BAB II

LANDASAN TEORI

Guna mempelajari lebih lanjut dan memudahkan pemahaman dalam penelitian perlu kiranya diadakan studi kepustakaan mengenai arti dan istilah yang digunakan dalam penelitian, sehingga memudahkan dalam pemecahan masalah dalam penelitian.

Adapun teori-teori yang penulis gunakan nantinya adalah sebagai berikut:

2.1 Sistem

Sistem adalah sekelompok elemen yang terintegrasi dengan maksud yang sama untuk mencapai suatu tujuan (McLeod, 1997)

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu (Jogiyanto, 2001).

Sistem dapat didefinisikan sebagai kumpulan elemen yang saling berhubungan satu sama lain yang membentuk satu kesatuan dalam usaha mencapai suatu tujuan (Sutedjo, 2002).

2.2 Informasi

Informasi merupakan kumpulan data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerima (Andri, 2003, hal.6).

Informasi adalah data yang sudah diolah, dibentuk, atau dimanipulasi sesuai dengan keperluan tertentu (Amzah, 1997, hal.5) Informasi dapat juga dibuat suatu prosedur, sehingga dapat diteruskan ke unit kerja yang lain.

2.3 Sistem Informasi

Sistem Informasi merupakan kumpulan dari perangkat keras dan perangkat lunak komputer serta perangkat manusia yang akan mengolah data menggunakan perangkat keras dan perangkat lunak tersebut (Andri, 2003, hal.11).

2.4 Pembayaran

Pembayaran adalah proses pertukaran mata uang atau nilai moneter untuk barang, jasa, atau informasi (Sing, 2004, hal.108)

Dapat disimpulkan bahwa pembayaran adalah perpindahan hak atas nilai antara pihak pembeli dan pihak penjual yang secara bersamaan terjadi pula perpindahan hak atas barang atau jasa secara berlawanan.

2.5 Sistem Pembayaran

Sistem pembayaran adalah suatu sistem yang mencakup seperangkat aturan, lembaga, dan mekanisme, yang digunakan untuk melaksanakan pemindahan dana guna memenuhi suatu kewajiban yang timbul dari suatu kegiatan ekonomi (BI, 2009, pasal 1).

Sistem Pembayaran merupakan sistem yang berkaitan dengan pemindahan sejumlah nilai uang dari satu pihak ke pihak lain. Media yang

digunakan untuk pemindahan nilai uang tersebut sangat beragam, mulai dari penggunaan alat pembayaran yang sederhana sampai pada penggunaan sistem yang kompleks dan melibatkan berbagai lembaga berikut aturan mainnya. Kewenangan mengatur dan menjaga kelancaran sistem pembayaran di Indonesia dilaksanakan oleh Bank Indonesia yang dituangkan dalam Undang Undang Bank Indonesia.

Dalam menjalankan mandat tersebut, Bank Indonesia mengacu pada empat prinsip kebijakan sistem pembayaran, yakni keamanan, efisiensi, kesetaraan akses dan perlindungan konsumen.

1. Aman berarti segala risiko dalam sistem pembayaran seperti risiko likuiditas, risiko kredit, risiko *fraud* harus dapat dikelola dan dimitigasi dengan baik oleh setiap penyelenggaraan sistem pembayaran.
2. Prinsip efisiensi menekankan bahwa penyelanggaran sistem pembayaran harus dapat digunakan secara luas sehingga biaya yang ditanggung masyarakat akan lebih murah karena meningkatnya skala ekonomi.
3. Kemudian prinsip kesetaraan akses yang mengandung arti bahwa Bank Indonesia tidak menginginkan adanya praktek monopoli pada penyelenggaraan suatu sistem yang dapat menghambat pemain lain untuk masuk.
4. Terakhir adalah kewajiban seluruh penyelenggara sistem pembayaran untuk memperhatikan aspek-aspek perlindungan konsumen.

Sementara itu dalam kaitannya sebagai lembaga yang melakukan pengedaran uang, kelancaran sistem pembayaran diejawantahkan dengan

terjaganya jumlah uang tunai yang beredar di masyarakat dan dalam kondisi yang layak edar atau biasa disebut *clean money policy*.

