BAB II

LANDASAN TEORI

2.1. Konsep Dasar Sistem

Konsep dasar sistem merupakan suatu rangkaian, jaringan atau konsep kerangka kerja dari prosedur-prosedur yang disusun sesuai dengan skema yang menyeluruh saling berhubungan, saling berpengaruh dan saling bergantung satu dengan yang lainnya untuk melaksanakan suatu kegiatan, fungsi atau proses tertentu sehingga dapat mencapai tujuan dengan hasil tersedianya informasi sesuai dengan yang dibutuhkan.

2.1.1. Pengertian Sistem

Secara umum sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur, komponen atau variabel yang terorganisir, saling berhubungan atau saling berinteraksi mempunyai ketergantungan satu dengan yang lainnya dan terpadu untuk mencapai suatu tujuan tertentu.

Menurut 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".

Model umum atau unsur-unsur yang mewakili suatu sistem terdiri dari masukan *(input)*, pengolahan *(processing)* dan keluaran *(output)*. Suatu sistem tidak lepas dari umpan balik *(feedback)* dilingkungan sekitarnya atau dari keluaran

(output) tersebut. Setiap sistem terdiri dari struktur dan proses, dimana struktur sistem merupakan unsur-unsur yang membentuk sistem tersebut, sedangkan proses sistem menjelaskan cara kerja setiap unsur sistem tersebut dalam mencapai tujuan sistem. Sub sistem merupakan bagian dari sistem lain yang lebih besar dan terdiri dari berbagai sistem yang lain atau lebih kecil.

2.1.2. Karakteristik Sistem

Suatu sistem memiliki karakteristik atau sifat tertentu, karakteristik dari suatu sistem merupakan unsur-unsur yang terpadu untuk mencapai tujuan tertentu. Karakteristik Sistem menurut Sutabri (2016:10) mengemukakan bahwa "Model umum sebuah sistem adalah masukan *(input)*, proses dan keluaran *(output)*, hal ini merupakan konsep sebuah sistem yang sangat sederhana karena sebuah sistem dapat memiliki beberapa keluaran". Karakteristik sistem menurut Sutabri (2016:10) terdiri menjadi beberapa bagian sebagai berikut:

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 sifat dari sistem yang menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan sehingga terjadi 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)*

   Lingkungan luar sistem merupakan segala sesuatu diluar batas sistem yang mempengaruhi operasi suatu 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, karena 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 yaitu masukkan sistem yang dapat berupa masukan pemeliharaan *(maintenance input)* agar sistem tersebut beroperasi. Contohnya program didalam suatu unit sistem komputer, masukan pemeliharaan *(maintenance input)* yang digunakan untuk mengoperasikan yaitu komputer dan data merupakan masukan sinyal *(signal input)* yang digunakan untuk pengolahan data menjadi informasi. Selain itu sistem masukan dapat

berupa sinyal masukan *(signal input)* yang bertujuan agar energi yang dimasukkan menghasilkan keluaran *(output)* yaitu informasi.

6. Keluaran Sistem *(Output)*

   Keluaran *(output)* adalah hasil energi yang diolah dan diklarifikasikan menjadi keluaran yang berguna, sehingga keluaran ini dapat menjadi masukan bagi sub sistem yang lainnya. Contohnya sistem informasi, keluaran yang dihasilkan tentunya adalah informasi yang mana informasi ini dapat digunakan sebagai masukan untuk pengambilan keputusan atau hal-hal lain yang menjadi masukan *(input)* bagi sub sistem lainnya.

7. Pengolah Sistem *(Process)*

   Suatu sistem harus memiliki suatu perangkat yang bertugas mengolah bagian atau melakukan proses yang akan mengubah dari masukan menjadi keluaran. Sebagai contoh, sistem persediaan barang yang akan melakukan proses pengolahan data barang yang terdiri dari barang, masuk, barang keluar sampai dengan laporan, informasi yang tersedia dan dibutuhkan sesuai dengan tingkatan manajemen.

8. Sasaran Sistem *(Objective)*

   Sasaran dari sistem sangat menentukan sekali masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem. Suatu sistem harus memiliki tujuan dan sasaran yang pasti karena akan menentukan masukan yang dibutuhkan sistem dan berpengaruh pada keluaran system. Sasaran merupakan hal-hal yang menjadi objek dan titik fokus untuk mencapai tujuan. Suatu sistem dapat dikatakan berhasil menjalankan fungsinya apabila berhasil mencapai sasaran dan tujuan yang telah direncanakan dari sistem tersebut.

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”.

1. Sistem Abstrak dan Sistem Fisik

   Sistem abstrak adalah sistem yang berupa pemikiran atau ide-ide yang tidak terlihat 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

Pengertian Informasi menurut Sutabri (2016:4) “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 tidak berguna menjadi berguna bagi penerimanya. Nilai atau kualitas informasi berhubungan dengan keputusan. Apabila tidak ada pilihan atau keputusan, maka informasi menjadi tidak diperlukan. Karakteristik dari nilai atau kualitas suatu informasi dapat dikukur apakah informasi tersebut memiliki kualitas atau tidak, kita dapat mengujinya dengan memperhatikan empat dimensi utama dasar informasi sebagai berikut:

1. 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.

2. 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.

3. 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.

4. 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 (SDLC)*)

Daur hidup sistem menurut Kadir (2014:244) "*System Development Life Cycle (SDLC)* merupakan metodologi klasik yang digunakan untuk mengembangkan, memelihara dan menggunakan sistem informasi". Pengertian lain menurut Rosa dan M. Shalahuddin (2016:26) "*System Development Life Cycle (SDLC)* adalah proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model dan metodologi yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya berdasarkan

cara-cara yang sudah teruji baik atau *best practice*". Daur hidup sistem terdiri menjadi beberapa tahapan *(fase)* yaitu sebagai berikut:

1. Mengenali Adanya Kebutuhan

   Sebelum segala sesuatunya terjadi, pasti terlebih dahulu adanya kebutuhan atau problema yang harus dapat dikenali sebagaimana adanya. Kebutuhan dapat terjadi sebagai hasil perkembangan suatu organisasi. Permintaan kebutuhan itu meningkat melebihi kapasitas dari sistem yang ada. Semua kebutuhan ini harus dapat didefinisikan dengan jelas, tanpa adanya kejelasan mengenai kebutuhan yang ada, pembangunan sistem akan kehilangan arah dan efektifitasnya.

