BAB II

LANDASAN TEORI

2.1. Aplikasi

Perangkat lunak/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 sebagai suatu paket atau suite aplikasi (*application suite*). Contohnya adalah *Microsoft Office* dan *OpenOffice.org*, 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 setiap aplikasi. Sering kali, aplikasi ini memiliki kemampuan 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.

2.2. Pemesanan

Proses pengolahan pesanan/order melibatkan penyiapan pesanan untuk pengiriman dan penerimaan pesanan ketika pengiriman-pengiriman tiba. Meliputi sejumlah kegiatan, seperti memeriksa kredit pelanggan, pencatatan penjualan, membuat catatan akuntansi yang sesuai, mengatur item yang akan dikirim, penyesuaian catatan persediaan, dan tagihan pelanggan.

2.3. Penjualan

Pengertian penjualan menurut beberapa ahli :

Menurut Winardi (1991, 2), penjualan adalah proses dimana sang penjual memuaskan segala kebutuhan dan keinginan pembeli agar dicapai manfaat baik bagi sang penjual maupun sang pembeli yang berkelanjutan dan yang menguntungkan kedua belah pihak.

Menurut William G. Nickels (1998, 10), penjualan tatap muka adalah interaksi antar individu, saling bertemu muka yang ditujukan untuk menciptakan, memperbaiki, menguasai atau mempertahankan hubungan pertukaran yang saling menguntungkan dengan pihak lain.

Dari definisi diatas dapat dilihat bahwa ruang lingkup pemasaran lebih luas dibandingkan dengan ruang lingkup penjualan karena penjualan merupakan salah satu kegiatan dalam pemasaran.

2.3.1. Jenis – jenis Penjualan

Ada beberapa jenis penjualan menurut Basu Swasta (1998 : 11) yaitu :

1. Trade Selling

Dapat terjadi bilamana produsen dan pedagang besar mempersilahkan pengecer untuk berusaha memperbaiki distributor produk-produk mereka. Hal ini melibatkan para penyalur dengan kegiatan promosi, peragaan, persediaan dan pengadaan produk baru, jadi titik beratnya pada "penjualan melalui" penyalur daripada "penjualan ke" pembeli akhir.

2. Missionary Selling

Dalam *missionary selling* penjualan berusaha ditingkatkan dengan mendorong pembeli untuk membeli barang-barang dari penyalur perusahaan. Dalam hal ini perusahaan yang bersangkutan memiliki penyalur sendiri dalam pendistribusian produknya.

3. Technical Selling

Berusaha meningkatkan penjualan dengan pemberian saran dan nasehat pada pembeli akhir dari barang dan jasanya dengan menunjukkan bagaimana produk dan jasa yang ditawarkan dapat mengatasi masalah tersebut.

4. New Business Selling

Berusaha membuka transaksi baru dengan merubah calon pembeli menjadi pembeli. Jenis penjualan ini sering dipakai oleh perusahaan asuransi.

5. Responsive Selling

Dua jenis penjualan utama disini adalah "*route driving*" dan "*retailling*". Jenis penjualan seperti ini tidak akan menciptakan penjualan yang terlalu besar

meskipun layanan yang baik dan hubungan pelanggan yang menyenangkan dapat menjurus pada pembeli ulang.

2.3.2. Langkah-langkah Dalam Proses Penjualan

Menurut Philip Kotler dialih bahasakan oleh Drs. Alexander Sindoro langkah-langkah dalam proses penjualan meliputi:

1. Memilih Prospek dan Menilai

Langkah pertama dalam proses penjualan adalah memilih prospek (*prospecting*), yaitu mencari siapa yang dapat masuk sebagai pelanggan potensial. Tenaga penjual perlu mengetahui cara menilai prospek (*qualify*) artinya cara mengenali calon yang baik dan menyisihkan calon yang jelek. Prospek dapat dinilai dengan meneliti kemampuan keuangan, *volume* bisnis, kebutuhan spesial, lokasi dan kemungkianan untuk tumbuh.

2. Prapendekatan.

Sebelum mengunjungi seorang calon pembeli, tenaga penjual sebaiknya mempelajari sebanyak mungkin mengenai organisasi (apa yang dibutuhkan, siapa yang terlibat dalam pembelian) dan pembelinya (karakteristik dan gaya membeli). Langkah-langkah ini dikenal dengan istilah prapendekatan. Wiraniaga sebaiknya menetapkan tujuan kunjungan yang mungkin untuk menilai calon, mengumpulkan informasi, atau membuat penjualan langsung.

3. Pendekatan

Dalam langkah ini, wiraniaga sebaiknya mengetahui caranya bertemu dan menyapa pembeli serta menjalin hubungan menjadi awal yang baik. Langkah ini mencakup penampilan wiraniaga, kata-kata pembukaan, dan tindak lanjutan.

4. Presentasi dan Demonstrasi

Dalam langkah presentasi dari proses penjualan, tenaga penjual menceritakan "riwayat" produk kepada pembeli, menunjukan bagaimana produk akan menghasilkan dan menghemat uang. Presentasi penjualan dapat diperbaiki dengan alat bantu demonstrasi, seperti buku kecil, pita *video*, dan sempel produk.

5. Mengatasi Keberatan

Pelanggan hampir selalu mempunyai keberatan selama presentasi atau kettika diminta untuk memesan. Dalam mengatasi keberatan wiraniaga harus menggunakan pendekatan positif, menggali keberatan tersembunyi, meminta pembeli untuk menjelaskan keberatan, menggunakan keberatan sebagai peluang untuk memberikan informasi lebih banyak dan mengubah keberatan menjadi alasan untuk membeli.

6. Menutup

Menutup merupakan langkah dalam proses penjualan ketika wiraniaga meminta pelanggan untuk memesan. Tenaga penjual harus mengetahui cara mengenali tanda-tanda penutupan dari pembeli termasuk gerakan fisik, komentar dan pertanyaan.

