BAB 2

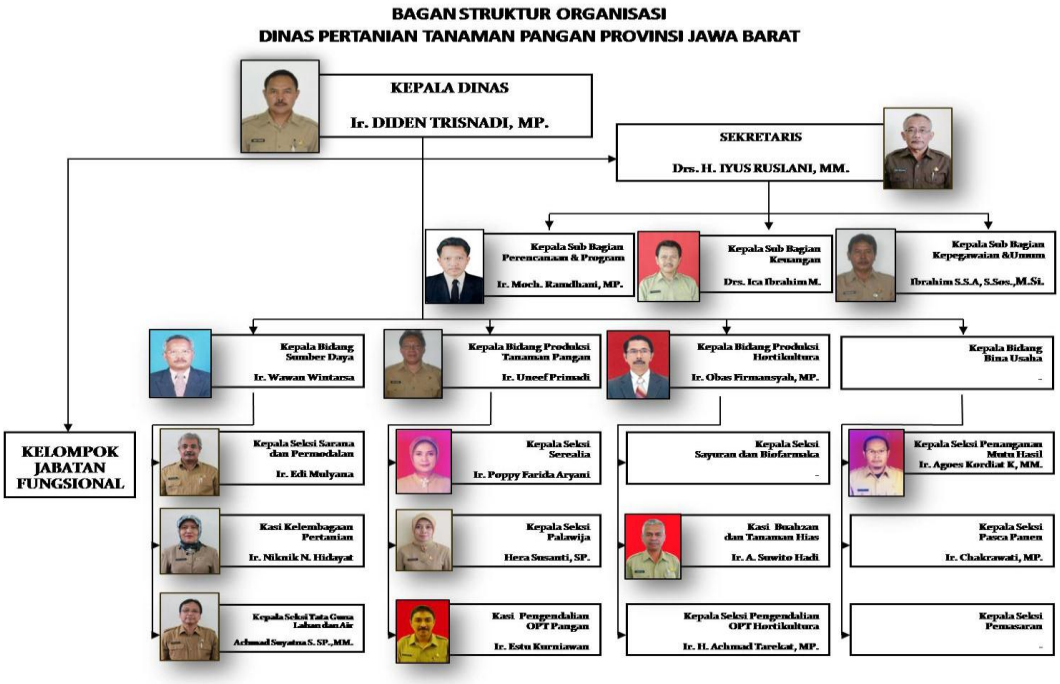
TINJAUAN PUSTAKA

2.1 Profil Dinas Pertanian Tanaman Pangan Prov.Jawa Barat

Dinas Pertanian Tanaman Pangan Provinsi Jawa Barat (Diperta Prov.Jabar) merupakan sebuah instansi yang bekerja untuk meningkatkan sumber daya manusia pertanian, mendukung ketahanan pangan, menambah nilai produk pertanian,meningkatkan sarana produksi serta mewujudkan kelestarian sumber daya alam melalui pembangunan pertanian yang berwawasan lingkungan,demi mewujudkan petani Jawa Barat yang mandiri,dinamis dan sejahtera, dan instansi ini berada di Jl.Surapati No.71 Bandung.

2.2 Struktur Organisasi Dinas Pertanian Tanaman Pangan Provinsi Jawa Barat

Adapun struktur organisasi dari Dinas Pertanian Tanaman Pangan Provinsi Jawa Barat dapat dilihat pada gambar :



Gambar 2-1 Struktur Organisasi

Deskripsi Jabatan:

Adapun Deskripsi Jabatan pegawai yang ada di Dinas Pertanian Tanaman Pangan, adalah sebagai berikut:

1. Kepala Dinas

Kepala Satuan Kerja Pemerintah Daerah (SKPD) selaku pejabat pengguna anggaran atau pengguna barang mempunyai tugas pokok meminjam, mengkoordinasi dan mengendalikan pelaksanaan tugas pokok Dinas. Tugas pokok Kepala Dinas Pertanian Tanaman Pangan mempunyai fungsi:

a. Pengaturan dan koordinasi pelaksanaan kebijaksanaan teknik operasional di bidang pertanian tanaman pangan.

b. Fasilitas dan pengendalian pelaksanaan tugas-tugas di bidang pertanian tanaman pangan yang meliputi program, pengelolaan sumber daya pertania, pengembangan produksi padi, palawija, holtikultura, aneka tanaman serta pengelolaan hasil panen dan pemasaran.

c. Penyelenggaraan koordinasi dan kerjasama dalam rangka perencanaan tugas pokok dan fungsiDinas.

2. Sekretaris

Sekretaris mempunyai tugas dan tanggung jawab sebagai berikut:

1. Menyusun Rincian Objek Penerimaan (ROP), pedoman atau petunjuk pelaksanaan kegiatan.
2. Menyiapkan naskah Surat Keputusan-Surat Keputusan (SK-SK) pemimpin kegiatan suratdan ketatausahaan kegiatan.
3. Menyusun laporan proyek secara berkala.
4. Menyusun evaluasi kegiatan.
5. Meyimpan dokumen dan mengolah data proyek.
6. Dalam pelaksanaan tugasnya dibantu oleh beberapa orang menurut kebutuhan.

