BAB 2
TINJAUAN PUSTAKA

2.1 Majmu' Syarif

Majmu' Syarif merupakan kumpulan dari surah-surah dalam *Al-Quran*, Doa dan Zikir. *Majmu' Syarif* biasa digunakan sebagai penuntun ibadah tambahan sehari-hari selain dari ibadah wajib, seperti digunakan dalam membaca surah *Yasin* pada malam Jumat, acara *Tahlil, Nisfu, Istigosah* dan lain-lain. *Majmu' Syarif* juga sering disebut sebagai *Yasin* lengkap, yang dalam artian tidak hanya berisi surah *Yasin*, berisi juga surah-surah lain dalam *Al-Quran* serta doa-doa dan zikir yang memiliki faedah yang baik jika di amalkan [6].

2.2 Aplikasi

Aplikasi adalah suatu subkelas perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna. Biasanya dibandingkan dengan perangkat lunak sistem yang mengintegrasikan berbagai kemampuan komputer, tapi tidak secara langsung menerapkan kemampuan tersebut untuk mengerjakan suatu tugas yang menguntungkan pengguna. Contoh utama perangkat lunak aplikasi adalah pengolah kata, lembar kerja, dan pemutar media. Beberapa aplikasi yang digabung bersama menjadi suatu paket kadang disebut suatu paket atau suite aplikasi (*Application Suite*). Contohnya adalah *Microsoft Office* dan *OpenOffice*, yang menggabungkan suatu aplikasi pengolah kata, lembar kerja, serta beberapa aplikasi lainnya. Aplikasi-aplikasi dalam suatu paket biasanya memiliki antarmuka pengguna yang memiliki kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan tiap aplikasi. Seringkali, mereka memiliki kemapuan untuk saling berinteraksi satu sama lain sehingga menguntungkan pengguna. Contohnya, suatu lembar kerja dapat dibenamkan dalam suatu dokumen pengolah kata walaupun dibuat pada aplikasi lembar kerja yang terpisah[14].

2.3 Perancangan Berorientasi Objek

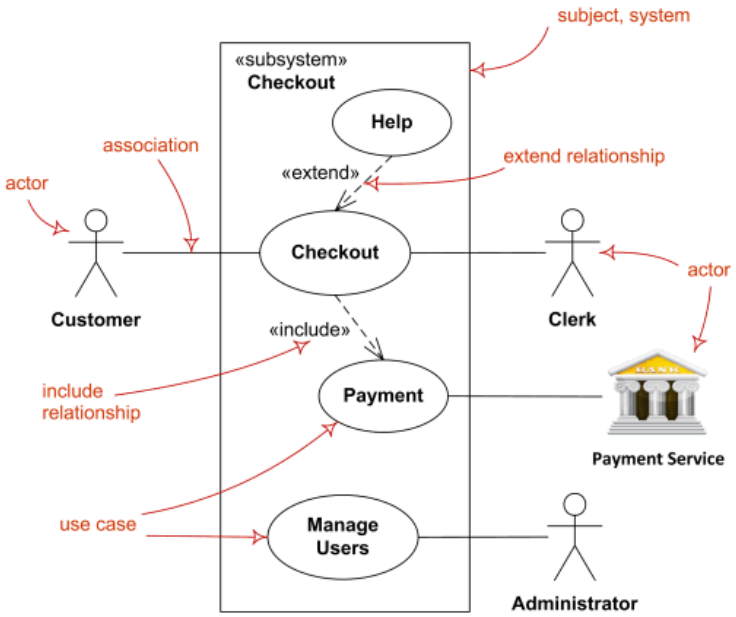
Teknologi objek menganalogikan sistem aplikasi seperti kehidupan nyata yang didominasi oleh objek. Didalam membangun sistem berorientasi objek akan menjadi lebih baik apabila langkah awalnya didahului dengan proses analisis dan perancangan yang berorientasi objek. Tujuannya adalah mempermudah *programmer* didalam mendesain program dalam bentuk objek-objek dan hubungan antar objek tersebut untuk kemudian dimodelkan dalam sistem nyata. Suatu perusahaan *software* yaitu *Rational Software*, telah membentuk konsarium dengan berbagai organisasi untuk meresmikan pemakaian *Unified Modelling Language* (UML) sebagai bahasa standar dalam *Object Oriented Analysist Design* (OOAD).

2.3.1 *Unified Modelling Language* (UML)

UML dalam sebuah bahasa untuk menentukan visualisasi, konstruksi, dan mendokumentasikan artifacts dari sistem *software*, untuk memodelkan bisnis, dan sistem *non-software* lainnya. UML merupakan sistem arsitektur yang bekerja dalam OOAD dengan satu bahasa yang konsisten untuk menentukan, visualisasi, konstruksi dan mendokumentasikan *artifact* yang terdapat dalam sistem. *Artifact* adalah sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa *software*. *Artifact* dapat berupa model, deskripsi atau *software*[13].

2.3.1.1 *Use Case Diagram*

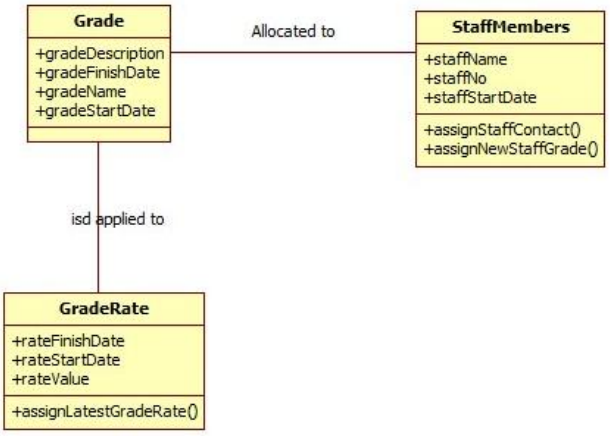
Use Case Diagram menjelakan manfaat sistem jika dilihat menurut pandangan orang yang berada diluar sistem (Aktor). Diagram ini menunjukan fungsionalitas suatu sistem yang berinteraksi dengan dunia luar. *Use Case Diagram* dapat digunakan selama proses analisis untuk menagkap *requitment* sistem dan untuk memahami bagaimana sistem bekerja.



Gambar 2.1 *Use Case Diagram*

2.3.1.2 *Class Diagram*

Class Diagram menjelaskan dalam visualisasi struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. *Class Diagram* memperlihatakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas dalam model desain dari suatu sistem. Selama proses analisis, *class diagram* memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan prilaku sistem. Selama tahap desain, *class diagram* berperan dalam menagkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat.



Gambar 2.2 Class Diagram

2.3.1.3 *Behaviour Diagram*

Behavior diagram dapat dikelompokan menjadi tiga diagram, yaitu :

1. *Statechart Diagram*

   Statechart Diagram berfungsi untuk memodelkan prilaku dinamis satu kelas satu objek.