Dalam Undang-Undang (UU) No. 11/1953, sistem pembayaran dibagi menjadi dua jenis, yaitu Sistem pembayaran tunai dan Sistem pembayaran non-tunai. Perbedaan mendasar dari kedua jenis sistem pembayaran tersebut terletak pada instrumen yang digunakan. Pada sistem pembayaran tunai instrumen yang digunakan berupa uang kartal, yaitu uang dalam bentuk fisik uang kertas dan uang logam, sedangkan pada sistem pembayaran non-tunai instrumen yang digunakan berupa Alat pembayaran menggunakan kartu (APMK), Cek, Bilyet Giro, Nota *Debet*, maupun uang elektronik.

2.6 BNI

BNI yang dahulu dikenal sebagai Bank Negara Indonesia, merupakan bank pertama yang didirikan dan dimiliki oleh Pemerintah Indonesia yang berdiri sejak 1946.

Bank Negara Indonesia mulai mengedarkan alat pembayaran resmi pertama yang dikeluarkan Pemerintah Indonesia, yakni ORI atau Oeang Republik Indonesia, pada malam menjelang tanggal 30 Oktober 1946, hanya beberapa bulan sejak pembentukannya. Hingga kini, tanggal tersebut diperingati sebagai Hari Keuangan Nasional, sementara hari pendiriannya yang jatuh pada tanggal 5 Juli ditetapkan sebagai Hari Bank Nasional.

Menyusul penunjukan *De Javsche Bank* yang merupakan warisan dari Pemerintah Belanda sebagai Bank Sentral pada tahun 1949, Pemerintah

membatasi peranan Bank Negara Indonesia sebagai bank sirkulasi atau bank sentral. Bank Negara Indonesia lalu ditetapkan sebagai bank pembangunan, dan kemudian diberikan hak untuk bertindak sebagai bank devisa, dengan akses langsung untuk transaksi luar negeri. Sehubungan dengan penambahan modal pada tahun 1955, status Bank Negara Indonesia diubah menjadi bank komersial milik pemerintah. Perubahan ini melandasi pelayanan yang lebih baik dan tuas bagi sektor usaha nasional.

Sejalan dengan keputusan penggunaan tahun pendirian sebagai bagian dari identitas perusahaan, nama Bank Negara Indonesia 1946 resmi digunakan mulai akhir tahun 1968. Perubahan ini menjadikan Bank Negara Indonesia lebih dikenal sebagai 'BNI 46'. Penggunaan nama panggilan yang lebih mudah diingat - 'Bank BNI' - ditetapkan bersamaan dengan perubahaan identitas perusahaan tahun 1988.

Tahun 1992, status hukum dan nama BNI berubah menjadi PT Bank Negara Indonesia (Persero), sementara keputusan untuk menjadi perusahaan publik diwujudkan melalui penawaran saham perdana di pasar modal pada tahun 1996.

Pada tahun 2004, identitas perusahaan yang diperbaharui mulai digunakan untuk menggambarkan prospek masa depan yang lebih baik, setelah keberhasilan mengarungi masa-masa yang sulit. Sebutan 'Bank BNI' dipersingkat menjadi 'BNI', sedangkan tahun pendirian - '46' - digunakan dalam logo perusahaan untuk meneguhkan kebanggaan sebagai bank nasional pertama yang lahir pada era Negara Kesatuan Republik Indonesia.

Pada akhir tahun 2011, Pemerintah Republik Indonesia memegang 60% saham BNI, sementara 40% saham selebihnya dimiliki oleh pemegang saham publik baik individu maupun institusi, domestik dan asing.

Sumber : http://www.bni.co.id/id-id/tentangkami/sejarah.aspx.

2.7 Kriptografi *Caesar*

Kata kriptografi berasal dari bahasa Yunani, "*kryptós*" yang berarti tersembunyi dan "*gráphein*" yang berarti tulisan. Sehingga kata kriptografi dapat diartikan berupa frase "tulisan tersembunyi". Menurut *Request for Comments* (RFC), kriptografi merupakan ilmu matematika yang berhubungan dengan transformasi data untuk membuat artinya tidak dapat dipahami (untuk menyembunyikan maknanya), mencegahnya dari perubahan tanpa izin, atau mencegahnya dari penggunaan yang tidak sah. Jika transformasinya dapat dikembalikan, kriptografi juga bisa diartikan sebagai proses mengubah kembali data yang terenkripsi menjadi bentuk yang dapat dipahami. Artinya, kriptografi dapat diartikan sebagai proses untuk melindungi data dalam arti yang luas (Oppliger, 2005).