7. Membantu pemimpin dalam pengendalian dan monitoring pelaksanaan kegiatan.
8. Bertanggung jawab terhadap kegiatan.

3. Sub Bagian Kepegawaian dan Umum

Sub Bagian Kepegawaian mempunyai tugas, tanggung jawab serta fungsi sebagai berikut:

a. Sub bagian kepegawaian mempunyai tugas pokok melaksanakan pengelolaan administrasi kepegawaian, kelembagaan, dan ketatalaksanaan serta pendokumentasian peraturan perundang-undangan.
b. Melaksanakan penyiapan rencana kebutuhan pegawai, formasi dan penunjukan jabatan di lingkungan Dinas.
c. Melaksanakan penyiapan dan pengusulan pegawai yang akan pensiun, peninjauan masa kerja serta pemberian penghargaan.
d. Melaksanakan penyiapan bahan kenaikan pangkat, Daftar Penilaian Pelaksanaan Pekerjaan (DP 3), Daftar Urutan Kepangkatan (DUK) sumpah atau gaji pegawi, gaji berkala dan peningkatan kesejahteraan pegawai.
e. Melaksanakan penyiapan bahan mutasi dan pemberhentian.
f. Melaksanakan penyiapan pegawai untuk mengikuti pendidikan pelatihan kepemimpinan teknis dan fungsional.
g. Melaksanakan penyiapan rencana pegawai yangakanmengikuti ujian Dinas dan ijin atau tugas belajar.
h. Melaksanakan penyiapan bahan pembinaan kepegawaian dan disiplin pegawai di lingkungan Dinas.
i. Melaksanakan penyajian bahan standar kompetensi pegawai, tenaga teknis dan fungsional.
j. Melaksanakan penyiapan bahan pembinaan kelembagaan dan ketatalaksanaan di lingkungan Dinas.

k. Melaksanakan penyiapan bahan rancangan kelembagaan dan ketatalaksanaan dilingkungan Dinas.

l. Melaksanakan pelayanan sistem informasi dan manajemen kepegawaian.

m. Melaksanakan penyusunan bahan evaluasi dan laporan kegiatan Sub bagian kepegawaian.

4. Sub Bagian Keuangan

Sub Bagian Keuangan mempunyai tugas dan tanggungjawab serta mempunyai fungsi sebagai berikut:

1).Sub Bagian Keuangan mempunyai tugas pokok melaksanakan pengelolaan administrasi keuangan.

2). Dalam menyelenggarakan tugas pokok sebagaimana dimaksud yang mempunyai tugas:

a. Penyiapan bahan dan penyusunan rencana anggaran pendapatan dan rutin daerah.

b. Pelaksanaan teknis administrasi keuangan.

5. Sub Bagian Umum

Sub bagian umum mempunyai wewenang dan tanggung jawab serta fungsi sebagai berikut:

1. Sub bagian umum mempunyai tugas pokok melaksanakan pengelolaan rumah tangga, perlengkapan dan umum serta pepustakaan dan kearsipan.

2. Dalam menyelenggarakan tugas pokok sebagaimana di maksud mempunyai fungsi:

a. Pelaksanaan urusan surat-menyurat, pengadaan, kearsipan, dan administrasi perjalanan Dinas.

b. Pelaksanaa urusan rumah dan perlengkapan Dinas.

c. Pengelolaan perpustakaan Dinashubungan masyarakat.

6. Sub Bagian Sumber Daya

Subdinas Sumber Daya mempunyai tugas, tanggung jawab serta fungsi:

1. Subdinas Sumber Daya mempunyai tugas pokok menyelenggarakan perumusan bahan kebijakan teknis di bidang pengembangan sumber daya pertanian tanaman pangan.
2. Untuk melaksanakan tugas pokok sebagaimana mestinya, subdinas sumber daya mempunyai fungsi:
   a. Perumusan bahan fasilitas pedoman standarisasi supervisi dan fasilitasi penyelenggaraan pengelolaan sumber daya lahan dan tata guna air.
   b. Perumusan bahan pengembangan sarana produksi dan mekanisme.
   c. Perumusan sumber daya manusia pertanian tanaman pangan.

7. Sub Produksi Tanaman Pangan

SubDinasproduksi tanaman pangan mempunyai tugas, tanggung jawab serta fungsi:

1. Subdinas produksi tanaman pangan mempunyai tugas pokok merumuskan bahan kebijakan operasional di bidang pengembangan teknologi dan produksi tanaman pangan serta perlindungannya.
2. Untuk melaksanakan tugas pokok subdinas produksi tanaman pangan mempunyai fungsi:
   a. Perumusan bahan kebijakan operasional fasilitasi, pedoman, standarisasi, supervisi pengembangan tanaman pangan.
   b. Perumusan bahan pengembangan teknologi produksi tanaman pangan.
   c. Perumusan bahan pengembangan perlindungan tanaman pangan.

8. Sub Produksi HoltiKultura

Subdinas produksi Holtikultura (Ilmu tentang perkebunan mencakup:padi, jagung, kacang tanah, ubi jalar, kentang, kubis, tomat, cabe merah, bawang merah,

jeruk, strawberry) yang mempunyai tugas, tanggung jawab serta fungsi sebagai berikut:

1. Mempunyai tugas pokok merumuskan bahan kebijakan operasional di bidang pengembangan produksi sayuran, buah-buahan, dan aneka tanaman serta perlindungan tanaman Holtikultura.
2. Untuk melaksanakan tugas pokok sebagaimana dimaksud, subdinas ini mempunyai fungsi:

   a. Perumusan bahan kebijakan operasional penyusunan pedoman pengembangan tanaman holtikultura.

   b. Perumusan bahan standarisai dan fasilitasi pengembangan produksi tanaman holtikultura.

   c. Perumusan bahan perlindungan tanaman holtikultura.

9. Sub Bagian Bidang Usaha

Seksibina usaha mempunyai tugas pokok melaksanakan penyusunan bahan kebijakan operasional kelembagaan usaha pertanian tanaman pangan.Tugas pokok seksi bina usaha pertanian tanaman pangan mempunyai fungsi:

a. Penyusunan bahan pedoman, standarisasi, superfisi, dan fasilitasi pengelolaan bina usaha pertanian tanaman pangan.
b. Penyusunan bahan pengembangan bina usaha pertanian tanaman pangan.

