BAB II

LANDASAN TEORI

2.1. Konsep Dasar Sistem

Menurut Nono (2013), "Mengungkapkan bahwa Sistem administrasi adalah kegiatan ketatausahaan yang terdiri dari berbagai kegiatan seperti pembukuan baik penghitungan, pencatatan atau yang lainnya dengan tujuan untuk menyediakan informasi yang dibutuhkan. Sedangkan dalam arti yang sempit, menurutnya administrasi merupakan kegiatan catat mencatat atau pembukuan, surat menyurat atau lainnya yang berkaitan dengan ketatausahaan".

Sistem Administrasi adalah "suatu kegiatan yang melibatkan aturan mencakup pekerjaan yang sistematis dan terarah". Keunggulan yang didapat dari kegiatan administrasi antara lain:

1. Aktivitas sistematis dan jelas
2. Kerapian yang memungkinkan informasi dapat dikelola dan didapatkan kembali sewaktu dibutuhkan.
3. Meminimalisasi kesalahan karena rutin yang sudah terstruktur.
4. Menyederhanakan kerumitan dan problematika yang mungkin muncul.

2.1.1. Pengertian Sistem

Sutabri (2016:2) mengemukakan bahwa, "Secara sederhana suatu sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur, komponen, atau

variabel yang terorganisir, saling berinteraksi saling tergantung satu sama lain dan terpadu".

Suatu sistem terdiri objek-objek, unsur-unsur atau komponen-komponen yang saling berkaitan antara satu dengan yang lainnya sehingga unsur-unsur tersebut menjadi satu kesatuan dalam pengolahan tertentu untuk mencapai tujuan bersama.

Adapun yang dimaksud dengan unsur dan komponen pembentuk organisasi disini bukan hanya bagian-bagian yang tampak secara fisik, tetapi juga hal-hal yang mungkin bersifat abstrak atau konseptual seperti misi, pekerjaan, kegiatan, kelompok informal dan lain sebagainya.

Unsur-unsur yang mewakili suatu sistem secara umum adalah masukan *(input)*, pengolahan *(processing)*, dan keluaran *(output)*. Disamping itu, suatu sistem senantiasa tidak lepas dari lingkungan sekitarnya, maka umpan balik *(feedback)* dapat berasal dari *output*, juga berasal dari lingkungan sistem yang dimaksud.

Dalam definisi sistem terdapat dua kelompok pendekatan, yakni kelompok pendekatan yang menekankan pada prosedur, metode, dan cara kerja yang saling berhubungan dan saling berinteraksi untuk melakukan suatu kegiatan dalam rangka mencapai tujuan bersama, dan kelompok pendekatan yang menekankan pada elemen atau komponen, dimana dalam pendekatan ini kumpulan elemen atau komponen saling berinteraksi dan berhubungan dalam rangka untuk mencapai tujuan bersama pula. Kedua definisi tersebut sama-sama bener dan tidak bertentangan hanya saja memiliki perbedaan pada cara pendekatannya.

2.1.2. Karakteristik Sistem

Sutabri (2016:10) mengemukakan bahwa, "Model umum sebuah sistem adalah *input*, proses, dan *output*, hal ini merupakan konsep sebuah sistem yang sangat sederhana sebab sebuah sistem dapat memiliki beberapa keluaran."

Adapun karakteristik-karakteristik sistem tersebut menurut Sutabri (2016:10), adalah:

1. Komponen Sistem *(Component)*

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, artinya saling bekerjasama membentuk satu kesatuan. Komponen-komponen sistem tersebut dapat berupa suatu bentuk sub sistem dimana setiap sub sistem memiliki sisfat dari sistem yang menjalankan suatu fungsi tertentu dan memengaruhi prosen sistem secara keseluruhan sehingga terjadilah suatu proses sistem tersebut untuk mencapai tujuan bersama.

2. Batasan Sistem *(Boundary)*

   Ruang lingkup sistem merupakan daerah yang membatasi antara sistem dengan sistem yang lain atau sistem dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan yang tidak dapat dipisahkan.

3. Lingkungan Luar Sistem *(Environtment)*

   Bentuk apapun yang ada diluar ruang lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut disebut lingkungan luar sistem. Lingkungan luar sistem ini dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut. Lingkungan luar yang menguntungkan merupakan energi bagi sistem tersebut. Dengan demikian, lingkungan luar

tersebut harus tetap dijaga dan dipelihara, sedangkan lingkungan luar yang merugikan harus dikendalikan. Kalau tidak, maka akan mengganggu kelangsungan hidup sistem tersebut.

4. Penghubung Sistem *(Interface)*

   Media yang menghubungkan sistem dengan sub sistem lain disebut penghubung sistem atau interface. Penghubung ini memungkinkan sumber-sumber daya mengalir dari satu sub sistem ke sub sistem lain. Bentuk keluaran dari satu sub sistem akan menjadi masukkan untuk sub sistem lain melalui penghubung tersebut. Dengan demikian, dapat terjadi suatu integrasi sistem yang membentuk satu kesatuan.

5. Masukan Sistem (Input)

   Energi yang dimasukkan kedalam sistem tersebut masukkan sistem, yang dapat berupa pemeliharaan *(maintenance input)* agar sistem tersebut beroprasi. Contoh, didalam suatu unit sistem komputer. "Program" adalah *maintenance* input yang digunakan untuk mengoprasikan komputernya dan "data" adalah sinyal input untuk diolah menjadi informasi. Selain itu sistem masukan dapat juga berupa masukan signal (signal input) yang bertujuan agar energi yang dimasukkan menghasilkan keluaran (output) contohnya informasi.

6. Keluaran Sistem (Ouput)

   Keluaran adalah hasil energi yang diolah dan diklarifikasikan menjadi keluaran yang berguna dan berupa sisa pembuangan sehingga keluaran ini dapat menjadi masukan bagi sub sistem yang lainnya. Contoh, sistem informasi, keluaran yang dihasilkan tentunya adalah informasi yang mana informasi ini dapat digunakan

sebagai masukkan untuk pengambilan keputusan atau hal-hal lain yang menjadi input bagi sub sistem lainnya.

7. Pengolah Sistem

   Suatu sistem harus memiliki suatu perangkat yang bertugas mengolah bagian pengolah ini yang akan berubah maskan menjadi keluaran. Sebagai contoh, sistem produksi akan mengolah masukan yang berupa bahan baku dan barang-barang lainnya menjadi barang jadi, sistem akuntansi akan mengolah data transaksi menjadi laporan-laporan termasuk laporan keuangan yang dibutuhkan oleh pihak manajemen.

8. Sasaran Sistem (Objektive)

   Tujuan dan sasaran merupakan sesuatu yang harus dimiliki oleh sistem. Suatu sistem harus memiliki tujuan dan sasaran yang pasti karena akan menentukan masukan yang dibutuhkan sistem yang tentunya juga berpengaruh pada keluaran sistem. Suatu dapat dikatakan berhasil bila dapat mengenai sasaran dan tujuan sesuai yang telah direncanakan.

2.1.3. Klasifikasi Sistem

Sistem merupakan suatu bentuk integrasi antara satu komponen dengan komponen lainnya, oleh karena itu sistem dapat dikasifikasikan beberapa sudut pandang.

Menurut Ladjamudin (2013:6), “Sistem adalah suatu bentuk integrasi antara satu komponen dengan komponen lain karena memiliki sasaran yang berbeda untuk setiap kasus yang terjadi didalam sistem”. Oleh sebab itu, sistem klasifikasi dari beberapa sudut pandang antaranya:

1. Sistem Abstrak dan Sistem Fisik

   Sistem abstrak adalahsistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik.

2. Sistem Alamiah dan Sistem Buatan Manusia

   Sistem alamiah adalah sistem yang terjadi melalui proses alam, tidak dibuat oleh manusia, misalnya sistem perputaran bumi, terjadinya siang dan malam serta pergantian musim. Sedangkan sistem buatan merupakan sistem yang melibatkan hubungan manusia dengan mesin, yang disebut Human Machine System. Salah satu contohnya adalah sistem informasi berbasis komputer, karena menyangkut penggunaan komputer yang berinteraksi dengan manusia.

3. Sistem Determinasi dan Sistem Probabilistic

   Sistem yang beroperasi dengan tingkah laku yang dapat diprediksi disebut sistem deterministic. Sistem yang kondisi masa depannya tidak dapat diprediksi karena mengandung unsur probabilistic.

4. Sistem Terbuka dan Sistem Tertutup Sistem terbuka adalah sistem yang berhubungan dan dipengaruhi oleh lingkungan luarnya, yang menerima masukan dan menghasilkan keluaran untuk sub sistem lainya. Sedangkan sistem tertutup adalah sistem yang tidak berhubungan dan tidak dipengaruhi oleh lingkungan luarnya, sistem ini bekerja secara otomatis tanpa adanya campur dari pihak luar.

