BAB II

LANDASAN TEORI

Pada bab ini akan diuraikan mengenai landasan teori yang berkaitan dengan topik yang dibahas. Dasar-dasar teori yang diuraikan merupakan dasar teori tentang Android dan yang berkaitan dengan aplikasi serta perangkat bergerak yang memiliki sistem operasi Android. Dasar-dasar teori ini juga diperlukan dari berbagai referensi yang memuat teori-teori yang berkaitan dengan topik yang dibahas.

2.1 Pengertian Android

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi.

Android merupakan generasi baru *platform mobile* yang memberikan kesempatan kepada pengembang untuk melakukan pengembangan sesuai dengan yang diharapkan. Sistem operasi yang mendasari Android merupakan lisensi di bawah naungan GNU, General *Public* License Versi 2(GPLv2), yang biasa dikenal dengan istilah *Copyleft*. Istilah *copyleft* ini merupakan lisensi yang setiap perbaikan oleh pihak ketiga harus terus jatuh di bawah *terms*.

Distribusi Android berada di bawah lisensi Apache Software (ASL/Apache2), yang memungkin untuk distribusi kedua atau seterusnya. Pengembang aplikasi Android diperbolehkan untuk mendistribusikan aplikasi mereka di bawah skema lisensi apapun yang mereka inginkan.

Pengembang memiliki beberapa pilihan dalam membuat aplikasi yang berbasis Android. Namun kebanyakan pengembang menggunakan Eclipse sebagai IDE untuk merancang aplikasi mereka. Hal ini diikarenakan Eclipse mendapat dukungan langsung dari Google untuk menjadi IDE pengembangan aplikasi Android.

Aplikasi Android dapat dikembangkan pada berbagai sistem operasi, diantaranya adalah:

a. Windows XP/Vista/7

b. Mac OS X (Mac OS X 10.48 atau yang lebih baru)

c. Linux

2.1.1 Sejarah Android

Pada saat perilisan perdana Android pada tanggal 5 November 2007, Android bersama Open Handset Alliance mendukung pengembangan standar terbuka pada perangkat seluler tersebut. Di sisi lain, Google merilis kode-kode Android di bawah lisensi Apache. Sehingga terdapat dua jenis *distributor* sistem operasi Android yaitu yang mendapat dukungan penuh dari Google dan yang mendapat dukungan penuh dari Open Handset Distribution (OHD).

Telepon selular pertama yang menggunakan sistem operasi Android adalah HTC Dream yang dirilis pada 22 Oktober 2008. Pada 9 Desember 2008, diumumkan anggota baru yang bergabung dalam program kerja Android ARM Holdings, Atheros Communication yang diproduksi oleh Asustek Computer Inc, Garmin Ltd, Softbank, Sony Ericsson, Toshiba Corp dan Vodafone Group Plc.

Hingga saat ini terdapat beberapa versi dari sistem operasi Android, antara lain:

a. Android versi 1.1

Dirilis pada 9 Maret 2009. Android versi ini dilengkapi dengan adanya jam, alarm, *voice search*, pengiriman pesan dengan Gmail dan pemberitahuan *email*.

b. Android versi 1.5 (Cupcake)

Dirilis pada Mei 2009. Terdapat pembaruan dari versi 1.1 diantaranya adalah fitur *upload* video ke Youtube dan gambar ke Picasa langsung dari telepon, dukungan *bluetooth* A2DP, kemampuan terhubung secara otomatis ke headset *bluetooth*, animasi layar, dan *keyboard* pada layar yang dapat disesuaikan dengan sistem.

c. Android versi 1.6 (Donut)

Dirilis pada September 2009. Pembaruan yang terdapat pada versi ini diantaranya adalah proses pencarian yang lebih baik, penggunaan baterai indikator dan kontrol applet VPN. Fitur lainnya adalah memungkinkan pengguna untuk memilih foto yang akan dihapus, kamera, camcorder dan galeri yang diintegrasikan, CDMA/EVDO, 802.1x, VPN, Gestures, *Text-to-speech engine*.

d. Android versi 2.1 (Éclair)

Dirilis pada 3 Desember 2009. Perubahan yang dilakukan adalah pengoptimalan *hardware*, peningkatan Google Maps 3.1.2, perubahan UI dengan *browser* baru dan dukungan HTML5, daftar kontak yang baru, dukungan *flash* untuk kamera 3.2 MP, digital zoom dan *bluetooth* 2.1.

