BAB II

LANDASAN TEORI

2.1 Android

2.1.1 Pengertian android

Android merupakan salah satu sistem operasi yang sangat berkembang saat ini, dengan berbasiskan Linux sistem operasi ini dirancang untuk mengembangkan perangkat seluler layar sentuh seperti *smartphone* dan juga komputer tablet. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi untuk digunakan oleh bermacam piranti gerak. [3]

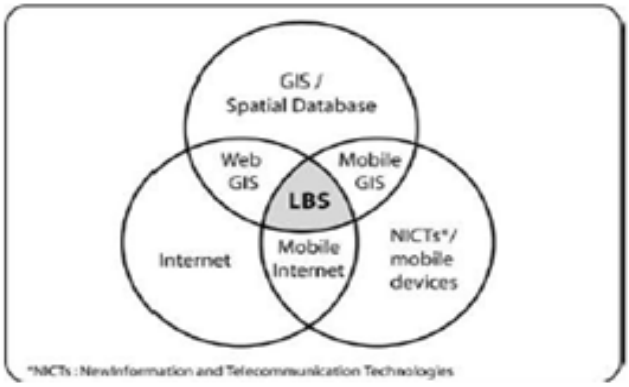
Salah satu penyebab mengapa sistem operasi Android begitu gampang diterima oleh pasar dan dengan cepatnya berkembang, itu dikarenakan android menggunakan bahasa pemrograman *Java* serta kelebihannya sebagai software yang menggunakan basis kode komputer yang bisa didistribusikan secara terbuka (*open source*) sehingga pengguna dapat membuat aplikasi baru didalamnya. Dan hal tersebut mengakibatkan banyaknya pengembang software yang berbondong untuk mengembangkan aplikasi berbasis Android. Sehingga saat ini bila dibanding dengan OS yang lain untuk perangkat handphone dan PC tablet. Android adalah yang mempunyai dukungan aplikasi dan game non berbayar terbanyak yang bisa diunduh oleh penggunanya melalui Google Play. Dengan terdapatnya fitur seperti browser, MMS, SMS, GPS, dan lain-lain maka sangat memudahkan penggunanya untuk mendapatkan informasi, mengetahui posisi, serta juga komunikasi antar para pengguna.

2.2 LOCATION BASED SERVICE

2.2.1 Pengertian Location Based Service

Location Based Service (LBS) merupakan sebuah layanan informasi yang dapat diakses dengan perangkat bergerak melalui jaringan dan mampu menampilkan posisi secara geografis keberadaan perangkat bergerak tersebut. LBS dapat berfungsi sebagai layanan untuk mengidentifikasi lokasi dari seseorang atau suatu objek tertentu, dan juga dapat beraksi aktif terhadap perubahan entitas posisi sehingga mampu mendeteksi letak objek dan memberikan layanan sesuai dengan letak objek yang telah diketahui tersebut. Pada teknologi LBS berbasis jaringan seluler, penentuan posisi sebuah peralatan komunikasi bergerak ditentukan berdasarkan posisi relatif peralatan tersebut terhadap lokasi BTS (*Base Transceiver Station*). LBS merupakan sebuah layanan *IP*- nirkabel yang menggunakan informasi geografi untuk memberikan layanan kepada pengguna perangkat *mobile*. [4]

Salah satu contoh aplikasi *local information* yang menerapkan konsep LBS di Indonesia adalah layanan informasi "*where am I*" dan "*I near you*" yang dimiliki oleh salah satu penyedia layanan GSM di Indonesia. Layanan informasi "*where am I*" dapat memberikan informasi kepada pengguna tentang lokasi pengguna dan juga dapat memberikan informasi lokasi dari pengguna lain dengan mekanisme tertentu. Layanan informasi "*I near you*" dapat memberikan informasi seperti : ATM, Restoran, SPBU yang paling dekat dengan lokasi pengguna. Kelebihan dalam aplikasi LBS ini adalah tetap berfungsi apabila kita berada di dalam gedung dan pengaruh medan elektromagnetik lain yang tidak terlalu besar, sedangkan kekurangan dari LBS ini adalah jangkauan area yang sangat bergantung pada jangkauan selular.



Gambar 2.1 Teknologi Location Based Service.[16]

2.2.2 Unsur Utama LBS

Dua unsur utama dari *Location Based Service* adalah:

1. ***Location Manager* (*API Maps*):** Menyediakan perangkat bagi sumber atau *source* untuk LBS, *Application Programming Interface* (API) *Maps* menyediakan fasilitas untuk menampilkan atau memanipulasi peta. Paket ini berada pada "com.google.android.maps;".[5]
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*. [5]

2.2.3 Komponen LBS

Terdapat lima komponen pendukung utama dalam teknologi Layanan Berbasis Lokasi, antara lain:

1. **Piranti *Mobile,*** adalah salah satu komponen penting dalam LBS. Piranti ini berfungsi sebagai alat bantu (*tool*) bagi pengguna untuk meminta informasi. Hasil dari informasi yang diminta dapat berupa teks, suara, gambar dan lain sebagainya. Piranti mobile yang dapat digunakan bisa berupa PDA,

smartphone, laptop. Selain itu, piranti *mobile* dapat juga berfungsi sebagai alat navigasi di kendaraan seperti halnya alat navigasi berbasis GPS.