2.1.4. Pengertian Informasi

Menurut Sutabri (2016:4) mengemukakan bahwa, "Informasi adalah data yang telah diklasifikasi atau diolah atau diinterpretasi untuk digunakan dalam proses pengambilan keputusan".

Sumber dari informasi adalah data. Data adalah kumpulan huruf atau angka yang belum diolah sehingga tidak memiliki arti. Secara konseptual, data adalah deskripsi tentang benda, kejadian, aktivitas dan transaksi yang tidak mempunyai makna atau tidak berpengaruh langsung kepada pemakai.

Sistem pengelolah informasi mengolah data menjadi informasi atau tepatnya mengolah data dari bentuk tak berguna menjadi berguna bagi penerimanya. Nilai atau kualitas informasi berhubungan dengan keputusan. Bila tidak ada pilihan atau keputusan, maka informasi menjadi tidak diperlukan.

Untuk mengukur apakah informasi tersebut memiliki kualitas atau tidak, kita dapat mengujinya dengan memperhatikan empat dimensi utama dasar informasi sebagai berikut:

2. Relevasi

   Relevasi atau keterkaitan yang dimiliki oleh suatu informasi dengan sebuah permasalahan menjadi penting karena hal itu bisa menjadi variabel-variabel yang menentukan pengambilan keputusan oleh penggunanya. Oleh karena itu, unformasi yang diberikan harus sesuai dengan yang dibutuhkan.

3. Akurasi

   Akurasi berarti informasi yang diberikan harus mencerminkan keadaan yang sebenarnya agar informasi yang disampaikan tidak biasa dan menyesatkan

peggunanya karena keakurasian sebuah informasi dapat menjadi tolak ukur ketepatan dan keberhasilan dalam pengambilan keputusan.

4. Ketepatan Waktu

   Informasi harus tersedia pada saat diperlukan dan tidak boleh terlambat karena didalam proses pengambilan keputusan, informasi yang sudah usang atau datangnya setelah suatu keputusan diambil tidak akan lagi memiliki nilai. Jadi, semakin baru atau *up to date* informasi tersebut maka akan semakin berguna.

5. Kelengkapan

   Para pengguna harus memperoleh informasi yang menyajikan suatu gambaran lengkap atas suatu masalah tertentu atau solusinya. Informasi yang diberikan harus lengkap secara keseluruhan sehingga dapat mendukung proses pengambilan keputusan disemua area dimana keputusan akan diambil.

2.1.5. Daur Hidup Sistem (*System Development Life Cycle*)

Dalam penulisan Tugas Akhir ini penulis mengembangkan sistem informasi dengan menggunakan metodologi pengembangan sistem informasi. Menurut Kadir (2014:244) mengemukakan bahwa, "*SDLC* merupakan metodologi klasik yang digunakan untuk mengembangkan, memelihara dan menggunakan sistem informasi".

2.1.6. Pemrograman

Menurut Kadir (2014:192) memberi pengertian bahwa, "Program adalah sekumpulan instruksi yang digunakan untuk mengatur perangkat keras komputer agar melaksanakan tindakan tertentu".

2.1.7. Pemrograman Berorientasi Objek

Menurut Kadir (2014:204) mengemukakan bahwa, "Pemrograman berorientasi objek adalah mengombinasikan data dan prosedur-prosedur untuk mengakses data menjadi sebuah kesatuan unit".

2.1.8. Karakteristik Pemrograman Berorientasi Objek

Metodologi pengembangan sistem berorientasi objek mempunyai tiga karakteristik utama:

1. Pengkapsulan (*Encapsulation*)
   a. *Encapsulation* merupakan dasar untuk pembatasan ruang lingkup program terhadap data yang diproses.
   b. Data dan prosedur atau fungsi dikemas bersama-sama dalam suatu objek, sehingga prosedur atau fungsi lain dari luar tidak dapat mengaksesnya.
   c. Data terlindungi dari prosedur atau objek lain, kecuali prodesur yang berada dalam objek itu sendiri.
2. Pewarisan (*Inheritance*)
   a. *Inheritance* adalah teknik yang menyatakan bahwa anak dari objek akan mewarisi data/atribut dan metode dari induknya langsung. Atribut dan metode dari objek induk diturunkan kepada anak objek, demiian seterusnya.
   b. *Inheritance* mempunyai arti bahwa atribut dan operasi yang dimiiliki bersama di antara kelas yang mempunyai hubungan secara hirarki.
   c. Suatu kelas dapat ditentukan secara umum, kemudian ditentukan spesifik menjadi subkelas. Setiap subkelas mempunyai hubungan atau mewarisi semua sifat yang dimilikinya.

d. Kelas objek dapat didefinisikan atribut dan *service* dari kelas objek lainnya.

e. *Inheritance* menggambarkan generalisasi sebuah kelas.

3. *Polimorfisme*

a. *Polimorfisme* yaitu konsep yang menyatakan bahwa sesuatu yang sama dapat mempunyai bentuk dan perilaku berbeda.

b. *Polimorfisme* mempunyai arti bahwa operasi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda.

c. Kemampuan objek-objek yang berbeda untuk melakukan metode yang pantas dalam merespon *message* yang sama.

d. Seleksi dari metode yang sesuai bergantung pada kelas yang seharusnya menciptakan objek.

2.1.9. Bahasa Pemrograman Visual Basic 2010

Menurut Hidayatullah (2015:05), "Visual Basic .NET adalah Visual Basic yang direkayasa kembali untuk digunakan pada *platform* .NET sehingga aplikasi yang dibuat menggunakan Visual Basic .NET dapat berjalan pada sistem komputer apa pun, dan dapat mengambil data dari *server* dengan tipe apa pun asalkan terinstal .NET Framework".

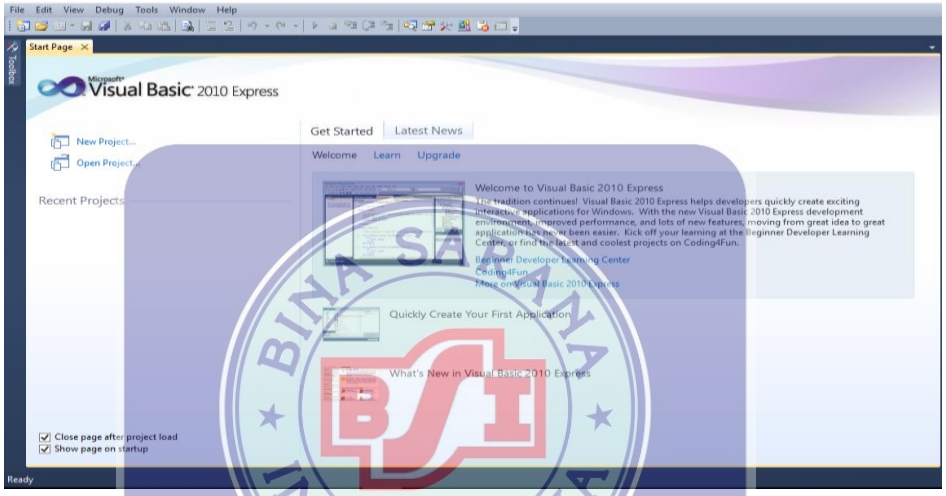
Berikut ini perkembangan Visual Basic .NET:

1. Visual Basic .NET 2002 (VB 7.0)
2. Visual Basic .NET 2003 (VB 7.1)
3. Visual Basic 2005 (VB 8.0)
4. Visual Basic 2008 (VB 9.0)
5. Visual Basic 2010 (VB 10.0)

6. Visual Basic 2012 (VB 11.0)
7. Visual Basic 2013
8. Visual Basic 2015

Berkenalan dengan Visual Basic .NET:

1. IDE *(Integrated Development Environment)* Visual Basis 2010



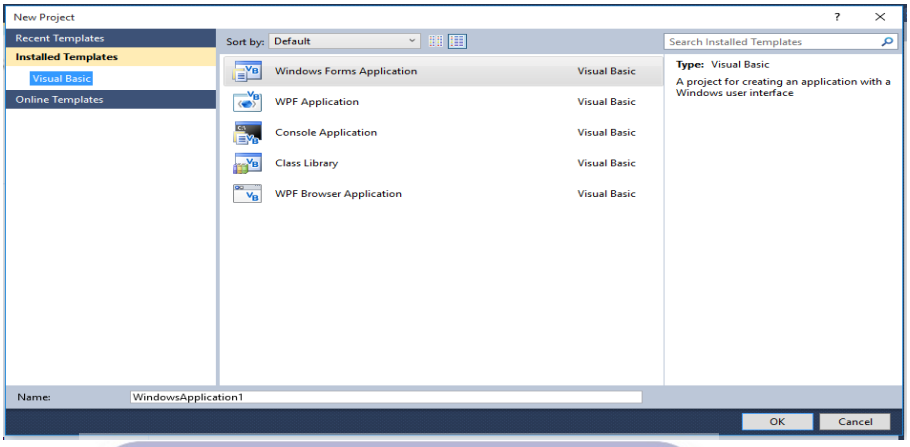
Sumber: Hidayatullah (2015:25)

Gambar II.1. *Star Page* dari Visual Studio 2010

Untuk membuka proyek yang ada, gunakan tombol **Open Project** atau langsung mengklik pada daftar proyek yang ditampilkan sedangkan untuk membuat sebuah proyek baru, klik tombol **New Project.**

Setelah itu akan muncul kotak dialog **New Project.** Pada kotak pilih **Other Languages > Visual Basic > Windows > Windows Forms Application**. Untuk memberi nama proyek dapat dilakukan pada bagian **Name,** tentukan posisi penyimpanan *file-file* proyek dan tentukan nama *solution*-nya dan tekan **OK**. Selanjutnya muncul **Visual Basic 2010 IDE** tempat untuk membangun aplikasi Visual Basic .NET. Pada **IDE Visual Studio 2010** untuk

Windowa Aplication secara *default* telah terdapat sebuah *form*. *Form* tersebut bernama Form1.