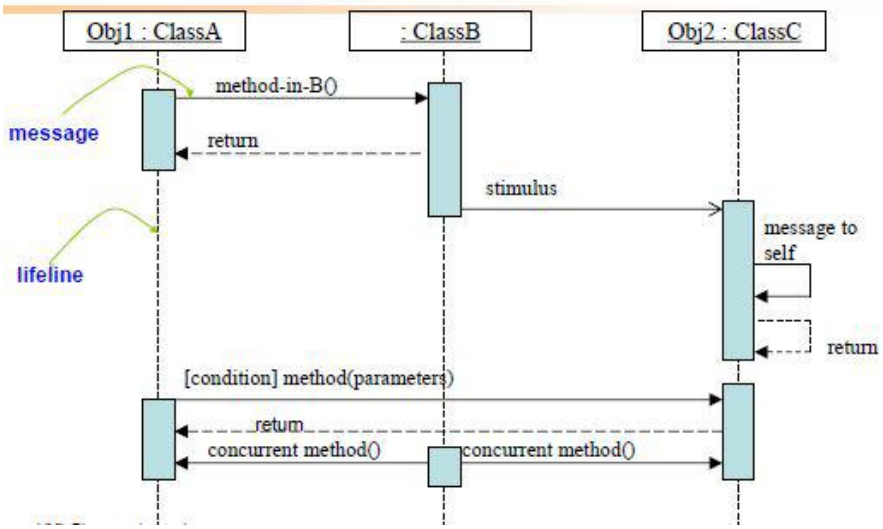
2. *Activity Diagram*

   Activity Diagram memodelkan alur kerja (work flow) sebuah proses bisnis dan urutan aktifitas dalam suatu proses.

3. *Interaction Diagram*

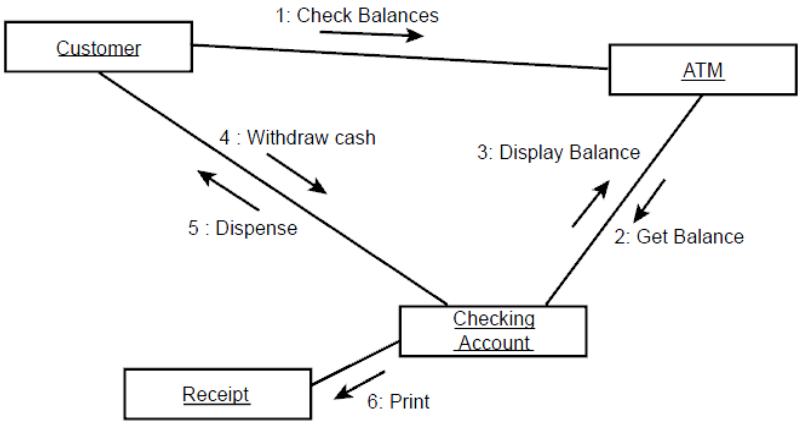
   Interaction Diagram dibagi menjadi dua model diagram yaitu :

   1. *Sequence Diagram* menjelaskan interaksi objek yang disusun dalam suatu urutan waktu. Diagram ini secara khusus bersosialisasi dengan *use case*. Sequence diagram, memperlihatkan tahap demi tahap apa yang seharusnya terjadi untuk menghasilkan sesuatu dalam *use case*.



Gambar 2.3 Sequence Diagram

2. *Collaboration Diagram* melihat pada interaksi dan hubungan terstruktur antar objek. Tipe diagram ini menekankan pada hubungan (*relationship*) antar objek, sedangkan *sequence diagram* menekankan pada urutan kejadian. Dalam *collaboration diagram* terdapat beberapa objek, *link*, dan *message*.



Gambar 2.4 *Collaboration Diagram*

2.3.1.4 *Implementation Diagram*

Implementation diagram dibagi menjadi dua diagram, yaitu :

1. *Component Diagram* menggambarkan alokasi semua kelas dan objek kedalam komponen-komponen dalam desain fisik sistem *software*. Diagram ini memperlihatkan pengaturan dan kebergantungan antara komponen-komponen *software*, seperti *source code*, *binary code*, dan komponen tereksekusi (*execute components*)
2. *Deployment Diagram* memperlihatkan pemetaan *software* kepada *hardware*. Dimana akan berjalan (di *server/multitier*, *standalone* atau lainnya), dan menggambarkan model koneksi dan kemampuan jaringan dan hal lainya yang bersifat fisik.

2.4 Android



Gambar 2.5 Logo Android

Android adalah sistem operasi yang berbasis *Linux* untuk telepon seluler seperti telepon pintar (*Smartphone*) dan komputer tablet. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti begerak. Awalnya, *Google Inc.* membeli *Android Inc.*, pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan *Android*, dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan piranti keras, piranti lunak dan telekomunikasi, termasuk *Google, HTC, Intel, Motorola, Qualcomm, T-Mobile* dan *Nvidia*[12].

Bebrapa pengertian lain dari *Android,* yaitu :

1. Merupakan *platform* terbuka (*Open Source*) bagi para programmer untuk membuat atau mengembangkan aplikasi.
2. Merupakan sistem operasi yang dibeli *Google Inc* dari *Android Inc*.
3. Bukan bahasa pemrograman, akan tetapi hanya menyediakan lingkungan hidup atau *runtime environment* yang disebut DVM (*Dalvik Virtual Machine*) yang telah dioptimasi untuk *device* dengan sistem memori yang kecil.

2.4.1 Versi *Android*

Banyak *smartphone* dan PC Tablet menggunakan sistem operasi dengan versi yang berbeda. Semakin tinggi versi, fiturnya semakin canggih dan banyak. Telepon pertama yang memakai sistem operasi *Android* adalah HTC Dream

yang dirilis pada tanggal 22 Oktober 2008. Beberapa uraian versi *android* seperti dibawah ini :

1. *Android* versi 1.1

   Pada 9 Maret 2009, *Google* merilis *Android* versi 1.1. *Android* versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam alarm, *voice search* (pencarian suara), pengiriman pesan dengan Gmail, dan pemberitahuan Email.

2. *Android* versi 1.5 (*Cupcake*)

   Pada pertengahan Mei 2009, *Google* kembali merilis telepon seluler dengan menggunakan *Android* dan SDK (*Software Development Kit*) dengan versi 1.5 (*Cupcake*). Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi ini yakni kemampuan merekam dan menonton video dengan modus kamera, mengunggah 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.

3. *Android* versi 1.6 (*Donut*)

   Donut (versi 1.6) dirilis pada September dengan menampilkan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indikator dan kontrol *applet* VPN. Fitur lainnya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus pada kamera, *camcorder* dan galeri yang diintegrasikan pada CDMA / EVDO, 802.1x, VPN, *Gestures*, dan *Text-to-spech engine*. Kemampuan dial kontak teknologi *text to change speech* (tidak tersedia pada semua ponsel).