7. Tindak Lanjut

Merupakan langkah terakhir dalam proses penjualan ketika wiraniaga melakukan tindak lanjut setelah penjualan untuk memastikan kepuasan pelanggan dan bisnis berulang.

2.4. Android

Android adalah sistem operasi yang berbasis Linux untuk telepon seluler seperti telepon pintar dan komputer tablet. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. 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 peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

Beberapa pendapat mendefinisikan android sebagai berikut :

Android is an operating system (OS) developed by the Open Handset Alliance (OHA). The Alliance is a coalition of more than 50 mobile technology companies ranging from handset manufactures and service providers to semiconductor manufacturers and software developers, including Acer, ARM, Google, eBay, HTC, Intel, LG Electronics, Qualcomm, Sprint, and T-Mobile. The stated goal of the OHA is to "accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience" (OHA, 2009, n.p.).

The basic architecture of Android is shown in Figure 1. At its core, Android OS builds are based on the Linux 2.6 kernel. When running on a hard drive, the Linux system device defaults to the first physical hard drive, or /dev/hd0. In addition, Linux only understands character and block devices, such as keyboards and disk drives, respectively. With Linux on flash, however, a Flash Transition layer provides the system evice functionality. A Memory Technology Device (MTD) is needed to provide an interface between the Linux OS and the physical flash device because flash memory devices are not seen as character or block devices (Dedekind, 2009).

The Android Runtime System utilizes the Dalvik virtual machine (VM), which allows multiple applications to be run concurrently as each application is

its own separate VM. Android applications (the apps of today's common parlance) are compiled into Dalvik executable (.dex) files (DalvikVM.com, 2008). During a forensic examination one will be mainly concerned with the Libraries and, in particular, the SQLite databases. This is where one will find the majority of data that could be of interest in an investigation. Files can be stored on either the device's storage or on the removable secure digital (SD) memory card (Android.com, 2009b).

Unlike the typical desktop operating system, data or other files created by one Android app cannot automatically be viewed by other applications by default. The VM nature of Android allows each application to run its own process. Security is permissions-based and attached at the process level by assigning user and group identifiers to the applications. Application cannot interfere with each other without being given the explicit permissions to do so (Android.com, 2009a).

The security mechanisms of the Android OS could impede a forensic examination although some of the basic tools and techniques could allow investigators to recover data from the device. The first, most obvious step is to perform a traditional forensics analysis of the microSD card from the phone. This is the least effective method as it can only is access the data that apps directly store on the SD card. SD cards use the FAT32 file system and are easily imaged and examined using traditional forensics tools (including write-blocking hardware) (TalkForensics, 2009).

The Android file system is Yet Another Flash File System 2 (YAFFS2). YAFFS, developed in 2002, was the first file system designed for NAND (Not-AND) flash memory devices. YAFFS2 was designed in 2004 in response to the availability of larger sized NAND flash devices; older chips support a 512 byte page size whereas newer NAND memory has 2096 byte pages. YAFFS2 is backward compatible with YAFFS (Manning, 2002).

Keunggulan utama Android adalah gratis dan *open source*, yang membuat smartphone Android dijual lebih murah dibandingkan dengan Blackberry atau iPhone meski fitur (hardware) yang ditawarkan Android lebih baik. Beberapa fitur utama dari Android antara lain WiFi hotspot, Multi-touch, Multitasking, GPS, support java, mendukung banyak jaringan (GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, and WiMAX) dan juga kemampuan dasar handphone pada umumnya.

Di dunia ini terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari Google atau Google Mail Services (GMS)

dan kedua adalah yang benar–benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai Open Handset Distribution (OHD).

2.4.1. Sejarah Android

Pada Juli 2000, Google bekerjasama dengan Android Inc., perusahaan yang berada di Palo Alto, California Amerika Serikat. Para pendiri Android Inc. Bekerja pada Google, di antaranya Andy Rubin, Rich Miner, Nick Sears, dan Chris White. Saat itu banyak yang menganggap fungsi Android Inc. hanyalah sebagai perangkat lunak pada telepon seluler. Sejak saat itu muncul rumor bahwa Google hendak memasuki pasar telepon seluler. Di perusahaan Google, tim yang dipimpin Rubin bertugas mengembangkan program perangkat seluler yang didukung oleh kernel Linux. Hal ini menunjukkan indikasi bahwa Google sedang bersiap menghadapi persaingan dalam pasar telepon seluler. versi android terbaru yaitu versi 3.0. Android juga sudah bergabung dengan beberapa smart *Mobile* seperti Nokia, Sony Ericsson, dan lainnya.

Sekitar September 2007 sebuah studi melaporkan bahwa Google mengajukan hak paten aplikasi telepon seluler (akhirnya Google mengenalkan Nexus One, salah satu jenis telepon pintar GSM yang menggunakan Android pada sistem operasinya. Telepon seluler ini diproduksi oleh HTC Corporation dan tersedia di pasaran pada 5 Januari 2010). Pada 9 Desember 2008, diumumkan anggota baru yang bergabung dalam program kerja Android ARM Holdings, Atheros Communications, diproduksi oleh Asustek Computer Inc, Garmin Ltd, Softbank, Sony Ericsson, Toshiba Corp, dan Vodafone Group Plc. Seiring

pembentukan Open Handset Alliance, OHA mengumumkan produk perdana mereka, Android, perangkat bergerak (*Mobile*) yang merupakan modifikasi kernel Linux 2.6. Sejak Android dirilis telah dilakukan berbagai pembaruan berupa perbaikan bug dan penambahan fitur baru. Telepon pertama yang memakai sistem operasi Android adalah HTC Dream, yang dirilis pada 22 Oktober 2008. Pada penghujung tahun 2009 diperkirakan di dunia ini paling sedikit terdapat 18 jenis telepon seluler yang menggunakan Android. Berikut adalah sejarah dan fitu sistem operasi Android semua versi:

1. Android Versi 1.0 (Apple pie)

Android 1.0 dirilis pada tanggal 23 September 2008 yang memiliki kode nama *Apple pie* serta ukuran layar 320×480 HVGA.