e. Android versi 2.2 (Froyo)

Dirilis pada 20 Mei 2010. Versi Android inilah yang sekarang banyak digunakan sebagai standar sistem operasi mereka. Terdapat perubahan yang cukup

signifikan dari versi sebelumnya diantaranya adalah kerangka aplikasi memungkinkan penggunaan dan penghapusan komponen yang tersedia, Dalvik Virtual Machine (DVM) yang dioptimalkan untuk perangkat *mobile*, grafik di 2D dan 3D berdasarkan *libraries* OpenGL, SQLite, mendukung berbagai format audio dan video, GSM, *bluetooth*, EDGE, 3G, Wifi, kamera, *Global Positioning System* (GPS), kompas dan accelerometer.

f. Android versi 2.3 (GingerBread)

Dirilis pada 6 Desember 2010. Beberapa perbaikan fitur dari versi sebelumnya adalah SIP-*based* VoIP, *Near Field Communications* (NFC), *gyroscope* dan sensor, *multiple cameras support*, *mixable audio effect* dan *download manager*.

g. Android versi 3.0 (Honeycomb)

Dirilis tahun 2011. Android versi ini dirancang khusus untuk tablet, sehingga terdapat perbedaan dari fitur UI (*User Interface*). Honeycomb sengaja dibuat untuk layar yang lebih besar dan juga dapat mendukung *multiprocessor*.

h. Android versi 4.0 (Ice Cream)

Versi ini masih dalam pengembangan. Dari berbagai informasi menyebutkan bahwa versi Ice Cream merupakan gabungan antara versi Gingerbread dengan Honeycomb. Sehingga bisa digunakan untuk ponsel maupun tablet. Dan kemungkinan dirilis pada quarter ke 4 tahun 2011.

2.1.2 Pengertian *Smartphone* (Telepone Pintar)

Smartphone adalah sebuah telepon yang kegunaan dasarnya sama dengan telepon biasa yang dapat dibawa kemana-mana dan tidak perlu disambungkan

dengan kabel, namun memiliki kemampuan tingkat tinggi dengan fungsi yang menyerupai komputer.

2.1.3 *Java Software Development Kit* (SDK Java)

SDK Java merupakan kebutuhan utama bagi programmer untuk membuat dan menjalankan java. Komponen JDK antara lain compiler(javac), interpreter(java) disebut juga java virtual machine atau java runtime environment, applet viewer(appletviewer), debugger(jdb), java class library(jcl), header dan stub generator(javah), dan yang paling penting yaitu java documentation(javadoc).

2.1.4 *Eclipse Software Development Kit* (SDK Eclipse)

SDK Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (*platform-independent*).

2.1.5 *Android Software Development Kit* (SDK Android)

SDK Android adalah tools API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java.

2.1.6 *Android Eclipse Plugin* (ADT)

Definisi *Android Eclipse Plugin* adalah suatu perangkat tambahan koneksi Eclipse dengan Android SDK, sehingga bisa lebih memudahkan membuat program untuk android di Eclipse.

2.1.7 *Android Package* (APK)

APK adalah paket aplikasi Android (**Android PacKage**). APK umumnya digunakan untuk menyimpan sebuah aplikasi atau program yang akan dijalankan pada perangkat Android. APK pada dasarnya seperti zip file, karena berisi dari kumpulan file, dapat diperoleh melalui berbagai metode, seperti menginstal sebuah aplikasi melalui Market, download dari sebuah situs web, atau membuat sendiri dengan bahasa Java.

2.1.8 *Android Virtual Devices* (AVD)

AVD adalah konfigurasi dari emulator sehingga kita dapat menjalankan perangkat Android sesuai model yang dipilih, misalkan Android 1.5 atau 2.2. Untuk dapat menjalankan emulator.

2.1.9 Emulator

Emulator adalah Aplikasi yang memungkinkan sebuah software dari sebuah platform berjalan di atas platflorm lain. Emulator menjalankan kode-kode software pada virtual machine. Sehingga software tersebut mengira bahwa ia sedang berjalan di atas platform aslinya. Dalam hal ini emulator android dapat membuat *virtual phone* di komputer seperti telepon genggam berbasis sistem operasi android yang sebenarnya.

2.2 Unified Modeling Language (UML)

Unified Modeling Language (UML) adalah bahasa pemodelan secara grafis untuk menspesifikasikan, menvisualisasikan, membangun, dan

mendokumentasikan seluruh rancangan sistem perangkat lunak. Penggunaan model ini bertujuan untuk mengidentifikasikan bagian-bagian yang termasuk dalam lingkup sistem yang dibahas dan bagaimana hubungan antara sistem dengan subsistem maupun sistem lain di luarnya.