4. *Android* versi 2.0/2.1 (*Eclair*)

   Pada 3 Desember 2009 kembali diluncurkan ponsel *Android* dengan versi 2.0/2.1 (*Eclair*), perubahan yang dilakukan adalah mengoptimalkan *hardware*, peningkatan Google Maps 3.1.2, perubahan UI dengan browser baru dan dukungan HTML 5, daftar kontak yang baru, dukungan *flash* untuk kamera 3.2 MP, *digital zoom*, dan *Bluetooth* 2.1.

Untuk bergerak cepat dalam persaingan perangkat generasi berikut, Google melakukan investasi dengan mengadakan kompetisi aplikasi *mobile* terbaik (*killer apps*-aplikasi unggulan). Kompetisi ini berhadiah $25,000 bagi setiap pengembang aplikasi terpilih. Kompetisi diadakan selama dua tahap yang tiap tahapnya dipilih 50 aplikasi terbaik.

Dengan semakin berkembangnya dan semakin bertambahnya jumlah *handset Android*, semakin banyak pihak ketiga yang berminat untuk menyalurkan aplikasi mereka kepada sistem operasi *Android*. Aplikasi terkenal yang diubah ke dalam sistem operasi *Android* adalah Shazam, *Backgrounds* dan *WeatherBug*.

Sistem operasi *Android* dalam situs internet juga dianggap penting untuk menciptakan aplikasi *Android* asli, contohnya oleh MySpace dan Facebook.

5. *Android* versi 2.2 (*Froyo* : *Frozen Yoghurt*)

   Pada 20 Mei 2010, *Android* versi 2.2 (Froyo) diluncurkan. Perubahan-perubahan umumnya terhadap versi-versi sebelumnya antara lain dukungan Adobe Flash 10.1, kecepatan kinerja dan aplikasi 2 sampai 5 kali lebih cepat, integrasi dengan V8 JavaScript *engine* yang dipakai Google Chrome yang mempercepat kemampuan rendering pada browser, pemasangan aplikasi dalam SD Card, kemampuan WiFi Hotspot portabel dan kemampuan *auto update* dalam aplikasi Android Market.

6. *Android* versi 2.3 (*Gingerbread*)

   Pada 6 Desember 2010, Android versi 2.3 (*Gingerbread*) diluncurkan. Perubahan-perubahan umum yang didapat dari versi ini antara lain peningkatan kemampuan permainan (*gaming*), peningkatan fungsi *copy paste*, layar antarmuka (*User Interface*) didesain ulang, dukungan format video VP8 dan WebM, efek audio baru (*reverb, equalization, headphone virtualization,* dan *bass boost*), dukungan kemampuan *Near Field Communication* (NFC) dan dukungan jumlah kamera yang lebih dari satu.

7. *Android* versi 3.1 (*Honeycomb*)

   Android Honeycomb dirancang khusus untuk tablet. *Android* versi ini mendukung ukuran layar yang lebih besar. *User Interface* pada *Honeycomb* juga berbeda karena sudah didesain untuk tablet. *Honeycomb* juga mendukung multi prosesor dan juga akselerasi perangkat keras (*hardware*) untuk grafis. Tablet peratam yang dibuat dengan menjalankan *Honeycomb* adalah Motorola Xoom.

8. *Android* versi 4.0 (*Ice Cream Sandwich*)

   Diumumkan pada tanggal 19 Oktober 2011, membawa fitur *Honeycomb* untuk *smartphone* dan menambahkan fitur baru termasuk membuka kunci dengan pengenalan wajah, jaringan data pemantauan penggunaan dan kontrol, terpadu kontak jaringan sosial, perangkat tambahan fotografi, mencari email secara *offline* dan berbagi informasi dengan menggunakan NFC.

9. *Android* versi 4.1 (*Jelly Bean*)

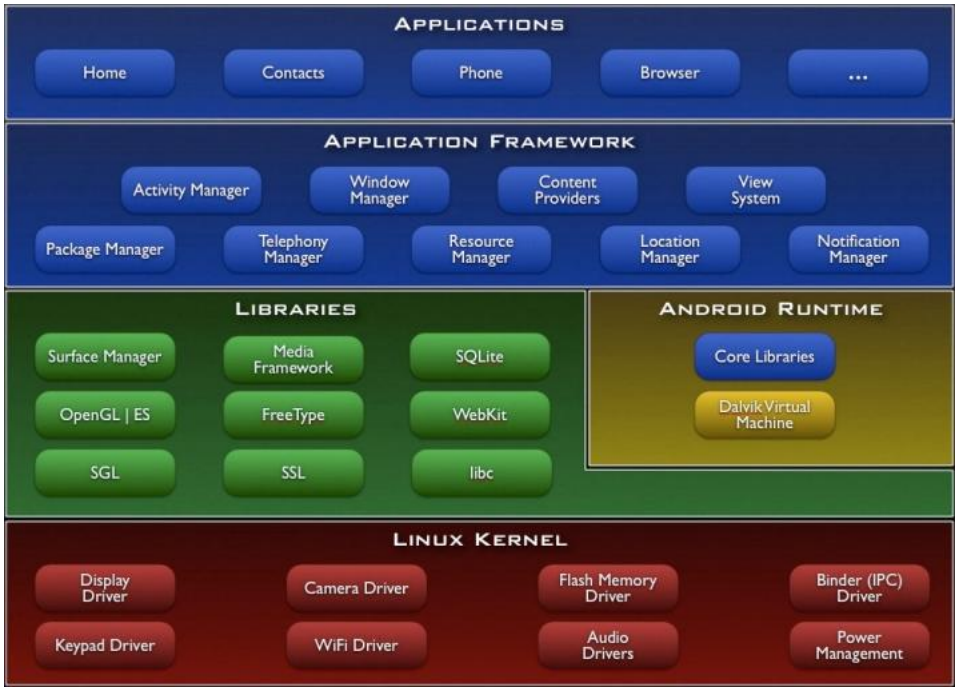
   Pada tanggal 27 Juni 2012 Google mengumumkan *Android* versi 4.1 (*Jelly Bean*) dalam konferensi Google I/O berdasarkan kernel Linux 3.0.31. *Jelly Bean* adalah pembaruan penting yang bertujuan untuk meningkatkan fungsi dan kinerja antarmuak pengguna (*User Interface*). Pembaruan ini diwujudkan dalam proyek *Butter*, perbaikan ini termasuk antisipasi sentuh, *triple buffering*, perpanjangan waktu vsync dan peningkatan *frame rate* hingga 60fps untuk menciptakan UI yang lebih halus. Perangkat pertama yang menggunakan sistem operasi ini adalah tablet Nexus 7, yang dirilis pada 13 Juli 2012.