Sumber: Hidayatullah (2015:26)

Gambar II.2. Kotak Dialog *New Project*

Sumber: Hidayatullah (2015:27)

Gambar II.3. IDE Visual Studio 2010

2. *Menu Bar*

Sumber: Hidayatullah (2015:27)

Gambar II.4. Menu Bar

Pada IDE Visual Studio 2010, terdapat sepuluh menu utama. Menu-menu tersebut antara lain sebagai berikut:

a. Menu **File** berisi perintah-perintah untuk membuat proyek, membuka proyek, menutup proyek, mencetak data dari proyek.
b. Menu **Edit** berisi perintah-perintah untuk *undo, cut, paste* dan lain-lain.
c. Menu **View** berisi perintah-perintah untuk menampilkan *window-window* dari IDE dan *toolbar*.
d. Menu **Project** berisi perintah-perintah untuk mengatur proyek dan *file-file*.
e. Menu **Build** berisi perintah-perintah untuk meng-*compile* program.
f. Menu **Debug** berisi perintah-perintah untuk men-*debug* dan menjalankan program.
g. Menu **Data** berisi perintah-perintah untuk berhubungan dengan basis data.
h. Menu **Tools** berisi perintah-perintah untuk mengakses komponen IDE tambahan dan mengubah IDE.
i. Menu **Window** berisi perintah-perintah untuk mengatur dan menampilkan Windows.
j. Menu **Help** berisi perintah-perintah untuk mengakses fasilitas bantuan.

3. *Toolbar*

Toolbar fungsinya sama seperti menu. Bedanya pada *toolbar* pilihan-pilihan berbentuk *icon*. Untuk memilih suatu proses yang akan dilakukan, kita tinggal menekan *icon* yang sesuai dengan proses yang kita inginkan.

Sumber: Hidayatullah (2015:29)

Gambar II.5 *Toolbar*

4. *Toolbox*

Toolbox adalah tempat di mana kontrol-kontrol dan komponen-komponen diletakkan. Kontrol dan komponen disimpan pada *Toolbox* dengan berbagai kategori:

a. *Common Controls,* berisi kontrol-kontrol umum yang sering digunakan seperti *button, label, textbox*,dsb.

b. *Containers,* berisi kontrol penyimpan kontrol lainya seperti *penel, group box, tabcontrol,* dsb.

c. *Menus & Toolbars,* berisi kontrol dan komponen *menu, context menu* dan *toolbar.*

d. *Data,* berisi kontrol dan komponen pengolahan data.

e. *Components,* berisi komponen-komponen seperti *timer, imagelist,* dsb.

f. *Printing,* berisi komponen untuk pencetakan dokumen.

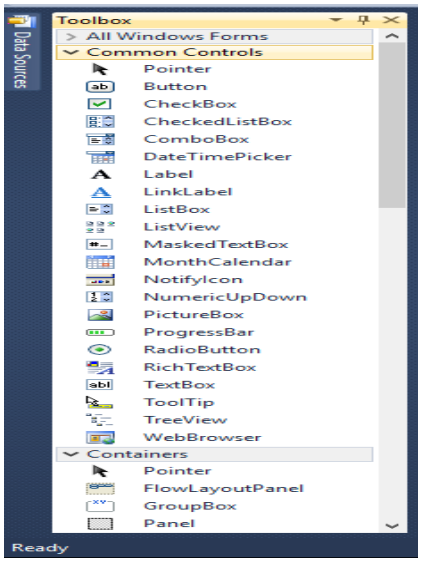
g. *Dialogs,* berisi komponen untuk berinteraksi dengan pengguna dalam hal membuka *file,* menyimpan *file,* membuka *folder* dll.

h. *WPF Interoperability,* berisi komponen untuk *Windows Presentation Foundation.*

i. *Reporting,* berisi kontrol untuk membuat laporan pada Visual Studio 2010. Untuk studi kasus pada buku ini, kita akan menggunakan fasilias *reporting* dari *Crytal Report* yang jauh lebih lengkap dan *powerful* namun gratis.

j. *Visual Basic PowerPacks,* berisi beberapa kontrol tambahan untuk menggambarkan (contoh: oval, garis, persegi) dan fungsi tambahan lainnya.

k. *General,* jika kontrol atau komponen yang mau ditambahkan dan belum memiliki kategori yang jelas, maka bisa dimasukkan ke kategori ini.

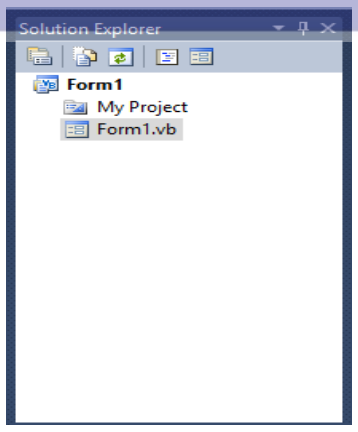


Sumber: Hidayatullah (2015:30)

Gambar II.6. *Toolbox*

5. *Solution Explorer*

Solution Explorer memberikan tampilan daftar *file-file* dari proyek yang sedang dibuat. Pada jendela *Solution Explorer* terdapat beberapa tombol dan *tree* yang berisi daftar dari *file-file* yang digunakan dalam proyek. Jika anda tidak dapat menemukan *Toolbox* di *IDE* anda, pilih menu **View > Solution Explorer** atau tekan **Ctrl + Alt + L**.

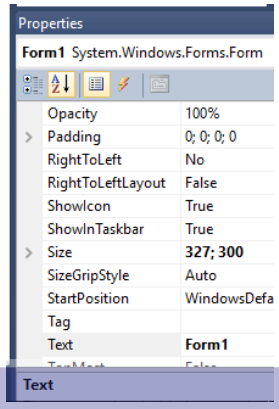


Sumber: Hidayatullah (2015:31)

Gambar II.7. *Solution Explorer*

6. *Properties Window*

Properties Window adalah tempat menyimpan *property* dari setiap objek kontrol dan komponen.

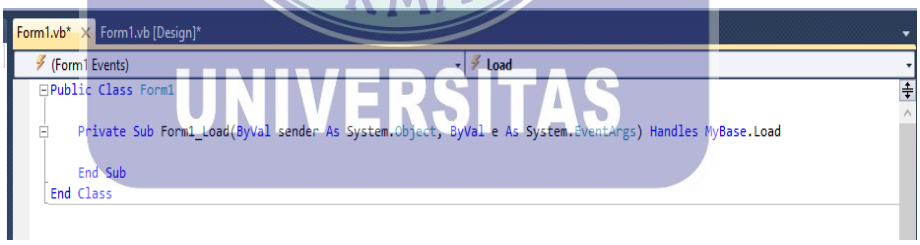


Sumber: Hidayatullah (2015:32)

Gambar II.8. *Properties Window*

7. *Code Editor*

Code editor adalah tempat di mana kita meletakkan atau menuliskan kode program dari program aplikasi kita.



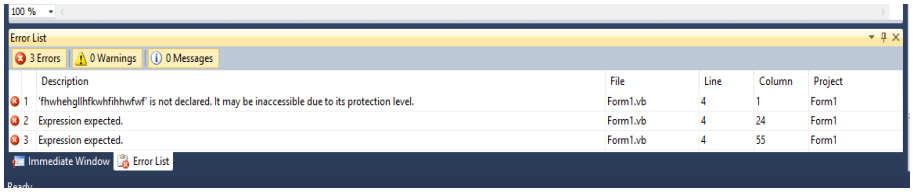
Sumber: Hidayatullah (2015:33)

Gambar II.9. *Code editor*

Untuk kembali melihat *form*, tekan tombol ***View Designer*** yang terletak pada bagian sebelah kanan *View Code*.