Sub Bagian Bina Usaha membawahi beberapa seksi, yaitu;

1. Seksi pengelolaan pasca panen.
2. Seksi pemasaran.

2.3 Landasan Teori

Landasan teori merupakan penjelasan berbagai konsep dasar dan teori-teori yang berkaitan dalam pembangunan aplikasi M-Petani berbasis android.

2.3.1 Web Sevice

Webservice adalah sekumpulan aplikasi logic beserta objek-objek dan metode-metode yang dimilikinya yang terletak di suatu server yang terhubung ke internet sehingga dapat diakses menggunakan protocol HTTP dan SOAP (*Simple Object Access Protocol*). Dalam penggunaannya, web service dapat digunakan dari hanya untuk memeriksa data user yang login ke sebuah website ataupun untuk digunakan pada transaksi perbankan online yang rumit. Tujuan dari teknologi ini adalah untuk memudahkan beberapa aplikasi atau komponennya untuk saling berhubungan dengan aplikasi lain dalam sebuah organisasi maupun diluar organisasi menggunakan standar yang tidak terikat platform(*platform-neutral*) dan tidak terikat akan bahasa pemrograman yang digunakan (language-neutral).Hal tersebut dapat terjadi karena penggunaan XML standar yang didukung oleh banyak perusahaan besar di dunia, yang digunakan untuk bertukar data. Selain daripada itu, penggunaan SOAP menjadikan metode-metode dari objek-objek yang ada dalam sebuah web service dapat di akses dari aplikasi lain seperti halnya aplikasi tersebut mengakses metode lokal.

2.3.2 JSON

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

JSON terbuat dari dua struktur:

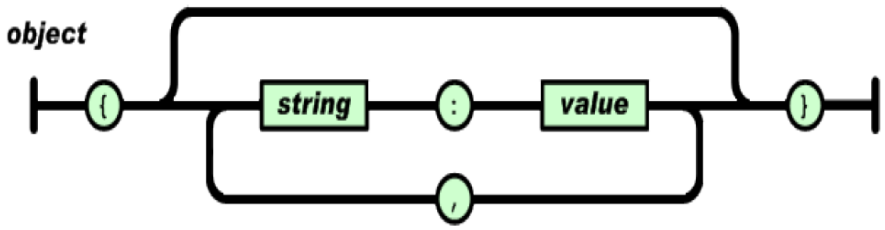
1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel *hash* (*hash table*), daftar berkunci (*keyed list*), atau *associative* array.

2. Daftar nilai terurutkan (*an ordered list of values*).Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (array), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

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

1. Objek

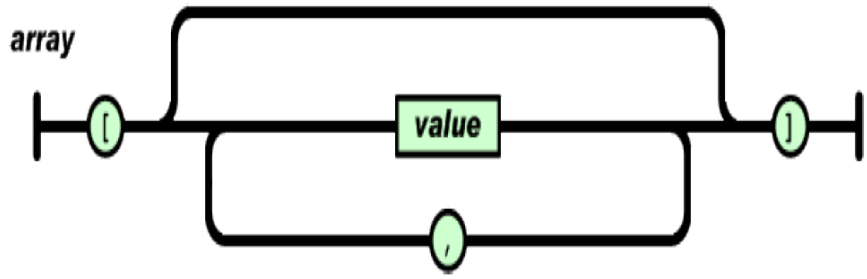
Objek adalah sepasang nama/nilai yang tidak terurutkan.Objek dimulai dengan {(kurung kurawal buka) dan diakhiri dengan} (kurung kurawal tutup).Setiap nama diikuti dengan: (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma). [3]



Gambar 2-2 Objek JSON

2. Larik

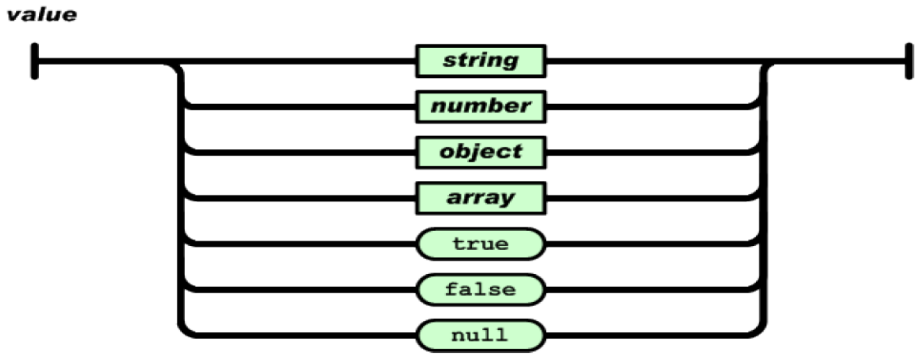
Larik adalah kumpulan nilai yang terurutkan.Larik dimulai dengan [(kurung kotak buka) dan diakhiridengan] (kurung kotak tutup).Setiap nilai dipisahkan oleh, (koma). [3]



