke tumoukan paling atas. Ketika *activity* dipanggil dan disimpan dalam *activity stack* terdapat 4 kemungkinan kondisi transisi yang akan terjadi:

   1. *Active*

      Setiap *activity* yang berada di tumpukan paling atas, maka *activity* tersebut akan terlihat dan menerima masukan dari *user*. Android akan membuat *activity* ini tetap hidup dengan menghentikan *activity* yang berada dibawah tumpukan jika diperlukan. Ketika *activity* sedang aktif maka yang lainnya akan dihentikan sementara.

   2. *Paused*

      Beberapa *activity* akan terlihat tapi tidak terfokus pada kondisi ini disebut *paused*, dia terlihat aktif namun tidak bias diakses oleh *user.*

   3. *Stopped*

      Ketika sebuah *activity* tidak terlihat maka itulah yang disebut *stopped. Activity* akan tetap berada dalam memori dengan semua keadaan dan informasi yang ada. Namun akan di eksekusi oleh system jika system membutuhkan sumberdaya lebih.

   4. *Inactive*

Kondisi ini ketika *activity* telah dihentikan atau dijalankan *inactive activity* telah ditiadakan dari *activity stack* sehingga perlu *restart* ulang agar dapat tampil dan digunakan kembali.

2. *Content Providers*

   Content providers digunakan untuk mengelola dan berbagi database. Data dapat disimpan dalam file system, dalam database SQLite, atau dengan cara lain yang pada prinsipnya sama. Dengan adanya content provider memungkinkan antar aplikasi untuk saling berbagi data.

3. *Services*

   Suatu *services* tidak memiliki tampilan antarmuka, melainkan berjalan di *background* untuk waktu yang tidak terbatas. Komponen *service* diproses tidak terlihat memperbaharui sumber data dan menampilkan notifikasi.

4. *Intens*

   *Intens* merupakan sebuah mekanisme untuk meng kan tindakan tertentu seperti memilih foto, menampilkan halaman web, dan lain sebagainya. *Intens* tidak selali dimulai dengan menjalankan aplikasi, namun juga digunakan oleh system untuk memberitahukan ke aplikasi bila terjadi sesuatu, semisal pesan masuk.

5. *Broadcast receivers*

   *Broadcast receivers* merupakan komponen yang sebenarnya tidak melakukan apa-apa kecuali menerima dan bereaksi menyampaikan pemberitahuan. Sebagian besar *broadcast* berasal dari sistem contohnya baterai sudah hampir habis, informasi zona waktu sudah berubah, atau *user* telah mengganti bahasa default pada perangkat Android. Sama halnya dengan service, *broadcast receivers* tidak menampilakan antarmuka user. Namun, *broadcast receivers* dapat menggunakan *notification manager* untuk memberitahukan sesuatu kepada *user*.

6. *Notifications*

   *Notifications* merupakan kerangka pemberitahuan untuk user. *Notifications* memberikan isyarat atau peringatan tanpa mengganggu aktivitas yang sedang dilakukan oleh user.

### 2.4.1 Perkembangan *Android* di Indonesia

Seiring dengan berkembangnya teknologi informasi menyebabkan semakin banyak orang mengerti akan pentingnya fungsi komputer dalam membantu pekerjaan mereka. Saat ini perkembangan komputer telah membawa perubahan besar dalam berbagai bidang diantaranya bidang Politik, Ilmu Pengetahuan, Ekonomi, Sosial, Budaya, dan Kesehatan. Perkembangan teknologi informasi, selain perkembangan aplikasi desktop pada komputer juga meliputi perkembangan aplikasi mobile. Seperti yang kita ketahui saat ini, kebutuhan manusia tidak pernah terbatas seperti kebutuhan komunikasi salah satunya. Sehingga, *handphone* yang kita kenal sebagai alat telpon (komunikasi) genggam semakin berkembang pesat dengan aplikasi-aplikasi terbaru dan bermanfaat untuk kebutuhan manusia di saat ini.

Semakin berkembang aplikasi mobile maka, terciptalah sebuah sistem operasi yang dikembangkan untuk perangkat mobile berbasi *linux* yaitu *Android*. Pada awalnya sistem operasi ini dikembangkan oleh *Android Inc*. yang kemudian dibeli oleh *Google* pada tahun 2005. Dalam usaha untuk mengembangkan *Android*, pada tahun 2007 dibentuklah *Open Handset Alliance(OHA)*, sebuah konsorsium dari beberapa perusahaan dengan tujuan untuk mengembangkan standar terbuka untuk perangkat mobile. Kelebihan dari *Android* sendiri bagi para pengembang aplikasi mobile adalah dengan *software Development Kits (SDK)* yang lengkap, dilengkapi dengan *emulator* yang membantu untuk menguji coba aplikasi yang dibuat serta dokumentasi yang lengkap. Serta tidak ada biaya lisensi untuk memperoleh *SDK* ini. *Android* merupakan pilihan yang tepat untuk pengembang.

Selain dari segi pengembang aplikasi *mobile*, *Android* juga mempunyai kelebihan dari sisi pengguna *Android (user). Android* menawarkan sebuah lingkungan yang berbeda untuk pengembang. Setiap aplikasi memiliki tingkatan yang sama. *Android* tidak membedakan antara aplikasi inti dengan aplikasi pihak ketiga. *API* yang disediakan menawarkan akses ke *hardware* maupun data-data ponsel sekalipun, atau data system sendiri. Bahkan pengguna dapat menghapus aplikasi inti dan menggantikannya dengan aplikasi pihak ketiga. Disinilah yang membuat *OS Android* berbeda dengan *OS mobile* lainnya. *User* dapat dengan

mudah mendapatkan berbagai aplikasi yang uptodate, hanya tinggal *mendownload* saja. Sehingga *user* dapat dengan leluasa menggunakan aplikasi pihak ketiga. Contohnya ada beberapa aplikasi yang sudah siap untuk di *download* oleh user *Android* seperti mapping, pariwisata, cara memasakdengan resep jitu dll. Aplikasi-aplikasi tersebut merupakan aplikasi pihak ketiga yang dikembangkan oleh pengembang menggunakan *Android*. Jadi *Android* mempunyai kelean bagi bihara pengembang aplkasi-aplikasi mobile (aplikasi pihak ketiga) dan juga sebagai pengguna mobile *Android* dapat dengan mudah mendapatkan berbagai aplikasi pihak ketiga tersebut. Cukup dengan mendownload nya di tempat *browser* yang tersedia di mobile tersebut.