2. Pembangunan Sistem

   Suatu proses atau seperangkat prosedur yang harus diikuti untuk menganalisa kebutuhan yang dibutuhkan dengan membangun suatu sistem untuk memenuhi kebutuhan tersebut.

3. Penerapan Sistem

   Setelah tahap pembangunan sistem selesai, kemudian sistem akan dioprasikan. Penerapan sistem merupakan tahap yang penting dalam daur ulang sistem, peralihan dari tahap pembangunan menuju tahap operasianal adalah penerapan sistem, yang merupakan tahap akhir dari suatu pembangunan sistem.

4. Pengoprasian Sistem

   Program-program komputer dan prosedur-prosedur pengoperasian yang membentuk suatu sistem informasi semuanya bersifat statis, sedangkan organisasi yang ditunjang oleh sistem informasi selalu mengalami perubahan karena pertumbuhan kegiatan, perubahan peraturan dan kebijaksanaan, ataupun

kemajuan teknologi. Untuk mengatasi perubahan-prubahan tersebut, sistem harus diperbaiki atau diperbaharui.

5. Sistem Menjadi Tidak Layak

   Perubahan-perubahan dengan kebutuhan yang begitu drastis sehingga tidak dapat diatasi hanya dengan melakukan perbaikan pada sistem yang sedang berjalan. Secara ekonomis dan teknis sistem yang ada sudah tidak layak lagi untuk dioperasikan dan sistem yang baru perlu dibangun untuk menggantikannya.

2.2. Konsep Dasar Program

Konsep dasar program merupakan bagian penentuan kebutuhan dari sistem informasi yang dibutuhkan. Pada umumnya program merupakan sederetan instruksi atau *statement* yang dapat dimengerti oleh komputer. Instruksi tersebut berfungsi untuk mengatur pekerjaan apa saja yang akan dilakukan oleh komputer agar mendapatkan dan menghasilkan suatu hasil atau keluaran yang diharapkan. Menurut Kadir (2014:192) memberi pengertian bahwa "Program adalah sekumpulan instruksi yang digunakan untuk mengatur perangkat keras komputer agar melaksanakan tindakan tertentu".

2.2.1. Pemograman Berorientasi Objek

Pemrograman berorientasi objek atau *Object Oriented Programming (OOP)* merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi didalam paradigma ini dibungkus dalam kelas-kelas atau

objek-objek. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya.

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. Aplikasi saat ini sangat beragam karena tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi kesemua jenis aplikasi tersebut.

Menurut Rossa dan M. Shalahuddin (2016:100) memberikan pengertian bahwa "Metodelogi berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya".

Objek merupakan entitas yang memiliki atribut, karakter *(behaviour)* dengan disertai kondisi *(state)*. Objek merepresentasikan sesuatu sistem yang nyata, seperti siswa, sensor atau mesin atau sesuatu dalam bentuk konsep seperti nasabah bank. Kelas *(Class)* merupakan penggambaran satu-kesatuan objek yang memiliki atribut dan karakter *(behaviour)* yang sama, Kelas dan objek merupakan jantung dari pemrograman berorientasi objek

2.2.2. Karakteristik Pemrograman Berorientasi Objek

Metodologi pengembangan sistem berorientasi objek Menurut Sugiarti (2013:5-6) 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 atau atribut dan metode dari induknya langsung. Atribut dan metode dari objek induk diturunkan kepada anak objek, demikian seterusnya.
   b. *Inheritance* mempunyai arti bahwa atribut dan operasi yang dimiiliki bersama diantara kelas yang mempunyai hubungan secara hirarki.
   c. Suatu kelas dapat ditentukan secara umum, kemudian ditentukan spesifik menjadi sub kelas. Setiap sub kelas 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.

2.2.3. Bahasa Pemrograman *Visual Basic 2010*

Bahasa pemrograman komputer merupakan sarana komunikasi yang menghubungkan antara manusia dengan komputer. Bahasa pemrograman komputer dikelompokan menjadi dua kelompok besar, yaitu bahasa pemrograman tingkat rendah (*Low level language)* dan bahasa pemrograman tingkat tinggi (*High level language).* Dengan bahasa pemrograman komputer, manusia dapat membuat sebuah program untuk menyelesaikan suatu masalah.

Menurut Hidayatullah (2015:5) memberikan pengertian bahwa *"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 ter*instal .NET Framework".*

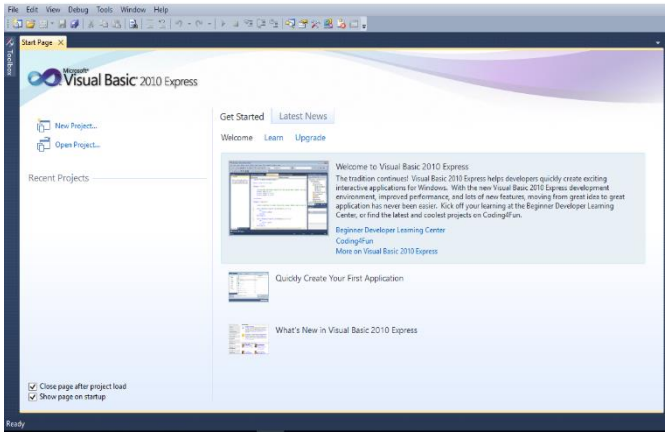
Perkembangan *visual basic.net* yaitu *visual basic.net* 2002 (*VB* 7.0), *visual basic.net* 2003 (*VB* 7.1), *visual basic* 2005 *(VB* 8.0), *visual basic* 2008 (*VB* 9.0), *visual basic* 2010 (*VB* 10.0), *visual basic* 2012 (*VB* 11.0), *visual basic* 2013 dan *visual basic* 2015. Kelebihan *visual basic.net* yaitu:

1. Sederhana dan mudah dipahami.
2. Mendukung *Graphical User Interface (GUI).*
3. Menyederhanakan program menjadi aplikasi *(deployment).*
4. Menyederhanakan pengembangan perangkat lunak.
5. Mendukung penuh *Object Oriented Programming (OOP).*
6. Mempermudah pengembangan aplikasi berbasis *Website.*
7. Migrasi ke *VB .NET* dapat dilakukan dengan mudah.
8. Banyak digunakan oleh *programmer-programmer* diseluruh dunia.