Selain itu UML adalah bahasa pemodelan yang menggunakan konsep orientasi object. UML dibuat oleh Grady Booch, James Rumbaugh, dan Ivar Jacobson di bawah bendera Rational Software Corp [HAN98]. UML menyediakan notasi-notasi yang membantu memodelkan sistem dari berbagai perspektif. UML tidak hanya digunakan dalam pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan.

Berikut ini definisi *Unified Modeling Language* (UML) menurut para ahli:

1. Menurut (Hend, 2006) "*Unified Modeling Language* (UML) adalah bahasa yang telah menjadi standard untuk visualisasi, menetapkan, membangun dan mendokumentasikan artifak suatu sistem perangkat lunak".
2. Menurut (Adi Nugroho : 2005). "*Unified Modeling Language* (UML) adalah alat bantu analisis serta perancangan perangkat lunak berbasis objek".
3. Menurut (Joomla dari http://soetrasoft.com : 2007). *"Unified Modeling Language* (UML) merupakan standard modeling language yang terdiri dari kumpulan-kumpulan diagram, dikembangkan untuk membantu para pengembang sistem dan *software* agar bisa menyelesaikan tugas-tugas seperti: Spesifikasi, Visualisasi, Desain Arsitektur, Konstruksi, Simulasi dan testing serta Dokumentasi".

Berdasarkan beberapa pendapat yang dikemukakan diatas dapat ditarik kesimpulan bahwa "*Unified Modeling Language* (UML) adalah sebuah bahasa

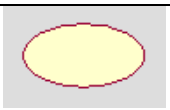
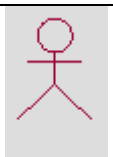
yang berdasarkan grafik atau gambar untuk menvisualisasikan, menspesifikasikan, membangun dan pendokumentasian dari sebuah sistem pengembangan perangkat lunak berbasis OO (*Object Oriented*)".

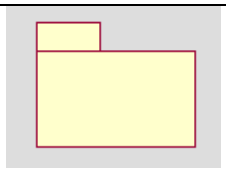
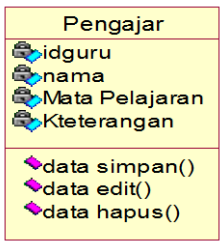
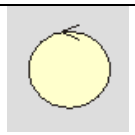
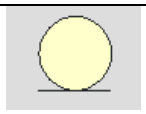
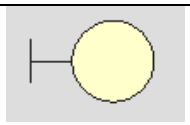
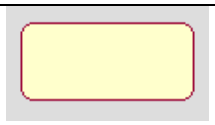
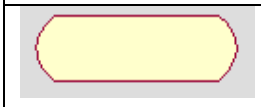
Unified Modelling Language (UML) adalah sebuah bahasa untuk menentukan, visualisasi, konstruksi, dan mendokumentasikan *artifacts* dari sistem *software*, untuk memodelkan bisnis, dan sistem *nonsoftware* lainnya. *Artifacts* adalah sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa *software*. *Artifacts* dapat berupa model, deskripsi, atau software. Untuk membuat suatu model, UML memiliki diagram grafis yang diberi nama berdasarkan sudut pandang yang berbeda-beda terhadap sistem dalam proses analisa atau rekayasa. Diagram grafis tersebut antara lain

Daftar simbol-simbol dalam UML (Unified Modelling Language)

Tabel 2.1 Daftar Simbol-Simbol UML

Sumber : http://freetechebooks.com/ebook-2011/daftar-simbol-uml.html

Gambar simbol	**Nama simbol**	**Keterangan**
	*Usecase*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
	*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.

	*Package*	suatu simbol yang memberikan batasan dan komentar yang dikaitkan pada suatu elemen atau kumpulan elemen
	*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
	*Control*	Mengkordinasikan aktifitas dalam sistem
	*Entity*	Kelas yang berhubungan data dan informasi yang dibutuhkan oleh sistem
	*Boundery*	Kelas yang memodelkan interaksi antar satu atau lebih aktor dengan sistem
	*Activity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
	*State*	Nilai atribut dan nilai link pada suatu waktu tertentu, yang dimiliki oleh suatu objek.