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 bisa 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, dan lain-lain.
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.

Gambar 2.2 Komponen Pendukung LBS.[17]

2.3 Metode pada LBS

2.3.1 Metode *Basic Positioning*

Metode *Basic Positioning* berbasis pada identifikasi cell (Cell ID), sehingga penentuan posisi didasarkan pada daerah geografis yang tercakup oleh sebuag *cell* yang berhubungan dengan daerah cakupan dari sinyal radio. Ketika sebuah handphone/selular terhubung secara aktif dengan sebuah *base station*, berarti handphone tersebut diasumsikan berada dalam *cell* dari *base station* tersebut. Untuk mengukur jarak dan arah handsphone dari *base station* tidak dapat diketahui dengan pasti, oleh karena itu untuk lebih meningkatkan lagi akurasi hasil pencarian, metode ini seringkali dikombinasikan dengan metode lain misalnya adalah *Timing Advance(TA)* dengan menambahkan fungsi untuk menghitung *Round Trip Time (RTT)*, yaitu waktu transmisi sebuah frame dari *Base Station* ke perangkat, ataupun waktu penerimaan dari perangkat ke *Base Station*.[5]

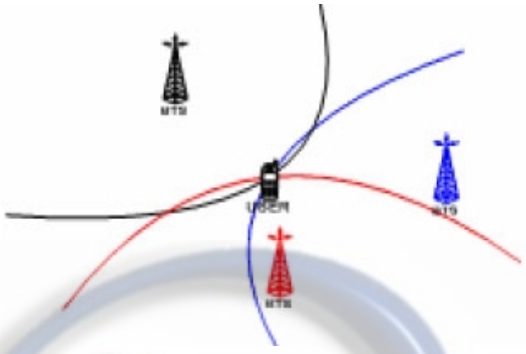
Tabel 2.1 Akurasi Metoda *Cell ID*

Metoda	**Rural/pedalaman**	**Suburban**	**Urban**	**Indoor**
Cell ID	Jangkauan 1km – 35km umumnya 15km maksimal 100km	Jangkauan 1km – 10km umumnya 5km	Macrocell: Jangkauan 500m – 5km umumnya 2km Microcell: Jangkauan 50m-500m , umumnya 200m	Jika menggunakan *picocell* biasanya mencapai 10m – 50m

Dari Data tabel tersebut dapat dilihat *Timing Advance(TA)* tidak memberikan peningkatan yang nyata dalam hal akurasi, namun bagus digunakan sebagai parameter yang baik untuk memeriksa apakah sebuah perangkat telah terhubung pada cell terdekat.

2.3.2 Metode *Enhanced Positioning*

Metode ini mengunakan pendekatan *Observe Time Difference (OTD*), namun pada jaringan GSM yang sering digunakan adalah *Enhanced-OTD (E-OTD*). E-OTD adalah metode pencarian posisi yang berdasarkan pada waktu. Untuk menentukan posisi relatif, sebuah *handphone* harus aktif terhadap tiga *base station* dan perlu ditentukan terlebih dahulu jarak *handphone* terhadap masing-masing *base station* berdsarkan waktu yang ditempuh oleh sebuah sinyal dari *handphone* ke masing-masing *base station*.



Gambar 2.3 Metode Enhanced Positioning.[19]

Tabel 2.2 Akurasi Metoda E-OTD

Metoda	**Rural/pedalaman**	**Suburban**	**Urban**	**Indoor**
E-OTD	50m – 150m	50m – 150m	50m – 150m	Bagus

Dari data tabel tersebut terlihat performansi rendah untuk daerah BTS dengan kerapatan yang rendah misalnya lingkungan pedalaman.

2.3.3 Metode Advanced Positioning

Metode ini merupakan metode penentuan posisi yang paling tinggi akurasinya dibandingkan kedua metode sebelumnya. Pada metode ini, akan dilakukan pengukuran waktu tiba dari sebuah sinyal yang dikirimkan dari 3 buah satelit GPS. Dalam hal ini handphone harus memiliki fasilitas untuk mengakses GPS, A-GPS juga dapat menghasilkan akurasi secara vertikal dan estimasi jarak yang baik.

Tabel 2.3 Akurasi Metode *Advanced Positioning*

Metode	**Rural/pedesaan**	**Suburban**	**Urban**	**Indoor**
Advanced Positioning / GPS	10m	10m – 20m	10m – 100m	Variabel

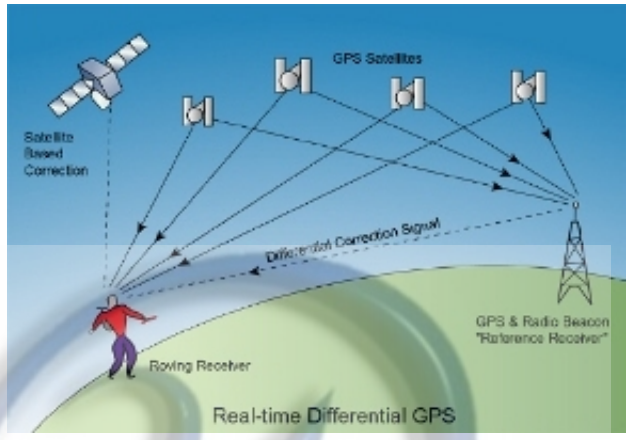
Dari data tabel tersebut dapat dilihat metode *Advanced Positioning* kurang baik dalam ruangan