2. Android Versi 1.1 (Banana bread)

Android 1.1 dirilis pada tanggal 9 Februari 2009 yang memiliki kode nama *Banana bread* serta ukuran layar 320×480 HVA.

3. Android 1.5 (Cupcake)

Android 1.5 dirilis pada tanggal 30 April 2009 yang memiliki kode nama *Cupcake*. Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur yakni kemampuan merekam dan menonton *video*, mengunggah *video* ke Youtube dan gambar ke Picasa langsung dari telepon seluler, dukungan *Bluetooth*, animasi layar, dan keyboard pada layar yang dapat disesuaikan dengan sistem.

4. Android 1.6 (Donut)

Android 1.6 dirilis pada tanggal 15 September 2009 yang memiliki kode nama *Donut*. Versi ini perbaikan dari Android *Cupcake* 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; kemampuan dial kontak; teknologi *text to change speech* (tidak tersedia pada semua ponsel); pengadaan resolusi VWGA.

5. Android versi 2.0/2.1 (Eclair)

Android versi 2.0/2.1 dirilis pada tanggal 3 Desember 2009 yang memiliki kode nama *Eclair*. 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.

6. Android versi 2.2 (Froyo)

Android versi 2.2 dirilis pada tanggal 20 Mei 2010 yang memiliki kode nama *Froyo*. *Froyo* akan memperkenalkan peningkatan kecepatan dengan JIT optimalisasi dan Chrome V8 mesin JavaScript, dan menambahkan Wi-Fi *hotspot* dan dukungan *Adobe Flash*.

7. Android versi 2.3 (Gingerbread)

Android versi 2.3 dirilis pada tanggal 6 Desember 2010 yang memiliki kode nama *Gingerbread*. Android *Gingerbread* adalah android yang mempermudah pengguna antarmuka , meningkatkan *keyboard* lunak dan fitur *copy / paste*, dan menambahkan dukungan untuk *Near Field Communication*. pada versi ini Android telah melakukan *update* sebanyak 5 kali yang dirilis pada tanggal yang berbeda-beda serta penambahan di fitur-fitur baru, mereka masing-masing adalah Android 2.3.3, Android 2.3.4, Android, 2.3.5, Android 2.3.6, dan terahir Android 2.3.7.

8. Android versi 3.0 (Honeycomb)

Android versi 3.0 dirilis pada tanggal 15 Juli 2011 yang memiliki kode nama *Honeycomb*. Di versi ini Android akhirnya jalan pada sebuah Tablet, dan Tablet pertama yang menggunakan sistem operasi *Honeycomb* adalah Motorola Xoom.

9. Android versi 4.0 (Ice Cream Sandwich)

Android versi 4.0 dirilis pada tanggal 19 Oktober 2011 yang memiliki kode nama *Ice Cream Sandwich*. Di versi Android menyempurnakan seluruh GUI yang dibangun dan penambahan fitur seperti *Facial recognition (Face Unlock),* UI *use Hardware acceleration, Better voice recognition (dictating/Voice typing), Web browser, allows up to 16 tabs, Updated launcher (customizable),* Android *Beam app to exchange data through NFC, Resizeable widgets.*

10. Android versi 4.1 (Jelly Bean)

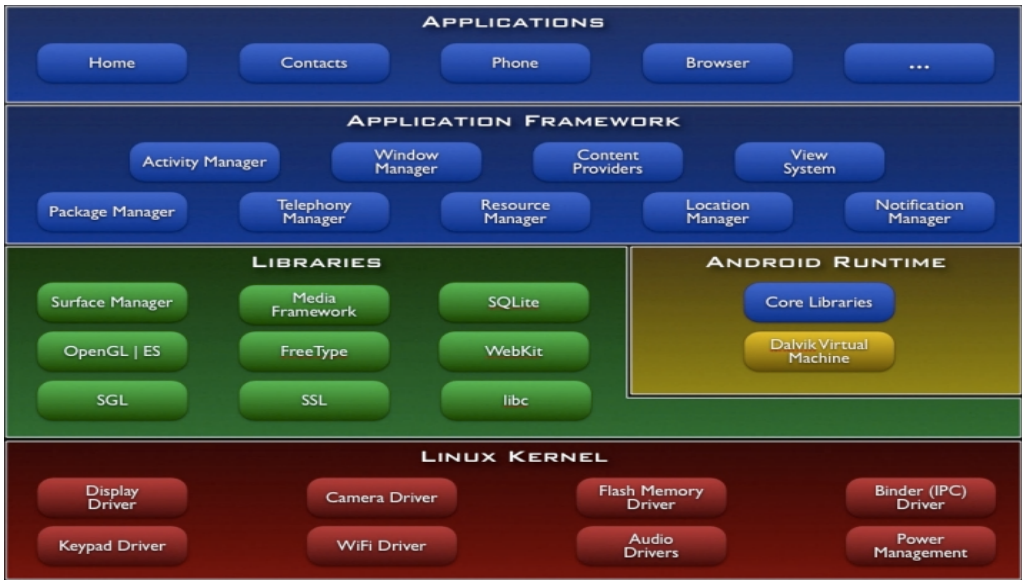
Android *Jelly Bean* yang diluncurkan pada acara Google I/O lalu membawa sejumlah keunggulan dan fitur baru. Penambahan baru diantaranya meningkatkan *input keyboard*, desain baru fitur pencarian, UI yang baru dan pencarian melalui *Voice Search* yang lebih cepat. Tak ketinggalan Google Now juga menjadi bagian yang diperbarui. Google Now memberikan informasi yang tepat pada waktu yang tepat pula. Salah satu kemampuannya adalah dapat mengetahui informasi cuaca, lalu-lintas, ataupun hasil pertandingan olahraga. Sistem operasi Android *Jelly Bean* 4.1 muncul pertama kali dalam produk tablet Asus, yakni Google Nexus 7.