Dalam bidang ilmu kriptografi terdapat algortima yang menjadi fungsi dasarnya, yaitu :

1. **Enkripsi**, merupakan pengamanan data yang dikirimkan agar terjaga kerahasiaannya. Pesan asli disebut *plaintext*, yang diubah menjadi kode – kode yang tidak bisa dimengerti. Dalam hal ini enkripsi disebut juga dengan *cipher* atau kode.

2. **Dekripsi**, merupakan kebalikan dari enkripsi. Pesan yang telah dienkripsi dikembalikan ke bentuk asalnya (teks-asli), disebut dengan dekripsi pesan. Algoritma yang digunakan untuk dekripsi tentu berbeda dengan algoritma yang digunakan untuk dekripsi.

3. **Kunci**, yang dimaksud di sini adalah kunci yang dipakai untuk melakukan enkripsi dan dekripsi. Kunci terbagi menjadi dua bagian yaitu kunci rahasia (*private key*) dan kunci umum (*public key*).

Dalam kriptografi, sandi *Caesar*, atau sandi geser, kode *Caesar* atau Geseran *Caesar* adalah salah satu teknik enkripsi menggunakan algoritma substitusi paling awal dan paling sederhana, yang digunakan oleh raja Yunani kuno, Julius Caesar. Caranya adalah dengan mengganti setiap karakter di dalam alfabet dengan karakter yang terletak pada tiga posisi berikutnya di dalam susunan alfabet (Munir, 2006).

Hal tersebut juga dapat dilakukan dengan menggunakan susunan karakter ASCII (*American Standard Code for Information Interchange*). Banyaknya pergeseran huruf yang dilakukan merupakan kunci untuk melakukan enkripsi dan dekripsi. Misalnya pergeseran huruf adalah 4, maka kuncinya adalah $k = 4$. Bila kita mengkodekan huruf abjad dengan angka yang berurutan, yaitu: A = 0, B = 1, dan seterusnya sampai Z = 25, maka akan didapat rumus enkripsi sebagai berikut:

$$\mathbf{Ci = E(Pi) = (Pi + k) \bmod 26}$$

Sedangkan rumus untuk dekripsinya adalah sebagai berikut:

$$\mathbf{Pi = D(Ci) = (Ci - k) \bmod 26}$$

dengan *C* adalah *cipherteks* yang akan diperoleh, *P* adalah *plainteks* mula-mula, *E(Pi)* adalah fungsi untuk melakukan enkripsi, *D(Ci)* adalah fungsi untuk melakukan dekripsi, dan *k* adalah kunci yang merupakan banyaknya pergeseran huruf.

Untuk melakukan enkripsi dan dekripsi dengan menggunakan karakter ASCII yang terdiri dari 256 buah karakter, maka persamaan enkripsi dan persamaan dekripsi di atas dapat ditulis kembali menjadi:

Ci = E(Pi) = (Pi + k) mod 256

Sedangkan rumus untuk dekripsinya adalah sebagai berikut:

Pi = D(Ci) = (Ci - k) mod 256

Cipher substitusi adalah jenis cipher yang sangat sederhana sehingga amat mudah untuk dipecahkan. Cara yang paling baik adalah menggunakan metode *exhaustive key search*, yaitu mendaftarkan semua kemungkinan kunci yang ada dan memilih kunci yang ada. Dengan menggunakan metode ini semua kemungkinan kunci yang ada didaftarkan di dalam sebuah tabel kemungkinan kunci, yaitu 26 buah kunci untuk karakter abjad dan 256 buah kunci untuk karakter ASCII.

Berdasarkan teori diatas maka penulis memilih menggunakan algoritma caesar karena selain mudah diterapkan, aplikasi yang dibangun adalah aplikasi berbasis desktop dengan jumlah pengguna yang terbatas sehingga tidak memerlukan keamanan data yang ketat. Algoritma caesar yang dipilih menggunakan karakter ASCII yang terdiri dari 256 buah karakter dan juga menggunakan metode *exhaustive key search* untuk

kuncinya, yaitu menggunakan jumlah karakter *plaintexts* dibagi 2 kemudian ditambah 60 sebagai kuncinya.