A. Komponen *Visual Basic 2010*

Komponen yang digunakan pada *visual basic* Dalam pemograman berbasiskan *Object Oriented Programming (OOP),* sebuah program dibagi menjadi bagian-bagian kecil yang disebut dengan obyek. Setiap obyek memiliki *entity* terpisah dengan *entity-entity* lain dalam lingkungannya. Obyek-obyek yang terpisah ini dapat diolah sendiri-sendiri, dan setiap obyek memiliki sekumpulan sifat dan metode yang melakukan fungsi tertentu sesuai dengan yang dibuat.

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



Sumber: Hidayatullah (2015:25)

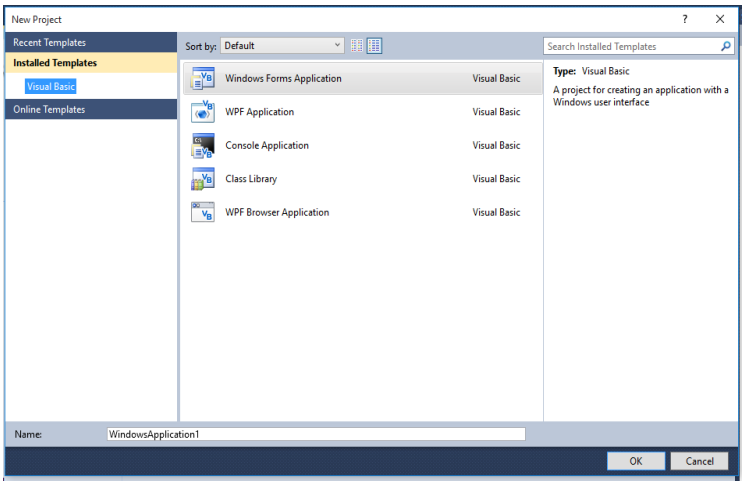
Gambar II. 1.

Tampilan awal *(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

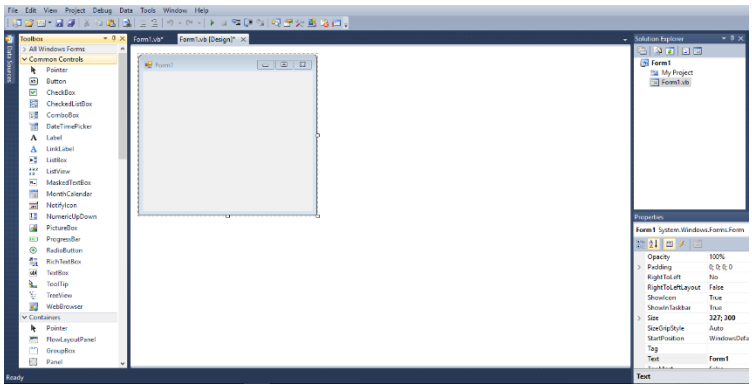
penyimpanan *file-file* proyek dan tentukan nama *solution*-nya dan tekan OK. Selanjutnya muncul *Visual Basic 2010 Integrated Development Environment (IDE)* tempat untuk membangun aplikasi *Visual Basic .NET*. Pada *IDE Visual Studio 2010* untuk *Window Aplication secara default* telah terdapat sebuah *form*, *form* tersebut bernama *Form1*.



Sumber: Hidayatullah (2015:26)

Gambar II. 2.

Kotak Dialog *New Project*



Sumber: Hidayatullah (2015:27)

Gambar II. 3.

Integrated Development Environment (IDE) Visual Studio 2010

2. *Menu bar*

Sumber: Hidayatullah (2015:27)

Gambar II. 4.

Menu Bar

Pada *Integrated Development Environment (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 *Integrated Development Environment (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 *Integrated Development Environment (IDE)* tambahan dan mengubah *Integrated Development Environment (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

1. *Toolbox*

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

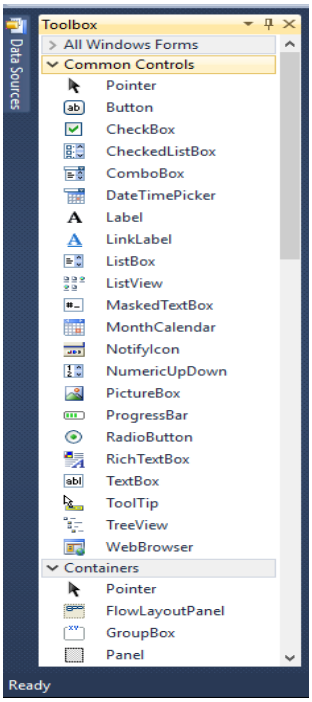
a. *Common Controls,* berisi kontrol-kontrol umum yang sering digunakan seperti *button, label, textbox* dan sebagainya.
b. *Containers,* berisi kontrol penyimpan kontrol lainya seperti *penel, group box, tabcontrol* dan sebagainya.
c. *Menus & Toolbars,* berisi kontrol dan komponen *menu, context menu* 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* dan lain-lain.

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, contohnya 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 pada kategori ini.



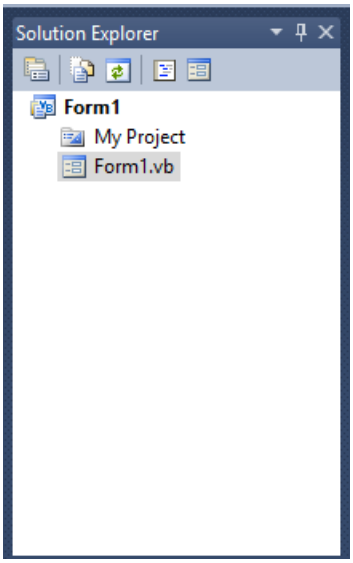
Sumber: Hidayatullah (2015:30)

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* pada *Integrated Development Environment (IDE)* , pilih menu *View* > *Solution Explorer* atau tekan *Ctrl* + *Alt* + *L*.



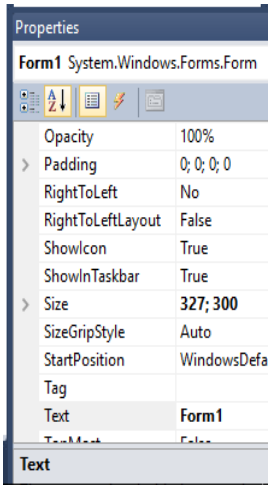
Sumber: Hidayatullah (2015:31)

Gambar II. 7.