2.4 GPS [6]

GPS (Global Positioning System) adalah sistem satelit navigasi dan penentuan posisi yang dimiliki dan dikelola oleh Amerika Serikat. Sistem ini didesain untuk memberikan posisi dan kecepatan tiga-dimensi serta informasi mengenai waktu, secara kontinyu di seluruh dunia tanpa bergantung waktu dan cuaca, bagi banyak orang secara simultan. Saat ini GPS sudah banyak digunakan orang di seluruh dunia dalam berbagai bidang aplikasi yang menuntut informasi tentang posisi, kecepatan, percepatan ataupun waktu yang teliti. GPS dapat memberikan informasi posisi dengan ketelitian bervariasi dari beberapa millimeter (orde nol) sampai dengan puluhan meter.

GPS memiliki kemampuan seperti dapat memberikan informasi tentang posisi, kecepatan, dan waktu secara cepat, akurat, murah, dimana saja di bumi ini tanpa tergantung cuaca. Hal yang perlu dicatat bahwa GPS adalah satu-satunya sistem navigasi ataupun sistem penentuan posisi dalam beberapa abad ini yang memiliki kemampuan handal seperti itu. Ketelitian dari GPS dapat mencapai beberapa mm untuk ketelitian posisinya, beberapa cm/s untuk ketelitian kecepatannya dan beberapa nanodetik untuk ketelitian waktunya. Ketelitian posisi yang diperoleh akan

tergantung pada beberapa faktor yaitu metode penentuan posisi, geometri satelit, tingkat ketelitian data, dan metode pengolahan datanya.



Gambar 2.4 Cara Kerja Real Time Differential GPS.[18]

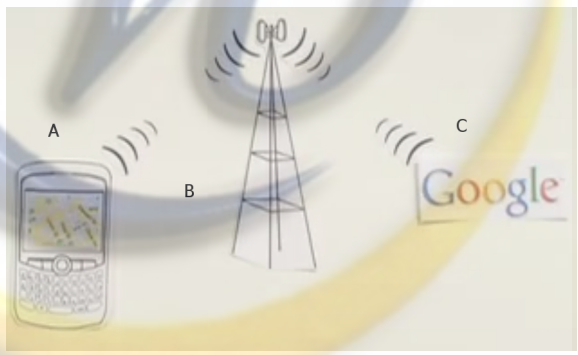
2.5 Google Maps

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 *JavaScript*. Cara membuat *Google Maps* untuk ditampilkan pada suatu web atau blog sangat mudah hanya dengan membutuhkan pengetahuan mengenai **HTML** serta ***JavaScript,*** 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. Dengan menggunakan

metode rumus *Haversine Formula* maka dapat dicari titik jarak antara lokasi dengan titik pengguna *handset* tersebut. [7]

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

1. *ROADMAP*, ini yang penulis pilih, 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)



Gambar 2.5 Sistem Kerja Google Maps.[20]

2.6 JAVA

2.6.1 Pengertian JAVA

Java adalah sebuah teknologi yang diperkenalkan oleh Sun Microsystems pada pertengahan tahun 1990. Menurut definisi dari Sun, Java adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringan. Kita lebih menyukai

menyebut Java sebagai sebuah teknologi dibanding hanya sebuah bahasa pemrograman, karena Java lebih lengkap dibanding sebuah bahasa pemrograman konvensional. Teknologi Java memiliki tiga komponen penting, yaitu:

- *Programming-language specification*
- *Application-programming interface*
- *Virtual-machine specification* [8]

2.6.2 Java Virtual Machine

Java Virtual Machine (JVM) adalah sebuah spesifikasi untuk sebuah komputer abstrak. JVM terdiri dari sebuah kelas pemanggil dan sebuah interpreter Java yang mengeksekusi kode arsitektur netral. Kelas pemanggil memanggil file .class dari kedua program Java dan Java API untuk dieksekusi oleh interpreter Java. Interpreter Java mungkin sebuah perangkat lunak interpreter yang menterjemahkan satu kode byte pada satu waktu, atau mungkin sebuah just-intime (JIT) kompiler yang menurunkan*bytecode* arsitektur netral kedalam bahasa mesin untuk *host computer*. [8]

2.6.3 Sistem Operasi Java

Sistem operasi biasanya ditulis dalam sebuah kombinasi dari kode bahssa C dan assembly, terutama disebabkan oleh kelebihan performa dari bahasa tersebut dan memudahkan komunikasi dengan perangkat keras.