Contoh :

Plaintexts = admin

Kunci = jumlah karakter *plaintexts* **div** 2 + 60 = 5 **div** 2 + 60 = 62

Plainteks tersebut akan dienkripsi menjadi *cipherteks* berikut:

Tabel 1 : *Plainteks* dienkripsi menjadi *cipherteks*

Plaintexts (Pi)	**Desimal ASCII**	**Kunci (k)**	**Rumus Ci=(Pi + k) mod 256**	**Ciphertext (Ci)**
a	97	62	159	Ÿ
d	100	62	162	¢
m	109	62	171	«
i	105	62	167	§
n	110	62	172	¬

Sedangkan rumus untuk dekripsinya adalah sebagai berikut:

Tabel 2 : *Cipherteks* dideskripsi menjadi *Plainteks*

Ciphertext (Ci)	**Desimal ASCII**	**Kunci (k)**	**Rumus Ci=(Pi - k) mod 256**	**Plaintexts (Pi)**
Ÿ	159	62	97	a
¢	162	62	100	d
«	171	62	109	m
§	167	62	105	i
¬	172	62	110	n

2.8 *Database*

Petroutsos (2002, hal.3) mendefenisikan, *Database* adalah sebuah objek yang kompleks untuk menyimpan informasi yang terstruktur, yang diorganisir dan disimpan dalam suatu cara yang mengizinkan pemakainya dapat mengambil informasi dengan cepat dan efisien.

Database sebagai kumpulan informasi disimpan dalam suatu atau tabel. Baris dalam tabel berisi satu unit data dan disebut record. Sedangkan kolom berisi atribut dari *record* dan disebut *field* Pada prinsipnya, perancangan *database* membutuhkan logika.

Banyak sekali kegiatan manusia yang menggunakan komputer sebagai sarana pengolahan data, sehingga diperlukan suatu perangkat lunak *database*. Jika dikaji lebih mendasar tentang batasan suatu *database*, maka dapat disebutkan bahwa segala bentuk koleksi data adalah suatu *database*. Mulai dari kelompok data pegawai, sampai dengan kelompok *file*, merupakan *database*.

2.8.1 Definisi dasar struktur *database* :

1. **Data** : Pengertian data menurut James A. O'Brien (2003) adalah sekumpulan fakta mengenai *objek* tertentu, orang dan lain-lain yang dinyatakan dengan angka, huruf, gambar, film, suara dan sebagainya yang relevan dan belum mempunyai arti.
2. **Informasi** : hasil pengolahan data yang konkrit dan sudah mempunyai arti untuk mencapai suatu tujuan tertentu.
3. **Tabel** : merupakan hal yang paling mendasar dalam hal penyimpanan data yang terdiri dari *field* dan *record*.
4. ***Field* (kolom)** : merupakan elemen dari tabel yang berisikan informasi tertentu yang spesifik tentang subjudul tabel pada sebuah item data.

   Syarat-syarat pembentukan *Field Name* pada tabel :

   a. Harus Unik atau Spesifik

b. Boleh disingkat

c. Pemisah sebagai pengganti spasi dalam pembentuk *field* adalah tanda lambang "_"

Contoh :

- Kode Barang menjadi KdBarang, KodeBrg, Kd_Brg, Kd_Barang, Kode_Brg
- Tanggal Lahir menjadi TglLahir, Tgl_Lahir, Tgl_Lhr

5. *Record* (baris) : merupakan sekumpulan data yang saling berkaitan tentang sebuah *subjek* tertentu, misalnya data seorang siswa akan disimpan dalam *record* yang terdiri dari beberapa kolom/*field*.

2.8.2 Komponen Dasar Dari Sistem *Database*

Menurut Connolly dan Begg (2002), terdapat 4 komponen pokok dari sistem *database* :

1. Data

a. Data disimpan secara terintegrasi (*integrated*)

Ter-*integrated* yaitu *database* merupakan kumpulan dari berbagai macam *file* dari aplikasi-aplikasi yang berbeda yang disusun dengan cara menghilangkan bagian-bagian yang rangkap (*redundant*).

b. Data dapat dipakai secara bersama-sama (*shared*)

Shared yaitu masing-masing bagian dari *database* dapat diakses oleh pemakai dalam waktu yang bersamaan, untuk aplikasi yang berbeda.