Gambar 2-3 Larik JSON

3. Nilai

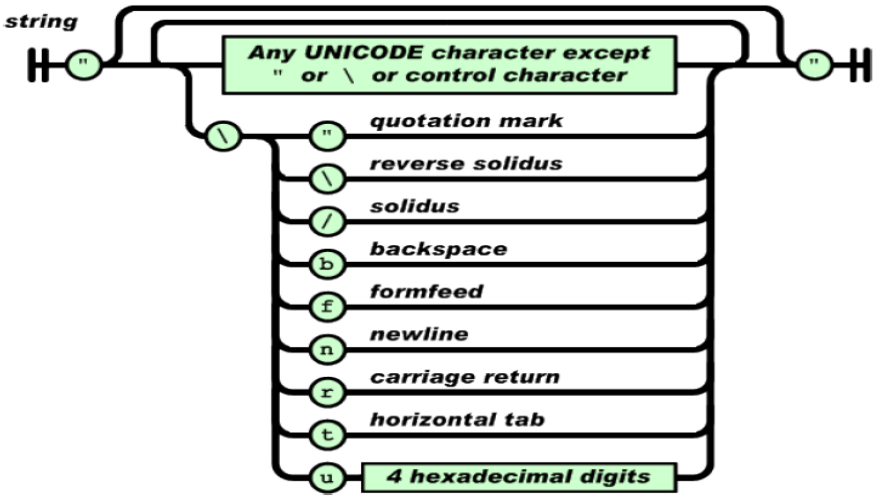
Nilai (*value*) dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau *true* atau *false* atau *null*, atau sebuah objek atau sebuah larik Struktur-struktur tersebut dapat disusun bertingkat.[3]



Gambar 2-4 Nilai JSON

4. String

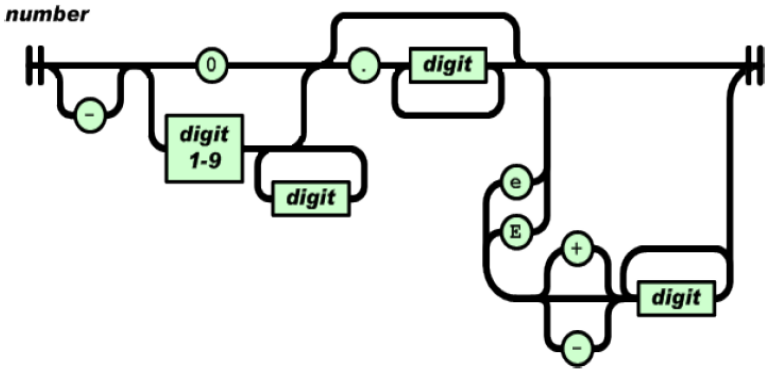
String adalah kumpulan dari nol atau lebih karakter *Unicode*, yang dibungkus dengan tanda kutip ganda.Di dalam string dapat digunakan backslash escapes "\" untuk membentuk karakter khusus.Sebuah karakter mewakili karakter tunggal pada string.



Gambar 2-5 String JSON

5. Angka

Angka adalah sangat mirip dengan angka di C atau Java ,kecualiformat oktal dan heksadesimal tidak digunakan. [3]

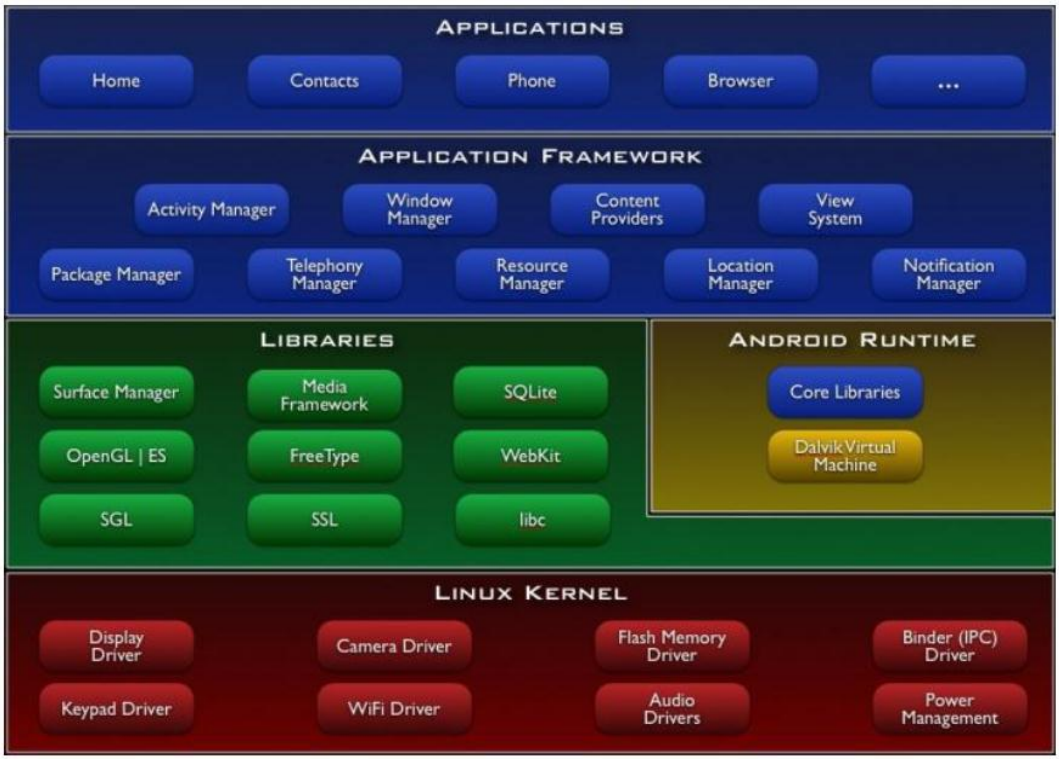


Gambar 2-6 Angka JSON

2.3.3 Android

Android adalah sistem operasi untuk telepon selular (*mobile)* seperti pada *smartphone* dan komputer tablet.[6] Sistem operasi Android berbasiskan Linux yang bersifat terbuka (*opensource)* bagi pengembang untuk menciptakan aplikasi maupun mengembangkan Android dengan *tools* yang disediakan yaitu Android SDK (*Software Development Kit)* dan API (*Application Programming Interface)* dengan menggunakan bahasa pemrograman Java. Dikembangkan Oleh Google Inc, HTC, Intel, Motorola, Qualcom, T-Mobile,dan Nvidia yang tergabung dalam *Open Handset Alliance* (OHA).Dengan tujuan mendukung sebuah standar terbuka pada perangkat seluler.Kode-kode yang dirilis oleh Google untuk Android berada dibawah *license Apache*, sebuah *license* perangkat lunak dan standar terbuka bagi perangkat seluler. [6]

Berikut adalah arsitektur android pada gambar 2-7: [7]



Gambar 2-7 Arsitektur Android

Deskripsi dari Arsitektur android:

1. Aplikasi

Android memilki satu set aplikasi inti yang terdiri dari email *client*, program SMS, kalender, peta, *browser*, kontak telepon, dan sebagainya.Semua aplikasi inti dibangun dengan menggunakan bahasa pemrograman Java.