10. *Android* versi 4.4 (KitKat)

    Android versi 4.4 (KitKat) direncanakan akan dirilis pada bulan Oktober 2013. Setelah sebelumnya beredar rumor bahwa Android versi berikutnya setelah *Jelly Bean* diperkirakan akan diberi nomor 5.0 dan dinamai Key Lime Pie.

2.4.2 Anatomi Aplikasi Android

Dalam paket sistem operasi Android tediri dari beberapa unsur seperti tampak pada gambar 2.1. Secara sederhana arsitektur Android merupakan sebuah kernel Linux dan sekumpulan pustaka C / C++ dalam suatu *Framework* yang menyediakan dan mengatur alur proses aplikasi.



Gambar 2.6 Anatomi Android

2.4.2.1 Linux Kernel

Android dibangun di atas kernel Linux 2.6. Namun secara keseluruhan Android bukanlah linux, karena dalam Android tidak terdapat paket standar yang dimiliki oleh linux lainnya. Linux merupakan sistem operasi terbuka yang handal dalam manajemen memori dan proses. Oleh karenanya pada Android hanya terdapat beberapa pelayanan yang diperlukan seperti keamanan, manajemen memori, manajemen proses, jaringan dan *driver*. Kernel linux menyediakan *driver* layar, kamera, *keypad*, *WiFi*, *Flash Memory*, audio, dan IPC (*Interprocess Communication*) untuk mengatur aplikasi dan lubang keamanan.

2.4.2.2 *Libraries*

Android menggunakan beberapa paket pustaka yang terdapat pada C/C++ dengan standar *Berkeley Software Distribution* (BSD) hanya setengah dari yang aslinya untuk tertanam pada kernel Linux. Beberapa pustaka diantaranya:

1. *Media Library* untuk memutar dan merekam berbagai macam format audio dan video.
2. *Surface Manager* untuk mengatur hak akses layer dari berbagai aplikasi.
3. *Graphic Library* termasuk didalamnya *SGL* dan *OpenGL*, untuk tampilan 2D dan 3D.
4. SQLite untuk mengatur relasi database yang digunakan pada aplikasi.
5. *SSl* dan *WebKit* untuk browser dan keamanan *internet*.

Pustaka-pustaka tersebut bukanlah aplikasi yang berjalan sendiri, namun hanya dapat digunakan oleh program yang berada di level atasnya. Sejak versi Android 1.5, pengembang dapat membuat dan menggunakan pustaka sendiri menggunakan *Native Development Toolkit* (NDK).

2.4.2.3 Android *Runtime*

Pada Android tertanam paket pustaka inti yang menyediakan sebagian besar fungsi Android. Inilah yang membedakan Android dibandingkan dengan sistem operasi lain yang juga mengimplementasikan Linux. Android *Runtime* merupakan mesin virtual yang membuat aplikasi Android menjadi lebih tangguh dengan paket pustaka yang telah ada. Dalam Android Runtime terdapat 2 bagian utama, diantaranya :

1. Pustaka Inti, Android dikembangkan melalui bahasa pemrograman Java, tapi Android *Runtime* bukanlah mesin virtual Java. Pustaka inti Android menyediakan hampir semua fungsi yang terdapat pada pustaka Java serta beberapa pustaka khusus Android.

Mesin Virtual Dalvik, Dalvik merupakan sebuah mesin virtual yang dikembangkan oleh Dan Bornstein yang terinspirasi dari nama sebuah perkampungan yang berada di Iceland. Dalvik hanyalah interpreter mesin virtual yang mengeksekusi file dalam format *Dalvik Executable* (*.dex).

Dengan format ini Dalvik akan mengoptimalkan efisiensi penyimpanan dan pengalamatan memori pada file yang dieksekusi. Dalvik berjalan di atas kernel Linux 2.6, dengan fungsi dasar seperti *threading* dan manajemen memori yang terbatas.

2.4.2.4 *Application Framework*

Kerangka aplikasi menyediakan kelas-kelas yang dapat digunakan untuk mengembangkan aplikasi Android. Selain itu, juga menyediakan abstraksi generik untuk mengakses perangkat, serta mengatur tampilan *User Interface* dan sumber daya aplikasi. Bagian terpenting dalam kerangka aplikasi Android adalah sebagai berikut:

1. *Activity Manager*, berfungsi untuk mengontrol siklus hidup aplikasi dan menjaga keadaan ”*Backstack*“ untuk navigasi penggunaan.
2. *Content Providers*, berfungsi untuk merangkum data yang memungkinkan digunakan oleh aplikasi lainnya, seperti daftar nama.
3. *Resuource Manager*, untuk mengatur sumber daya yang ada dalam program. Serta menyediakan akses sumber daya diluar kode program, seperti karakter, grafik, dan *file layout.*
4. *Location Manager*, berfungsi untuk memberikan informasi detail mengenai lokasi perangkat Android berada.
5. *Notification Manager*, mencakup berbagai macam peringatan seperti, pesan masuk, janji, dan lain sebagainya yang akan ditampilkan pada *status bar*.

2.4.2.5 *Application Layer*

Puncak dari diagram arsitektur Android adalah lapisan aplikasi dan *widget*. Lapisan aplikasi merupakan lapisan yang paling tampak pada pengguna ketika menjalankan program. Pengguna hanya akan melihat program ketika digunakan tanpa mengetahui proses yang terjadi dibalik lapisan aplikasi. Lapisan ini berjalan dalam Android *runtime* dengan menggunakan kelas dan service yang tersedia pada *Framework* aplikasi.

Lapisan aplikasi Android sangat berbeda dibandingkan dengan sistem operasi lainnya. Pada Android semua aplikasi, baik aplikasi inti (*native*) maupun

aplikasi pihak ketiga berjalan diatas lapisan aplikasi dengan menggunakan pustaka API (*Application Programming Interface*) yang sama.

2.4.3 Komponen Aplikasi Android

Fitur penting Android adalah bahwa satu aplikasi dapat menggunakan elemen dari aplikasi lain (untuk aplikasi yang memungkinkan). Sebagai contoh, sebuah aplikasi memerlukan fitur *scroller* dan aplikasi lain telah mengembangkan fitur *scroller* yang baik dan memungkinkan aplikasi lain menggunakannya. Maka pengembang tidak perlu lagi mengembangkan hal serupa untuk aplikasinya, cukup menggunakan *scroller* yang telah ada.

Agar fitur tersebut dapat bekerja, sistem harus dapat menjalankan aplikasi ketika setiap bagian aplikasi itu dibutuhkan, dan pemanggilan objek java untuk bagian itu. Oleh karenanya Android berbeda dari sistem-sistem lain, Android tidak memiliki satu tampilan utama program seperti fungsi main() pada aplikasi lain. Sebaliknya, aplikasi memiliki komponen penting yang memungkinkan sistem untuk memanggil dan menjalankan ketika dibutuhkan.