Solution Explorer

6. *Properties Window*

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



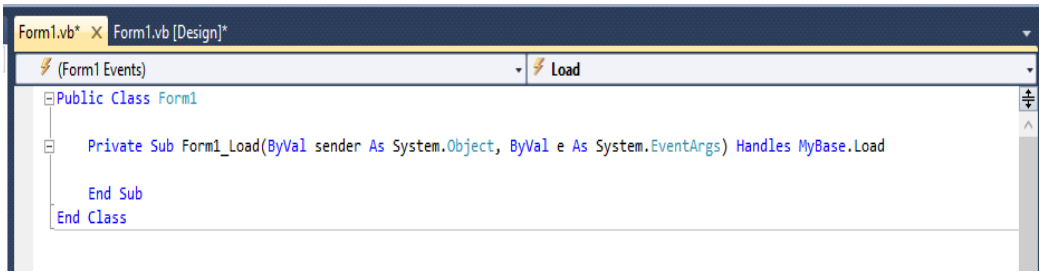
Sumber: Hidayatullah (2015:32)

Gambar II. 8.

Properties Window

7. *Code Editor*

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



Sumber: Hidayatullah (2015:33)

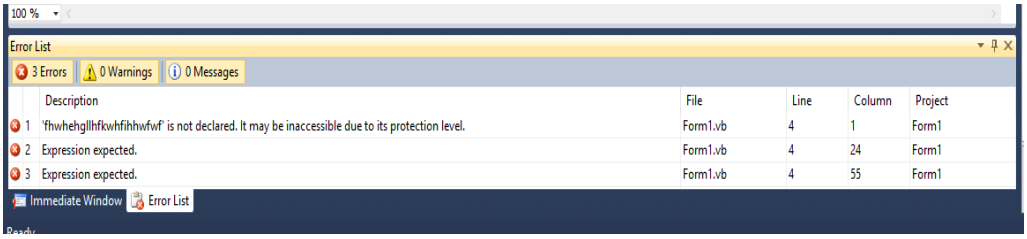
Gambar II. 9.

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

2.2.4. Basis Data *(Database)*

Pada kehidupan sehari-hari didunia komputer, basis data akan menggunakan media penyimpanan *(storage),* yaitu berhubungan 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.*

Menurut Rosa dan M. Shalahuddin (2016:43) memberikan pengertian bahwa "Basis data adalah media untuk menyimpan data agar dapat diakses dengan mudah dan cepat".

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. Perangkat Keras *(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 masukan *(input)* dan keluaran *(output).*
   c. Peralatan komunikasi data.

3. Perangkat Lunak *(Software)*

   Berfungsi sebagai perantara antara pemakai dengan data fisik pada basis data *(database)* yang terdiri dari:

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

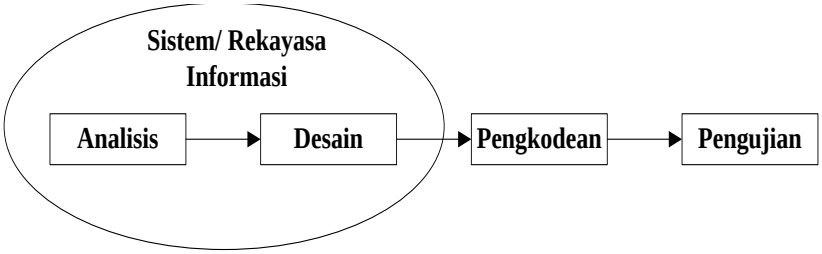
2.2.5. Aplikasi Basis Data *(Database) My Structure Query Language (MySQL)*

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 *My Structure Query Language (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), "*My Structure Query Language (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*."

2.2.6. Model Pengembangan Perangkat Lunak (*System Development Life Cycle (SDLC))*

Metode yang digunakan pada pengembangan perangkat lunak ini menggunakan model air terjun *(waterfall).* Menurut Rosa dan M. Shalahuddin (2016:28) menjelaskan bahwa "Model *System Development Life Cycle (SDLC)* air terjun (*waterfall)* sering juga disebut model sekuensial linier (*seqeuntial linear)* atau alur hidup klasik (*classic life cycle)"*. Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengodean, pengujian dan tahap pendukung (*support*).'



Sumber: Rosa dan M. Shalahuddin (2016:29)

Gambar II. 11.

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 pengguna *(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.

2.3. 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.

2.3.1. *Unified Modeling Language (UML)*

Pemodelan menggunakan *Unified Modeling Language (UML)* merupakan metode pemodelan berorientasi objek dan berbasis visual. Pemodelan menggunakan *Unified Modeling Language (UML)* merupakan pemodelan objek yang fokus pada pendefinisian struktur statis dan model sistem informasi yang dinamis daripada mendefinisikan data dan model proses yang tujuannya adalah pengembangan tradisional.

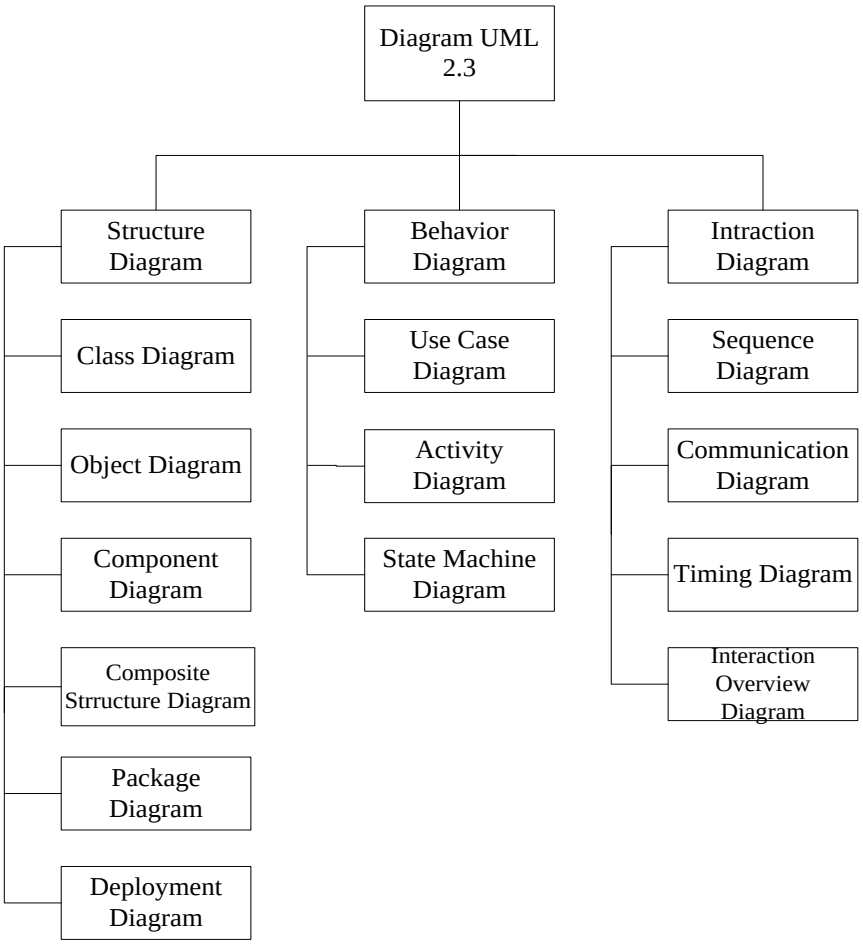
Menurut Rosa dan M. Shalahuddin (2016:133), "*Unified Modeling Language (UML)* adalah salah standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement,* membuat analisis dan *desain,* serta menggambarkan *arsitektur* dalam pemrograman berorientasi objek".