11. Android versi 4.2 (Jelly Bean)

Fitur *photo sphere* untuk panorama, *daydream* sebagai *screensaver*, *power control*, *lock screen widget*, menjalankan banyak *user* (dalam tablet saja), *widget* terbaru. Android 4.2 Pertama kali dikenalkan melalui LG Google Nexus 4.

2.4.2. Anatomi Aplikasi Android

Dalam paket sistem operasi android tediri dari beberapa unsur seperti tampak pada gambar 2.8. Secara sederhana arsitektur android merupakan sebuah kernel Linux dan sekumpulan pustaka C / C++ dalam suatu *Framework* yang menyediakan dan mengatur alur proses aplikasi.



Gambar 2.1 *Detail* Anatomi Android
Sumber: Small Scale Digital Device Forensics Journal Vol.4, No.1, September 2010, ISSN#1941-6164

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 servis 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.
2. 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 generic 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.

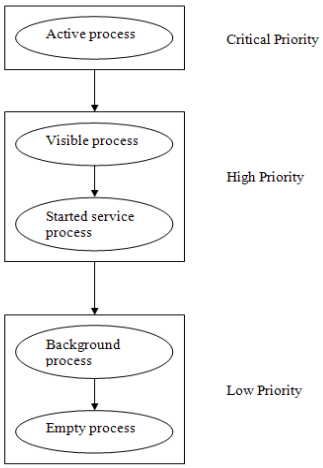
Lapisan aplikasi android sangat berbeda dibandingkan dengan sistem operasi lainnya. Pada android semua aplikasi, baik aplikasi inti (*native*) maupun aplikasi pihak ketiga berjalan diatas lapisan aplikasi dengan menggunakan pustaka API (*Application Programming Interface*) yang sama.

2.4.3. Siklus Hidup Aplikasi Android

Siklus hidup aplikasi android dikelola oleh sistem, berdasarkan kebutuhan pengguna, sumber daya yang tersedia, dan sebagainya. Misalnya Pengguna ingin

menjalankan *browser web*, pada akhirnya sistem yang akan menentukan menjalankan aplikasi. Sistem sangat berperan dalam menentukan apakah aplikasi dijalankan, dihentikan sementara, atau dihentikan sama sekali. Jika pengguna ketika itu sedang menjalankan sebuah *Activity*, maka sistem akan memberikan perioritas utama untuk aplikasi yang tersebut. Sebaliknya, jika suatu *Activity* tidak terlihat dan sistem membutuhkan sumber daya yang lebih, maka *Activity* yang prioritas rendah akan ditutup.

Android menjalankan setiap aplikasi dalam proses secara terpisah, yang masing-masing memiliki mesin *virtual* pengolah sendiri, dengan ini melindungi penggunaan memori pada aplikasi. Selain itu juga android dapat mengontrol aplikasi mana yang layak menjadi prioritas utama. Karenanya android sangat sensitive dengan siklus hidup aplikasi dan komponen-komponennya. Perlu adanya penanganan terhadap setiap kondisi agar aplikasi menjadi stabil. Gambar 2.2 menunjukkan prioritas dari aplikasi.



Gambar 2.2 Prioritas Aplikasi berdasarkan *Activity*

Sumber : http://riyan214.wordpress.com

2.4.4. Kelebihan Android

Sudah banyak *platform* untuk perangkat selular saat ini, termasuk didalamnya Symbian, iPhone, Windows *Mobile*, BlackBerry, Java *Mobile* Edition, Linux *Mobile* (LiM), dan banyak lagi. Namun ada beberapa hal yang menjadi kelebihan Android. Walaupun beberapa fitur-fitur yang ada telah muncul sebelumnya pada platform lain, Android adalah yang pertama menggabungkan hal seperti berikut :

1. Keterbukaan, Bebas pengembangan tanpa dikenakan biaya terhadap sistem karena berbasiskan Linux dan *open source*. Pembuat perangkat menyukai hal ini karena dapat membangun *platform* yang sesuai yang diinginkan tanpa harus membayar royality. Sementara pengembang *software* menyukai karena android dapat digunakan diperangkat manapun dan tanpa terikat oleh *vendor* manapun.
2. Arsitektur komponen dasar android terinspirasi dari teknologi internet *Mashup*. Bagian dalam sebuah aplikasi dapat digunakan oleh aplikasi lainnya, bahkan dapat diganti dengan komponen lain yang sesuai dengan aplikasi yang dikembangkan.
3. Banyak dukungan *service*, kemudahan dalam menggunakan berbagai macam layanan pada aplikasi seperti penggunaan layanan pencarian lokasi, database SQL, browser dan penggunaan peta. Semua itu sudah tertanam pada android sehingga memudahkan dalam pengembangan aplikasi.
4. Siklus hidup aplikasi diatur secara otomatis, setiap program terjaga antara satu sama lain oleh berbagai lapisan keamanan, sehingga kerja sistem menjadi

lebih stabil. Pengguna tak perlu kawatir dalam menggunakan aplikasi pada perangkat yang memorinya terbatas.