Satu kesulitan dalam merancang sistem basis bahasa adalah dalam hal proteksi memori, yaitu memproteksi sistem opeasi dari pemakai program yang sengaja memproteksi pemakai program lainnya. Sistem operasi tradisional mengaharapkan pada tampilan perangkat keras untuk menyediakan proteksi memori. Sistem basis bahasa mengandalkan pada tampilan keamanan dari bahasa. Sebagai hasilnya, sistem basis bahasa menginginkan pada alat perangkat keras kecil, yang mungkin kekurangan tampilan perangkat keras yang menyediakan proteksi memori. [8]

2.6.4 Dasar Pemrograman

Java2 adalah generasi kedua dari Java *platform* (generasi awalnya adalah Java Development Kit). Java berdiri di atas sebuah mesin interpreter yang diberi nama JVM. JVM inilah yang akan membaca *bytecode* dalam file *.class* dari suatu program sebagai representasi langsung program yang berisi bahasa mesin. Oleh karena itu, bahasa Java disebut sebagai bahasa pemrograman yang *portable* karena dapat dijalankan pada berbagai sistem operasi, asalkan pada sistem operasi tersebut terdapat JVM.

Platform Java terdiri dari kumpulan *library*, JVM, kelas- kelas *loader* yang dipaket dalam sebuah lingkungan rutin Java, dan sebuah *compiler*, *debuger*, dan perangkat lain yang dipaket dalam Java Development Kit (JDK). Java2 adalah generasi yang sekarang sedang berkembang dari *platform*Java. Agar sebuah program Java dapat dijalankan, maka file dengan ekstensi ".java" harus dikompilasi menjadi file *bytecode*. Untuk menjalankan *bytecode* tersebut dibutuhkan JRE (*Java Runtime Environment*) yang memungkinkan pemakai untuk menjalankan program Java, hanya menjalankan, tidak untuk membuat kode baru lagi. JRE berisi JVM dan *library*Java yang digunakan.

*Platform*Java memiliki tiga buah edisi yang berbeda, yaitu J2EE (*Java2 Enterprose Edition*), J2ME (*Java2 Micro Edition*) dan J2SE (*Java2 Second Edition*). J2EE adalah kelompok dari beberapa API (*Application Programming Interface*) dari Java dan teknologi selain Java. J2EE sering dianggap sebagai *middleware* atau teknologi yang berjalan di *server*, namun sebenarnya J2EE tidak hanya terbatas untuk itu. Faktanya J2EE juga mencakup teknologi yang dapat digunakan di semua lapisan dari sebuah sistem informasi. Implementasi J2EE menyediakan kelas dasar dan API dari Java yang mendukung pengembangan dari rutin standar untuk aplikasi klien maupun *server*, termasuk aplikasi yang berjalan di *web browser*. J2SE adalah lingkungan dasar dari Java, sedangkan J2ME merupakan edisi *library* yang dirancang untuk digunakan pada *device* tertentu seperti *pagers* dan *mobile phone*.

Java merupakan bahasa pemrogrman yang bersifat *case sensitive* yang berarti penulisan menggunakan huruf besar ataupun huruf kecil pada kode program dapat berarti lain. Misalnya penulisan "System" akan diartikan berbeda dengan "system" oleh interpreter. Java tidak seperti C++, Java tidak mendukung pemrograman prosedural, tapi mendukung pemrograman berorientasi objek sehingga ada sintaks *class* pada kode programnya. [8]

2.6.5 JDK (*Java Development Kit*)

Java Development Kit (JDK) merupakan bagian terpenting dalam pengembangan aplikasi android, karena Android merupakan aplikasi yang dibangun dengan menggunakan bahasa pemrograman Java. Untuk mendapatkan installer JDK bisa mengunduh / medownload langsung di situs resminya. Pilihlah installer JDK (java) yang sesuai dengan sistem operasi komputer kita. JDK yang bisa digunakan untuk membuat program Android adalah JDK 5 dan 6 atau versi terbarunya. [8]

2.6.6 SDK *(Software Development Kit)*

Android SDK adalah tools API (*Aplication Programming Interface*) yang digunakan untuk mulai mengembangkan aplikasi pada platform android menggunakan bahasa pemrograman Java. Android merupakan subset perangkat lunak untuk ponsel yang meliputi sistem operasi, middleware dan aplikasi kunci yang di release oleh Google. Saat ini disediakan android SDK (*Software Development Kit*) sebagai alat bantu dan API untuk mulai mengembangkan aplikasi pada platform android menggunakan bahasa pemrograman Java. Sebagai platform aplikasi netral, android memberi anda kesempatan untuk membuat aplikasi yang kita butuhkan yang bukan merupakan aplikasi bawaan hanphone/smartphone. Beberapa fitur android yang paling penting adalah :

- *Framework* aplikasi yang mendukung penggantian komponen dan reusable.
- Mesin *Virtual Dalvik* dioptimalkan untuk perangkat mobile.