Unified Modeling Language (*UML*) dapat digunakan untuk kebutuhan sebagai berikut:

1. Menggambarkan batasan sistem dan fungsi-fungsi sistem secara umum, dibuat dengan *use case* dan *actor.*
2. Menggambarkan kegiatan atau proses bisnis yang dilaksanakan secara umum, dibuat dengan *iteraction diagrams.*
3. Menggambarkan representasi struktur statik sebuah sistem dalam bentuk *class diagrams.*
4. Membuat model *behavior* yang menggambarkan kebiasaan atau sifat sebuah sistem dengan *state transition diagrams.*
5. Menyatakan arsitektur implementasi fisik menggunakan *and development diagrams.*

A. Diagram *Unified Modeling Language* (*UML*)

Diagram Unified Modeling Language (UML) 2.3 terdiri dari tiga belas (13) macam diagram yang dikelompokan 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 *Unified Modeling Language* (*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.

B. *Use Case Diagram*

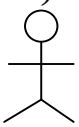
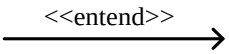
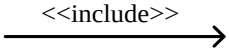
Use case diagram menurut Rosa dan Shalahuddin (2016:155) *"Use Case Diagram* merupakan pemodelan untuk kelakuan *(behavior)* sistem informasi yang akan dibuat. *Use Case* mendeskripsi sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat". *Use case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut.

Simbol-simbol yang terdapat didalam *use case diagram*, sebagai berikut:

Tabel II. 1.

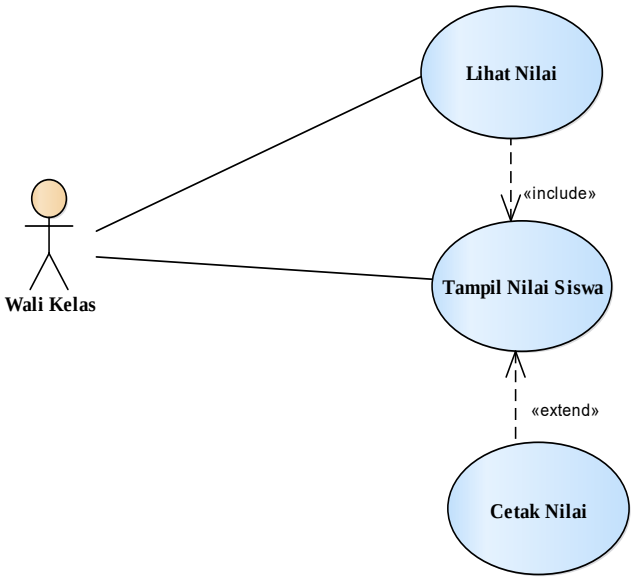
Simbol-Simbol *Use Case Diagram*

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 	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* berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan.

Sumber: Rosa dan Shalahuddin (2016:156)

Contoh dari *Use Case Diagram* sebagai berikut:



Sumber: Dermawan dan Hartini (2017)

Gambar II. 13.

Contoh *Use Case Diagram*

C. *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.

Simbol-simbol yang terdapat dalam kelas *(class)* diagram sebagai berikut:

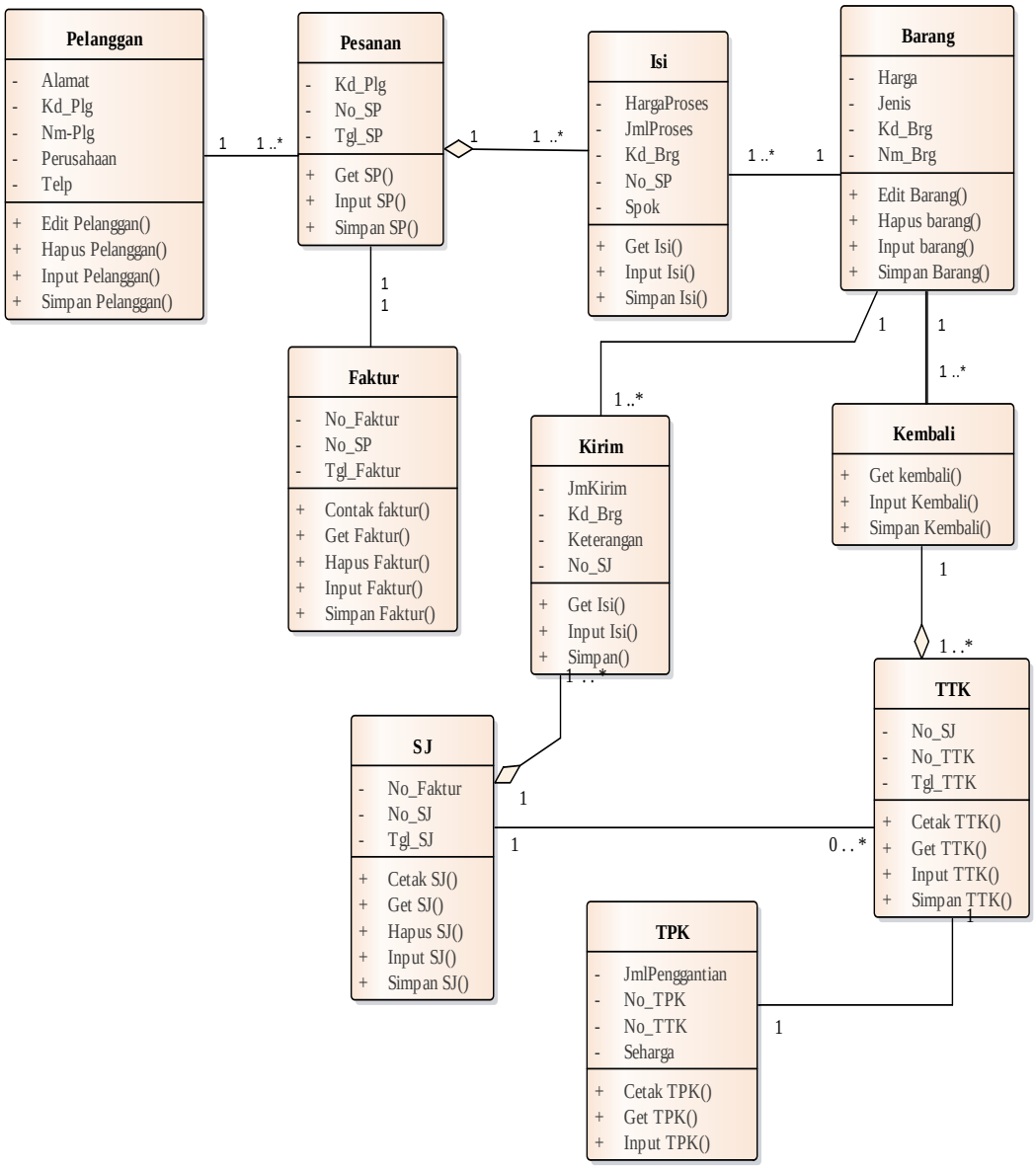
Tabel II. 2.