UML menyediakan 10 macam diagram yang merupakan salah satu alat bantu yang sangat handal dalam mengembangkan system berorientasi objek. Ada 9 jenis diagram yang ditangani oleh UML, yakni:

2.2.1 Use Case Diagram

Use Case Diagram adalah suatu kumpulan urutan interaksi di antara user dengan system untuk mencapai suatu tujuan di mana use case ini menggambarkan kebutuhan fungsional suatu system tanpa menampilkan struktur internal system.

2.2.2 Sequence Diagram

Sequence diagram adalah suatu diagram yang menggambarkan interaksi antar obyek dan mengindikasikan komunikasi diantara obyek-obyek tersebut. Diagram ini juga menunjukkan serangkaian pesan yang dipertukarkan oleh obyek-obyek yang melakukan suatu tugas atau aksi tertentu. Obyek-obyek tersebut kemudian diurutkan dari kiri ke kanan, aktor yang menginisiasi interaksi biasanya ditaruh di paling kiri dari diagram.

2.2.3 Collaboration Diagram

Collaboration diagram adalah perluasan dari objek diagram. Objek diagram menunjukkan objek-objek dan hubungannya dengan yang lain. Collaboration diagram menunjukkan pesan-pesan objek yang dikirim satu sama lain.

2.2.4 Activity Diagram

Activity diagram adalah representasi secara grafis dari proses dari proses dan control flow dan berfungsi untuk memperlihatkan alur dari satu aktivitas ke aktivitas yang lain serta menggambarkan perilaku yang kompleks.

2.2.5 Deplyoment Diagram

Deployement adalah penggambaran tugas-tugas kongkrit dari setiap node/software yang terlibat dalam jaringan system, menampilkan keseluruhan node dalam jaringan serta hubungan dari node-node tersebut termasuk proses-proses yang terlibat di dalamnya.

2.2.6 Component Diagram

Component diagram adalah diagram yang menunjukkan organisasi dan kebergantungan di antara sekumpulan komponen. Diagram ini memodelkan pandangan implementasi fisik dari sistem.

2.2.7 Langkah-langkah Penggunaan *Unified Modeling Language* (UML)

Menurut (Afif Amrullah:2002). "Langkah-langkah penggunaan *Unified Modeling Language* (UML) sebagai berikut:

1. Buatlah daftar *business process* dari *level* tertinggi untuk mendefinisikan aktivitas dan proses yang mungkin muncul.
2. Petakan *use case* untuk setiap *business process* untuk mendefinisikan dengan tepat fungsional yang harus disediakan oleh sistem, kemudian perhalus *use case diagram* dan lengkapi dengan *requirement, constraints* dan catatan-catatan lain.
3. Buatlah *deployment diagram* secara kasar untuk mendefinisikan arsitektur fisik sistem.
4. Definisikan *requirement* lain *non fungsional, security* dan sebagainya yang juga harus disediakan oleh sistem.

5. Berdasarkan *use case diagram,* mulailah membuat *activity diagram.*
6. Definisikan obyek-obyek level atas *package* atau *domain* dan buatlah *sequence* dan/atau *collaboration* utuk tiap alir pekerjaan, jika sebuah *use case* memiliki kemungkinan alir normal dan *error*, buat lagi satu diagram untuk masing-masing alir.
7. Buatlah rancangan *user interface model* yang menyediakan antamuka bagi pengguna untuk menjalankan *skenario use case.*
8. Berdasarkan model-model yang sudah ada, buatlah *class diagram*. Setiap *package* atau *domian* dipecah menjadi *hirarki class* lengkap dengan *atribut* dan metodenya. Akan lebih baik jika untuk setiap *class* dibuat *unit test* untuk menguji *fungsionalitas class* dan interaksi dengan *class* lain.
9. Setelah *class diagram* dibuat, kita dapat melihat kemungkinan pengelompokkan *class* menjadi komponen-komponen karena itu buatlah *component diagram* pada tahap ini. Juga, definisikan *test integrasi* untuk setiap komponen meyakinkan ia bereaksi dengan baik.
10. Perhalus *deployment diagram* yang sudah dibuat. Detilkan kemampuan dan *requirement* piranti lunak, sistem operasi, jaringan dan sebagainya. Petakan komponen ke dalam node.
11. Mulailah membangun sistem. Ada dua pendekatan yang tepat digunakan:

a. Pendekatan *use case* dengan meng*assign* setiap *use case* kepada tim pengembang tertentu untuk mengembangkan unit kode yang lengkap dengan *test.*
b. Pendekatan *komponen* yaitu *mengassign* setiap komponen kepada tim pengembang tertentu.