2. *Framework* Aplikasi

Pengembang memiliki akses penuh menuju API framework yang sama dengan yang digunakan oleh aplikasi inti.Arsitektur aplikasi dirancang agar komponen dapat digunakan kembali (*reuse*) dengan mudah.Setiap aplikasi dapat memanfaatkan kemampuan ini (sesuai dengan batasan keamanan yang didefinisikan oleh *framework*).Mekanisme yang sama memungkinkan komponen untuk diganti oleh pengguna.

Semua aplikasi merupakan rangkaian set layanan dan sistem, termasuk:

a. *Views* yang kaya dan extensible yang dapat digunakan untuk membangun aplikasi, termasuk *list*, *grids*, kotak teks, tombol, dan bahkan sebuah embedded web.
b. *Content Provider* yang memungkinkan aplikasi untuk mengakses data (seperti dari daftar kontak telepon) atau membagi data tersebut.
c. *Resource Manager*, yang menyediakan akses ke kode nonsumber lokal seperti string, gambar, dan tata letak file.
d. *Notifications* Manager yang memungkinkan semua aplikasi menampilkan alert yang bisa dikostumisasi di dalam status bar.
e. *Activity Manager* yang mengelola siklus hidup aplikasi dan menyediakan navigasi umum *backstack*.

3. Libraries

Android menyertakan libraries C/C++ yang digunakan oleh berbagai komponen dari sistem Android.Kemampuan ini disediakan kepada *Developer*

aplikasi melalui *Framework* Aplikasi Android.Beberapa inti libraries tercantum dibawah ini:

- *System* C *Library* – Variasi dari implementasi BSD-berasal pelaksana sistem standar C library (*libc*), sesuai untuk perangkat *embedded* berbasis Linux.
- Media *Libraries* – *Packet Video* berdasarkan OpenCORE; *library* medukung pemutaran rekaman dan populer banyak format audio dan video, serta file gambar, termasuk MPEG4, H.264, MP3, AAC, AMR, JPG, dan PNG.
- *Surface* Manager – mengelola akses ke subsistem layar, lapisan komposit 2D dan grafis 3D dari beberapa aplikasi.
- LibWebCore – mesin web modern yang *powerfull* yang baik browser Android *embedded* web.
- 3D *Libraries* – sebuah pelaksana berdasarkan openGL ES 1,0 API; perpustakaan baik menggunakan perangkat keras akselerasi 3D (apabila tersedia) atau yang disertakan, sangat optimal 3D *software rasterizer*.
- FreeType – bitmap dan vektor *font rendering*.
- SQLite – mesin database yang kuat dan ringan, dan penghubung tersedia untuk semua aplikasi.

4. Android Runtime

Android terdiri dari satu set core *libraries* yang menyediakan sebagian besar fungsi yang sama dengan yang terdapat pada *core libraries* bahasa pemrograman Java.Setiap aplikasi menjalankan prosesnya sendiri dalam Android, dengan masing-masing instan dari mesin virtual Dalvik (Dalvik VM).Dalvik dirancang agar perangkat dapat menjalankan *multiple* VMs secara efisien. Mesin Virtual Dalvik mengeksekusi file dalam Dalvik executable (.dex), sebuah format yang dioptimalkan untuk memori yang kecil. Dalvik VM berbasis, berjalan, dan dikompilasi oleh compiler bahasa Java yang telah ditransformasikan ke dalam .dex format oleh tool “dx” yang telah disertakan.Dalvik VM bergantung pada

kernel Linux untuk berfungsi seperti *threading* dan manajemen memori tingkat rendahnya.

5. Kernel Linux

Pada layanan sistem inti Android bergantung pada Linux versi 2.6 seperti layanan keamanan, manjemen memori,manajemen proses, *stack* jaringan, dan model *driver*. Kernel bertindak sebagai lapisan abstrak antara perangkat keras dengan seluruh *stack software*.

2.3.4 Pengenalan UML

Pada perkembangan teknologi perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang di berbagai negara dapat mengerti pemodelan perangkat lunak. Seperti yang kita ketahui bahwa menyatukan banyak kepala untuk menceritakan sebuah ide dengan tujuan untuk memahami, merupakan hal yang tidak mudah, oleh karena itu diperlukan sebuah bahasa pemodelan perangkat lunak yang dapat dimengerti oleh banyak orang.[9]

Banyak orang yang telah membuat bahasa pemodelan pembangunan perangkat lunak sesuai dengan teknologi pemrograman yang berkembang pada saat itu, misalnya yang sempat berkembang dan digunakan banyak pihak adalah *Data Flow Diagram* (DFD) untuk memodelkan perangkat lunak yang menggunakan pemrograman prosedural atau struktural, kemudian juga ada *State Transition Diagram* (STD) yang digunakan untuk memodelkan sistem *real time* (waktu nyata).