8. *Output Window*

Output Window menujukkan langkah-langkah dalam mengompilasi aplikasi.



Sumber: Hidayatullah (2015:33)

Gambar II.10. *Output Window*

2.1.10. Basis Data *(Database)*

Menurut Lubis (2016:3), "Basis Data adalah tempat berkumpulnya data yang saling berhubungan dalam suatu wadah (organisasi/ perusahaan) bertujuan agar dapat mempermudah dan mempercepat untuk pemanggilan atau pemanfaatan kembali data tersebut.

Pada kehidupan sehari-hari didunia komputer, basis data akan menggunakan media penyimpanan *(storage)*, yaitu berkaitan dengan setiap alat yang dapat menerima data yang dapat disimpan, dan dapat dipanggil kembali data itu pada waktu berikutnya atau setiap alat yang dapat digunakan untuk menyimpan data. Adapun media penyimpanan yang dapat digunakan terdiri dari *harddisk*, *diskette* atau *floppy disk*, *tape* maupun dengan *compact disk* (CD) atau DVD dan kini juga dapat digunakan *flashdisk*.

Dengan bantuan basis data ini diharapkan bahwa sistem informasi yang dibuat dapat terintegrasi antara bagian yang satu dengan yang lainnya, sehingga pada akhirnya tidak ada pembatas area dalam perusahaan. Dalam pembuatan dan penggunaan basis data, terdapat 4 (empat) komponen dasar sistem basis data, yaitu:

1. Data

   Data yang digunakan dalam sebuah basis data, haruslah mempunyai ciri sebagai berikut:

   a. Data disimpan secara integrasi, yaitu *database* merupakan kumpulan dari berbagai macam *file* dari aplikasi-aplikasi yang berbeda yang disusun dengan cara menghilangkan bagian-bagian yang rangkap.
   b. Data dapat dipakai secara bersama-sama, yaitu masing-masing bagian dari *database* dapat diakses oleh pemakai dalam waktu yang bersamaan, untuk aplikasi yang berbeda.

2. *Hardware*

   Terdiri dari semua peralatan perangkat keras komputer yang digunakan untuk pengelolaan sistem *database*, seperti:

   a. Peralatan untuk penyimpanan, *disk*, drum dan lain-lain.
   b. Peralatan *input* dan *output*.
   c. Peralatan komunikasi data.

3. *Software*

   Berfungsi sebagai perantara antara pemakai dengan data fisik pada *database*, dapat berupa :

   a. *Database Management System* (DBMS).
   b. Program-program aplikasi dan prosedur-prosedur yang lain seperti Oracle, SQL Server, MySQL.

      MySQL adalah salah satu jenis *database server* yang sangat terkenal dan banyak digunakan untuk membangun aplikasi *web* yang menggunakan *database* sebagai sumber dan pengolahan datanya. MySQL

dikembangkan oleh perusahaan swedia bernama MySQL AB yang pada saat ini benama Tex Data Konsult AB sekitar tahun 1994-1995.

MySQL merupakan *database* yang pertama kali didukung oleh bahasa pemrograman *script* untuk internet (PHP dan Perl). MySQL dan PHP dianggap sebagai pasangan *software* pembangun aplikasi *web* yang *ideal*. MySQL lebih sering digunakan untuk membangun aplikasi berbasis *web*, umumnya pengembangan aplikasinya menggunakan bahasa pemrograman *script* PHP. Adapun kelebihan-kelebihan dari MySQL yaitu:

1) *Source* MySQL dapat diperoleh dengan mudah dan gratis.
2) Sintaksnya lebih mudah dipahami dan tidak rumit.
3) Pengaksesan basis data dapat dilakukan dengan mudah.
4) MySQL merupakan program yang *multithreaded*, sehingga dipasang pada server yang memiliki multi CPU.
5) Didukung program-program umum seperti C, C++, Java, Perl, PHP, Python, dsb.
6) Bekerja pada berbagai *platform*. (tersedia berbagai versi untuk berbagai sistem operasi).
7) Memiliki jenis kolom yang cukup banyak sehingga memudahkan konfigurasi sistem *database*.
8) Mendukung ODBC untuk sistem operasi *Windows*.
9) Mendukung *record* yang memiliki kolom dengan panjang tetap atau panjang bervariasi.

4. *User*

Terdiri menjadi 3 klasifikasi:

a. *Database Administrator,* yaitu orang yang bertugas mengelola sistem *database* secara keseluruhan.
b. *Programmer,* yaitu orang yang membuat program aplikasi yang mengakses *database* dengan menggunakan bahasa pemograman.
c. *End User,* orang yang mengakses *database* melalui terminal dengan menggunakan *query language* atau program aplikasi yang dibuat oleh *programmer.*

Penggunaan data pada basis data mempunyai peran yang kuat pada masing-masing bagian, yang dikelompokkan dalam 3 jenis data pada sistem basis data, yaitu:

1. Data operasional dari suatu organisasi, berupa data yang disimpan di dalam basis data.
2. Data masukan *(input data),* data dari luar sistem yang dimasukkan melalui peralatan *input* yang dapat mengubah data operasional.
3. Data keluaran *(output data),* berupa laporan melalui peralatan *output* sebagai hasil dari dalam sistem yang mengakses daa operasional.

2.1.11. Aplikasi *Database* MYSQL

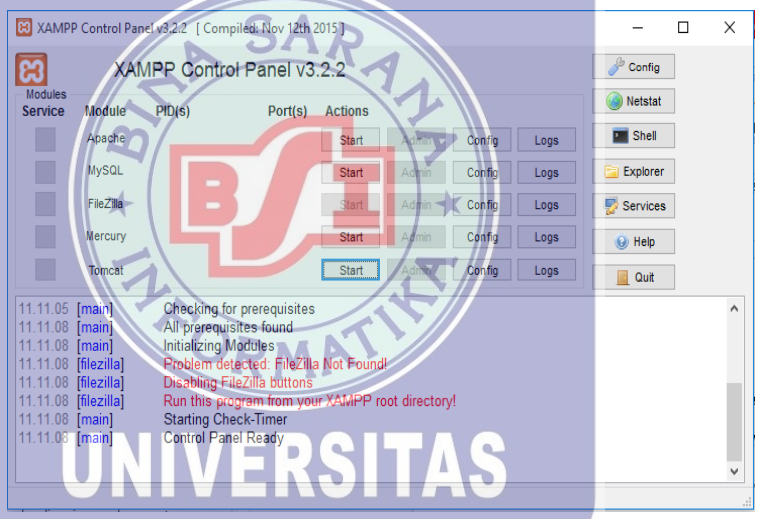
Aplikasi basis data sering digunakan oleh para pembuat aplikasi sebagai media pengolahan basis data. Aplikasi basis data yang sering digunakan dalam pengolahan basis data yaitu *My Structure Query Language (MySQL).*

Salah satu aplikasi basis data yang sering digunakan untuk mengolah dan menata *file-file* yaitu *MySQL*. Menurut Sadeli (2013:10) memberikan pengertian bahwa, "*MySQL* adalah *database* yang menghubungkan *script php* menggunakan

perintah *query* dan *escaps character* yang sama dengan *php*". Menurut Madcome (2013), "*Mysql* merupakan *databases* yang sangat popular. Beberapa keuntungan yang dimiliki *mysql* yaitu: bersifat *open source*, menggunakan bahasa *Structure Query Language (SQL)*, super *perfomance* dan *reliable*, mudah dipelajari, mampu bekerja dilintas *platform* dan *multi user*.

2.1.12. XAMPP

XAMPP adalah aplikasi *web server* bersifat instan (siap saji) yang dapat digunakan baik di sistem operasi Linux maupun di sistem operasi Windows.



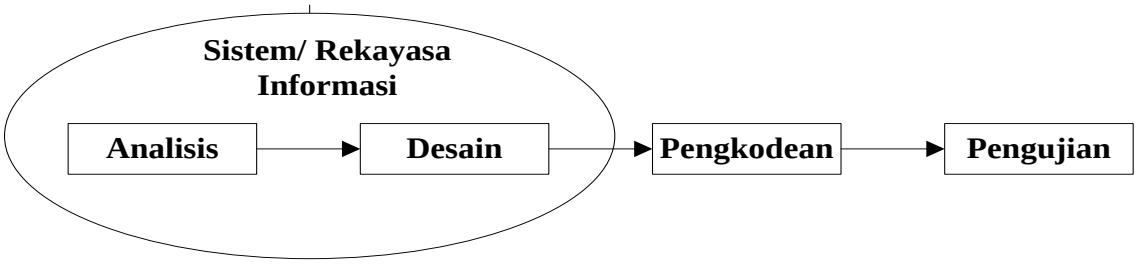
Sumber: Pratama (2014:440)

Gambar: II.11. Tampilan XAMPP

2.1.13. Model Pengembangan Perangkat Lunak

Metode yang digunakan pada pengembangan perangkat lunak ini menggunakan model *waterfall*. Menurut Rosa A.S dan M. Shalahuddin (2013:28) menjelaskan bahwa, "Model *waterfall* sering juga disebut model sekuensial linier atau alur hidup klasik. "Model air terjun menyediakan pendekatan alur hidup

perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengodean, pengujian dan tahap pendukung.