12. Lakukan uji modul dan uji integrasi serta perbaiki model beserta *code*nya. Model harus selalu sesuai dengan *code* yang aktual.

Perangkat lunak siap dirilis".

2.3 Object Oriented Programming (OOP)

Object Oriented Progamming (OOP) atau Pemrograman Berorientasi Objek adalah konsep pemrograman yang difokuskan pada penciptaan kelas yang merupakan abstraksi/ *blueprint*/ *prototype* dari suatu objek. Kelas ini harus mengandung sifat (data) dan tingkah laku (method) umum yang dimiliki oleh objek-objek yang kelak akan dibuat (diinstansiasi). Data dan method merupakan anggota (*member*) dari suatu kelas.

Pemrograman prosedural murni yang tidak menerapkan konsep *object oriented* (karena ada bahasa pemrograman prosedural yang juga sudah berorientasi objek, meskipun belum sepenuhnya) banyak menitikberatkan ke arah pembentukan fungsi-fungsi, sehingga di dalam program akan terdapat banyak sekali fungsi dan variabel yang menyulitkan pemrogram untuk mengelola dan mengembangkannya. Oleh karena itu, dengan memperhatikan kekurangan-kekurangan tersebut, maka dibentuklah bahasa yang menerapkan pendekatan *object oriented* untuk menyederhanakan fungsi-fungsi dan variabel-variabel ke dalam bentuk objek.

Dalam OOP dibutukan *memory* lebih besar dibandingkan dengan program prosedural (tradisional). Dua objek yang identik akan memerlukan dua area *memory* berbeda walaupun dari sisi data dan proses keduanya memiliki jumlah

dan jenis yang sama. Hal ini disebabkan karena data dan proses pada kedua objek tersebut dipisahkan oleh komputer.

Secara garis besar yang menjadi ciri dari OOP adalah adanya proses abstraksi (*abstraction*), pengkapsulan (*encapsulation*), penurunan sifat (*inheritance*), dan polimorfisme (*polymorphism*) pada objek-objek yang dibentuk.

2.4 Bahasa Pemrograman

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.

Java2 adalah generasi kedua dari Java platform (generasi awalnya adalah Java Development Kit). Java berdiri di atas sebuah mesin interpreter yang diberi nama JVM (*Java Virtual Machine*). 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 berbagai system operasi, asalkan pada system operasi tersebut terdapat JVM.

Platform Java terdiri dari kumpulan library, JVM, kelas-kelas yang dipaket dalam sebuah lingkungan rutin Java, dan sebuah *compiler*, *debugger*, 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.

2.5 Definisi Perangkat Yang Digunakan

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua *platform* (*platform-independent*). 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 popular adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug-in*.

Karakteristik Eclipse Galileo:

1. *Components*

   Eclipse terdiri dari model komponen-komponen (*plug-ins*) yang dapat digunakan lebih dari satu aplikasi.

2. *Middleware and Infrastructure*

   Komponen utama dalam eclipse dibuat berdasarkan *framework* dan fasilitas, sehingga mempermudah penulisan kode oleh pengguna. Fasilitas yang diperoleh, seperti: paradigma UI yang fleksibel, aplikasi yang dapat diekstensi, *help support*, *scalable* UI, *context-sensitive help*, *network updates*, *error handling*, dan lain-lainnya.

3. *Native User Experience*

   Eclipse Standard Widget Toolkit menyediakan *toolkit* GUI untuk java yang efisien dan akses yang portable ke fasilitas bawaan (*native*) UI di OS.

4. *Portability*

   Eclipse memiliki aplikasi yang fleksibel di berbagai OS dan *client environments* (syaratnya: *hardware* itu dapat diinstal *Java Runtime Environment*).

5. *Intelligent Install and Update*

   Aplikasi di eclipse memiliki fitur *update plugins* melalui HTTP, *Java Web Start, Update Site, copy file,* atau system manajemen perusahaan yang canggih.

6. *Disconnected Operation*

   Tidak membutuhkan koneksi internet, karena aplikasi eclipse berjalan di local komputer.

7. *Development Tooling Support*

   Eclipse menyediakan class pertama di IDE java yang dapat diintegrasikan untuk *develop, test,* dan *package* aplikasi *rich clients*.