- *Integrated browser* berdasarkan engine open source webkit.
- Grafis yang dioptimalkan dan didukung oleh libraries grafis 2D, grafis 3D berdasarkan spesifikasi *openGL* ES 1.0 (Opsional Akselerasi Hardware).
- *SQLite* untuk penyimpanan data.
- Media Support yang mendukung audio, video dan gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF), GSM Telephony (tergantung hardware).
- Bluetooth, EDGE, 3G, WiFi (tergantung hardware).
- Kamera, GPS, kompas dan Accelerator (tergantung hardware).
- Lingkungan development yang lengkap dan kaya, termasuk perangkat emulator, tools untuk debuging, profil dan kinerja memori, dan plug in untuk *IDE Eclipse*. [8]

2.6.7 AVD (Android Virtual Device)

Android Virtual Devices (AVD) adalah konfigurasi dari emulator sehingga kita dapat menjalankan perangkat Android sesuai model yang dipilih, misal Android 1.5 atau 2.2. Untuk dapat menjalankan emulator, Anda harus terlebih dahulu memiliki Android SDK yang telah ter install.

Setiap AVD terdiri dari:

1. Sebuah profil perangkat keras. Anda dapat mengatur opsi untuk menentukan fitur hardware emulator. Misalnya, Anda dapat menentukan apakah menggunakan perangkat kamera, apakah menggunakan keyboard QWERTY fisik atau tidak, berapa banyak memori internal, dan lain-lain.
2. Sebuah pemetaan versi Android. Anda dapat menentukan versi dari platform Android akan berjalan pada emulator.
3. Pilihan lainnya. Anda dapat menentukan skin yang ingin Anda gunakan pada emulator, yang memungkinkan Anda menentukan dimensi layar, tampilan,

dan sebagainya. Anda juga dapat menentukan SD Card virtual untuk digunakan dengan di emulator. [8]

2.6.8 Eclipse

Eclipse adalah sebuah **IDE** (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (*platform-independent*). Berikut ini adalah sifat dari Eclipse:

- **Multi-platform**: Target sistem operasi Eclipse adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.
- **Mulit-language**: Eclipse dikembangkan dengan bahasa pemrograman Java, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, Python, Perl, PHP, dan lain sebagainya.
- **Multi-role**: Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya.

Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan *open source*, yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari Eclipse yang membuatnya populer adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug-in*.

Penggunaan Eclipse sebenarnya bersifat optional, artinya bisa digunakan atau pun tidak. Karena Eclipse bisa digantikan oleh editor lainnya. Namun saya lebih menyarankan Eclipse karena Eclipse bisa memudahkan kita dalam hal pembuatan program Android dan juga karena Eclipse masih berbasis teks. Jadi kita bakal melakukan pemrograman dengan menulis *source code* terus menerus bukan dengan

cara *drag 'n drop*. Hal ini membantu kita dalam memahami setiap baris kode yang kita butuhkan untuk membuat aplikasi Android. Kalau belajar lebih baik pilih yang di tengah-tengah saja, jangan terlalu mudah dan jangan juga terlalu susah. Klo kita memilih metode yang terlalu susah juga, kemungkinan kita tidak akan memulainya karena sudah terbayang di kepala kita betapa sulitnya hal tersebut.

Eclipse bisa di download di situs resminya. Eclipse yang bisa digunakan adalah Eclipse yang mendukung pengembangan pemrograman berbasis Java. Versi yang direkomendasikan adalah Eclipse versi 3.5 Galileo atau versi 3.4 Ganymade. Hal ini karena terdapat sedikit masalah dengan Eclipse 3.6 Helios, walaupun ada beberapa pengembang yang pernah mencoba menggunakan Helios dan dapat berjalan dengan baik untuk Android. [8]

2.7 Konsep Dasar *Database*

2.7.1 Pengertian *Database*

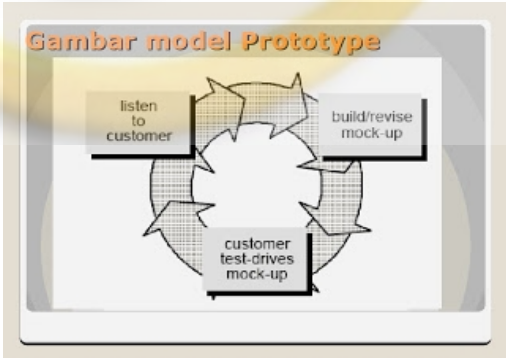
1. *Database* adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut.
2. *Database* adalah representasi kumpulan fakta yang saling berhubungan disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redudansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan.
3. *Database* merupakan sekumpulan informasi yang saling berkaitan pada suatu subjek tertentu pada tujuan tertentu pula.
4. *Database* adalah susunan record data operasional lengkap dari suatu organisasi atau perusahaan, yang diorganisir dan disimpan secara terintegrasi dengan menggunakan metode tertentu dalam komputer sehingga mampu memenuhi informasi yang optimal yang dibutuhkan oleh para pengguna. [9]

2.8 Metode pengembangan *software* dengan model *Prototyping*

Prototyping adalah proses pembuatan model sederhana software yang mengijinkan pengguna memiliki gambaran dasar tentang program serta melakukan pengujian awal. Prototyping memberikan fasilitas bagi pengembang dan pemakai untuk saling berinteraksi selama proses pembuatan, sehingga pengembang dapat dengan mudah memodelkan perangkat lunak yang akan dibuat. Prototyping merupakan salah satu metode pengembangan perangat lunak yang banyak digunakan.