Berdasarkan kelebihan *Android* seperti yang telah di sebutkan di atas maka, kini saya akan membahas perkembangan *Android* khusus di Negara Indonesia yaitu, pasar *Android* di Indonesia akan berkembang seiring dari banyaknya operator selular dan Produsen *Smartphone* gencar menyuarakan *Open source Android*. Pangsa Pasar *Smartphone* Indonesia yang besar memungkinkan *Smartphone* yang murah dan mempunyai feature yang lengkap sesuai dengan karateristik dari masyarakat Indonesia

Saat ini kita bahas salah satu operator yang gencar menyuarakan *Android* yaitu Indosat. Pada bulan Maret lalu Indosat melakukan road show di 7 kota yakni Jakarta, Bandung, Semarang, Jogyakarta, Malang, Surabaya, dan Makassar. Indosat memperkenalkan Teknologi *Smartphone* yang mempermudah komunikasi, tampilan yang menarik, serta aplikasi yang mudah dioperasikan. Dalam Road Show ini Indosat juga melakukan seminar dan edukasi kepada pengunjung. Dalam seminar ini, pengunjung bisa lebih mengetahui apa itu *Android*, sejarah, kelebihan serta platform dan variasi aplikasi yang dimiliki. Apa yang dilakukan Indosat akan diikuti oleh operator selular lain yang melihat potensi dan peluang bisnis dari *Open Source Android* ini.

Pada saat ini, perkembangan *Android* di Indonesia dipengaruhi oleh banyaknya *Smartphone* yang telah beredar di Indonesia dan keinginan berbagai produsen *Smartphone* tersebut untuk memangkas biaya produksi sehingga menghasilkan produk *Smartphone* yang berkualitas dan mempunyai harga jual yang

lebih terjangkau daripada menggunakan *OS* yang lainnya. Persebaran *Smartphone* ber*Android* di Indonesia yang besar memungkinkan *Smartphone* yang murah dan mempunyai *feature* yang lengkap sesuai dengan karateristik dari masyarakat Indonesia.

Penyebab mengapa *Android* dapat berkemban cepat di Indonesia.

1. *Update* rutin

   *Android* selalu melakukan update secara terus menerus, melakukan perbaikan perbaikan berbagai bugs dan penambahan fitur yang menjadikan *OS* semakin lebih bagus dari versi sebelumnya.

2. *Open source*

   *Android* adalah *OS open source* yang gratis jadi dilihat dari segi harganya akan lebih murah daripada *Smartphone* yang memiliki *OS* tidak gratis, disamping itu *OS Android* memungkinkan para programmer programmer untuk mengembangkan atau membuat aplikasi berbasis *Android*.

3. Didukung oleh *Vendor* Kelas Atas

   Dukungan penuh dari vendor-vendor kelas atas seperti *Samsung, HTC, Motorola dll* dalam menghasilkan *Smartphone* yang berkelas akan membantu menaikkan pamor *Android*.

4. Merek *Google*

   Reputasi *Google* yang tidak diragukan lagi menjadi keunggulan tersendiri bagi *Android*. Hal ini membuat konsumen yakin bahwa *OS Android* adalah *OS* yang benar benar bagus dan berkualitas.

5. *User Friendly*

   Teknologi layar sentuh, membuat mudah dalam penggunaannya serta didukung oleh tampilan yang menarik

### 2.4.2 The Dalvik Virtual Machine (DVM)

Salah satu elemen kunci dari *Android* adalah *Dalvik Virtual Machine* (DVM). *Android* berjalan di dalam *Dalvik Virtual Machine* (DVM) bukan di *Java Virtual Machine* (*JVM*), sebenarnya banyak persamaan dengan *Java Virtual Machine* (*JVM*) seperti *Java ME* (*Java Mobile Edition*), tetapi *Android* menggunakan *Virtual Machine* sendiri yang menurut saya dikustomisasi dan dirancang untuk

memastikan bahwa beberapa *feature-feature* berjalan lebih efisien pada perangkat *mobil*[10].

*Dalvik Virtual Machine* (DVM) adalah "*register bases"* sementara *Java Virtual Machine* (JVM) adalah "*stack based", DVM* didesain dan ditulis oleh Dan Bornsten dan beberapa engineers *Google* lainnya. Jadi bisa kita katakana "*Dalvik equals(Java) == False".Dalvik Virtual Machine* menggunakan kernel Linux untuk menangani fungsionalitas tingkat rendah termasuk keamanan, *threading*, dan proses serta manajemen memori. Ini memungkinkan kita untuk menulis Aplikasi *C/C+* sama halnya seperti *OS Linux* kebanyakan. Meskipun dalam kenyataannya kita harus banyak memahami Arsitektur dalam proses *system* dari kernel linux yang digunakan dalam *Android* tersebu.