Simbol-Simbol Diagram Kelas *(Class)*

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 antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)
Kebergantungan *(dependency)*	Relasi antar kelas dengan makna saling kebergantungan satu sama lainnya.
Agregasi *(aggregation)*	Relasi antar kelas dengan makna semua bagian (*whole-part*)

Sumber: Rosa dan Shalahuddin (2016:146).

Contoh kelas *(class)* diagram sebagai berikut:



Sumber: Sunarti (2014)

Gambar II. 14.

Contoh Class Diagram

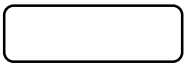
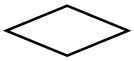
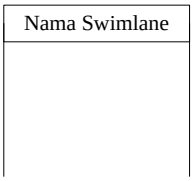
D. *Activity Diagram*

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

Simbol-simbol yang terdapat dalam *activity* diagram sebagai berikut:

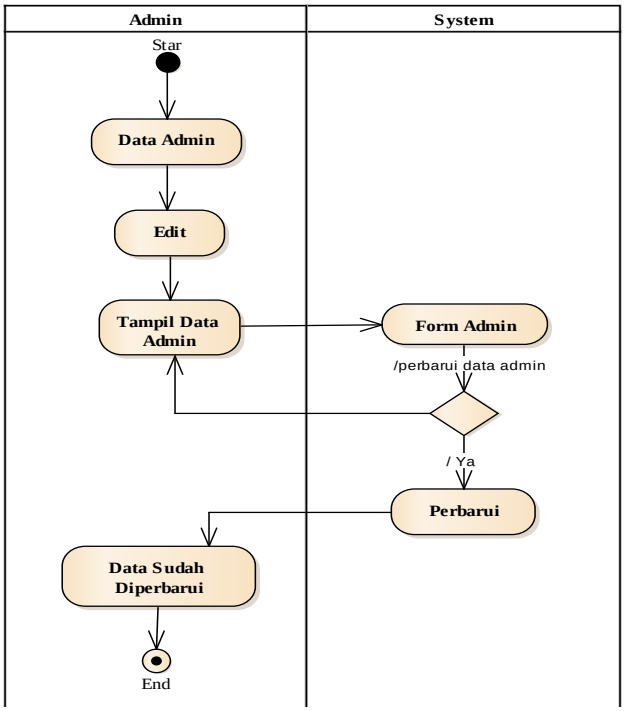
Tabel II. 3.

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)

Contoh *activity* diagram sebagai berikut:



Sumber: Dermawan dan Hartini (2017)

Gambar II. 15.

Contoh *Activity* Diagram

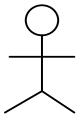
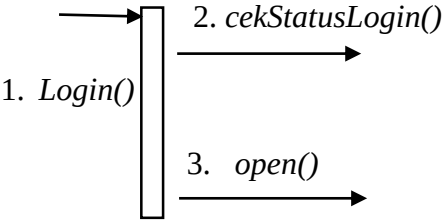
E. *Sequence Diagram*

Sequence diagram 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 *sequence* harus diketahui objek-objek yang terlibat dalam sebuah *use case* terlebih dahulu.

Simbol-simbol yang terdapat dalam *sequence* diagram sebagai berikut:

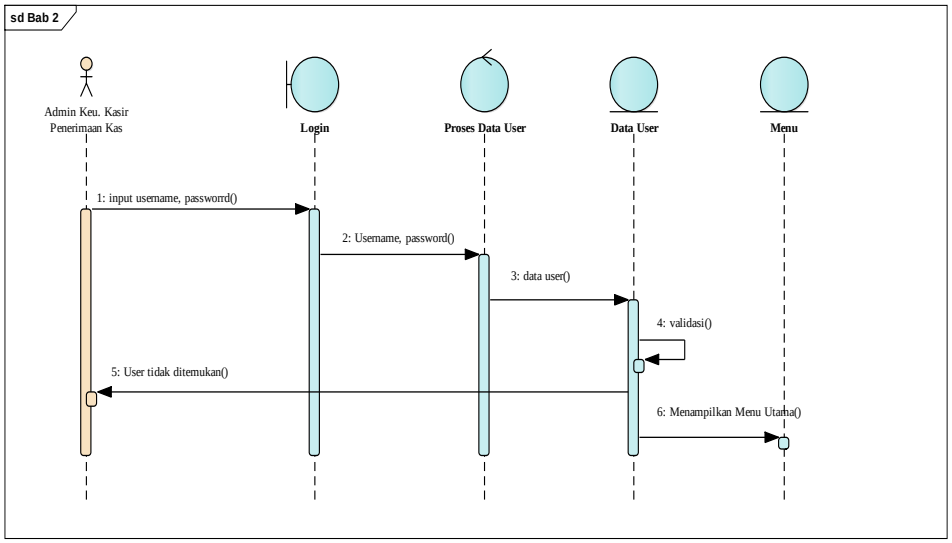
Tabel II. 4.