5. Dukungan grafis dan suarat terbaik, dengan adanya dukungan 2D grafis dan animasi yang diilhami oleh *Flash* menyatu dalam 3D menggunakan *OpenGL* memungkinkan membuat aplikasi maupun game yang berbeda.
6. Portabilitas aplikasi, aplikasi dapat digunakan pada perangkat yang ada saat ini maupun yang akan datang. Semua program ditulis dengan menggunakan bahas pemrograman Java dan dieksekusi oleh mesin *virtual* Dalvik, sehingga kode program portabel antara ARM, X86, dan arsitektur lainnya. Sama halnya dengan dukungan masukan seperti penggunaan *Keyboard*, layar sentuh, *trackball* dan resolusi layar semua dapat disesuaikan dengan program.

2.5. Rational Unified Process (RUP)

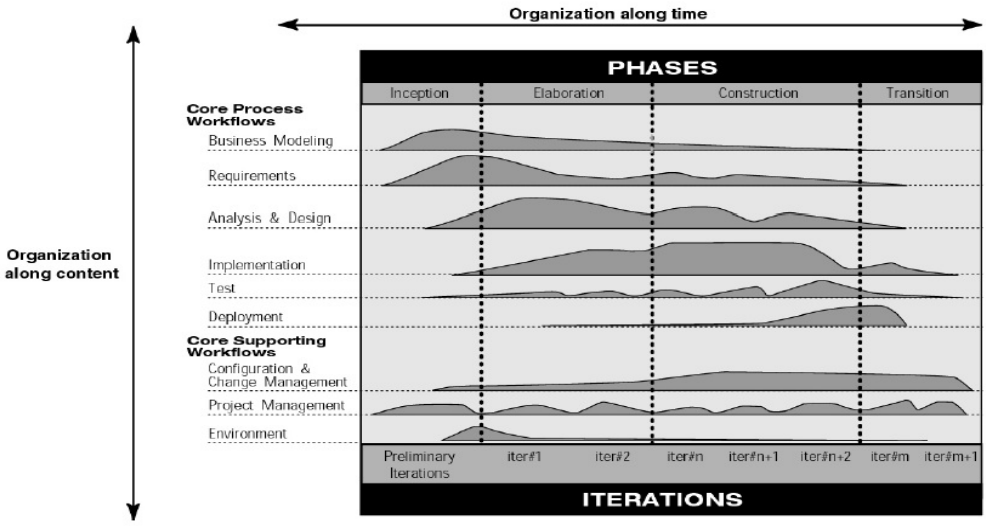
Rational Unified Process (RUP) merupakan suatu metode rekayasa perangkat lunak yang dikembangkan dengan mengumpulkan berbagai *best practises* yang terdapat dalam industri pengembangan perangkat lunak. Ciri utama metode ini adalah menggunakan *use-case driven* dan pendekatan iteratif untuk siklus pengembangan perangkat lunak.

RUP menggunakan konsep *object oriented*, dengan aktifitas yang berfokus pada pengembangan model dengan menggunakan *Unified Model Language* (UML). Melalui gambar dibawah dapat dilihat bahwa RUP memiliki, yaitu:

1. **Dimensi pertama** digambarkan secara horizontal. Dimensi ini mewakili aspek-aspek dinamis dari pengembangan perangkat lunak. Aspek ini

dijabarkan dalam tahapan pengembangan atau fase. Setiap fase akan memiliki suatu *major milestone* yang menandakan akhir dari awal dari phase selanjutnya. Setiap phase dapat berdiri dari satu atau beberapa iterasi. Dimensi ini terdiri atas *Inception*, *Elaboration*, *Construction*, dan *Transition*.

2. **Dimensi kedua** digambarkan secara vertikal. Dimensi ini mewakili aspek-aspek statis dari proses pengembangan perangkat lunak yang dikelompokkan ke dalam beberapa disiplin. Proses pengembangan perangkat lunak yang dijelaskan kedalam beberapa disiplin terdiri dari empat elemen penting, yakni *who is doing*, *what, how* dan *when*. Dimensi ini terdiri atas *Business Modeling, Requirement, Analysis and Design, Implementation, Test, Deployment, Configuration* dan *Change Manegement, Project Management, Environtment.*



Gambar 2.3. Arsitektur *Rational Unified Process* (RUP)

(Sumber : *Rational-The Software Development Company, Rational Unified Process: Best Practices for Software Development Teams*)

2.6. Konsep 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 *Unifed Modelling Language (UML)* sebagai bahasa standar dalam *Object Oriented Analysist Design (OOAD).*

2.6.1. 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*.

2.6.1.1. 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.

2.6.1.2. 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.

2.6.1.3. Behavior Diagram

Behavior diagram dapat dikelompokan menjadi tiga diagram, yaitu :

1. *Activity Diagram*

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

2. *Interaction Diagram*

Interaction Diagram dibagi menjadi dua model diagram yaitu :

a. *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*.
b. *Colaboration 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*.

2.6.1.4. 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.

2.6.1.5. Kelebihan UML

Kelebihan UML dibandingkan dengan bahasa permodelan yang lain antara lain :

1. Menyediakan bahasa pemodelan visual yang *ekspresif* dan siap pakai untuk mengembangkan dan pertukaran model-model yang berarti.
2. Menyediakan mekanisme perluasan dan spesialisasi untuk memperluas konsep-konsep inti.
3. Mendukung spesifikasi independen bahasa pemrograman dan proses pengembangan tertentu.
4. Menyediakan basis formal untuk bahasa pemodelan.
5. Memadukan praktek-praktek terbaik di industri perangkat lunak menjadi terminologi dan notasi yang diterima luas.
6. Menyediakan kemampuan merepresentasikan semua konsep yang relevan untuk sistem perangkat lunak.
7. Menyediakan fleksibilitas yang diperlukan bagi konsep-konsep perangkat lunak yang baru.