Semua *hardware* yang berbasi *Android* dijalankan menggunakan *Virtual Machine* untuk eksekusi aplikasi, pengembang tidak perlu khawatir tentang implementasi perangkat keras tertentu.*Dalvik Virtual Machine* mengeksekusi *executablefile*, sebuah format yang dioptimalkan untuk memastikan memori yang digunakan sangat kecil. *The Executable file* diciptakan dengan mengubah kelas bahasa java dan dikompilasi menggunakan *tools* yang disediakan dalam SDK *Android.*

### 2.4.3 Android SDK (Software Development Kit)

*Android SDK tools API* (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform Android* menggunakan bahasa pemrograman Java [11]. *Android* merupakan subset perangkat lunak untuk ponsel yang meliputi system operasi, *middleware* dan aplikasi kunci yang di rilis oleh Google. Saat ini disediakan *Android SDK* (*Software Development Kit*) sebagai alat bantu dan *API* untuk mulai mengembangkan aplikasi pada *platform* menggunakan Bahasa pemrograman *Java*.

*Android* memberi kita kesempatan untuk membuat aplikasi yang kita butuhkan yang bukan merupakan aplikasi bawaan *Handphone/Smartphone.* Beberapa fitur-fitur *Android* yang paling penting adalah:

1. *Framework* Aplikasi yang mendukung penggantian komponen dan *reusable.*
2. Mesin *Virtual Dalvik* dioptimalkan untuk perangkat *mobile*

3. *Integrated Browser* berdasarkan *engine open source WebKit*
4. *Grafis* yang dioptimalkan dan dudukung oleh *libraries* grafis 2D, grafis 3D berdasarkan spesifikasi *openglES 1,0(Opsional akselerasi hardware)*
5. *SQLite* untuk penyimpanan data
6. Media Support yang mendukung audio, video, dan gambar *(MPEG4, H.164, MP3, AAC, AMR, JPG, PNG, GIF), GSM Telephony* (tergantumg *hardare*)
7. *Bluetooth, EDGE, 3G, dan WiFi* (tergantung *hardware*)
8. Kamera, *GPS*, Kompas, dan *accelerometer* (tergantung *hardware*)
9. Lingkungan *Development* yang lengkap dan kaya termasuk perangkat emulator, *tools* untuk *debugging*, profil dan kinerja memori, dan plugin untuk *IDE Eclipse*.

### 2.4.4 ADT (Andoid Development Tools)

*Android Development Tools* (ADT) adalah *plugin* yang didesain untuk *IDE Eclipse* yang memberikan kita kemudahan dalam mengembangkan aplikasi *Android* dengan menggunakan *IDE Eclipse* [10]. Dengan menggunakan *ADT* untuk *Eclipse* akan memudahkan kita dalam membuat aplikasi project *Android*, membuat *GUI* aplikasi, dan menambahkan komponen-komponen yang lainnya, selain itu juga dapat melakukan *running* aplikasi menggunakan *Android SDK* melalui *Eclipse*. Dengan *ADT* juga kita dapat melakukan pembuatan *package Android* (.apk) yang digunakan untuk distribusi aplikasi *Android* yang kita rancang.

Mengembangkan aplikasi *Android* dengan menggunakan ADT di *Eclipse* sangat dianjurkan dan sangat mudah untuk memulai pengembangan aplikasi *Android*. Berikut adalah versi *ADT* untuk *Eclipse* yang sudah dirilis:

1. *ADT* 12.0.0 (July 2011)
2. *ADT* 11.0.0 (June 2011)
3. *ADT* 10.0.1 (March 2011)
4. *ADT* 10.0.0 (February 2011)
5. *ADT* 9.0.0 (January 2011)
6. *ADT* 8.0.1 (December 2010)
7. *ADT* 8.0.0 (December 2010)
8. *ADT* 0.9.9 (September 2010)

9. *ADT* 0.9.8 (September 2010)
10. *ADT* 0.9.7 (May 2010)
11. *ADT* 0.9.6 (March 2010)
12. *ADT* 0.9.5 (December 2009)
13. *ADT* 0.9.4 (October 2009)

Semakin tinggi *platform Android* yang kita gunakan, semakin lebih terbaru pula *ADT* yang digunakan, Karena biasanya munculnya *platform* baru diikuti oleh munculnya versi *ADT* yang terbaru.

***2.5 Framework.***

*Framework* secara sederhana dapat diartikan kumpulan dari fungsi-fungsi/prosedur-prosedur dan class-class untuk tujuan tertentu yang sudah siap digunakan sehingga bisa lebih mempermudah dan mempercepat pekerjaan seorang programer, tanpa harus membuat fungsi atau class dari awal [11],[12]. Ada beberapa alasan mengapa menggunakan *Framework*:

1. Mempercepat dan mempermudah pembangunan sebuah aplikasi *web*.
2. Relatif memudahkan dalam proses *maintenance* karena sudah ada pola tertentu dalam sebuah *framework* (dengan syarat programmer mengikuti pola standar yang ada).
3. Umumnya *framework* menyediakan fasilitas-fasilitas yang umum dipakai sehingga kita tidak perlu membangun dari awal (misalnya validasi, *ORM, pagination, multiple database, scaffolding*, pengaturan session, *error handling*, dll.
4. Lebih bebas dalam pengembangan jika dibandingkan CMS.

*Framework* menganut konsep *Model View Controller (MVC)* merupakan suatu konsep yang cukup populer dalam pembangunan aplikasi *web* (Mulyanto, 2012). Berawal pada bahasa pemrograman *Small Talk*, *MVC* memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, *user interface*, dan bagian yang menjadi kontrol aplikasi . Terdapat 3 jenis komponen yang membangun suatu *MVC pattern* dalam suatu aplikasi yaitu:

1. *View*, merupakan bagian yang menangani presentation logic. Pada suatu

aplikasi *web* bagian ini biasanya berupa file *template HTML*, yang diatur oleh *controller*. *View* berfungsi untuk menerima dan merepresentasikan data kepada user. Bagian ini tidak memiliki akses langsung terhadap bagian model.