2.4.3.1 *Activities*

Activity merupakan bagian yang paling penting dalam sebuah aplikasi, karena *Activity* menyajikan tampilan visual program yang sedang digunakan oleh pengguna. Setiap *Activity* dideklarasikan dalam sebuah kelas yang bertugas untuk menampilkan antarmuka pengguna yang terdiri dari *Views* dan respon terhadap *Event*. Setiap aplikasi memiliki sebuah *activity* atau lebih. Biasanya pasti akan ada *activity* yang pertama kali tampil ketika aplikasi dijalankan.

Perpindahan antara *activity* dengan *activity* lainnya diatur melalui sistem, dengan memanfaatkan *activity stack*. Keadaan suatu *activity* ditentukan oleh posisinya dalam tumpukan *acitivity*, LIFO (*Last In First Out*) dari semua aplikasi yang sedang berjalan. Bila suatu *activity* baru dimulai, *activity* yang sebelumnya digunakan maka akan dipindahkan ketumpukan paling atas. Jika pengguna ingin menggunakan *activity* sebelumnya, cukup menekan tombol *Back*, atau menutup *activity* yang sedang digunakan, maka *activity* yang berada diatas akan aktif kembali. *Memory Manager* Android

menggunakan tumpukkan ini untuk menentukan prioritas aplikasi berdasarkan *activity*, memutuskan untuk mengakhiri suatu aplikasi dan mengambil sumber daya dari aplikasi tersebut.

Ketika *activity* diambil dan disimpan dalam tumpukkan *activity* terdapat 4 kemungkinan kondisi transisi yang akan terjadi :

1) ***Active***, setiap *activity* yang berada ditumpukan paling atas, maka dia akan terlihat, terfokus, dan menerima masukkan dari pengguna. Android akan berusaha untuk membuat *activity* aplikasi ini untuk untuk tetap hidup dengan segala cara, bahkan akan menghentikan *activity* yang berada dibawah tumpukkannya jika diperlukan. Ketika *activity* sedang aktif, maka yang lainnya akan dihentikan sementara.
2) ***Paused***, dalam beberapa kasus *activity* akan terlihat tapi tidak terfokus pada kondisi inilah disebut *paused*. Keadaan ini terjadi jika *activity* transparan dan tidak *fullscreen* pada layar. Ketika *activity* dalam keadaan *paused*, dia terlihat *active* namun tidak dapat menerima masukkan dari pengguna. Dalam kasus ekstrim, Android akan menghentikan *activity* dalam keadaan *paused* ini, untuk menunjang sumber daya bagi *activity* yang sedang aktif.
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 menjadi kandidat utama untuk dieksekusi oleh sistem ketika membutuhkan sumberdaya lebih. Oleh karenanya ketika suatu *activity* dalam kondisi *stopped* maka perlu disimpan data dan kondisi antarmuka saat itu. Karena ketika *activity* telah keluar atau ditutup, maka dia akan menjadi *inactive.*
4) ***Inactive***, kondisi ketika *activity* telah dihentikan dan sebelum dijalankan. *Inactive activity* telah ditiadakan dari tumpukan *activity* sehingga perlu *restart* ulang agar dapat tampil dan digunakan kembali.

Kondisi transisi ini sepenuhnya ditangani oleh manajer memori Android. Android akan memulai menutup aplikasi yang mengandung *activity inactive*, kemudian *stopped activity*, dan dalam kasus luar biasa *paused activity* juga akan ditutup.

2.4.3.2 *Services*

Suatu *service* tidak memiliki tampilan antarmuka, melainkan berjalan di *background* untuk waktu yang tidak terbatas. Komponen *service* diproses tidak terlihat, memperbarui sumber data dan menampilkan notifikasi. *Service* digunakan untuk melakukan pengolahan data yang perlu terus diproses, bahkan ketika *Activity* tidak aktif atau tidak tampak.

2.4.3.3 *Intents*

Intens merupakan sebuah mekanisme untuk menggambarkan tindakan tertentu, seperti memilih foto, menampilkan halaman *web*, dan lain sebagainya. *Intents* tidak selalu dimulai dengan menjalankan aplikasi, namun juga digunakan oleh sistem untuk memberitahukan ke aplikasi bila terjadi suatu hal, misal pesan masuk. *Intents* dapat eksplisit atau implisit, contohnya jika suatu aplikasi ingin menampilkan *URL*, sistem akan menentukan komponen apa yang dibutuhkan oleh *Intents* tersebut.

2.4.3.4 *Broadcast Receiver*

Broadcast Receivers merupakan komponen yang sebenarnya tidak melakukan apa-apa kecuali menerima dan bereaksi menyampaikan pemberitahuan. Sebagian besar *Broadcast* berasal dari sistem misalnya, Batre sudah hampir habis, informasi zona waktu telah berubah, atau pengguna telah merubah bahasa *default* pada perangkat. Sama halnya dengan *service*, *Broadcast Receivers* tidak menampilkan antarmuka pengguna. Namun, *Broadcast Receivers* dapat menggunakan *Notification Manager* untuk memberitahukan sesuatu kepada pengguna.

2.4.3.5 *Content Providers*

Content Providers digunakan untuk mengelola dan berbagi *database*. Data dapat disimpan dalam file sistem, dalam database *SQLite*, atau dengan cara lain yang pada prinsipnya sama. Dengan adanya *Content Provider* memungkinkan antar aplikasi untuk saling berbagi data. Komponen ini sangat berguna ketika sebuah aplikasi membutuhkan data dari aplikasi lain, sehingga mudah dalam penerapannya.

2.5 Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Dikembangkan oleh Sun Microsystems dan diterbitkan tahun 1995. Java tidak boleh disalahpahami sebagai JavaScript. JavaScript adalah bahasa scripting yang digunakan oleh Web Browser[8].

2.5.1 Perkembangan Java

Bahasa pemrograman Java pertama lahir dari *The Green Project*, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992. Proyek tersebut belum menggunakan versi yang dinamakan Oak. Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy, beserta Sembilan programer lainnya dari Sun Microsystems. Salah satu hasil proyek ini adalah maskot Duke yang dibuat oleh Joe Palrang. Pertemuan proyek berlangsung di sebuah gedung perkantoran Sand Hill Road di Menlo Park. Sekitar musim panas 1992 proyek ini ditutup dengan menghasilkan sebuah program Java Oak pertama, yang ditujukan sebagai pengendali sebuah peralatan dengan teknologi layar sentuh (*touch screen*), seperti pada PDA sekarang ini. Teknologi baru ini dinamai "*7" (*Star Seven*).

Setelah era *Star Seven* selesai, sebuah anak perusahaan Tv kabel tertarik ditambah beberapa orang dari proyek *The Green Project*. Mereka memusatkan kegiatannya pada sebuah ruangan kantor di 100 Hamilton Avenue, Palo Alto. Perusahaan baru ini bertambah maju, jumlah karyawan meningkat dalam waktu singkat dari 13 menjadi 70 orang. Pada rentang waktu ini juga ditetapkan pemakaian Internet sebagai medium yang menjembatani kerja dan ide di antara mereka.