Pada perkembangan teknik pemrograman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modeling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk

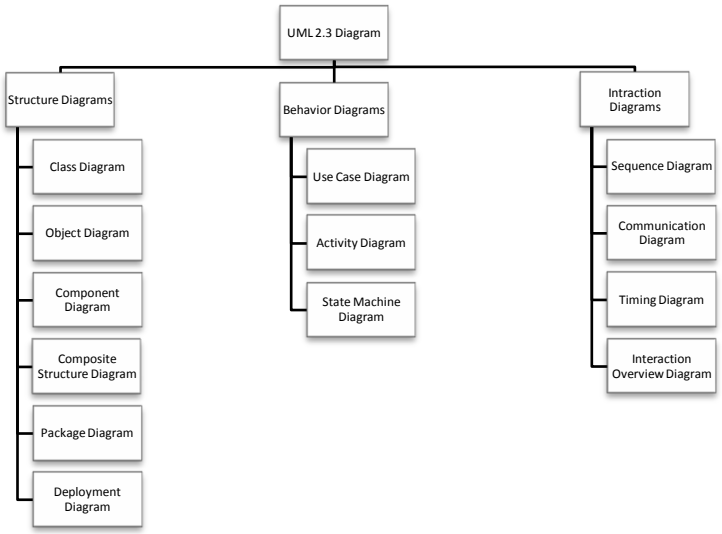
pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek.

Seperti yang kita ketahui di dunia sistem informasi yang tidak dapat dibakukan, semua tergantung kebutuhan, lingkungan dan konteksnya. Begitu juga dengan perkembangan penggunaan UML bergantung pada level abstraksi penggunaannya. Jadi, belum tentu pandangan yang berbeda dalam penggunaan UML adalah suatu yang salah, tapi perlu ditelaah dimanakah UML digunakan dan hal apa yang ingin digambarkan. Secara analogi jika dengan bahasa yang digunakan sehari-hari, belum tentu penyampaian bahasa dengan puisi adalahhal yang salah. Sistem informasi bukanlah ilmu pasti, maka jika ada banyak perbedaan dan interpretasi di dalam bidang sistem informasi merupakan hal yang sangat wajar.

2.2.4.1. Diagram UML

Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar dibawah.[2][5]



Gambar 2-8 Diagram UML

Berikut ini penjelasan singkat dari pembagian kategori tersebut:

1. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
3. *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antarsubsistem pada suatu sistem.

1. Use Case Diagram

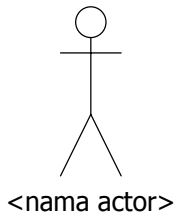
Use Case atau diagram *Use Case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. [5]

Syarat penamaan pada *Use Case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *Use Case* yaitu pendefinisian apa yang disebut aktor dan *Use Case*.

2. Aktor

Aktor adalah sesuatu (entitas) yang berhubungan dengan sistem dan berpartisipasi dalam *use case*. Aktor menggambarkan orang, sistem atau *entitas eksternal* yang secara khusus membangkitkan sistem dengan *input* atau masukan kejadian-kejadian, atau menerima sesuatu dari sistem. Aktor dilukiskan dengan peran yang mereka mainkan dalam *use case*, seperti Staff, Kurir dan lain-lain.[5]

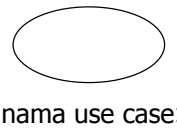
Dalam *use case diagram* terdapat satu aktor pemulai atau *initiator actor* yang membangkitkan rangsangan awal terhadap sistem, dan mungkin sejumlah aktor lain yang berpartisipasi atau *participating actor*. Akan sangat berguna untuk mengetahui siapa aktor pemulai tersebut.



Gambar 2-9 Aktor

3. Use Case

Use case yang dibuat berdasarkan keperluan aktor merupakan gambaran dari "apa" yang dikerjakan oleh sistem, bukan "bagaimana" sistem mengerjakannya. *Use case* diberi nama yang menyatakan apa hal yang dicapai dari interaksinya dengan aktor.[5]

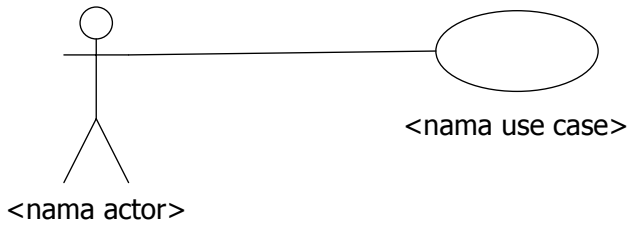


Gambar 2-10 Use Case

4. Relationship

Relasi (*relationship*) digambarkan sebagai bentuk garis antara dua simbol dalam *use case diagram*. Relasi antara *actor* dan *use case* disebut juga dengan asosiasi (*association*). Asosiasi ini digunakan untuk menggambarkan bagaimana hubungan antara keduanya.[5]

Relasi-relasi yang terjadi pada *use case diagram* bisa antara *actor* dengan *use case* atau *use case* dengan *use case*.



Gambar 2-11 Relationship

Relasi antara *use case* dengan *use case* :

a. ***Include*****,** pemanggilan *use case* oleh *use case* lain atau untuk menggambarkan suatu *use case* termasuk di dalam *use case* lain (diharuskan). Contohnya adalah pemanggilan sebuah fungsi program. Digambarkan dengan garis lurus berpanah dengan tulisan <<include>>.
b. ***Extend*****,** digunakan ketika hendak menggambarkan variasi pada kondisi perilaku normal dan menggunakan lebih banyak kontrol *form* dan mendeklarasikan ekstension pada *use case* utama. Atau dengan kata lain adalah perluasan dari *use case* lain jika syarat atau kondisi terpenuhi. Digambarkan dengan garis berpanah dengan tulisan <<extend>>.
c. ***Generalization/Inheritance*****,** dibuat ketika ada sebuah kejadian yang lain sendiri atau perlakuan khusus dan merupakan pola berhubungan *base-parent use case*. Digambarkan dengan garis berpanah tertutup dari *base use case* ke *parent use case*.