Model tersebut dapat berupa tiga bentuk :

1. Prototype kertas atau model berbasis komputer yang menjelaskan bagaimana interaksi antara pemakai dan komputer.
2. Prototype yang mengimplementasikan beberapa bagian fungsi dari perangkat lunak yang sesungguhnya. Dengan cara ini pemakai akan lebih mendapatkan gambaran tentang program yang akan dihasilkan, sehingga dapat menjabarkan lebih rinci kebutuhannya.
3. Menggunakan perangkat lunak yang sudah ada. Seringkali pembuat software memiliki beberapa program yang sebagian dari program tersebut mirip dengan program yang akan dibuat. [10]



Gambar 2.6 Model Prototype.[10]

Prototyping merupakan *Javascript Framework* yang dibuat untuk lebih memudahkan proses dalam membangun aplikasi berbasis web. Metode prototyping

sebagai suatu paradigma baru dalam pengembangan sistem informasi, tidak hanya sekedar suatu evolusi dari metode pengembangan sistem informasi yang sudah ada, tetapi sekaligus merupakan revolusi dalam pengembangan sistem informasi manajemen.

Proses-proses tersebut dapat dijelaskan sebagai berikut: [10]

1. Pengumpulan kebutuhan: *developer* dan klien (*user)* bertemu dan menentukan tujuan umum, kebutuhan yang diketahui dan gambaran bagian-bagian yang akan dibutuhkan berikutnya.
2. Perancangan: perancangan dilakukan cepat dan rancangan mewakili semua aspek software yang diketahui, dan rancangan ini menjadi dasar pembuatan prototype.
3. Evaluasi Prototype: klien mengevaluasi prototype yang dibuat dan digunakan untuk memperjelas kebutuhan software

Protoyping memiliki tiga pendekatan utama, yaitu: [10]

1. THROW-AWAY

Prototype dibuat dan dites. Pengalaman yang diperoleh dari pembuatan prototype digunakan untuk membuat produk akhir (final), kemudian prototype tersebut dibuang (tak dipakai).

2. INCREMENTAL

Produk finalnya dibuat sebagai komponen-komponen yang terpisah. Desain produk finalnya secara keseluruhan haya ada satu tetapi dibagi dalam komonen-komponen lebih kecil yang terpisah (independent).

3. EVOLUTIONARY

Pada metode ini, prototipenya tidak dibuang tetapi digunakan untuk iterasi desain berikutnya. Dalam hal ini, sistem atau produk yang sebenarnya dipandang sebagai evolusi dari versi awal yang sangat terbatas menuju produk final atau produk akhir.

Berikut adalah penjelasan tahapan dalam prototyping :

1. Pengumpulan kebutuhan

User dan *Developer* bersama-sama mendefinisikan format seluruh perangkat lunak, mengidentifikasikan semua kebutuhan, dan garis besar sistem yang akan dibuat.

2. Membangun prototyping

Membangun prototyping dengan membuat perancangan sementara yang berfokus pada penyajian kepada *user* (misalnya dengan membuat input dan format output).

3. Evaluasi protoptyping

Evaluasi ini dilakukan oleh *user* apakah prototyping yang sudah dibangun sudah sesuai dengan keinginan *user*. Jika sudah sesuai maka langkah 4 akan diambil. Jika tidak prototyping direvisi dengan mengulangi langkah pengumpulan kebutuhan, membangun prototyping , dan evaluasi prototyping.

4. Mengkodekan sistem

Dalam tahap ini prototyping yang sudah di sepakati diterjemahkan ke dalam bahasa pemrograman yang sesuai.

5. Menguji system

Setelah sistem sudah menjadi suatu perangkat lunak yang siap pakai, harus dites dahulu sebelum digunakan. Pengujian ini dilakukan dengan *White Box*, *Black Box*, dan lain-lain.

6. Evaluasi Sistem.

User mengevaluasi apakah sistem yang sudah jadi sudah sesuai dengan yang diharapkan . Jika ya, gunakan sistem; jika tidak, kembali menkodekan system dan menguji sitem.

7. Menggunakan system

Perangkat lunak yang telah diuji dan diterima *user* siap untuk digunakan .

Keunggulan dan Kelemahan Prototyping adalah sebagai berikut :

A. Keunggulan prototyping :

1. Adanya komunikasi yang baik antara *developer* dan *user*.
2. *developer* dapat bekerja lebih baik dalam menentukan kebutuhan *user*.
3. *User* berperan aktif dalam pengembangan system.
4. Lebih menghemat waktu dalam pengembangan system.
5. Penerapan menjadi lebih mudah karena pemakai mengetahui apa yang diharapkannya

B. Kelemahan prototyping :

1. *User* kadang tidak melihat atau menyadari bahwa perangkat lunak yang ada belum mencantumkan kualitas perangkat lunak secara keseluruhan dan juga belum memikirkan kemampuan pemeliharaan untuk jangka waktu lama.
2. *Developer* biasanya ingin cepat menyelesaikan proyek. Sehingga menggunakan algoritma dan bahasa pemrograman yang sederhana untuk membuat prototyping lebih cepat selesai tanpa memikirkan lebih lanjut bahwa program tersebut hanya merupakan cetak biru sistem .
3. Hubungan *user* dengan komputer yang disediakan mungkin tidak mencerminkan teknik perancangan yang baik.