2. *Model*, biasanya berhubungan langsung dengan *database* untuk memanipulasi data *(insert, update, delete, search)*, menangani validasi dari bagian *controller*, namun tidak dapat berhubungan langsung dengan bagian *view*.
3. *Controller*, merupakan bagian yang mengatur hubungan antara bagian model dan bagian *view*, *controller* berfungsi untuk menerima *request* dan data dari user kemudian menentukan apa yang akan diproses oleh aplikasi.

Dengan menggunakan prinsip *MVC* suatu aplikasi dapat dikembangkan sesuai dengan kemampuan developernya, yaitu programmer yang menangani bagian *model* dan *controller*, sedangkan designer yang menangani bagian *view*, sehingga penggunaan arsitektur *MVC* dapat meningkatkan *maintanability* dan organisasi kode. Walaupun demikian dibutuhkan komunikasi yang baik antara programmer dan designer dalam menangani variabel-variabel yang akan ditampilkan.

### *2.6 Codeigneiter*

*Codeigniter* adalah sebuah *framework PHP* yang dapat membantu mempercepat developer dalam pengembangan aplikasi *web* berbasis *PHP* dibandingkan jika menulis semua kode program dari awal. *Codeigniter* menyediakan banyak *library* untuk mengerjakan tugas-tugas yang umumnya ada pada sebuah aplikasi berbasis *web*..Selain itu, struktur dan susunan logis dari *codeigniter* membuat aplikasi yang dibuat menjadi semakin teratur dan rapi. Dengan demikian, dapat fokus pada fitur-fitur apa yang dibutuhkan aplikasi dengan membuat kode program seminimal mungkin [13],[14].

Ada beberapa kelebihan *CodeIgniter* (CI) dibandingkan dengan *Framework PHP* lain yaitu:

1. Performa cepat.

   Salah satu alasan tidak menggunakan *framework* adalah karena eksekusinya

yang lebih lambat daripada *PHP from the scracth*, tapi *Codeigniter* sangat cepat bahkan mungkin bisa dibilang *codeigniter* merupakan *framework* yang paling cepat dibanding *framework* yang lain.

2. Konfigurasi yang sangat minim (*nearly zero configuration*).

   Tentu saja untuk menyesuaikan dengan *database* dan keleluasaan *routing* tetap diizinkan melakukan konfigurasi dengan mengubah beberapa file konfigurasi seperti *database*, *PHP* atau *Autoload PHP*, namun untuk menggunakan *codeigniter* dengan setting standard, anda hanya perlu mengubah sedikit saja file pada folder *config*.

3. Banyak komunitas.

   Dengan banyaknya komunitas *CI* ini, memudahkan kita untuk berinteraksi dengan yang lain, baik itu bertanya atau teknologi terbaru.

4. Dokumentasi yang sangat lengkap.

   Setiap paket instalasi *codeigniter* sudah disertai *user guide* yang sangat bagus dan lengkap untuk dijadikan permulaan, bahasanya pun mudah dipahami.

5. Gratis.

   *Codeigniter* berlisensi dibawah *Apache/BSD open source*, jadi Anda bisa menggunakannya secara bebas. Untuk informasi lebih lanjut, anda bisa membaca *license agreement* yang dapat dibaca saat instalasi *codeigniter*.

6. Menggunakan *PHP* 4.

   Meskipun *codeigniter* dapat berjalan pada *PHP* 5, namun sampai saat ini kode program *codeigniter* masih dibuat dengan menggunakan *PHP* 4. Hal ini dilakukan agar *codeigniter* dapat tersebar lebih luas di komunitas *PHP*. Karena hingga saat ini, sebagian besar *web hosting* masih menggunakan *PHP* 4. Jika *codeigniter* dibuat dengan *PHP* 5, tentu saja hasilnya akan jauh lebih canggih, karena bisa memanfaatkan teknologi *PHP* 5 yang saat ini masih belum dapat dilakukan oleh *PHP* 4, misalnya untuk menerapkan *konsep OOP Multiple Inheritance*.

7. Berukuran kecil.

   Ukuran *codeigniter* yang kecil merupakan keunggulan tersendiri. Dibanding *framework* lainnya yang berukuran besar, serta membutuhkan *resource* yang

besar pula untuk berjalan. Pada *codeigniter*, bisa diatur agar sistem meload *library* yang dibutuhkan saja, sehingga dapat berjalan ringan dan cepat.

8. Menggunakan konsep *M-V-C.*

   *Codeigniter* menggunkan konsep *M-V-C* (*Model-View-Controller*) yang memungkinkan pemisahan antara *layer aplication-logic* dan *presentation.*

9. URL yang sederhana.

   Secara *default*, URL yang dihasilkan *Codeigniter* sangat bersih (*Clean*) dan *Search Engine Friendly* (SEF).

10. Memiliki paket *library* yang lengkap.

    *Codeigniter* memiliki *library* yang lengkap untuk mengerjakan operasi-operasi yang umum dibutuhkan oelh sebuah aplikasi berbasis *web*, misalnya mengakses *database*, mengirim email, memvalidasi form, menangani *session*, dan sebagainya.

11. *Extensible*.

    Sistem dapat dikembangkan dengan mudah dengan menggunkan *plugin* dan *helper*, atau dengan menggunakan hooks.

12. Tidak memerlukan *Template Engine.*

## 2.7 *PHP*

PHP merupakan bahasa skrip yang ditanam dalam HTML, yang berarti kode PHP dan HTML dapat digabungkan dalam file yang sama. *PHP* disebut juga pemrograman *Server Side Programming*, hal ini dikarenakan seluruh prosesnya dijalankan pada *server*. *PHP* adalah suatu bahasa dengan hak cipta terbuka atau yang juga dikenal dengan *open source* yaitu pengguna data mengembangkan kode-kode fungsi sesuai kebutuhannya[13],[14].