2. *Hardware* (Perangkat Keras)

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

a. Peralatan untuk penyimpanan misalnya *disk, drum, tape*.

b. Peralatan *input* dan *output*.

c. Peralatan komunikasi data, dll

3. *Software* (Perangkat Lunak)

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

a. *Database Management System* (DBMS) .

b. Program-program aplikasi & prosedur-prosedur

4. *User* (Pemakai)

Terbagi menjadi 3 klasifikasi :

a. *Database Administrator* (DBA), orang/tim yang bertugas mengelola sistem *database* secara keseluruhan.

b. *Programmer*, orang/tim membuat program aplikasi yang mengakses *database* dengan menggunakan bahasa pemprograman.

c. *End user*, orang yang mengakases *database* melalui *terminal* dengan menggunakan *query language* atau program aplikasi yang dibuat oleh *programmer.*

2.8.3 Keuntungan Pemakaian Sistem *Database*

1. Terkontrolnya kerangkapan data dan inkonsistensi

2. Terpeliharanya keselarasan data
3. Data dapat dipakai secara bersama-sama
4. Memudahkan penerapan standarisasi
5. Memudahkan penerapan batasan-batasan pengamanan.
6. Terpeliharanya intergritas data
7. Terpeliharanya keseimbangan atas perbedaan kebutuhan data dari setiap aplikasi
8. Program / data independent

2.8.4 Kerugian Pemakaian Sistem *Database*

1. Mahal dalam implementasinya
2. Rumit/komplek
3. Penanganan proses recovery & backup sulit
4. Kerusakan pada sistem basis data dapat mempengaruhi departemen yang terkait.

2.8.5 Istilah-Istilah Yang Dipergunakan Dalam Sistem *Database*

1. *Enterprise* yaitu suatu bentuk organisasi

   Contoh :

   Sekolah --> data_mhs

   Rumah sakit --> data_pasien

2. Entitas yaitu suatu obyek yang dapat dibedakan dengan objek lainnya

   Contoh :

   Bidang administrasi siswa --> entitas mahasiswa, buku pembayaran

   Bidang kesehatan --> entitas pasien, dokter, obat

3. *Atribute/field* yaitu setiap entitas mempunyai atribut atau suatu sebutan untuk mewakili suatu entitas.

   Contoh :

   Entitas siswa -> *field* = Nim, nama_siswa,alamat,dll

   Entitas nasabah --> *field* = Kd_nasabah,nama_nasabah,dll

4. *Data value* yaitu data aktual atau informasi yang disimpan pada tiap data elemen atau *atribute*.

   Contoh :

   Atribut nama_karyawan --> sutrisno, budiman, dll

5. *Record/tuple* yaitu kumpulan elemen-elemen yang saling berkaitan menginformasikan tentang suatu entitas secara lengkap.

   Contoh :

   record mahasiswa --> nim, nm_mhs, alamat.

6. *File* yaitu kumpulan *record-record* sejenis yang mempunyai panjang elemen sama, *atribute* yang sama namun berbeda-beda data *value*-nya

7. Kunci elemen data yaitu tanda pengenal yang secara unik mengindentifikasikan entitas dari suatu kumpulan entitas.

2.9 Borland Delphi

Delphi adalah sebuah bahasa pemrograman dan lingkungan pengembangan perangkat lunak. Produk ini dikembangkan oleh CodeGear sebagai divisi pengembangan perangkat lunak milik Embarcadero, divisi tersebut sebelumnya adalah milik Borland. Bahasa Delphi, atau dikenal pula

sebagai *object pascal* (pascal dengan ekstensi pemrograman berorientasi objek (PBO/OOP)) pada mulanya ditujukan hanya untuk Microsoft Windows, namun saat ini telah mampu digunakan untuk mengembangkan aplikasi untuk Linux dan Microsoft .NET *framework*. (Chandraleka, 2003)

Delphi membawa keuntungan-keuntungan berikut:

1. Dapat mengkompilasi menjadi *single executable*, memudahkan distribusi dan meminimalisir masalah yang terkait dengan *versioning*.
2. Banyaknya dukungan dari pihak ketiga terhadap VCL (biasanya tersedia berikut *source code*-nya) ataupun *tools* pendukung lainnya (dokumentasi, *tool debugging*).
3. Optimasi kompiler yang cukup cepat.
4. Mendukung *multiple platform* dari *source code* yang sama.
5. IDE (*Integrated Development Environment*) yakni lingkungan aplikasi yang didalamnya terdapat menu menu yang memudahkan kita untuk membuat suatu proyek program.
6. Mudah digunakan, *source* kode delphi yang merupakan turunan dari pascal.
7. Sifatnya *multi purphase* yakni mudah digunakan untuk mengembangkan berbagai keperluan pengembangan aplikasi

Selain memiliki keunggulan seperti yang diterangkan diatas, pemrograman Delphi juga mempunyai beberapa karateristik, yaitu :

1. Pemrograman delphi tidak *case sensitive*, artinya delphi tidak membedakan huruf besar dan huruf kecil.

2. Delphi merupakan pemrograman berorientasi *object,* artinya hampir seluruhnya merupakan *object*.
3. Pemrograman delphi merupakan pengembangan dari pemrograman bahasa pascal, sehingga bahasanya hampir mirip, tatapi memiliki kelebihan yang sangat banyak, seperti tipe data yang lebih fleksibel dan besar.
4. Setiap aplikasi yang kita buat dengan delphi akan memiliki banyak sekali *file modul* yang terpisah. Bisa kita lihat pada *clausa uses* pada setiap *unit*, misalkan dalam contoh kita kali ini, kita menggunakan *modul form, window*, dll. Tetapi dalam pengembangan yang lebih lanjut kita bisa membuat *modul* khusus untuk aplikasi kita, misalnya *modul .dll*

Pada dasarnya IDE milik Delphi dibagi menjadi enam bagian utama yaitu :

1. Menu

Gambar 2. Menu

Menu adalah sekumpulan perintah didalam menubar yang terletak di bagian atas *window* utama. Menu pada delphi ini memiliki kegunaan seperti menu pada aplikasi *windows* lainnya. Menu di dephi ini disediakan sepuluh menu, yaitu *file, edit, search, view, project, run, component, database, tools,windows* dan *help*. Masing-masing menu memiliki menu *pull down* yang berisikan perintah-perintah.

2. *Speed Bar*

Gambar 3. *Speed Bar*

Biasa juga disebut dengan *tool bar*, berisi kumpulan tombol yang tidak lain adalah beberapa menu yang sering digunakan setiap tombol pada *speed bar* menggatikan salah satu item menu.

3. *Component Palette*

Gambar 4. *Component Palette*

Merupakan kumpulan *icon* yang digunakan untuk merancang suatu aplikasi, dimana mereka akan membentuk suatu *interface* sebuah aplikasi. Dalam setiap komponen *pallete* semua *icon* dikelompokkan dalam berbagai komponen sesuai dengan fungsi dan kegunaannya. Berikut beberapa fungsi dari suatu *pallete* standar :

No	Icon	Name	Fungsi
1		Pointer	Mengembalikan fungsi mouse ke defaultnya
2		Frame	Membentuk suatu frame terhadap obyek yang ada didalamnya
3		Main menu	Membuat menu Utama
4		Popup Menus	
5	A	label	Hanya untuk menampilkan Teks
6	ab|	Edit	Untuk menampilkan dan input data (1 baris)
7		Memo	Sama seperti edit tetapi mempunyai kapasitas lebih besar (lebih dari 1 baris)
8	OK	Button	Digunakan untuk melakukan eksekusi terhadap suatu proses
9		Checkbox	Digunakan untuk menentukan pilihan lebih dari satu
10		Radio Button	Digunakan untuk menentukan pilhan, tetapi hanya satu pilhan yang bisa digunakan
11		List Box	Menampilkan pilihan dalam bentuk list
12		Combo Box	Menampilkan pilihan dalam bentuk popup
13		Scroll Bar	Merupakan icon yang berupa baris status
14		Group Box	Digunakan untuk mengelompokan suatu icon
15		Radio Group	Digunakan untuk mengelompokan pilihan