Simbol-simbol *Sequence* Diagram

Simbol	Deskripsi
Aktor  Nama aktor Atau Nama aktor Tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang. Tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda diawal *frase* nama aktor.
Garis hidup *(lifeline)*	Menyatakan kehidupan suatu objek.
Objek Nama objek:nama kelas	Menyatakan objek yang berinterkasi pesan.
Waktu aktif	Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukandidalamnya, misalnya  Maka *cekStatusLogin()* dan *open()* dilakukan didalam *metode login()* Aktor tidak memiliki waktu aktif.

Pesan tipe create <<create>>	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.
Pesan tipe call 1:nama_metode()	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri. 1. Nama_metode() Arah panah menagarah pada objek yang memiliki metode atau operasi, karena ini memanggil metode atau operasi maka metode atau operasi yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.

Contoh *sequence* diagram sebagai berikut:



Sumber: Syara, Chintya (2018).

Gambar II. 16.

Contoh *Sequence Diagram*

2.3.2. *Entity Relationship Diagram (ERD)*

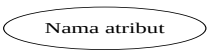
Model *Entity Relationship Diagram (ERD)* menurut Rosa dan Shalahuddin (2016:50) "*Entity Relationship Diagram* (*ERD*) adalah bentuk paling awal dalam melakukan perancangan basis data relasional. Jika menggunakan *Object Oriented Database Management System (OODMBS)* maka perancangan *Entity Relationship Diagram (ERD)* tidak perlu dilakukan".

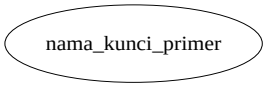
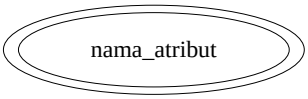
Model *Entity Relationship Diagram (ERD)* merupakan pemodelan yang menggambarkan komponen-komponen himpunan entitas dan himpunan relasi yang masing-masing dilengkapi dengan atribut-atribut yang mempresentasikan fakta atau kejadian yang terjadi pada dunia nyata.

Simbol-simbol yang terdapat dalam *Entity Relationship Diagram (ERD)* diagram sebagai berikut:

Tabel II. 5.

Simbol-Simbol *Entity Relationship Diagram (ED)*

Simbol	Deskripsi
Entitas *(entity)* 	Entitas merupakan data inti yang akan disimpan, pada basis data, benda yang memiliki data dan harus disimpan datanya agar dapat diakses oleh aplikasi computer, penamaan entitas biasanya lebih ke kata benda dan belum merupakan nama tabel
Atribut 	Kolom atau *Field* data yang akan disimpan dalam suatu entitas

Atribut Kunci Primer atau Kunci Utama *(Primary Key (PK))* 	Kolom atau *Field* data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses *record* yang diinginkan, biasanya berupa *id*, kunci primer atau kunci utama dapat lebih dari satu kolom dengan kombinasi dari kolom tersebut dapat bersifat unik yaitu berbeda tanpa ada yang sama
Atribut Multinilai *(multivalue)* 	Kolom atau *Field* 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)

Derajat kardinalitas merupakan nilai hubungan diantara entitas-entitas yang saling terhubung dengan tujuan untuk mengetahui nilai keterhubungan yang terjadi diantara entitas. Derajat kardinalitas yang terdapat dalam *Entity Relationship Diagram (ERD)* yaitu:

1. Satu ke Satu *(One to One)* (1:1)

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

   Contoh derajat kardinalitas satu ke satu *(one to one)* (1:1), sebagai berikut:

Sumber: Rosa dan Shalahuddin (2016:166)

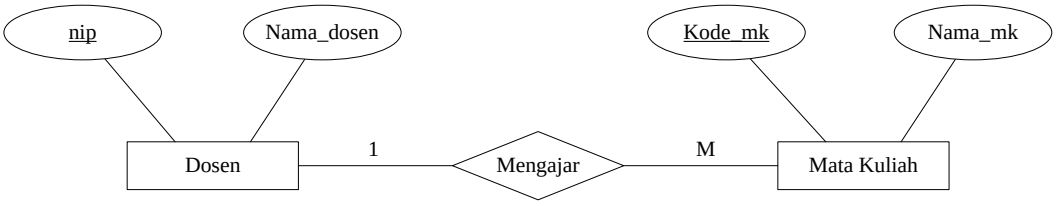
Gambar II. 17.

Kardinalitas Relasi Satu Ke Satu *(One to One)* (1:1)

2. Satu ke Banyak *(One to Many)* (1:N)

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

   Contoh derajat kardinalitas satu ke banyak *(one to many)* (1:N), sebagai berikut:



Sumber: Rosa dan Shalahuddin (2016:166)

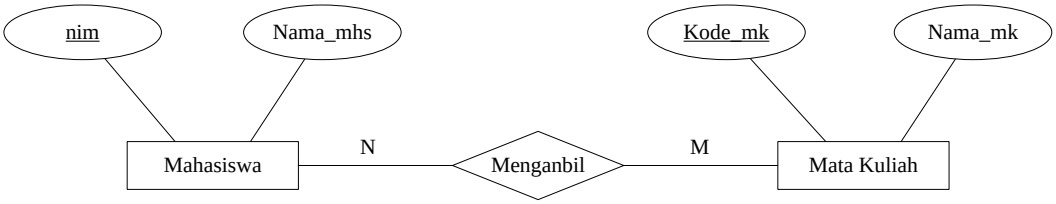
Gambar II. 18.

Kardinalitas Relasi Satu Ke Banyak *(One to Many)* (1:N)

3. Banyak ke Banyak (*Many to Many)* (N:M)

   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.