Sumber: Rosa dan M. Shalahuddin (2013:29)

Gambar II.12. Gambar Tahapan model *waterfall*

1. Analisa kebutuhan perangkat lunak

   Proses pengumpulan kebutuhan dilakukan secara *intensif* untuk menspesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user*. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu untuk didokumentasikan.

2. Desain

   Desain perangkat lunak adalah proses multi langkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antar muka, dan prosedur pengkodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan kerepresentasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya. Desain perangkat lunak yang dihasilkan pada tahap ini juga perlu didokumentasikan.

3. Pembuatan kode program

   Desain harus ditranslasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

4. Pengujian

   Pengujian fokus pada perangkat lunak secara segi *logic* dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan *(error)* dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

5. Pendukung *(support)* atau pemeliharaan *(maintenance)*

   Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user*. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari tahap analisis spesifikasi untuk perubahan perangkat lunak baru.

2.2. Teori Pendukung

Peralatan pendukung merupakan alat yang tepat digunakan untuk menggambarkan model logika dari suatu program, model logika dari program lebih menjelaskan dari pemakaian bagaimana nantinya fungsi-fungsi dari program secara logika akan bekerja. Adapun perlatan yang dimaksud adalah UML, Diagram UML, *Use Case Diagram*, *Activity Diagram*, *Entity Relationship Diagram* (ERD), *Logical*

Record Structure (LRS), Teknik pengkodean, *Class Diagram*, dan *Sequence Diagram*.

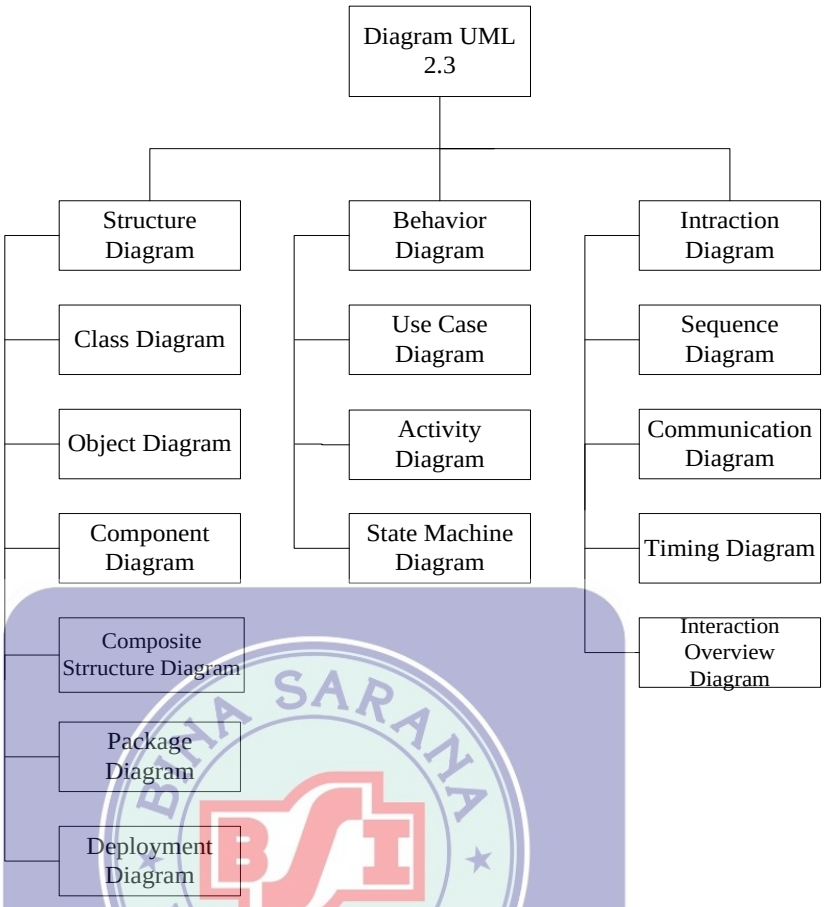
2.2.1. *Unified Modeling Language (UML)*

Menurus Rosa dan Shalahuddin (2016:133), "UML *(Unified Modeling Language)* salah satu strandar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requierment*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemragraman objek."

Secara fisik, UML merupakan sekumpulan spesifikasi yang dikeluarkan oleh OMG. UML, terbaru adalah 2.3 yang terdiri dari empat macam spesifikasi, yaitu *Diagram Interchange Specification*, UML *Infranstructure*, UML *Superstructure*, dan *Object Constraint Language* (OCL).

2.2.2. Diagram UML

Pada UML 2.3 terdiri dari tiga belas (13) macam diagram yang dikelompokkan kedalam tiga kategori. Adapun pembagian kategori dan macam-macam diagram tersebut akan digambarkan pada diagram berikut.



Sumber: Rosa dan Shalahuddin (2016:140)

Gambar II.12 Diagram UML

Secara garis besar penjelasan gambar diatas adalah sebagi berikut:

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

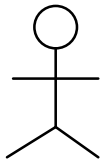
2.2.3. *Use Case Diagram*

Menurut Rosa dan Shalahuddin (2016:155), *"Use Case Diagram* merupakan pemodelan untuk kelakuann (*behavior*) sistem informasi yang aan dibuat. *Use Case* mendeskripsi 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 didalam sebuah informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Berikut adalah simbol-simbol yang terdapat didalam *use case diagram*:

Tabel II.1

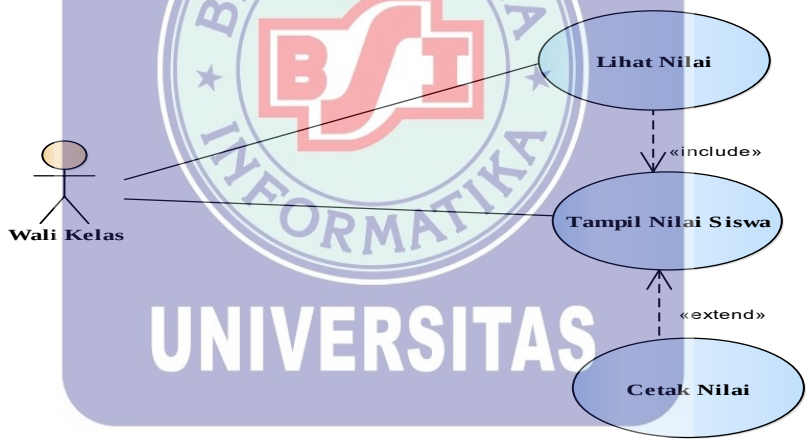
Simbol-Simbol *Use Case Diagram*

Simbol	Deskripsi
Use Case **Nama use case**	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja diawal nama *use case*.
Aktor/*actor* 	Merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri.
Asosiasi ———————	Menunjukkan *use case* memiliki interaksi dengan aktor.

Extensi <<entend>>	Relasi *use case* tambahan kesebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu.
Generalisasi	Menunjukkan hubungan generalisasi-spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu lebih umum dari lainnya.
Include <<include>>	*Include* berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan.

Sumber: Rosa dan Shalahuddin (2016:156)

Berikut contoh dari *Use Case Diagram:*



Sumber: Juniardi Dermawan dan Sri Hartini dari *Jurnal Paradigma BSI , Vol. 19, No. 2,September 2017*

Gambar II.13. Contoh *Use Case Diagram*

2.2.4. *Class Diagram*

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem, dimana

kelas ini memiliki apa yang disebut dengan atribut atau variabel-variabel yang dimiliki oleh suatu kelas dan operasi atau metode yang berarti fungsi-fungsi yang dimiliki oleh suatu kelas.

Diagram kelas dibuat agar pembuat atau *programmer* membuat kelas-kelas yang sesuai dengan rancangan didalam diagram kelas sehingga terjadi kesesuaian antara dokumentasi perancangan dan perangkat lunak, dengan demikian kelas-kelas yang ada pada struktur sistem dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem tersebut. Susunan struktur kelas dalam diagram kelas terdiri dari beberapa jenis kelas sebagai berikut:

1. Kelas *Main* adalah kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
2. Kelas *View* adalah kelas yang menangani tampilan sistem ke pemakai.
3. Kelas *Coniroller* adalah kelas yang menangani fungsi-fungsi yang diambil dari pendefinisian *Use Case* dan menangani proses bisnis pada perangkat lunak.
4. Kelas Model adalah kelas yang digunakan untuk membungkus data menjadi sebuah kesatuan yang diambil maipun disimpan dalam basis data.