*PHP (Perl Hypertext Preprocessor)* adalah bahasa *server-side-scripting* yang menyatu dengan *HTML* untuk membuat halaman *web* yang dinamis". Dengan menggunakan program *PHP*, sebuah *website* akan lebih interaktif dan dinamis. Kelebihan-kelebihan dari *PHP* yaitu:

1. *PHP* merupakan sebuah bahasa *script* yang tidak melakukan sebuah kompilasi dalam penggunaannya. Tidak seperti halnya bahasa pemrograman aplikasi yang lainnya.

2. *PHP* dapat berjalan pada *web server* yang dirilis oleh Microsoft, seperti *IIS* atau *PWS* juga pada *apache* yang bersifat *open source*.
3. Karena sifatnya yang *open source*, maka perubahan dan perkembangan interpreter pada *PHP* lebih cepat dan mudah, karena banyak milis-milis dan *developer* yang siap membantu pengembanganya.
4. Jika dilihat dari segi pemahaman, *PHP* memiliki referensi yang begitu banyak sehingga sangat mudah untuk dipahami.
5. *PHP* dapat berjalan pada 3 operating sistem, yaitu: *Linux*, *unux,* dan *windows,* dan juga dapat dijalankan secara *runtime* pada suatu *console*.

### *2.8 MYSQL.*

*MySQL* adalah *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 *closed source* atau komersial. *MySQL* sebenarnya merupakan turunan salah satu konsep utama dalam *database* sejak lama, yaitu *SQL (Structured Query Language). SQL* adalah sebuah konsep pengoperasian *database*, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis [14],[15],[16],[17]. Keandalan suatu sistem *database* (*DBMS*) dapat diketahui dari cara kerja optimizer-nya dalam melakukan proses perintah-perintah *SQL*, yang dibuat oleh user maupun program-program aplikasinya. *MySQL* biasanya digunakan atau diinstall bersamaan dengan *XAMPP* sehingga untuk melihat isi tabel bisa menggunakan *PHPmyAdmin*.

### *2.9 HTML.*

*Hypertext Markup Language (HTML)* adalah bahasa yang digunakan untuk menulis halaman *web*. *HTML* merupakan pengembangan dari standar pemformatan dokumen teks yaitu *Standard Generalized Markup Language* (SGML) [18],[19]. *HTML* sebenarnya adalah dokumen *ASCII* atau teks biasa, yang dirancang untuk tidak tergantung pada suatu sistem operasi tertentu.

Mendesain *HTML* berarti melakukan suatu tindakan pemrograman. Namun *HTML* bukanlah sebuah bahasa pemrograman. Namun *HTML* hanyalah berisi

perintah-perintah yang telah terstruktur berupa tag-tag penyusun. Menuliskan tag-tag *HTML* tidaklah sebatas hanya memasukkan perintah-perintah tertentu agar *HTML* kita dapat di akses oleh browser. Mendesain *HTML* adalah sebuah seni tersendiri. *Homepage* yang merupakan implementasi dari *HTML* adalah refleksi dari orang yang membuatnya. Untuk itu kita perlu mendesainnya dengan baik agar para pengunjung homepage yang kita buat merasa senang dan bermanfaat. Mendesain *HTML* dapat dilakukan dengan cara menggunakan *HTML Editor*, seperti notepad++, adobe dreamweaver dan lain-lain.

### 2.10 *Cascading Style Sheets* (*CSS*).

*Cascading Style Sheets (CSS)* adalah suatu bahasa *stylesheet* yang digunakan untuk mengatur tampilan suatu dokumen yang ditulis dalam bahasa markup (Kadir, 2014). Penggunaan yang paling umum dari CSS adalah untuk memformat halaman *web* yang ditulis dengan *HTML* dan *XHTML*. Walaupun demikian, bahasanya sendiri dapat dipergunakan untuk semua jenis dokumen *XML* termasuk *SVG dan XUL*. *Spesifikasi CSS* diatur oleh *World Wide Web Consortium (W3C)*[18].

*CSS* digunakan oleh penulis maupun pembaca halaman *web* untuk menentukan warna, jenis huruf, tata letak, dan berbagai aspek tampilan dokumen. *CSS* digunakan terutama untuk memisahkan antara isi dokumen (yang ditulis dengan *HTML* atau *bahasa markup* lainnya) dengan presentasi dokumen (yang ditulis dengan *CSS*). Pemisahan ini dapat meningkatkan aksesibilitas isi, memberikan lebih banyak keleluasaan dan kontrol terhadap tampilan, dan mengurangi kompleksitas serta pengulangan pada stuktur isi.

### *2.11 Black Box Testing.*

*Tester* menggunakan *behavioral test* (disebut juga *Black Box Test*), sering digunakan untuk menemukan *bug* dalam *high level operations*, pada tingkatan fitur, profil operasional dan skenario *customer*. *Tester* dapat membuat pengujian fungsional *blackbox* berdasarkan pada apa yang harus sistem lakukan. *Behavioraltesting* melibatkan pemahaman rinci mengenai domain aplikasi, masalah bisnis yang dipecahkan oleh sistem dan misi yang dilakukan sistem[20]. *Behavioraltest* paling baik dilakukan oleh penguji yang memahami desain sistem,

setidaknya pada tingkat yang tinggi sehingga mereka dapat secara efektif menemukan bug umum untuk jenis desain. *Black box testing* juga disebut *functional testing*, sebuah teknik pengujian fungsional yang merancang *test case* berdasarkan informasi dari spesifikasi.