Contoh derajat kardinalitas banyak ke banyak *(many to many)* (N:M), sebagai berikut:

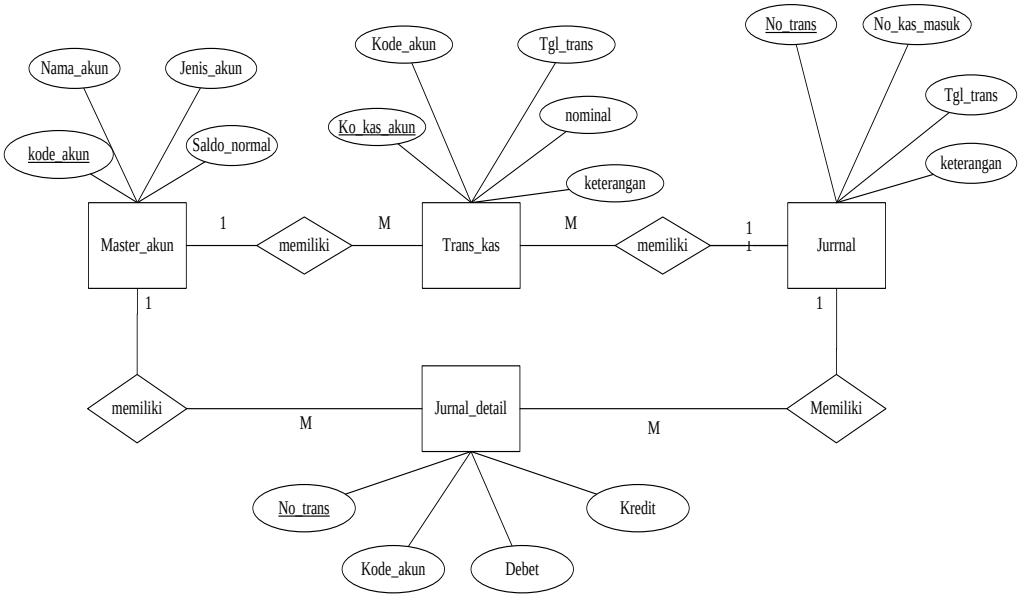


Sumber: Rosa dan Shalahuddin (2016:166)

Gambar II. 19.

Kardinalitas Relasi Banyak Ke Banyak *(Many to Many)* (N:N)

Contoh *Entity Relationship Diagram (ERD)* sebagai berikut*:*



Sumber: Syara, Chintya (2018).

Gambar II. 20.

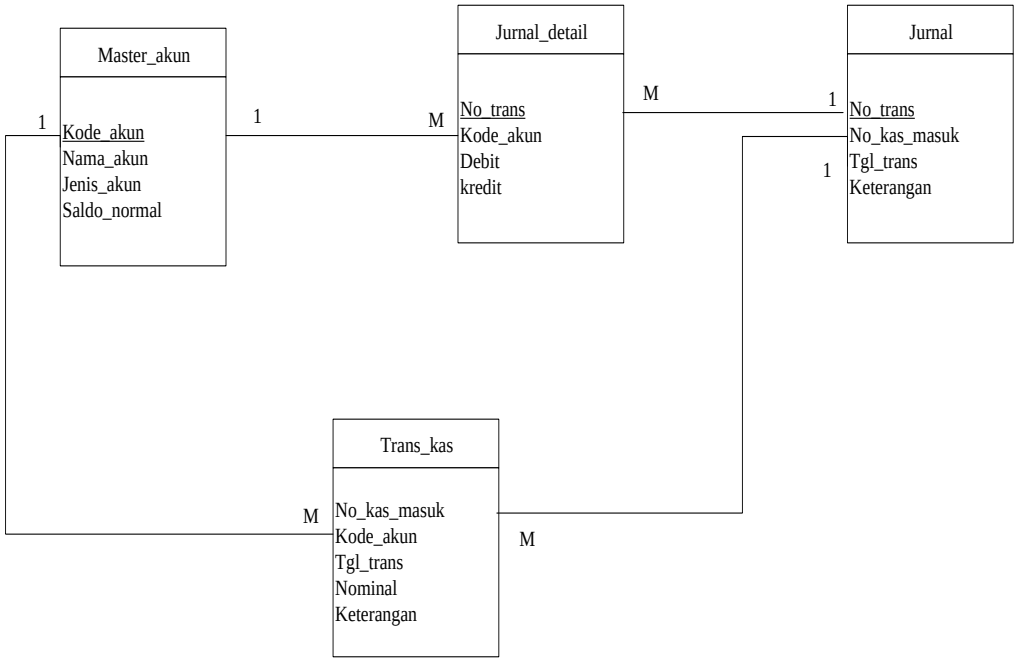
Contoh *Entity Relationship Diagram (ERD)*

2.3.3. *Logical Record Structure (LRS)*

Menurut Ladjamudin (2013:159) *Logical Record Structure (LRS)* adalah "Representasi dari *structure record-record* pada tabel-tabel yang berbentuk dari hasil antar himpunan entitas". *Logical Record Structure (LRS)* dapat menentukan kardinalitas, jumlah tabel dan *Foreign Key (FK).* Aturan-aturan dalam melakukan transformasi E-R diagram ke *Logical Record Structure (LRS)* menurut Ladjamudin (2013:159) sebagai berikut:

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 atau menjadi entitas baru dalam bentuk belah ketupat.

Contoh dari *Logical Record Structure (LRS)* sebagai berikut*:*



Sumber: Syara, Chintya (2018).

Gambar II. 21.

Contoh *Logical Record Structure (LRS)*

2.3.4. Pengkodean

Menurut 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, sebagai berikut:

1. Harus Mudah Diingat

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

2. Unik

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

3. Fleksibel

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

4. Efisien

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

5. Konsisten

   Bilamana mungkin kode konsisten dengan kode yang telah digunakan.

6. Harus distandarisasi

   Kode harus distandarisasi untuk seluruh tingkatan dan departemen dalam organisasi.

7. 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.

8. Hindarkan karakter yang mirip

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

9. 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.

2.3.5. Model Perancangan Tampilan *(Prototype)*

Model *Prototype* merupakan salah satu metode pengembangan perangkat lunak yang banyak digunakan. Dengan metode *prototyping* ini pengembang dan pelanggan dapat saling berinteraksi selama proses pembuatan sistem. Paradigma *Prototyping* dimulai dengan mengumpulkan kebutuhan. Pengembang dan pelanggan bertemu dan mendefinisikan obyektif keseluruhan sistem meliputi perangkat lunak yang akan dibuat, mengidentifikasi semua kebutuhan yang diketahui dengan ruang lingkup garis besar didefiniskan secara keseluruhan.

Menurut Kadir (2014:357) “Prototipe merupakan suatu metode dalam pengembangan sistem yang menggunakan pendekatan untuk membuat sesuatu program dengan cepat dan bertahap sehingga segera dapat dievaluasi oleh pemakai”. Prototipe dapat membantu dalam proses pembuatan pengembangan sistem informasi menjadi lebih cepat dan lebih mudah, terutama pada keadaan kebutuhan pemakai sulit untuk diidentifikasi. Dalam pembuatan prototipe ini penulis menggunakan perangkat lunak *Visual Basic 2010.*