Berikut adalah simbol-simbol yang terdapat dalam kelas diagram:

Tabel II.2.

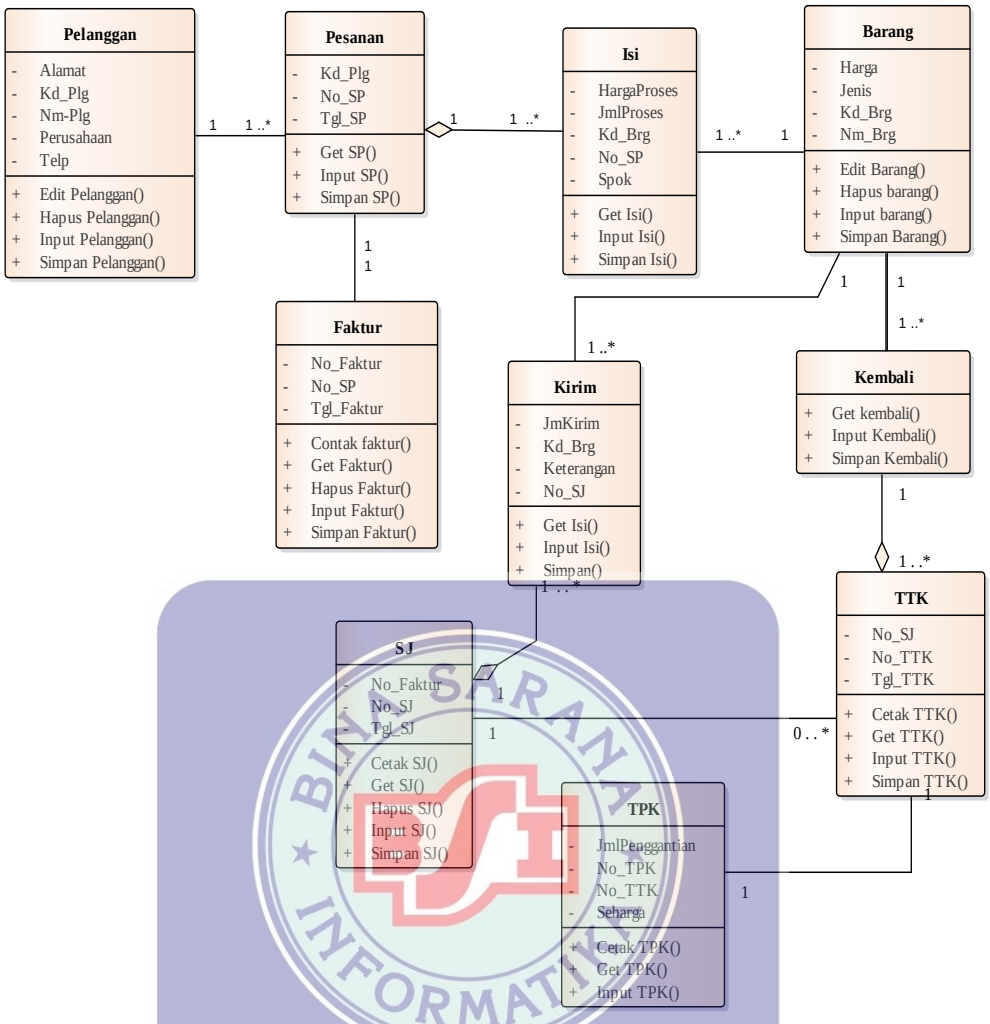
Simbol-simbol Pada Diagram Kelas

Simbol	Deskripsi
Kelas **nama_kelas** +atribut +operasi()	Kelas pada struktur sebuah sistem

Antarmuka/*interface*  **nama_interface**	Sama dengan konsep *interface* dalam pemrogramman berorientasi onjek
Asosiasi/*association* 	Menunjukkan relasi atau hubungan antarkelas dengan makna umum
Asosiasi berarah/*directed association* 	Menunjukkan relasi kelas yang satu digunakan oleh kelas lain
Generanlisasi 	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum-khusus)
Kebergantungan/*dependency* 	Relasi antarkelas dengan makna saling kebergantungan satu sama lainnya.
Agregasi/aggregation 	Relasi antarkelas dengan makna semua bagian (*whole-part*)

Sumber: Rosa dan Shalahuddin (2016:146).

Berikut contoh dari *Class Diagram:*



Sumber: Sunarti *Jurnal Paradigma BSI, Vol XVI no.2 September 2014*

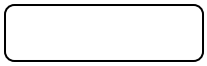
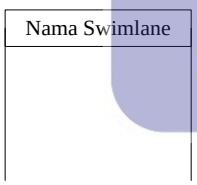
Gambar II.14. *Class Diagram*

2.2.5. *Activity Diagram*

Menurut Rosa dan Shalahuddin (2016:157) "*Activity diagram* atau diagram aktivitas adalah *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak". Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Berikut simbol-simbol dalam diagram aktivitas:

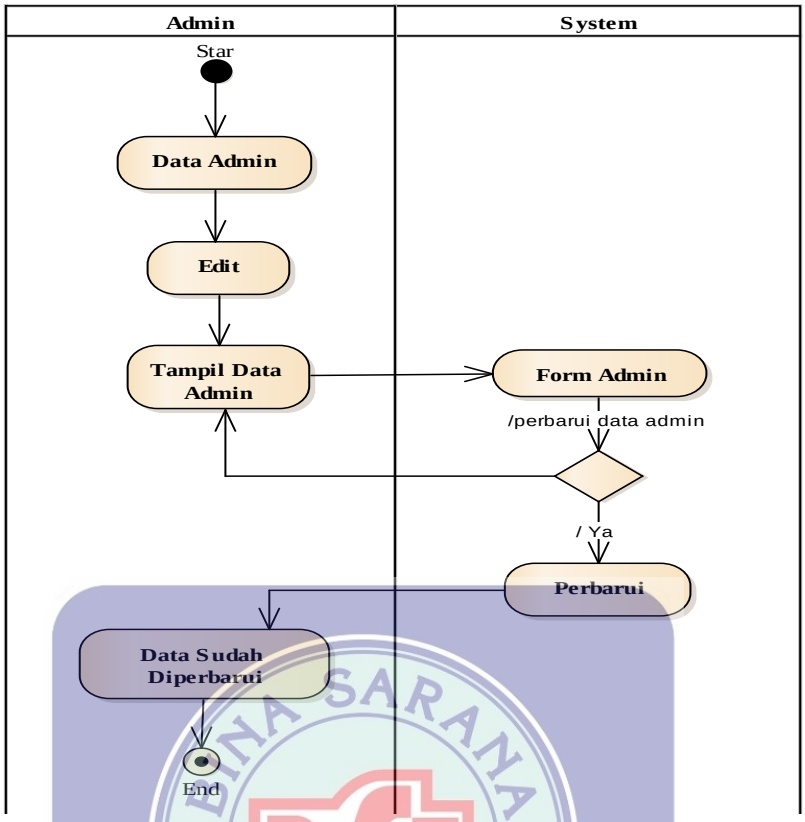
Tabel II.3.

Simbol-simbol *Activity Diagram*

Simbol	Deskripsi
Status awal 	Sebuah diagram aktivitas memiliki sebuah status awal
Aktivitas 	Aktivitas yang dilakukan sistem biasanya diawali dengan kata kerja
Percabangan/*decision* 	Percabangan terjadi jika ada pilihan lebih dari satu
Penggabungan/*join*	Ketika ada lebih dari satu aktivitas yang akan digabungkan
Status akhir 	Status akhir yang dilakukan sistem karena sebuah diagram aktivitas pasti memiliki status akhir
Swimlane 	Memisahkan organisasi yang bertanggung jawab terhadap aktivitas yang terjadi

Sumber: Rosa dan Shalahuddin (2016:162)