2.6.1.6. Kekurangan UML

Sedangkan kekurangan UML antara lain :

1. UML bukanlah bahasa pemrograman visual, melainkan bahasa pemodelan visual.
2. UML bukan spesifikasi dari tool, tapi spesifikasi bahasa pemodelan.
3. UML bukanlah proses, tapi yang memungkinkan proses-proses.

2.7. 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*.

2.7.1. Sejarah Singkat

Pada 1991, sekelompok insinyur Sun dipimpin oleh Patrick Naughton dan James Gosling ingin merancang bahasa komputer untuk perangkat konsumer seperti *cable TV Box*. Karena perangkat tersebut tidak memiliki banyak memori, bahasa harus berukuran kecil dan mengandung kode yang liat. Juga karena manufaktur – manufaktur berbeda memilih *processor* yang berbeda pula, maka bahasa harus bebas dari manufaktur manapun. Proyek diberi nama kode ”Green”.

Kebutuhan untuk fleksibilitas, kecil, liat dan kode yang netral terhadap *platform* mengantar tim mempelajari implementasi Pascal yang pernah dicoba. Niklaus Wirth, pencipta bahasa Pascal telah merancang bahasa portabel yang menghasilkan *intermediate code* untuk mesin hipotesis. Mesin ini sering disebut dengan mesin maya *(virtual machine)*. Kode ini kemudian dapat digunakan di sembarang mesin yang memiliki *interpreter*. Proyek Green menggunakan mesin maya untuk mengatasi isu utama tentang netral terhadap arsitektur mesin.

Karena orang-orang di proyek Green berbasis C++ dan bukan Pascal maka kebanyakan sintaks diambil dari C++, serta mengadopsi orientasi objek dan bukan prosedural. Mulanya bahasa yang diciptakan diberi nama ”Oak” oleh James

Gosling yang mendapat inspirasi dari sebuah pohon yang berada pada seberang kantornya, namun dikarenakan nama Oak sendiri merupakan nama bahasa pemrograman yang telah ada sebelumnya, kemudian SUN menggantinya dengan JAVA. Nama JAVA sendiri terinspirasi pada saat mereka sedang menikmati secangkir kopi di sebuah kedai kopi yang kemudian dengan tidak sengaja salah satu dari mereka menyebutkan kata JAVA yang mengandung arti asal bijih kopi. Akhirnya mereka sepakat untuk memberikan nama bahasa pemrograman tersebut dengan nama Java.

Produk pertama proyek Green adalah Star 7 (*7), sebuah kendali jarak jauh yang sangat cerdas. Dikarenakan pasar masih belum tertarik dengan produk konsumer cerdas maka proyek Green harus menemukan pasar lain dari teknologi yang diciptakan. Pada saat yang sama, implementasi WWW dan Internet sedang mengalami perkembangan pesat. Di lain pihak, anggota dari proyek Green juga menyadari bahwa Java dapat digunakan pada pemrograman internet, sehingga penerapan selanjutnya mengarah menjadi teknologi yang berperan di web.

Sebagai sebuah bahasa pemrograman, Java dapat membuat seluruh bentuk aplikasi, *desktop, web* dan lainnya, sebagaimana dibuat dengan menggunakan bahasa pemrograman konvensional yang lain.

Java adalah bahasa pemrograman yang berorientasi objek (OOP) dan dapat dijalankan pada berbagai *platform* sistem operasi. Perkembangan Java tidak hanya terfokus oada satu sistem operasi, tetapi dikembangkan untuk berbagai sistem operasi dan bersifat *open source.* Sebagai sebuah peralatan pembangun, teknologi Java menyediakan banyak *tools* : *compiler, interpreter,* penyusun dokumentasi,

paket kelas dan sebagainya. Aplikasi dengan teknologi Java secara umum adalah aplikasi serbt a guna yang dapat dijalankan pada seluruh mesin yang memiliki *Java Runtime Environment* (JRE).

Terdapat dua komponen utama dari *Deployment Environment*. Yang pertama adalah JRE, yang terdapat pada paket J2SDK, mengandung kelas – kelas untuk semua paket teknologi Java yang meliputi kelas dasar dari Java, komponen GUI dan sebagainya. Komponen yang lain terdapat pada Web Browser. Hampir seluruh Web Browser komersial menyediakan *interpreter* dan *runtime environment* dari teknologi Java.

2.7.2. 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.

2.7.3. Kelebihan Java

Kelebihan-kelebihan dari Java antara lain, yaitu :

1. *Multiplatform*. Kelebihan utama dari Java ialah dapat dijalankan di beberapa *platform* / sistem operasi komputer, sesuai dengan prinsip tulis sekali, jalankan di mana saja. Dengan kelebihan ini pemrogram cukup menulis sebuah program Java dan dikompilasi (diubah, dari bahasa yang dimengerti manusia menjadi bahasa mesin / *bytecode*) sekali lalu hasilnya dapat dijalankan di atas beberapa platform tanpa perubahan. Kelebihan ini memungkinkan sebuah program berbasis java dikerjakan diatas operating system Linux tetapi dijalankan dengan baik di atas Microsoft Windows. *Platform* yang didukung sampai saat ini adalah Microsoft Windows, Linux, Mrac OS dan Sun Solaris. Penyebanya adalah setiap sistem operasi menggunakan programnya sendiri-sendiri (yang dapat diunduh dari situs Java) untuk menginterpretasikan *bytecode* tersebut.
2. OOP (*Object Oriented Programming* - Pemrogram Berorientasi Objek) yang artinya semua aspek yang terdapat di Java adalah Objek. Java merupakan salah satu bahasa pemrograman berbasis objek secara murni. Semua tipe data diturunkan dari kelas dasar yang disebut Object. Hal ini sangat memudahkan pemrogram untuk mendesain, membuat, mengembangkan dan mengalokasi kesalahan sebuah program dengan basis Java secara cepat, tepat, mudah dan terorganisir. Kelebihan ini menjadikan Java sebagai salah satu bahasa pemograman termudah, bahkan untuk fungsi fungsi yang advance seperti komunikasi antara komputer sekalipun.