2.9 *Unified Modelling Language (UML)*

UML adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan mendokumentasi sebiah system pengembangan software berbasis *object-oriented*. Serta memberikan standar penulisan sebuah system blue print, yan meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema database, dan komponen-komponen yang diperlukan dalam system *software*. Tidak hanya merupakan sebuah

bahasa pemograman visual saja, namun juga dapat secara langsung dihubungkan ke berbagai bahasa pemograman, seperti *JAVA, C++, Visual Basic*, atau bahkan dihubungkan secara langsung ke dalam sebuah *object-oriented database*. Begitu juga mengenai pendokumentasian dapat dilakukan seperti; *requirements*, arsitektur, *design, source code, project plan, tests,* dan *prototypes.* [11]

UML menawarkan sebuah standar untuk merancang model sebuah system. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi, deimana aplikasi tersebut dapat berjalan pada piranti keras, system operasi dan jaringan apapun, sertra ditulis dalam bahasa pemrograman apapun. Berikut adalah jenis diagram yang ada pada UML :

2.9.1 *Use Case Diagram*

Rangkaian/uraian sekelompok yang saling terkait dan membentuk system secara teratur yang dilakukan atau diawasi oleh sebuah actor. "use case" digunakan untuk membentuk tingkah-laku benda dalam sebuah model serta direalisasikan oleh sebuah *collaboration diagram. Use case* diagram dapat sangat membantu saat sedang menyusun *requirement* pada sebuah system, mengkomunikasikan rancangan dengan *user*, dan menguji semua *feature* yang ada pada system tersebut. [11]

2.9.2 *Class Diagram*

Diuraikan sebagai sekelompok dar object yang mempunyai *attribute*, operasi, hubungan yang sematik, sehingga suatu *class* dapat mengimplementasikan 1 atau lebih *interfaces*. Sebuah class dapat berhubungan dengan yang lain melalui berbagai cara, yaitu: *associated* (terhubunga satu sama lain), *dependent* (satu class tergantung/menggunakan *class* yang lain), *specialed* (satu *class* merupakan spesialisasi dari *class* lainnya), *package* (grup bersama sebagai satu unit), yang juga dapat digambarkan sebagai sebuah persegi panjang, yang mempunyai sebuah nama, attribute, dan metoda pengoperasiannya, sehingga sangat membantu dalam visualisasi struktur *class* dari suatu system. [11]

2.9.3 *State Diagram*

Menggambarkan semua *state* (kondisi) yang dimiliki oleh suatu object dari suatu *class* dan keadaan yang menyebabkan *state* berubah. Kejadian dapat berupa object lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class*, hanya yang mempunyai sejumlah *state* yang terdefinisi dengan baik dan kondisi *class* berubah oleh *state* yang berbeda. [11]

2.9.4 *Activity Diagram*

Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *use case* atau interaksi. *Activity* diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi dan juga merupakan *state* diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity* diagram tidak menggambarkan *behaviour internal* sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari *level* atas secara umum. Menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses. Dipakai pada *business modeling* untuk memperlihatkan urutan aktifitas proses bisnis. Struktur diagram ini mirip *flowcha*rt atau *Data Flow* Diagram pada perancangan terstruktur. Sangat bermanfaat apabila kita membuat diagram ini terlebih dahulu dalam memodelkan sebuah proses untuk membantu memahami proses secara keseluruhan. [11]

2.9.5 *Sequence Diagram*

suatu diagram yang memperlihatkan atau menampilkan interaksi-interaksi antar objek di dalam sistem yang disusun pada sebuah urutan atau rangkaian waktu.

Interaksi antar objek tersebut termasuk *user*, *display*, dan sebagainya berupa pesan/*message*. *Sequence* Diagram digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai sebuah respon dari suatu kejadian/*event* untuk menghasilkan output tertentu.

Sequence Diagram diawali dari apa yang me-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan. Diagram ini secara khusus berasosiasi dengan *use case* diagram. *Sequence* diagram juga memperlihatkan tahap demi tahap apa yang seharusnya terjadi untuk menghasilkan sesuatu didalam *use case*. *Sequence* diagram juga dapat merubah atribut atau *method* pada *class* yang telah dibentuk oleh *class* diagram, bahkan menciptakan sebuah *class* baru. *Sequence* diagram memodelkan aliran logika dalam sebuah system dalam cara yang visual. *Sequence* diagram biasanya digunakan untuk tujuan analisa dan desain, memfokuskan pada identifikasi method didalam sebuah system. *Sequence* diagram biasanya dipakai untuk memodelkan :

- Deskripsi tentang system yang ada pada sebuah / beberapa *use case* pada *use case* diagram, yang menggambarkan hubungan antara *actor* dan *use case* diagram.
- Logika dari *method* (*operation, function* atau *procedure*).
- Logika dari *service* (*high level method*). [11]