Berikut contoh dari *Acivity Diagram:*



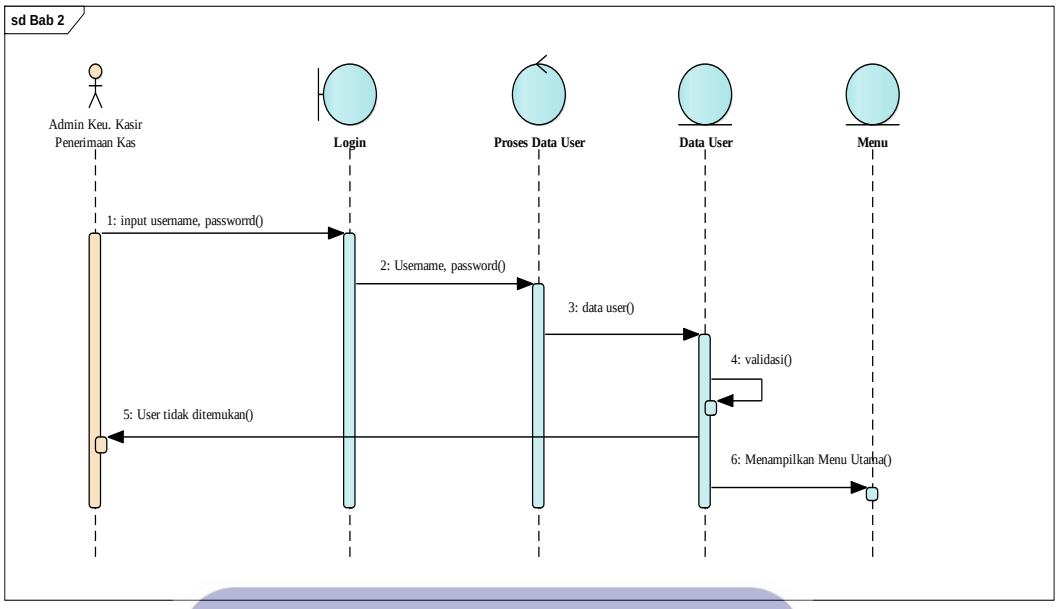
Sumber: Juniardi Dermawan dan Sri Hartini dari *Jurnal Paradigma BSI , Vol. 19, No. 2,September 2017*

Gambar II.15. Contoh *Activity Diagram*

2.2.6. *Sequence Diagram*

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Oleh karena itu, dalam menggambar diagram sekuen harus diketahui objek-objek yang terlibat dalam sebuah *use case* terlebih dahulu.

Berikut contoh dari *Sequence Diagram:*



Sumber: Syara, Chintya. *Jurnal Paradigma BSI, Vol. XX, No. 1, Maret 2018.*

Gambar II.16. Contoh *Sequence Diagram*

2.2.7. *Component Diagram*

Component Diagram atau diagram komponen menunjukkan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem yang berfokus pada komponen sistem yang dibutuhkan dan ada dalam sistem.

Fungsi lain dari diagram komponen adalah untuk memodelkan *source code* perangkat lunak, komponen *executable* yang dilepas ke *user*, basis data secara fisik, sistem yang harus beradaptasi dengan sistem lain dan *framework* sistem atau perangkat lunak yang merupakan kerangka kerja yang dibuat untuk memudahkan pengembangan dan pemeliharaan aplikasi. Berikut adalah komponen dasar yang biasanya ada dalam suatu sistem:

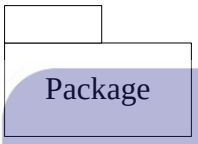
1. Komponen *user interface* yang menangani tampilan.
2. Komponen *bussines processing* yang menangani fungsi-fungsi proses bisnis.
3. Komponen *data* yang menangani manipulasi data.

4. Komponen *security* yang menangani keamanan sistem.

Komponen lebih terfokus pada penggolongan seara umum fungsi-fungsi yang diperlukan. Berikut adalah simbol-simbol yang terdapat pada diagram komponen:

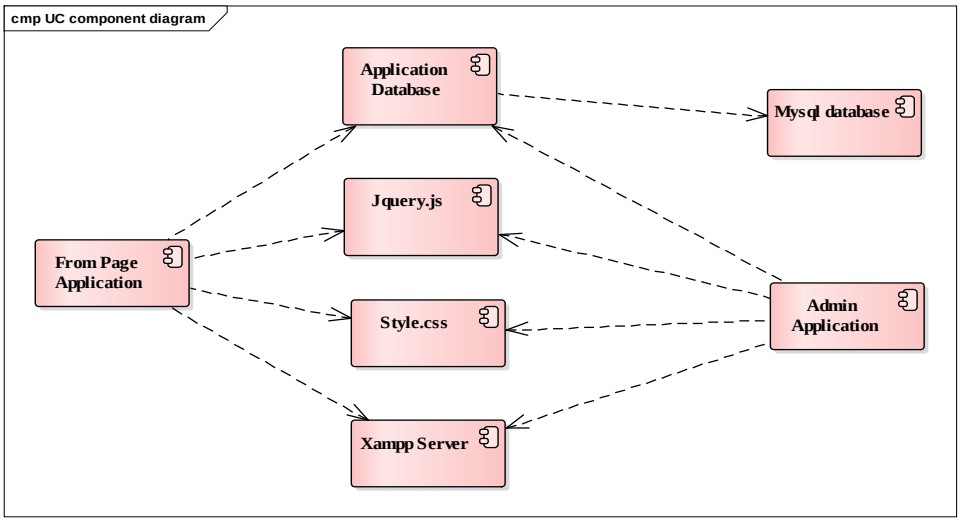
Tabel II.4.

Simbol-simbol Pada *Component Diagram*

Simbol	Deskripsi
Package 	Package merupakan sebuah bungkusan dari satu atau lebih komponen
Komponen 	Komponen sistem
Kebergantungan/ *dependency* 	Pada simbol ini arah panahnya mengarah pada komponen yang dipakai
Antarmuka/*interface*  **nama_interface**	Sebagai antar muka komponen agar tidak mengakses komponen secara langsung
Link	Menunjukkan relasi antar komponen

Sumber: Rosa dan Shalahuddin (2016:149)

Berikut contoh dari *Component Diagram:*



Sumber: Juniardi Dermawan dan Sri Hartini dari *Jurnal Paradigma BSI , Vol. 19, No. 2,September 2017*

Gambar II.16. Contoh *Component Diagram*

2.2.8. *Entity Relationship Diagram (ERD)*

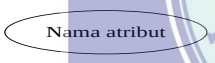
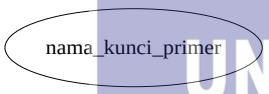
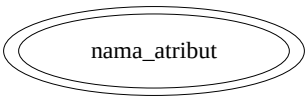
Model *Entity Relationship Diagram* merupakan pemodelan yang menggambarkan komponen-komponen himpunan entitas dan himpunan relasi yang masing-masing dilengkapi dengan atribut-atribut yang mempresentasikan fakta atau kejadian yang terjadi pada dunia nyata.

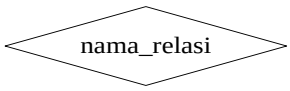
Entity Relationship Diagram merupakan notasi grafis dalam pemodelan data konseptual yang mendeskripsikan hubungan antar penyimpanan. Dengan menggunakan *Entity Relationship Diagram* dapat memodelkan struktur data dan hubungan antar data yang begitu kompleks, sehingga mudah untuk dibaca dan dimengerti. Pengguna *Entity Relationship Diagram* dalam pemodelan struktur data dapat mengabaikan proses yang harus dilakukan, selain itu pemodelan struktur data dengan menggunakan *Entity Relationship Diagram* dapat menjawab pertanyaan seperti, data apa saja yang kita butuhkan dan bagaimana data yang satu

berhubungan dengan data yang lainnya. Adapun simbol-simbol yang digunakan dalam *Entity Relationship Diagram*:

Tabel II.5.

Simbol-simbol *Entity Relationship Diagram*

Simbol	Deskripsi
Entitas/*entity* 	Entitas merupakan data inti yang akan disimpan; bakal 44ompu pada basis data; benda yang memiliki data dan harus disimpan datanya agar dapat diakses oleh aplikasi 44omputer; penamaan entitas biasanya lebih ke kata benda dan belum merupakan nama table
Atribut 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas
Atribut Kunci Prime 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses *record* yang diinginkan; biasanya berupa id; kunci primer dapat lebih dari satu kolom, asalkan kombinasi dari kolom tersebut dapat bersifat unik (berbeda tanpa ada yang sama)
Atribut Multinilai/*multivalue* 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki nilai lebih dari satu

Relasi  	Relasi yang menghubungkan antar entitas; biasanya diawali dengan kata kerja

Sumber: Rosa dan Shalahuddin (2016:165)

Adapun derajat kardinalitas yang terdapat dalam *Entity Relationship Diagram* adalah:

1. *One To one*

   Adalah derajat kardinalitas yang menunjukkan adanya relasi himpunan entitas yang satu dengan satu entitas lainnya.

   Sumber: Rosa dan Shalahuddin (2016:166)

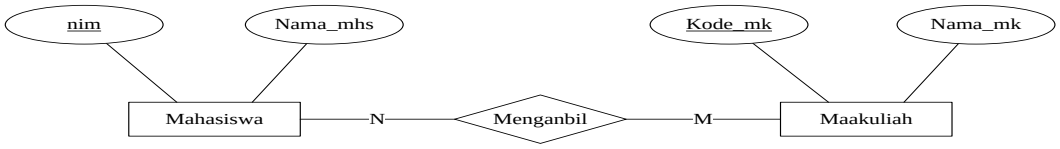
2. *One to many*

   Adalah derajat kardinalitas yang menunjukkan adanya relasi antar himpunan entitas yang satu dengan entitas banyak entitas lainnya.