5. Activity Diagram

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.[5]

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut ini:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan
2. Urutan atau pengelompokan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya

6. Sequence Diagram

Diagram *sequence* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram *sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.[5]

Banyaknya diagram *sequence* yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram *sequence* sehingga semakin banyak *use case* yang didefinisikan maka diagram *sequence* yang harus dibuat juga semakin banyak.

Penomoran pesan berdasarkan urutan interaksi pesan. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu.

7. Class Diagram

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.[5]

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem. Susunan struktur kelas yang baik pada diagram kelas ebaiknya memiliki jenis-jenis kelas berikut:

1) Kelas Main
   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.

2) Kelas yang menangani tampilan sistem
   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.

3) Kelas yang diambil dari pendefinisian *usecase*
   Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*.

4) Kelas yang diambil dari pendefinisian data
   Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Jenis-jenis kelas diatas juga dapat digabungkan satusama lain sesuai dengan pertimbangan yang dianggap baik asalkan fungsi-fungsi yang sebaiknya ada pada struktur kelas tetap ada. Susunan kelas juga dapat ditambahkan kelas utilitas seperti koneksi ke basis data, membaca *file* teks, dan lain sebagainya sesuai kebutuhan.

Dalam mendefinisikan metode yang ada di dalam kelas perlu memperhatikan apa yang disebut dengan *cohesion* dan *coupling*. *Cohesion* adalah ukuran seberapa dekat keterkaitan instruksi di dalam sebuah metode terkait satu sama lain

sedangkan *coupling* adalah ukuran seberapa dekat keterkaitan instruksi antara metode yang satu dengan metode yang lain dalam sebuah kelas. Sebagai aturan secara umum maka sebuah metode yang dibuat harus memiliki kadar*choesion* yang kuat dan kadar *coupling* yang lemah.

Dalam class diagram terdapat beberapa relasi (hubungan antar class) yaitu:

1. *Generalization dan Inheritence*

   Diperlukan untuk memperlihatkan hubungan pewarisan (inheritance) antar unsur dalam diagram kelas. Pewarisan memungkinkan suatu kelas mewarisi semua atribut, operasi, relasi, dari kelas yang berada dalam hirarki pewarisannya.

2. *Associations*

   Hubungan statis antar class. Umumnya menggambarkan class yang memiliki atribut berupa class lain, atau class yang harus mengetahui ekstensi class lain. Dalam notasi UML kita mengenal asosiasi 2 arah (bidirectional) dan 1 arah (undirectional).

3. *Aggregation*

   Hubungan antar-class dimana class yang satu (part class) adalah bagian dari class lainnya (whole class).

4. *Composition*

   Aggregation dengan ikatan yang lebih kuat. Di dalam composite aggregation, siklus hidup part class sangat bergantung pada whole classsehingga bila objek instance dari whole class dihapus maka objek instance dari part calss juga akan terhapus.

5. *Depedency*

   Hubungan antar-class dimana sebuah class memiliki ketergantungan pada class lainnya tetapi tidak sebaliknya.

6. *Realization*

   Hubungan antar-class dimana sebuah class memiliki keharusan untuk mengikuti aturan yang ditetapkan class lainnya. Biasanya realization digunakan untk menspesifikasikan hubungan antara sebuah interface dengan class yang mengimplementasikan interface tersebut.

2.3.5 Pemograman Berorientasi Objek

Metodologi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya.[5] Metodologi berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis. Metode berorientas objek didasarkan pada penerapan prinsip-prinsip pengelolaan kompleksitas. Metode berorientasi objek meliputi rangkaian aktivitas analisis berorientasi objek, perancangan berorientasi objek, pemrograman berorientasi objek, dan pengujian berorientasi objek.

Pada saat ini, metode berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekatan terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan pada saat ini beragam dengan *platform* yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi tersebut.

Keuntungan menggunakan metodologi berorientasi objek adalah sebagai berikut:

1. Meningkatkan produktivitas

   Karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut (*reusable*).

2. Kecepatan pengembangan

   Karena sistem yang dibangun dengan baik dan benar pada saat analisis dan perancangan akan menyebabkan berkurangnya kesalahan pada saat pengkodean.

3. Kemudahan pemeliharaan

   Karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.

4. Adanya konsistensi

   Karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.

5. Meningkatkan kualitas perangkat lunak

   Karena pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan mampu memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

2.2.5.1 Konsep Dasar Berorientasi Objek

Pendekatan berorientasi objek merupakan suatu teknik atau cara pendekatan dalam melihat permasalahan dan sistem (sistem perangkat lunak, sistem informasi, atau sistem lainnya.[5] Pendekatan berorientasi objek akan memandang sistem yang akan dikembangkan sebagai suatu kumpulan objek yang berkorespondensi dengan objek-objek dunia nyata.

Ada banyak cara untuk mengabstraksikan dan memodelkan objek-objek tersebut, mulai dan abstraksi objek, kelas, hubungan antar kelas sampai abstraksi sistem. Saat mengabstraksikan dan memodelkan objek, data dan proses-proses yang dipunyai oleh objek akan dienkapsulasi (dibungkus) menjadi suatu kesatuan.

Dalam rekayasa perangkat lunak, konsep pendekatan berorientasi objek dapat diterapkan pada tahap analisis, perancangan, pemrograman, dan pengujian

perangkat lunak. Ada berbagai teknik yang dapat digunakan pada masing-masing tahap tersebut, dengan aturan dan alat bantu pemodelan tertentu.