2.9.6 *Collaboration Diagram*

Didefinisikan dengan interaksi dan sebuah kumpulan/kelompok dari *class*/elemen yang bekerja secara bersama serta mempunyai struktur dan dimensi. Pemberian sebuah *class* memungkinkan berpartisipasi didalam beberapa *collaboration* dan digambarkan dengan sebuah "*elips*" dengan garis berpotong. Jika menggunakan diagram ini maka setiap pesan memiliki *sequence number* yang dimana pesan dari *level* tertinggi memiliki nomor 1. Pesan dari *level* yang sama memiliki prefix yang sama. [11]

2.9.7 *Component Diagram*

Adalah diagram UML yang menampilkan komponen dalam system dan hubungan antara mereka. Pada *component View*, akan difokuskan pada organisasi fisik system. Pertama, diputuskan bagaimana kelas-kelas akan diorganisasikan menjadi kode pustaka. Kemudian akan dilihat bagaimana perbedaan antara berkas eksekusi, berkas *dynamic link library (DDL*), dan berkas *runtime* lainnya dalam system. Merupakan Komponen peranti lunak yang berisikan modul *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time, link time* maupun *run time*. Pada umumnya komponen terbentuk dari bebrapa *class*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen tersebut dapat berupa *interface*, yang merupakan kumpulan layanan yang disediakan sebuah komponen untuk komponen lain. [11]

2.9.8 *Deployment Diagram*

Deployment View adalah pandangan yang terkait dengan penyebaran fisik aplikasi. Hal ini termasuk persoalan layout jaringan dan lokasi komponen-komponen dalam jaringan. *Deployment View* berisikan prosesor-prosesor, peralatan-peralatan, proses-proses dan hubungan antar prosesor dan antar perlatan. Semua informasi ini digambarkan dalam suatu *deployment* Diagram. Hanya ada satu *deployment* Diagram dalam setiap system , sehingga hanya satu *deployment* dalam setiap model. Suatu *deployment* diagram menampilkan semua titik (*node*) dalam suatu jaringan, hubungan antar mereka, dan proses-proses yang dijalankan pada masing-masing *node*. [11]

2.10 MySQL

MySQL adalah sebuah implementasi dari sistem manajemen basis data relasional (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL(*General Public License*). Setiap pengguna dapat secara bebas menggunakan MySQL, namun

dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basisdata yang telah ada sebelumnya; SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian basis data, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis. [12]

Kehandalan suatu sistem basis data (DBMS) dapat diketahui dari cara kerja pengoptimasi-nya dalam melakukan proses perintah-perintah SQL yang dibuat oleh pengguna maupun program-program aplikasi yang memanfaatkannya. Sebagai peladen basis data, MySQL mendukung operasi basisdata transaksional maupun operasi basisdata non-transaksional. Pada modus operasi non-transaksional, MySQL dapat dikatakan unggul dalam hal unjuk kerja dibandingkan perangkat lunak peladen basisdata kompetitor lainnya. Namun demikian pada modus non-transaksional tidak ada jaminan atas reliabilitas terhadap data yang tersimpan, karenanya modus non-transaksional hanya cocok untuk jenis aplikasi yang tidak membutuhkan reliabilitas data seperti aplikasi blogging berbasis web (*wordpress*), CMS, dan sejenisnya. Untuk kebutuhan sistem yang ditujukan untuk bisnis sangat disarankan untuk menggunakan modus basisdata transaksional, hanya saja sebagai konsekuensinya unjuk kerja MySQL pada modus transaksional tidak secepat unjuk kerja pada modus non-transaksional.[12]

2.11 Teknik Pengujian Sistem

2.11.1 Whitebox

Ujicoba *Whitebox* merupakan metode desain uji kasus yang menggunakan struktur kontrol dari desain prosedural untuk menghasilkan kualitas suatu pengujian. Dengan menggunakan metode ujicoba *whitebox*, para pengembang software dapat menghasilkan kualitas suatu pengujian yang :

1. Menjamin bahwa seluruh independent path dalam modul telah dilakukan sedikitnya satu kali.
2. Melakukan seluruh keputusan logikal baik dari sisi benar maupun salah.
3. Melakukan seluruh perulangan sesuai batasannya dan dalam batasan operasionalnya.
4. Menguji struktur data internal untuk memastikan validitasnya.[14]

2.11.2 Blackbox

Metode uji coba *blackbox* memfokuskan pada keperluan fungsional dari *software*. Karna ujicoba *blackbox* memungkinkan pengembang *software* untuk membuat himpunan kondisi input yang akan melatih seluruh syarat-syarat fungsional suatu program. Ujicoba *blackbox* bukan merupakan alternatif dari ujicoba *whitebox*, tetapi merupakan pendekatan yang melengkapi untuk menemukan kesalahan lainnya, selain menggunakan metode *whitebox*. Ujicoba *blackbox* berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya :

1. Fungsi-fungsi yang salah atau hilang
2. Kesalahan *interface*
3. Kesalahan dalam struktur data atau akses database eksternal
4. Kesalahan performa
5. kesalahan inisialisasi dan terminasi.[15]