   Sumber: Rosa dan Shalahuddin (2016:166)

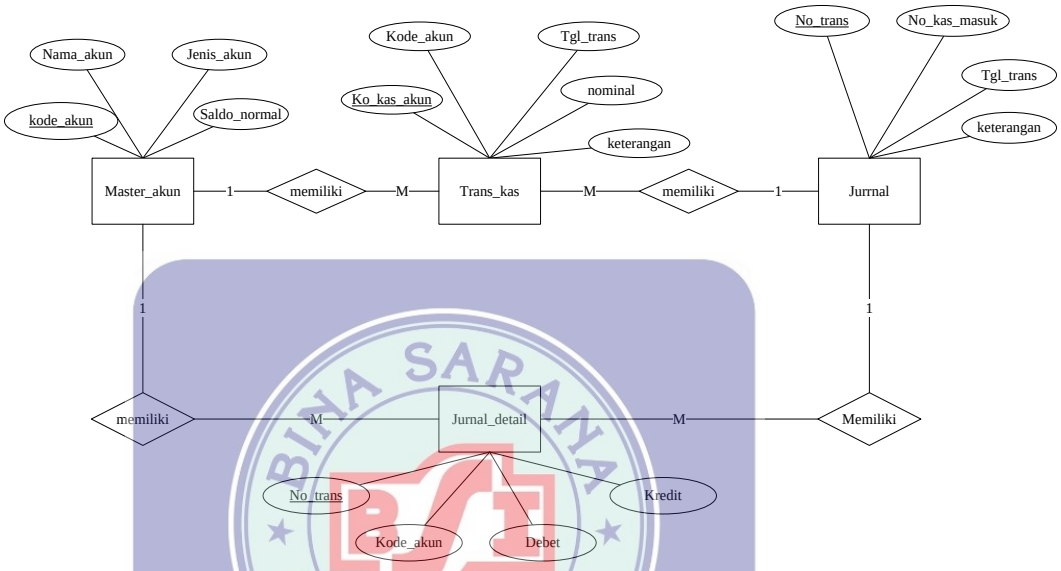
3. *Many to many*

   Adalah derajat kardinalitas yang menunjukkan adanya relasi antar himpunan banyak entitas dengan banyak entitas lainnya. Jika terjadi relasi *many to many* maka akan menghasilkan sebuah entitas baru.



Sumber: Rosa dan Shalahuddin (2016:166)

Berikut contoh dari *Entity Relationship Diagram:*



Sumber: Syara, Chintya. *Jurnal Paradigma BSI, Vol. XX, No. 1, Maret 2018.*

Gambar II.17. *Entity Relationship Diagram*

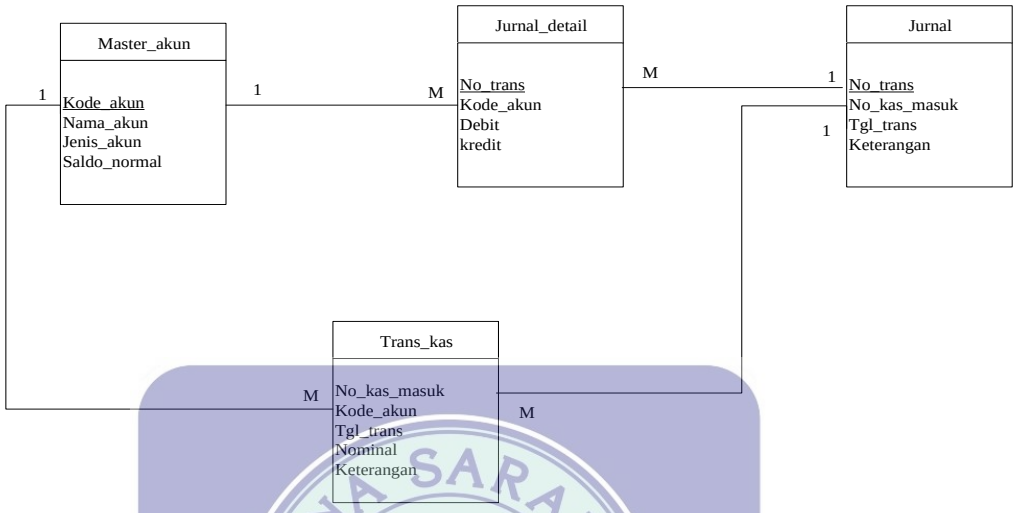
2.2.9. *Logical Record Structure (LRS)*

Logical Record Structure (LRS) adalah representasi dari s*structure record-record* pada tabel-tabel yang berbentuk dari hasil antar himpunan entitas. Menentukan kardinalitas, jumlah tabel dan *Foreign Key* (FK). Aturan pembuatan LRS adalah:

1. Setiap entitas akan diubah menjadi bentuk kotak.
2. Sebuah atribut disatukan dalam sebuah kotak bersama entitas.

3. Sebuah relasi dipisah dalam sebuah kotak tersendiri (menjadi entitas baru) dalam bentuk belah ketupat.

Berikut contoh dari *Logical Record Structure:*



Sumber: Syara, Chintya. *Jurnal Paradigma BSI, Vol. XX, No. 1, Maret 2018.*

Gambar II.18. *Logical Record Structure*

2.2.10. Kamus Data *(Data Dictionary)*

Menurut Rosa dan M. Shalahuddin (2013:73), "Kamus Data adalah kumpulan daftar elemen data yang mengalir pada sistem perangkat lunak sehingga masukan (*input*) dan keluaran (*output*) dapat dipahami secara umum (memiliki standar cara penulisan). Kamus data dalam implementasi program dapat menjadi parameter masukan atau keluaran dari sebuah fungsi atau prosedur".

2.2.11. Pengkodean

Fathansyah (2015:34) mengemukakan bahwa, "Strukur kode bertujuan untuk mengklasifikasikan data, memasukkan data kedalam komputer dan untuk

mengambil bermacam-macam informasi yang berhubungan dengan data tersebut. Kode data dibentuk dari kumpulan angka, huruf dan karakter-karakter khusus".

Dalam merancang kode yang baik ada beberapa hal yang harus diperhatikan, yaitu sebagai berikut:

a. Harus mudah diingat

   Supaya kode mudah diingat maka dapat dilakukan dengan cara menghubungkan kode tersebut dengan obyek yang diwakili dengan kodenya.

b. Unik

   Kode harus unik untuk masing-masing item yang diwakilinya. Unik berarti tidak ada kode yang kembar.

c. Fleksibel

   Kode harus fleksibel sehingga memungkinkan perubahan-perubahan atau penambahan item baru dapat tetap diwakili oleh kodenya masing-masing.

d. Efisien

   Kode harus sekecil mungkin, selain mudah diingat juga akan efisien apabila direkam dan disimpan diluar komputer.

e. Konsisten

   Bilamana mungkin kode konsisten dengan kode yang telah digunakan.

f. Harus distandarisasi

   Kode harus distandarisasi untuk seluruh tingkatan dan departemen dalam organisasi.

g. Spasi dihindarkan

Spasi dalam kode sebaiknya dihindarkan, karena dapat menyebabkan kesalahan dalam menggunakannya. Karakter-karakter yang hampir serupa bentuk dan bunyi pengucapannya sebaiknya tidak digunakan dalam kode.

h. Hindarkan karakter yang mirip

Karakter-karakter yang hampir mirip bentuk dan bunyi pengucapannya sebaiknya tidak digunakan dalam kode.

i. Panjang kode harus sama

Masing-masing kode yang sejenis harus memiliki panjang kode yang sama.

Ada beberapa macam tipe dari kode yang dapat digunakan didalam sistem informasi, antara lain:

1. Kode Mnemonik *(mnemonik code)*

   Bertujuan supaya mudah diingat, dibuat dengan dasar singkatan atau mengambil sebagai karakter dari item yang akan diwakili dengan kode ini.

2. Kode Urut (*squential code*)

   Disebut juga dengan kode seri, merupakan kode yang nilainya urut anatara kode dengan kode berikutnya.

3. Kode Blok *(block code)*

   Mengklasifikasi item kedalam kelompok blok tertentu yang mencerminkan satu klasifikasi tertentu atas dasar pemakaian maksimum yang diharapkan.

4. Kode Group *(group code)*

   Kode berdasarkan field-field dan tiap-tiap kode mempunyai arti tersendiri.

5. Kode Desimal *(decimal code)*

Kode desimal adalah kode yang mengklasifikasikan kode atas dasar sepuluh unit angka desimal dimulai dari angka nol sampai dengan sembilan atau 00 sampai dengan 99 tergantung banyaknya kelompok.