Sistem berorientasi objek merupakan sebuah sistem yang dibangun dengan berdasarkan metode berorientasi objek adalah sebuah sistem yang komponennya dibungkus (dienkapsulasi) menjadi kelompok data dan fungsi. Setiap komponen dalam sistem tersebut dapat mewarisi atribut dan sifat dan komponen lainnya, dan dapat berinteraksi satu sama lain.

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek:

a. Kelas (*Class*)

   Kelas adalah sekumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statis dan himpunan objek yang sama yang mungkin lahir atau diciptakan dan kelas tersebut. Sebuah kelas akan mempunyai sifat (a*tribut*), kelakuan (metode/operasi), hubungan (*relationship*) dan arti. Suatu kelas dapat diturunkan dan kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru. Secara teknis kelas adalah sebuah struktur dalam pembuatan perangkat lunak. Kelas merupakan bentuk struktur pada kode program yang menggunakan metodologi berorientasi objek.

b. Objek (*object*)

   Objek adalah abstraksi dari sesuatu yang mewakili dunia nyata benda, manusia, satu organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu *entitas* yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.

   Secara teknis, sebuah kelas saat program dieksekusi maka akan dibuat sebuah objek. Objek dilihat darisegi teknis adalah elemen pada saat *runtime* yang akan diciptakan, dimanipulasi, dan dihancurkan saat eksekusi sehinga sebuah objek hanya ada saat sebuah program dieksekusi. Jika masih dalam bentuk

kode, disebut sebagai kelas jadi pada saat *runtime* (saat sebuah program dieksekusi), yang kita punya adalah objek, di dalam teks program yang kita lihat hanyalah kelas.

c. Metode (*method*)

Operasi atau metode pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek.

Metode atau operasi dapat berasal dari *event*, aktifitas atau aksi keadaan, fungsi, atau kelakuan dunia nyata. Contoh metode atau operasi misalnya *Read, Write, Move, Copy,* dan sebagainya.

d. Atribut (*attribute*)

Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama, dan sebagainya.

e. Abstraksi (*abstraction*)

Prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.

f. Enkapsulasi (*encapsulation*)

Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerja-nya.

g. Pewarisan (*inheritance*)

Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dan dirinya.

h. Antarmuka (*interface*)

Antarmuka sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah *interface* dapat diimplementasikan oleh kelas lain.

i. *Reusability*

Pemanfaatan kembali objek yang sudah didefinisikan untuk suatu permasalahan pada permasalahan lainnya yang melibatkan objek tersebut.

j. Generalisasi dan Spesialisasi

Menunjukkan hubungan antara kelas dan objek yang umum dengan kelas dan objek yang khusus. Misalnya kelas yang lebih umum (generalisasi) adalah kendaraan darat dan kelas khususnya (spesialisasi) adalah mobil, motor, dan kereta.

k. Komunikasi Antarobjek

Komunikasi antarobjek dilakukan lewat pesan (*message*) yang dikirim dari satu objek ke objek lainnya.

l. Polimorfisme (*polymorphism*)

Kemampuan suatu objek digunakan di banyak tujuan yang berbeda dengan nama yang sehingga menghemat baris program.

m. *Package*

Package adalah sebuah kontainer atau kemasan yang dapat digunakan untuk mengelompokkan kelas-kelas sehingga memungkinkan beberapa kelas yang bernama sama disimpan dalam *package* yang berbeda.

2.2.6 Database

Database dapat diartikan sebagai kumpulan data yang terdiri atas satu atau lebih tabel yang terintegrasi satu sama lain, di mana setiap pemakai (user) diberi wewenang (otorisasi) untuk dapat mengakses (mengubah, menghapus, menganalisis, menambah, memperbaiki) data dalam tabel-tabel tersebut.

1. Fungsi Database

Adapun fungsi database diantaranya adalah sebagai berikut:

1. salah satu komponen penting dalam sistem informasi, karena merupakan dasar dalam menentukan informasi.
2. Menentukan kualitas informasi. Informasi dapat dikatakan bernilai apabila manfaatnya lebih efektif dibandingkan dengan biaya mendapatkannya.
3. Mengurangi duplikasi data *(data redudancy).*
4. Hubungan data dapat ditingkatkan *(data relatability).*
5. Mengurangi pemborosan tempat simpanan luar.

2. Kriteria Database

Dari penjelasan sebelumnya dapat disimpulkan bahwa basis data mempunyai beberapa kriteria penting, yaitu:

1. bersifat data *oriented* dan bukan *program oriented*
2. dapat digunakan oleh beberapa program aplikasi tanpa perlu mengubah basis datanya.
3. Dapat dikembangkan dengan mudah, baik volume maupun strukturnya.
4. Dapat memenuhi kebutuhan sistem-sistem baru secara mudah, dapat digunakan dengan cara-cara yang berbeda.

2.2.7 Skala Likert

Skala likert merupakan skala yang menyatakan tingkat persetujuan individu terhadap suatu pernyataan. Skala ini sering digunakan dalam berbagai penelitian yang menggunakan pendekatan survei, dimana kuesioner dijadikan sebagai alat untuk memperoleh data.Ada beberapa tingkat Skala Likert yang sering digunakan dalam penelitian, diantaranya adalah 5-level, 7-level dan 9-level[8][9].

Tabel 2-1 Skala Likert

No.	Keterangan	Skor Positif	Skor Negatif
1	Sangat Setuju	5	1
2	Setuju	4	2
3	Ragu-ragu	3	3
4	Tidak Setuju	2	4
5	Sangat Tidak Setuju	1	5

Apabila data yang diperoleh melalui jawaban kuesioner telah terkumpul, maka angka dalam bentuk skala liker yang dipilih responden (pada setiap pertanyaan kuesioner) selanjutnya dikalkulasi. Perhitungan ini menggunakan teknik statistik sehingga dapat dianalisis lebih lanjut.