## 2.12 Global Positioning System (GPS)

GPS atau Global Positioning System, merupakan sebuah alat atau sistem yang dapat digunakan untuk menginformasikan penggunanya berada (secara global) di permukaan bumi yang berbasiskan satelit. Data dikirim dari satelit berupa sinyal radio dengan data digital. Dimanapun posisi saat ini, maka GPS bisa membantu menunjukan arah, selama masih terlihat langit. Layanan GPS ini tersedia gratis bahkan tidak perlu mengeluarkan biaya apapun kecuali membeli GPS *reciever*-nya. Awalnya GPS hanya digunakan hanya untuk kepentingan militer, tapi pada tahun 1980-an dapat digunakan untuk kepentingan sipil. GPS dapat digunakan dimanapun juga dalam 24 jam. Posisi unit GPS akan ditentukan berdasarkan titik-titik koordinat derajat lintang dan bujur [21],[22].

### 2.12.5 Pengertian GPS

GPS (*Global Positioning System*) adalah sistem navigasi yang berbasiskan satelit yang saling berhubungan yang berada di orbitnya. Satelit-satelit itu milik Departemen Pertahanan (*Departemen of Defense*) Amerika Serikat yang pertama kali diperkenalkan mulai tahun 1978 dan pada tahun 1994 sudah memakai 24 satelit [21].

Untuk dapat mengetahui posisi seseorang maka diperlukan alat yang diberinama GPS *reciever* yang berfungsi untuk menerima sinyal yang dikirim dari satelit GPS. Posisi di ubah menjadi titik yang dikenal dengan nama *Way-point* nantinya akan berupa titik-titik koordinat lintang dan bujur dari posisi seseorang atau suatu lokasi kemudian di layar pada peta elektronik.

Satelit-satelit ini mengorbit pada ketinggian sekitar 12.000 mil dari permukaan bumi. Posisi ini sangat ideal karena satelit dapat menjangkau area coverage yang lebih luas. Satelit-satelit ini akan selalu berada posisi yang bisa

menjangkau semua area di atas permukaan bumi sehingga dapat meminimalkan terjadinya blank spot (area yang tidak terjangkau oleh satelit) .

GPS reciever sendiri berisi beberapa integrated circuit (IC) sehingga murah dan teknologinya mudah untuk di gunakan oleh semua orang. GPS dapat digunakan utnuk berbagai kepentingan, misalnya mobil, kapal, pesawat terbang, pertanian dan di integrasikan dengan komputer maupun laptop.

Berikut beberapa contoh perangkat GPS *receiver* dapat dilihat pada Gambar 2.5 :


Gambar 2.5 GPS

### 2.12.6 Sistem Satelit GPS

Untuk menginformasikan posisi user, 24 satelit GPS yang ada di orbit sekitar 12,000 mil di atas kita. Bergerak konstan bergerak mengelilingi bumi 12 jam dengan kecepatan 7,000 mil per jam. Satelit GPS berkekuatan energi sinar matahari, mempunyai baterai cadangan untuk menjaga agar tetap berjalan pada saat gerhana matahari atau pada saat tidak ada energi matahari. Roket penguat kecil pada masing-masing satelit agar dapat mengorbit tepat pada tempatnya [21]. Satelit GPS harus selalu berada pada posisi orbit yang tepat untuk menjaga akurasi data yang dikirim ke GPS reciever, sehingga harus selalu dipelihara agar posisinya tepat. Stasiun-stasiun pengendali di bumi ada di Hawaii, Ascension Islan, Diego Garcia, Kwajalein dan Colorado Spring. Stasiun bumi tersebut selalu memonitor posisi orbit jam jam satelit dan di pastikan selalu tepat.

### 2.12.7 Location Based Services

Location Based Service (LBS) atau layanan berbasis lokasi adalah sebuah layanan informasi yang dapat diakses dengan perangkat bergerak melalui jaringan dan mampu menampilkan posisi secara geografis keberadaan perangkat bergerak

tersebut. Location Based Service dapat berfungsi sebagai layanan untuk mengidentifikasi lokasi dari seseorang atau suatu objek tertentu, seperti menemukan lokasi tempat yang diinginkan terdekat atau lokasi lainnya.

Hal paling penting dari Location Based Service (LBS) dapat bekerja sesuai yang diinginkan oleh pengembang aplikasi Android. Android pun memungkinkan pengembang menentukan metode pencarian lokasi yang dibutuhkan dan juga dapat mengatur kebutuhan daya, biaya dan akurasi berdasarkan spesifik yang akan dibuat untuk aplikasi tersebut [23],[24],[25].

2.12.7.1 Unsur-Unsur Location Based Services

Dua unsur utama dari *Location Based Service* adalah:

1. *Location Manager* (API Maps): Menyediakan perangkat bagi sumber untuk *Location Based Service* (LBS), *Application Programming Interface* (API) Maps menyediakan fasilitas untuk menampilkan atau memanipulasi peta. Paket ini berada pada "com.google.android.maps;".
2. *Location Providers* (API Location): Menyediakan teknologi pencarian lokasi yang digunakan oleh perangkat. API Location berhubungan dengan data GPS (*Global Positioning System*) dan data lokasi *real-time*. API *Location* berada pada paket Android yaitu dalam paket "android.location". Lokasi, perpindahan, serta kedekatan dengan lokasi tertentu dapat ditentukan melalui *Location Manager*.

2.12.7.2 Komponen Location Based Services

Terdapat lima komponen pendukung utama dalam teknologi Location Based Service, antara lain:

1. Piranti Mobile, adalah salah satu komponen penting dalam LBS. Piranti ini berfungsi sebagai alat bantu bagi pengguna untuk meminta informasi. Hasil dari informasi yang diminta dapat berupa teks, suara, gambar dan lain sebagainya. Piranti mobile yang dapat digunakan bias berupa PDA, smartphone, laptop.
2. Jaringan Komunikasi, Komponen ini berfungsi sebagai jalur penghubung yang dapat mengirimkan data-data yang dikirim oleh pengguna dari piranti mobile-nya untuk kemudian dikirimkan ke penyedia layanan dan kemudian

hasil permintaan tersebut dikirimkan kembali oleh penyedia layanan kepada pengguna.