Pada awal tahun 1990-an, Internet masih merupakan rintisan, yang dipakai hanya di kalangan akademisi dan militer. Mereka menjadikan perambah *(browser*) Mosaic sebagai landasan awal untuk membuat perambah Java pertama yang dinamai *Web Runner*, terinsipirasi dari film 1980-an, Blade Runner. Pada perkembangan rilis pertama, Web Runner berganti nama menjadi Hot Java.

Pada sekitar bulan Maret 1995, untuk pertama kali kode sumber Java versi 1.0a2 dibuka. Kesuksesan mereka diikuti dengan untuk pemberitaan pertama kali pada surat kabar San Jose Mercury News pada tanggal 23 Mei 1995. Sayang terjadi perpecahan di antara mereka suatu hari pada pukul 04.00 di sebuah ruangan hotel Sheraton Palace. Tiga dari pimpinan utama proyek, Eric Schmidt dan George Paolini dari Sun Microsystems bersama Marc Andreessen, membentuk Netscape. Nama Oak, diambil dari pohon Oak yang tumbuh di depan jendela ruangan kerja "bapak java", James Gosling. Nama Oak ini tidak dipakai untuk versi *release* Java karena sebuah perangkat lunak sudah terdaftar dengan merek dagang tersebut, sehingga diambil nama penggantinya menjadi "Java". Nama ini diambil dari kopi murni yang digiling langsung dari biji (kopi tubruk) kesukaan Gosling. Konon kopi ini berasal dari Pulau Jawa. Jadi nama bahasa pemrograman Java tidak lain berasal dari kata Jawa (bahasa Inggris untuk Jawa adalah *Java*).

2.5.2 Versi Awal

Versi awal Java ditahun 1996 sudah merupakan versi *release* sehingga dinamakan Java Versi 1.0. Java versi ini menyertakan banyak paket standar awal yang terus dikembangkan pada versi selanjutnya:

1. Java.lang: Peruntukan kelas elemen-elemen dasar.
2. Java.io: Peruntukan kelas *input* dan *output*, termasuk penggunaan berkas.
3. Java.util: Peruntukan kelas pelengkap seperti kelas struktur data dan kelas-kelas penanggalan.
4. Java.net: Peruntukan kelas TCP/IP, yang memungkinkan berkomunikasi dengan komputer lain menggunakan jaringan TCP/IP.
5. Java.awt: Kelas dasar untuk aplikasi antarmuka dengan pengguna (GUI).
6. Java.applet: Kelas dasar aplikasi antar muka untuk diterapkan pada penjelajah web.

2.5.3 Kelebihan Java

a) *Multiplatform*. Kelebihan utama dari Java ialah dapat dijalankan di beberapa platform / sistem operasi komputer, sesuai dengan prinsip tulis sekali, jalankan di mana saja.

b) OOP (*Object Oriented Programming* - Pemrogram Berorientasi Objek) yang artinya semua aspek yang terdapat di Java adalah Objek. Hal ini sangat memudahkan pemrogram untuk mendesain, membuat, mengembangkan dan mengalokasi kesalahan sebuah program dengan basis Java secara cepat, tepat, mudah dan terorganisir.

c) Perpustakaan Kelas yang Lengkap, Java terkenal dengan kelengkapan library/perpustakaan (kumpulan program program yang disertakan dalam pemrograman java) yang sangat memudahkan dalam penggunaan oleh para pemrogram untuk membangun aplikasinya.

d) Bergaya C++, memiliki sintaks seperti bahasa pemrograman C++ sehingga menarik banyak pemrogram C++ untuk pindah ke Java.

e) Pengumpulan sampah otomatis, memiliki fasilitas pengaturan penggunaan memori sehingga para pemrogram tidak perlu melakukan pengaturan memori secara langsung (seperti halnya dalam bahasa C++ yang dipakai secara luas).

2.5.4 Kekurangan Java

1. Tulis sekali, perbaiki di mana saja - Masih ada beberapa hal yang tidak kompatibel antara *platform* satu dengan *platform* lain. Untuk J2SE, misalnya SWT-AWT *bridge* yang sampai sekarang tidak berfungsi pada Mac OS X.
2. Mudah didekompilasi. Dekompilasi adalah proses membalikkan dari kode jadi menjadi kode sumber. Ini dimungkinkan karena kode jadi Java merupakan *bytecode* yang menyimpan banyak atribut bahasa tingkat tinggi, seperti nama-nama kelas, metode, dan tipe data. Hal yang sama juga terjadi pada Microsoft .NET *Platform*. Dengan demikian, algoritma yang digunakan program akan lebih sulit disembunyikan dan mudah dibajak/di-*reverse engineer*.
3. Penggunaan memori yang banyak. Penggunaan memori untuk program berbasis Java jauh lebih besar daripada bahasa tingkat tinggi generasi sebelumnya seperti C/C++ dan Pascal.

2.5.5 Tahap Kompilasi Java

1. Tulis / Ubah. Pemrogram menulis program dan menyimpannya di media dalam bentuk berkas '.java'.
2. Kompilasi. Pengkompilasi membentuk *bytecodes* dari program menjadi bentuk berkas '.*class*'.
3. Muat. Pemuat kelas memuat *bytecodes* ke memori.
4. Verifikasi. Peng-verifikasi memastikan *bytecodes* tidak mengganggu sistem keamanan Java.
5. Jalankan. Penerjemah menerjemahkan bytecodes ke bahasa mesin tidak bisa di pakai.

2.6 *Application Programming Interface* (API)

Antarmuka pemrograman aplikasi (*Application Programming Interface*) adalah sekumpulan perintah, fungsi, dan protokol yang dapat digunakan oleh pengembang saat membangung perangkat lunak untuk sistem operasi tertentu. API memungkinkan pengembang untuk menggunakan fungsi standar untuk berinteraksi dengan sistem operasi[12].

2.7 Skala Likert (*Likert Scale*)

Skala likert adalah suatu skala psikometrik yang digunakan dalam kuesioner dan merupakan salah satu teknik yang dapat digunakan dalam evaluasi suatu program atau kebijakan perencanaan. Rensis Likert telah mengembangkan sebuah skala untuk mengukur sikap masyarakat di tahun 1932 yang sekarang terkenal dengan nama skala Likert[5]. Skala Likert ini merupakan skala yang dapat dipergunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang mengenai suatu gejala atau fenomena. Dan pada evaluasi, skala likert digunakan untuk (a) Menilai keberhasilan suatu kebijakan atau program (b) Menilai manfaat pelaksanaan suatu kebijakan atau program (c) Mengetahui kepuasan stakeholder terhadap pelaksanaan suatu kebijakan atau program[5].