3. 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. Kelengkapan perpustakaan ini ditambah dengan keberadaan komunitas Java yang besar yang terus menerus membuat perpustakaan-perpustakaan baru untuk melingkupi seluruh kebutuhan pembangunan aplikasi.
4. Bergaya C++, memiliki sintaks seperti bahasa pemrograman C++ sehingga menarik banyak pemrogram C++ untuk pindah ke Java. Saat ini pengguna Java sangat banyak, sebagian besar adalah pemrogram C++ yang pindah ke Java. Universitas-universitas di Amerika Serikat juga mulai berpindah dengan mengajarkan Java kepada murid-murid yang baru karena lebih mudah dipahami oleh murid dan dapat berguna juga bagi mereka yang bukan mengambil jurusan komputer.
5. 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).

2.7.4. Kekurangan Java

Selain kelebihan, java memiliki kekurangan yang antara lain adalah :

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 (lebih spesifik lagi, Delphi dan *Object* Pascal). Biasanya ini bukan merupakan masalah bagi pihak yang menggunakan teknologi terbaru (karena *trend* memori terpasang makin murah), tetapi menjadi masalah bagi mereka yang masih harus berkutat dengan mesin komputer berumur lebih dari 4 tahun.

2.8. Database

Database pada dasarnya memiliki pengertian kumpulan data-data dan informasi yang terstruktur dalam suatu tabel dan relasi sehingga memudahkan dalam pengolahan data. *Database* ini digunakan khususnya untuk arus informasi atau data dalam jumlah besar. *Database* dipakai untuk menyimpan data sehingga dapat dimanipulasi dengan mudah. *Database* dipakai untuk aplikasi sederhana sampai yang rumit dimana melibatkan beberapa *use*r. Oleh karena itu *Database* dibagi sesuai dengankompleksitasnya. Ada dua model *Database*, yaitu :

1. *Database* yang berdiri sendiri *(stand alon*e). Merupakan *database* yang sangat sederhana karena disimpan dalam sistem *file* lokal dan mengakses pada mesin yang sama. *Database* ini hanya dipakai untuk satu aplikasi saja.
2. *Database* terbagi *(file shar*e). Hampir sama dengan *database stand-alone,*tetapi dapat diakses oleh beberapa *use*r. *Database* ini akan mengalami masalah jika aplikasi memerlukan banyak perhitungan dan pada saat yang bersamaan ada pengaksesan ke dalam *databas*e.

2.8.1. Structured Query Language (SQL)

SQL adalah Cara berinteraksi/berkomunikasi antara pemakai dengan basis data diatur dalam suatu bahasa khusus yang ditetapkan oleh perusahaan pembuat DBMS. Bahasa itu disebut Bahasa Basis Data. Bahasa Basis Data dapat dipilah ke dalam 2 bentuk yaitu :

1. *Data Definition Language* (DDL)

Dengan bahasa ini kita dapat membuat tabel baru, membuat indeks, mengubah tabel dan sebagainya. Perintah DDL dalam SQL meliputi :

1. CREATE TABLE – membuat tabel baru dalam database
2. ALTER TABLE – mengubah definisi tabel
3. DROP TABLE – menghapus tabel
4. CREATE INDEX – membuat index (search key)
5. DROP INDEX - menghapus index

2. Data Manipulation Language (DML)

Merupakan bentuk bahasa basis data yang berguna untuk melakukan manipulasi dan pengambilan data pada suatu basis data. Manipulasi data dapat berupa :

1. Penyisipan /Penambahan data baru ke suatu basis data
2. Penghapusan data dari suatu basis data
3. Pengubahan data di suatu basis data

Perintah SQl untuk DML (Data Manipulation Language) antara lain :

1. SELECT – Mengekstrak data dari tabel-tabel di database
2. UPDATE – Mengupdate data dalam tabel di database
3. DELETE – Menghapus data dari tabel di database
4. INSERT INTO – Menyisipkan/menambah data baru ke dalam tabel di database

2.8.2. Primary Key

Pada tabel dalam database yang didesain dengan baik, terdapat sebuah atau lebih kolom yang digunakan sebagai *primary key*. *Primary key* adalah suatu informasi yang menjadikan suatu satu *record* adalah unik terhadap *record* yang lain dalam satu tabel.

Contoh informasi yang dapat digunakan sebagai *primary key* adalah informasi Nomor Induk Pegawai atau Nomor Induk Mahasiswa. Pada saat pembuatan tabel, *primary key* dibuat dari kolom unik yang biasanya merupakan suatu tipe *integer* dan hindari untuk menggunakan tipe yang lain. Sebab *indeks* yang dibuat untuk *primary key*, ukurannnya menjadi besar. Dengan membuat

informasi tambahan yang berupa *integer* sebagai *primary key* selain *indeks* yang digunakan berukuran kecil, juga meningkatkan kecepatan proses *query*. *Primary key* yang ditambahkan ini sering juga disebut kolom *internalID*.