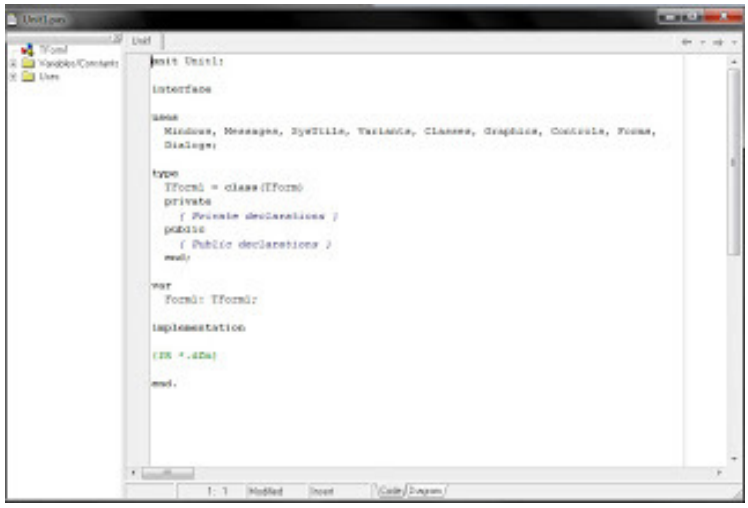
Gambar 5. Fungsi *Component Palette Standart*

4. *Form Designer*

Gambar 6. *Form Designer*

Merupakan tempat yang digunakan untuk merancang suatu tampilan aplikasi program yang dapat dimasukkan komponen-komponen *pallete*.

5. *Code Editor*



Gambar 7. *Code Editor*

Bagian dari delphi yang kita gunakan dalam penulisan kode program. disinilah kita akan menuliskan kode program kita. secara otomatis delphi akan membuat struktur dari program unit ini, seperti penulisan *clausa uses,unit, type* dll.

Untuk menampilkan *window* ini anda bisa klik dua kali pada komponen yang ingin anda isikan kodenya, misalkan pada komponen *button*. Jalan lain anda bisa menekan F12 pada *keyboard* anda. Penjelasan dari *code editor* :

unit *Unit1;* (nama unit)

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs; (*modul-modul* yang dipakai dalam program, *modul* ini telah disediakan oleh delph).

Type (mendeklarasikan *type* utama, setiap unit paling tidak terdiri dari sebuah t*ype)*

TForm1 = ***class****(TForm)*

Button1: TButton;

procedure *Button1Click(Sender: TObject);* (*procedur* dari *button* jika di klik)

Private

(Tuliskan disini *procedure*, variabel yang diperlukan yang hanya bisa diakses oleh unit ini saja)

{ Private declarations }

public

(Tuliskan disini *procedure*, variabel yang diperlukan yang bisa diakses oleh seluruh unit dalam *project*)

{ Public declarations }

end;

var

Form1: TForm1;

implementation

*{$R *.dfm}*

(mulai disinilah anda akan menuliskan kode)

procedure *TForm1.Button1Click(Sender: TObject);*

begin

(program yang akan dijalankan ketika *button* 1 di klik)

end;

end.

6. Object Inspector & Object Tree View

a. *Object Inspector* merupakan jendela yang digunakan guna mengatur tampilan komponen pada form, seperti menulisan tampilan *caption*, penamaan dari suatu komponent. *object inspector* secara umum terbagi menjadi 2 tab, Yakni :

1) *Object Properties* yang digunakan untuk mengatur tampilan dalam suatu komponen baik itu nama, warna, *font, border* dan lain sebagainya.

2) *Object Event* yang digunakan untuk memberikan fungsi yang lebih detail dari perintah-perintah *coding* yang kita buat didalam komponent, agar berjalan seperti yang kita inginkan.

Gambar 8. *Object Inspector*

b. *Object Tree View* yaitu diagram pohon yang menggambarkan hubungan logis komponen kompenen dalam *project program* meliputi *form, modul, frame* yang sesuai dengan penempatannya. *Window* ini digunakan untuk melihat komponen apa saja yang digunakan dalam *form*. Setiap komponen yang berada dalam *form* akan terlihat disini. Seperti dalam contoh terdapat sebuah komponen *button*. Jika dalam delphi anda tidak menampilkan *window* ini maka anda bisa memunculkan dengan klik *Window* | *Object TreeView* pada *menu bar*.