Dalam skala Likert terdapat dua bentuk pernyataan yaitu pernyataan positif yang berfungsi untuk mengukur sikap positif, dan pernyataan negatif yang berfungsi untuk mengukur sikap negatif objek. Skor pernyataan positif

dimulai dari 1 untuk sangat tidak setuju (ST), 2 untuk tidak setuju (T), 3 untuk tak memutuskan (N), 4 untuk setuju (S), dan 5 untuk sangat setuju (SS). Skor pernyataan negatif dimulai dari 1 untuk sangat setuju (SS), 2 untuk setuju (S), 3 untuk tak memutuskan (N), 4 untuk tidak setuju (T), dan 5 untuk sangat tidak setuju (ST). Skala Likert digunakan untuk mengukur **kesetujuan dan ketidaksetujuan** seseorang terhadap sesuatu rencana program, pelaksanaan program ataupun tingkat keberhasilan suatu program.

Tabel 2.1 Bobot Penilaian Skala Likert

Arah Pertanyaan	Bobot Penilaian				
	Sangat Setuju (SS)	Setuju (S)	Tak Memutuskan (N)	Tidak Setuju (T)	Sangat Tidak Setuju (ST)
Positif	5	4	3	2	1
Negatif	1	2	3	4	5

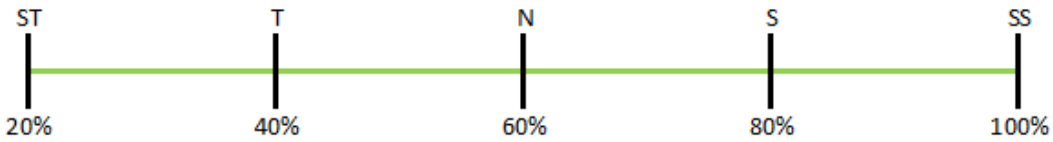
Berdasarkan data hasil kuesioner tersebut, dapat dicari persentase masing-masing jawaban dengan menggunakan rumus :

$$Y = \frac{X}{Skor\,ideal}\ x\ 100\%$$

Keterangan :

Y	=	Nilai persentase yang dicari
X	=	Jumlah nilai kategori jawaban dikalikan dengan frekuensi ($\sum$ = N . R)
N	=	Nilai dari setiap jawaban
R	=	Frekuensi
Skor ideal	=	Nilai tertinggi dikalikan dengan jumlah sampel

Setelah diperoleh nilai Y maka interpretasi nilai dapat dilihat sebagai berikut :



Gambar 2.7 Skala Penilaian Kuesioner

Setelah didapatkan jumlah yang pasti dari setiap kuesioner, selanjutnya penulis menarik kesimpulan dari masing-masing kuesioner yang disediakan.

2.7.1 Kelebihan Skala Likert

Beberapa Kelebihan Skala Likert dalam riset berupa survei, antara lain :

1. Mudah dibuat dan diterapkan.
2. Skala Likert lebih mudah membuatnya dibandingkan dengan skala yang lain seperti skala Thurstone.
3. Terdapat kebebasan dalam memasukan pertanyaan- pertanyaan, asalkan sesuai dengan konteks permasalahan yang diteliti.
4. Jawaban suatu item dapat berupa alternatif, sehingga informasi mengenai item tersebut diperjelas.
5. Reliabilitas pengukuran bisa diperoleh dengan jumlah item tersebut diperjelas.
6. Karena jangka responsi yang lebih besar membuat skala Likert dapat memberikan keterangan yang lebih jelas dan nyata tentang pendapatan atau sikap responden tentang isu yang dipertanyakan.

2.7.2 Kekurangan Skala Likert

Beberapa Kekurangan Skala Likert dalam riset berupa survei, antara lain :

1. Karena ukuran yang digunakan adalah ukuran ordinal, skala Likert hanya dapat mengurutkan individu dalam skala, tetapi tidak dapat membandingkan berapa kali satu individu lebih baik dari individu yang lain.
2. Kadangkala total skor dari individu tidak memberikan arti yang jelas, karena banyak pola respons terhadap beberapa item akan memberikan skor yang sama. Adanya kelemahan di atas sebenarnya dapat dipikirkan sebagai *error* dari respons yang terjadi.

2.8 *Proximity Sensor*

Sensor *proximity* adalah sebuah sensor yang bisa mendeteksi keberadaan benda tanpa kontak fisik. Sensor *proximity* memancarkan medan elektromagnetik atau sinar radiasi elektromagnetik (misalnya inframerah) dan mendeteksi perubahan bidang dengan mengembalikan sinyal[15]. Ada empat jenis teknologi sensor *proximity*, diantaranya :

1. *Electrical* (*Inductive* dan *Capacitive*)
2. *Optical* (IR dan Laser)
3. *Magnetic*
4. *Sonar*

Dari beberapa jenis sensor *proximity*, jenis sensor *proximity* dengan tipe *optical* paling banyak digunakan, karena jenis sensor *proximity* dengan tipe *optical* lebih murah dan lebih sederhana dibandingkan dengan jenis sensor yang lainnya, dan bisa mendeteksi benda hingga jarak 5CM. Sensor *proximity* pada ponsel android biasanya terdapat di depan ponsel, tepatnya diatas layar ponsel seperti pada Gambar 2.8.



Gambar 2.8 Letak Sensor *Proximity*

Pada ponsel android sensor *proximity* biasa digunakan untuk mematikan layar saat terjadi panggilan masuk. Hal ini bertujuan untuk menghemat konsumsi batere saat pengguna sedang melakukan panggilan telepon. Sensor *proximity* akan aktif otomatis dan mendeteksi ketika benda

dalam jangkauan sensor, kemudian sensor akan mengirimkan informasi ke sistem ketika benda yang mendekati sensor, dan sistem akan mematikan layar selama benda dekat dengan sensor selama panggilan terjadi

2.9 Adobe Photoshop

Adobe Photoshop merupakan suatu aplikasi yang paling populer dikalangan desainer grafis dan pengolahan citra (*image processing*). Sejak pertama diperkenalkan oleh Adobe corporation pada dekade 90-an, Photoshop langsung mendapatkan hati dikalangan profesional dan praktisi *imege-editing* dengan segala kecanggihan fitur dan kemampuan yang maksimal serta kemudahan dalam penggunaannya (*user friendly*).

Adobe Photoshop yang dirilis pada tahun 2003 yang lalu oleh Adobe corp., adalah Adobe Photoshop CS dengan slogan yang cukup menjanjikan *The Professional Standard in Desktop Digital Imaging* merupakan pengembangan dari Adobe Photoshop generasi sebelumnya, dan pada edisi terakhir ini sudah dilengkapi dengan fasilitas *Image Ready CS*. Kelengkapan ini dimaksudkan untuk memberikan fleksibelitas, efisiensi dan kemudahan dalam penggunaan, sehingga dapat dengan mudah membuat dan menyunting *image* dengan kualitas tinggi yang siap untuk dicetak, *publisher*, ataupun untuk ditempatkan di *web*, yang selama ini merupakan kendala untuk menempatkan *image* berkualitas tinggi pada *web*, karena berhubungan erat dengan beban *loading* data di sisi *client* pengguna.