8. *Component Libraries*

   Komponen di *framework* tidak akan lengkap tanpa set plugins yang komprehensif. Sehingga eclipse akan memproduksi plugins yang dibutuhkan dalam membuat aplikasi secara utuh. [14]

2.6 *Location Based Service* (LBS)

Location Based Service (LBS) atau layanan berbasis lokasi adalah istilah umum yang digunakan untuk menggambarkan teknologi yang digunakan untuk menemukan lokasi perangkat yang kita gunakan. Dua unsur utama LBS adalah:

1. *Location Manager* (API Maps)

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

2. *Location Providers* (API Location)

   Menyediakan teknologi pencarian lokasi yang digunakan oleh device/perangkat. API Location berhubungan dengan data GPS (*Global Positioning System*) dan data lokasi real-time. API *Location* berapa pada paket android yaitu dalam paket *android.location.* Dengan Location Manager, kita dapat menentukan lokasi kita saat ini, *Track* gerakan/perpindahan, serta kedekatan dengan lokasi tertentu dengan mendeteksi perpindahan. [2]

2.7 Arsitektur Andorid

Secara garis besar arsitektur android dapat dijelaskan dan digambarkan sebagai berikut:

1. *Applications dan Widgets*

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

2. *Applications Frameworks*

   Android adalah "*Open Development Platform*" yaitu Android menawarkan kepada pengembang atau memberi kemampuan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *resources*, menjalankan *service background*, mengatur alarm, dan menambahkan status *notifications*, dan sebagainya.

3. *Libraries*

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

4. *Android Run Time*

Layer yang membuat aplikasi Android dapat dijalankan dimana dalam prosesnya menggunakan Implementasi Linux. *Dalvik Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi Android.

5. *Linux Kernel*

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

2.8 Android Lifecycle

Setiap aktifitas yang ada di dalam aplikasi pasti akan melalui *lifecycle.* Misalnya jika menjalankan aplikasi fungsi *onCreate* akan digunakan untuk membuat aplikasi tampil di layar Android, jika keluar dari aplikasi fungsi *OnDestroy* akan digunakan.

2.9 Database SQLite

Android juga memiliki fasilitas untuk membuat *database* yang dikenal dengan *SQLite* yaitu salah satu *software* yang *embedded* yang sangat popular, kombinasi *SQL interface* dan penggunaan memory yang sangat sedikit dengan kecepatan yang sangat cepat. SQLite di android termasuk dalam *Android runtime*, sehingga setiap versi dari android dapat membuat *database* dengan *SQLite*.

Dalam system android terdapat beberapa teknik untuk melakukan penyimpanan data. Teknik yang umum digunakan adalah sebagai berikut:

1. *Shared prefences* yaitu menyimpan data beberapa nilai (*value*) dalam bentuk *groups key* yang dikenal dengan *prefences*.
2. *Files* yaitu menyimpan data dalam file, dapat berupa menulis ke file atau membaca dari file.
3. *SQLite Database*, yaitu menyimpan data dalam bentuk *database.*
4. *Content Providers,* yaitu menyimpan data dalam bentuk *content providers service.*

2.10 Eclipse

Eclipse adalah sebuah IDE (*Integrated Development Environment*) untuk mengembangkan perangkat lunak dan dijalankan di semua *platform*. Eclipse sendiri juga merupakan sebuah komunitas *open source*, yang memiliki proyek yang berfokus pada membangun sebuah *platform* pengembangan terbuka dari extensible *Framework, tools* dan runtime untuk membangun, menyebarkan dan mengelola perangkat lunak di seluruh siklus hidup perangkat lunak tersebut.

Umumnya Eclipse digunakan untuk membuat sebuah program yang menggunakan bahasa pemrograman Java. Namun, Eclipse juga bisa digunakan untuk penggunaan bahasa pemrograman lainnya seperti C, C++, COBOL, Perl, PHP, Python, dan sebagainya.

2.10.1 Versi Eclipse

Berikut ini adalah versi Eclipse yang telah dirilis:

Tabel 2.2 Versi *Eclipse*

Nama	**Tanggal rilis**	**Versi**
Eclipse 3.0	21 Juni 2004	3.0
Eclipse 3.1	28 Juni 2008	3.1
Callisto	30 Juni 2006	3.2
Europa	29 Juni 2007	3.3
Ganymade	25 Juni 2008	3.4
Galileo	24 Juni 2009	3.5
Helios	23 Juni 2010	3.6
Indigo	22 Juni 2011	3.7

Sumber: Eclipse *Foundation* (2011)