3. Komponen Positioning (Penunjuk Posisi/Lokasi), Setiap layanan yang diberikan oleh penyedia layanan biasanya akan berdasarkan pada posisi pengguna yang meminta layanan tersebut. Oleh karena itu diperlukan komponen yang berfungsi sebagai pengolah/pemroses yang akan menentukan posisi pengguna layanan saat itu. Posisi pengguna tersebut bisa didapatkan melalui jaringan komunikasi mobile atau juga menggunakan Global Positioning System (GPS).
4. Penyedia layanan dan aplikasi, merupakan komponen LBS yang memberikan berbagai macam layanan yang bisa digunakan oleh pengguna. Sebagai contoh ketika pengguna meminta layanan agar bias tahu posisinya saat itu, maka aplikasi dan penyedia layanan langsung memproses permintaan tersebut, mulai dari menghitung dan menentukan posisi pengguna, menemukan rute jalan, mencari data di Yellow Pages sesuai dengan permintaan, dan masih banyak lagi yang lainnya.
5. Penyedia data dan konten, Penyedia layanan tidak selalu menyimpan seluruh data dan informasi yang diolahnya. Karena bisa jadi berbagai macam data dan informasi yang diolah tersebut berasal dari pengembang/pihak ketiga yang memang memiliki otoritas untuk menyimpannya. Sebagai contoh basis data geografis dan lokasi bisa saja berasal dari badan-badan milik pemerintah atau juga data-data perusahaan/bisnis/industri bisa saja berasal dari Yellow Pages, maupun perusahaan penyedia data lainnya. Secara lengkap kelima komponen pendukung LBS tersebut dapat dilihat pada gambar berikut.

## 2.13 *Application Programming Interface* (API)

*Application Programming Interface* (API) adalah sekumpulan perintah, fungsi, komponen, dan protokol yang disediakan oleh sistem operasi ataupun bahasa pemograman tertentu yang dapat digunakan oleh programmer saat membangun perangkat lunak [11],[26].

Dalam suatu pemograman dibutuhkan setidaknya ribuan system call per detik oleh karena itu banyak programmer yang menggunakan API. Didalam API terdapat fungsi-fungsi/perintah untuk menggantikan bahasa yang digunakan dalam system calls dengan bahasa yang lebih mudah dimengerti oleh programmer. Fungsi yang dibuat dengan menggunakan API tersebut kemudian akan memanggil system call sesuai dengan sistem operasinya. Tidak tertutup kemungkinan nama dari system call sama dengan nama API.

Kelebihan pemograman menggunakan API adalah:

1. Portabilitas

   *Programmer* yang menggunakan API dapat menjalankan programnya dengan sistem operasi mana saja asalkan sudah terinstal API tersebut.
2. Dapat dengn mudah dipahami oleh pengguna

### 2.12.8 Restful Web Services

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.

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. Berikut adalah keuntungan dari REST :

1. Bahasa dan platform agnostic

2. Lebih sederhana/simpel untuk dikembangkan ketimbang 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

Sedangkan untuk kelemahan dari REST adalah sebaga berikut ini :

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

Berikut metode HTTP 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 adalah standar yang digunakan untuk melakukan pertukaran data antar aplikasi atau sistem, karena aplikasi yang melakukan pertukaran data bisa ditulis dengan bahasa pemrograman yang berbeda atau berjalan pada platform yang berbeda. Contoh implementasi dari web service antara lain adalah SOAP dan REST. Web service yang berbasis arsitektur REST kemudian dikenal sebagai RESTful web services. Layanan web ini menggunakan metode HTTP untuk menerapkan konsep arsitektur REST.

### 2.12.9 Cara Kerja Restful Web Services

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

1. Verb, HTTP method yang digunakan misalnya GET, POST, DELETE, PUT dll.
2. *Uniform Resource Identifier* (URI) untuk mengidentifikasikan lokasi resource pada server.
3. HTTP Version, menunjukkan versi dari HTTP yang digunakan, contoh HTTP v1.1.
4. Request Header, berisi metadata untuk HTTP Request. Contoh, type client/browser, format yang didukung oleh client, format dari body pesan, seting cache dll.
5. Request Body, konten dari data.

Sedangkan komponen dari http respon**se** :

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.
4. Response Body, konten dari data yang diberikan.

## 2.14 JavaScript Object Notation (JSON)

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

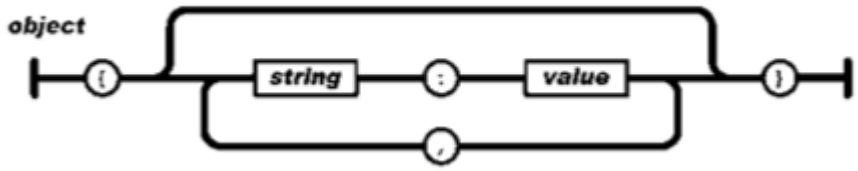
JSON terbuat dari dua struktur [27]:

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).