Adobe Photoshop CS, yang kemudian hanya disebut Photoshop CS saja, merupakan program aplikasi yang canggih yang telah membantu pekerjaan para profesional dalam bidang *image processing*, seperti fotografi, pracetak, periklanan dan *web designer* serta berbagai disiplin pekerjaan lainnya yang berhubungan dengan *imaging*.

Adobe Photoshop CS3 disusun untuk membantu pekerjaan yang fokus pada pembuatan dan pengeditan gambar. Menu ruang kerja yang ada berupa *tools* dan *palettes* yang berfungsi untuk melihat, mengedit, dan menambahkan elemen ke gambar yang diinginkan[1].

2.9.1 Area Kerja Photoshop

Beberapa Area kerja Photoshop, diantaranya :

a. Menu Bar

Berisi perintah utama untuk membuka *file*, *save*, mengubah ukuran gambar, *filter* dan lain-lain.

b. *Option*

Berisi pilihan dari *tool* yang dipilih. Misalnya dipilih kuas/*brush*, maka ukuran/diameter *brush* ada di sini.

c. Image

Menampilkan gambar yang sedang dibuat atau diperbaiki.

d. *Pallete Well*

Cara cepat untuk mengakses *palet brushes, tool resets* dan *Layer Comps*. Juga dapat digunakan untuk meletakkan palet yang sering digunakan.

e. *Toolbox*

Berisi *tool* untuk menyeleksi dan memodifikasi gambar.

f. *Palette*

Berisi jendela-jendela kecil yang di dalamnya terdapat perintah dan pilihan untuk dokumen/gambar yang sedang dikerjakan.

2.10 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:

1. *Multi-platform*: Target sistem operasi Eclipse adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.
2. *Multi-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.
3. *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 popular adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan *plug-in*.

2.10.1 Arsitektur Eclipse

Sejak versi 3.0, Eclipse pada dasarnya merupakan sebuah *kernel*, yang mengangkat *plug-in*. Apa yang dapat digunakan di dalam Eclipse sebenarnya adalah fungsi dari *plug-in* yang sudah diinstal. Ini merupakan basis dari Eclipse yang dinamakan *Rich Client Platform* (RCP). Berikut ini adalah komponen yang membentuk RCP:

1. *Core platform*
2. OSGi
3. SWT (*Standard Widget Toolkit*)
4. JFace
5. Eclipse *Workbench*

Secara standar Eclipse selalu dilengkapi dengan JDT (*Java Development Tools*), *plug-in* yang membuat Eclipse kompatibel untuk mengembangkan program Java, dan PDE (*Plug-in Development Environment*) untuk mengembangkan *plug-in* baru. Eclipse beserta *plug-in*-nya diimplementasikan dalam bahasa pemrograman Java.

Konsep Eclipse adalah IDE yang terbuka (*open*), mudah diperluas (*extensible*) untuk apa saja, dan tidak untuk sesuatu yang spesifik. Jadi, Eclipse tidak saja untuk mengembangkan program Java, akan tetapi dapat digunakan untuk berbagai macam keperluan, cukup dengan menginstal *plug-in* yang dibutuhkan. Apabila ingin mengembangkan program C/C++ terdapat *plug-in* CDT (C/C++ *Development Tools*). Selain itu, pengembangan secara visual bukan hal yang tidak mungkin oleh Eclipse, *plug-in* UML2 tersedia

untuk membuat diagram UML. Dengan menggunakan PDE setiap orang bisa membuat *plug-in* sesuai dengan keinginannya.

2.10.2 Perkembangan Eclipse

Sejak tahun 2006, Eclipse Foundation mengkoordinasikan peluncuran Eclipse secara rutin dan simultan yang dikenal dengan nama *Simultaneous Release*. Setiap versi peluncuran terdiri dari Eclipse Platform dan juga sejumlah proyek yang terlibat dalam proyek Eclipse. Tujuan dari sistem ini adalah untuk menyediakan distribusi Eclipse dengan fitur-fitur dan versi yang terstandarisasi. Hal ini juga dimaksudkan untuk mempermudah deployment dan maintenance untuk sistem enterprise. Adapun versi Eclipse yang telah diluncurkan yaitu :

1. Eclipse 3.0
2. Eclipse 3.1
3. Callisto
4. Europa
5. Ganymede
6. Galileo
7. Helios
8. Indigo
9. Juno
10. Kepler

2.11 *Software Development Kit* (SDK)

Software Development Kit (SDK) adalah suatu kit atau library dari bahasa pemrograman untuk pengembangan atau pembangunan suatu perangkat lunak dan biasanya SDK terdiri dari kumpulan tools yang dibutuhkan. Misalnya bahasa pemrograman java, mempunyai SDK yang berisi suatu *library* yang dapat digunakan untuk membuat suatu aplikasi berbasis java.

2.12 *Java Development Kit* (JDK)

Java Development Kit (JDK) adalah sekumpulan perangkat lunak yang dapat kamu gunakan untuk mengembangkan perangkat lunak yang

berbasis Java, Sedangkan JRE adalah sebuah implementasi dari Java Virtual Machine yang benar-benar digunakan untuk menjalankan program java.Biasanya, setiap JDK berisi satu atau lebih JRE dan berbagai alat pengembangan lain seperti sumber kompiler java, bundling, debuggers, development libraries dan lain sebagainya. Perbedaan JDK dengan SDK *(Software Development Kit)* yaitu JDK adalah sebuah SDK tetapi sebuah SDK tidak harus menjadi sebuah JDK.

2.13 *Android Development Tools* (ADT)

Android Development Tools (*ADT*) adalah *plugin* untuk Eclipse *Intergrated Development Environment* (*IDE*) yang dirancang untuk memberikan lingkungan yang terpadu di mana untuk membangun aplikasi Android.

ADT memperluas kemampuan Eclipse untuk membiarkan para *developer* lebih cepat dalam membuat proyek baru Android, membuat aplikasi UI, menambahkan komponen berdasarkan Android *Framework* API, *debug* aplikasi dalam pengunaan Android SDK, dan membuat file APK untuk mendistribusikan aplikasi.

Mengembangkan aplikasi di Eclipse dengan ADT sangat dianjurkan dan merupakan cara tercepat untuk memulai membuat aplikasi Android, karena banyak kemudahan-kemudahan sebagai tools yang terintegrasi seperti, custom XML editor, dan *debug panel ouput*. Selain itu ADT memberikan dorongan luar biasa dalam mengembangkan aplikasi Android.