2.8.3. Foreign Key

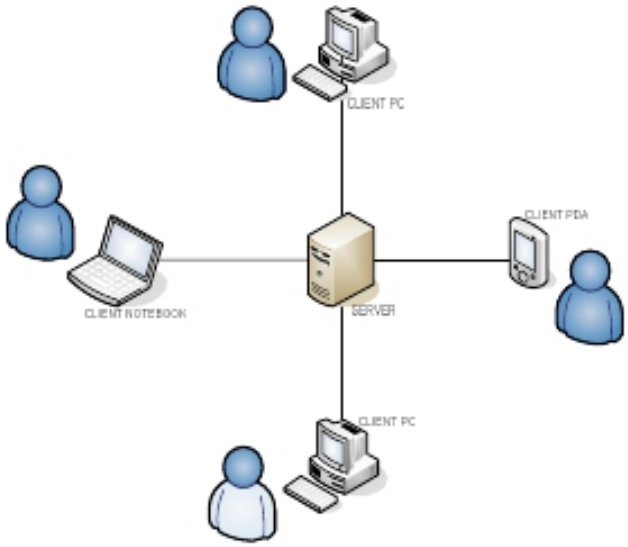
Tabel juga dapat berisi kolom *foreign key*. *Foreign key* pada suatu tabel dapat digunakan untuk mengidentifiksi informasi pada tabel yang lain. *Foreign key* juga digunakan untuk menciptakan *referenstial integrity*, yaitu jika pada suatu tabel di-*update*, maka tabel lain di *update* pula. *Field* yang menjadi *foreign key* pada tabel pertama tidaklah unik, tetapi pada tabel lain *field* tersebut harus menjadi *primary key*. Karena *foreign key* akan menjadi *primary key* pada tabel kedua, maka *primary key* harus dibuat terlebih dahulu.

2.8.4. MySQL

MySQL adalah salah satu jenis *database server* yang sangat terkenal. Kepopulerannya disebabkan MySQL menggunakan SQL *(Structured QueryLanguag*e) sebagai bahasa dasar untuk mengakses *database*-nya. Selain itu *software* ini bersifat *open source* sehingga tidak perlu membayar untuk menggunakannya. MySQL termasuk jenis RDBMS *(Relational Database -Management Syste*m). Itulah sebabnya istilah seperti tabel, baris dan kolom digunakan pada MySQL. Pada MySQL, sebuah *database* mengandung satu atau sejumlah tabel. Tabel terdiri dari sejumlah baris dan setiap baris mengandung satu atau beberapa kolom.

2.9. Client Server

Client-server merupakan arsitektur komputasi yang memisahkan antara klien dengan server, dan kebanyakan arsitektur ini diimplementasikan pada jaringan komputer. Namun bisa juga diimplementasikan antar perangkat lain seperti juga *handphone* dengan *handphone*, ataupun *handphone* dengan komputer. Setiap klien atau server yang terhubung ke jaringan disebut sebagai *node.* Umumnya arsitektur ini memiliki 2 tipe *node* yaitu: klien dan server. Tipe ini biasanya disebut pula *two-tier*. Dengan arsitektur ini, memungkinkan *device* untuk melakukan *sharing files* dan *resources.*



Gambar 2.4 Arsitektur Umum Client Server

Sumber : http://www.scribd.com/

Setiap *instance* dari *software* pada klien dapat mengirimkan *request* kepada satu atau beberapa server yang terhubung kepadanya. Server kemudian menerima *request* tersebut, memprosesnya, setelah itu mengirimkan informasi yang

dihasilkan kepada klien. Meskipun konsep ini diaplikasikan pada berbagai jenis aplikasi, namun secara garis besar pola *request* dan *response* nya tetap sama. Beberapa penerapan arsitektur *client server*, yaitu pada *web browser* sebagai klien, serta *web server*, *database* server, dan *mail* server sebagai server.

2.9.1. Karakteristik Client Server

Secara umum karakteristik dari *client* adalah sebagai berikut:

1. Aktif
2. Menginisialisasi *request*
3. Menunggu balasan dari *request*
4. Umumnya terhubung dengan sedikit server dalam suatu waktu
5. Umumnya berinteraksi secara langsung dengan *end-user* menggunakan *GUI (Graphical User Interface).*

Sedangkan karakteristik server secara umum yaitu:

1. Pasif
2. Menunggu *request* dari klien
3. Seketika terdapat *request* yang masuk kepadanya, server akan langsung memproses dan mengirimkan hasil informasinya kepada klien yang memintanya
4. Umumnya menerima koneksi dari klien dalam jumlah yang besar
5. Umumnya tidak berinteraksi secara langsung dengan *enduser.*
6. Dapat bersifat *stateless* ataupun *stateful.*

2.9.2. Keuntungan dan Kerugian Arsitektur Client Server

Keuntungan dari penerapan arsitektur *client server* antara lain:

1. Semua data disimpan pada server, sehingga manajemen sekuritas penuh berada pada server, hal ini menjadi baik dibandingkan dengan manajemen sekuritas data yang tersebar, server menjamin siapa-siapa saja yang berhak mengakses data.
2. Oleh karena *storage* terpusat, proses *update* data menjadi mudah untuk dilakukan daripada melakukan *update* data yang tersebar. Pada kasus *peer to peer*, update data dilakukan di setiap *peer*, hal ini tentunya sangat memakan waktu dan rentan terjadinya inkonsistensi data.

Sedangkan kerugian dari penerapan arsitektur klien server antara lain:

1. Kongesti trafik menjadi masalah utama arsitektur klien server. Semakin banyak jumlah *request client* secara simultan, dapat mengakibatkan server dan jaringan kelebihan beban.
2. Jika server mengalami kegagalan, maka *request* dari klien tidak dapat dipenuhi, hal ini berlaku untuk semua klien yang terkoneksi dengan server.

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. 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.
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. Sejarah Eclipse

Eclipse awalnya dikembangkan oleh IBM untuk menggantikan perangkat lunak IBM Visual Age for Java 4.0. Produk ini diluncurkan oleh IBM pada tanggal 5 November 2001, yang menginvestasikan sebanyak US$ 40 juta untuk pengembangannya. Semenjak itu konsursium Eclipse Foundation mengambil alih untuk pengembangan Eclipse lebih lanjut dan pengaturan organisasinya.

2.10.2. 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.

2.10.3. 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

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.