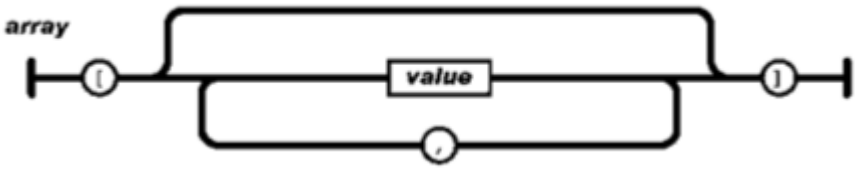
Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga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

JSON menggunakan bentuk sebagai berikut[27]:

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) seperti pada Gambar 2.6 berikut:

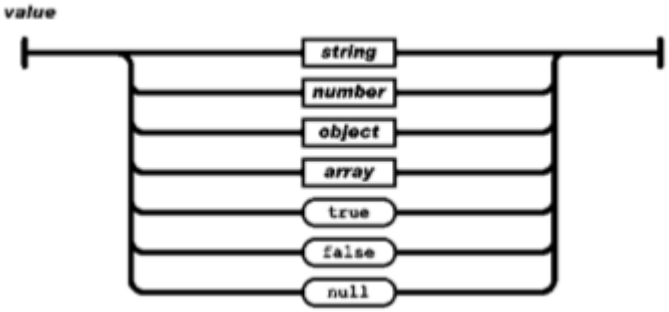



**Gambar 2 6 Objek JSON**




**Gambar 2 7 Array JSON**

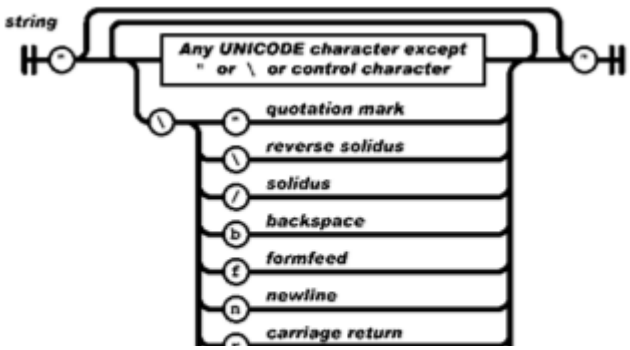
Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [ (kurung kotak buka) dan diakhiri dengan ] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma) seperti pada Gambar 2.8 berikut:.




**Gambar 2 8 Value JSON**

**Nilai** (*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.

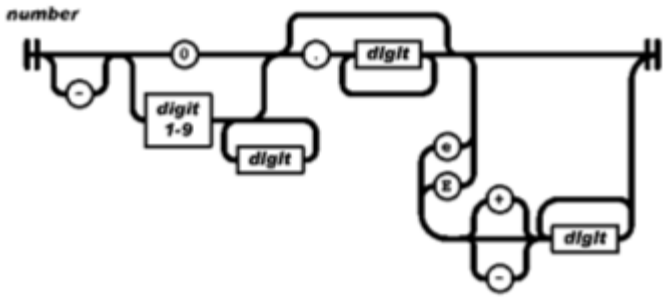
String adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan backslash




**Gambar 2 9 Sting JSON**

escapes "\" untuk membentuk karakter khusus.. String sangat mirip dengan string C atau Java.

**Angka** adalah sangat mirip dengan angka di C atau Java, kecuali format oktal dan heksadesimal tidak digunakan seperti pada gambar 2.10 berikut:




**Gambar 2 10 Number JSON**

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

## 2.15 Android Studio

Android Studio adalah sebuah IDE untuk Android Development yang diperkenalkan google pada acara Google I/O 2013. Android Studio merupakan pengembangkan dari Eclipse IDE, dan dibuat berdasarkan IDE Java populer, yaitu IntelliJ IDEA. Android Studio merupakan IDE resmi untuk pengembangan aplikasi Android[28].

Sebagai pengembangan dari Eclipse, Android Studio mempunyai banyak fitur-fit ur baru dibandingkan dengan Eclipse IDE. Berbeda dengan Eclipse yang menggunakan Ant, Android Studio menggunakan Gradle sebagai build environment. Fitur-fitur lainnya adalah sebagai berikut :

1. Menggunakan Gradle-based build system yang fleksibel.
2. Bisa mem-build multiple APK.
3. Template support untuk Google Services dan berbagai macam tipe perangkat.
4. Layout editor yang lebih bagus.

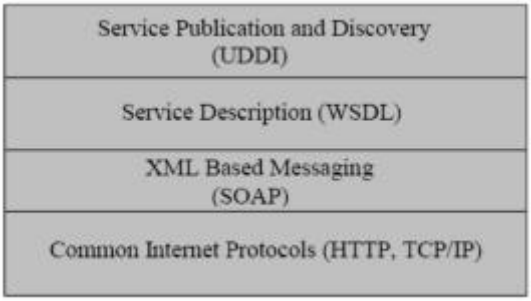
5. Built- in support untuk Google Cloud Platform, sehingga mudah untuk integras idengan Google Cloud Messaging dan App Engine.
6. Import library langsung dari Maven repository.

Satu hal tambahan lagi yang membuat Android Studio unggul adalah dukungan layout xml editor secara visual yang jauh lebih baik daripada Eclipse.

## 2.16 Web Service

Web service merupakan suatu komponen software yang merupakan selfcontaining, aplikasi modular self-describing yang dapat dipublikasikan, dialokasikan, dan dilaksanakan pada web. Web service adalah teknologi yang mengubah kemampuan internet dengan menambahkan kemampuan transactional web, yaitu kemampuan web untuk saling berkomunikasi dengan pola program-toprogram (P2P). Fokus web selama ini didominasi oleh komunikasi program-to-user dengan interaksi business-to-consumer (B2C), sedangkan transactional web akan didominasi oleh program-to-program dengan interaksi business-to-business [29]. Gambar 5 merupakan blok bangunan web service yang mana menyediakan fasilitas komunikasi jarak jauh antara dua aplikasi yang merupakan layer arsitektur web service.

1. Layer 1 : Protokol internet standar yang digunakan sebagai sarana transportasi adalah HTTP dan TCP/IP.
2. Layer 2 : Simple Object Access Protocol (SOAP) berbasiskan XML dan digunakan untuk pertukaran informasi antar sekelompok layanan.
3. Layer 3 : Web service Definition Language (WSDL) digunakan untuk mendiskripsikan attribute layanan.
4. ayer 4 : Universal Description, Discovery and Integration, yang mana merupakan direktori pusat untuk deskripsi




**Gambar 2 11 Blok Bangunan Web Service**