Gambar 9. *Object Tree View*

Dalam penelitian ini penulis menggunakan bahasa pemrograman Delphi 7 untuk pembuatan aplikasi sistem pembayaran kuliah. Aplikasi pembayaran ini dibangun berbasis *desktop* karena mempertimbangkan dari sisi jumlah pengguna dan dari sisi keamanan datanya. Alasan penulis memilih Delphi 7 sebagai bahasa pemrograman pembuatan program aplikasi sistem pembayaran ini karena Delphi 7 dapat mengkompilasi menjadi *single executable*, memudahkan distribusi dan meminimalisir masalah yang terkait dengan *versioning.* Selain itu juga karena penulis cukup menguasai bahasa ini.

2.10 MySQL

MySQL merupakan salah satu perangkat lunak sistem manajemen basis data *(database management system)* atau DBMS *(Database Management System*) yang menggunakan perintah standar SQL *(Structured Query Language).* Dimana MySQL mampu untuk melakukan banyak eksekusi perintah *query* dalam satu permintaan *(multithread)*, baik itu menerima dan mengirimkan data. MySQL juga *multi-user* dalam arti dapat dipergunakan oleh banyak pengguna dalam waktu bersamaan. (Achmad Solichin, 2010).

MySQL tersedia dalam perangkat lunak gratis dibawah lisensi GNU *General Public Lisence* (GPL). Penggunaan MySQL yang merupakan sebuah *database server* sekaligus dapat sebagai *client*, dan dapat berjalan di multi-OS *(operating system)* memiliki keunggulan lainnya. Dapat mendukung database dengan kapasitas yang sangat besar. Merupakan

database management system (DBMS) yang mudah digunakan. Didukung oleh *driver* ODBC, sehingga *database* MySQL dapat diakses oleh aplikasi apa saja. Bahasa pemrograman yang dapat digunakan untuk mengakses MySQL diantaranya adalah dengan C, C++, Java, Perl, PHP, Phyton, dan APIs.

2.10.1 Keunggulam MySQL

Mysql adalah suatu database popular dengan pengembag *Web* (*Web developers*). Kecepatan dan ukuran yang kecil membuatnya ideal untuk *website*. Ditambah lagi dengan fakta bahwa MySQL adalah *open source*, yang berarti gratis. (Achmad Solichin, 2010).

Dibawah ini adalah beberapa keuntungannya :

- **Cepat**. Tujuan utama dari pengembangan MySQL adalah kecepatan, sebagai konsekuensi *software* yang dirancang dari awal untuk kecepatan.
- **Tidak mahal**. MySQL adalah cuma-cuma di bawah lisensi GPL open source, sementara pembiayaan untuk lisensi komersialnya sangatlah pantas.
- **Mudah digunakan**. Kita pun dapat membangun dan berinteraksi dengan *database* MySQL hanya dengan menggunakan sedikit pernyataan (*statement*) sederhana di dalam bahasa SQL, yang menjadi bahasa standar untuk komunikasi dengan RDBMS.
- **Dapat berjalan pada beberapa sistem operasi**. MySQL berjalan pada sistem operasi yang beragam, seperti Windows,

Linux, Mac OS, kebanyakan versi Unix (termasuk Solaris, AIX, dan EDC Unix), FreeBSD, OS/2, Irix, dan lainnya.

- **Dukungan teknis secara luas tersedia**. MySQL menyediakan dukungan cuma-cuma untuk pengguna *via mailing list*. Pengembang MySQL juga berpartisipasi di dalam *e-mail list*. Anda juga dapat membeli dukungan teknis dari MySQL AB.
- **Aman**. MySQL adalah sistem otorisasi fleksibel yang mengijinkan beberapa atau semua *privilege database* (sebagai contoh, *privilege* untuk menciptakan suatu *database* atau menghapus data) untuk pengguna khusus atau kelompok pengguna.
- **Mendukung *database* yang besar**. MySQL menangani *database* sampai 50 juta baris atau lebih. Batas ukuran file secara *default* untuk tabel adalah 4 GB, tetapi dapat dinaikkan (jika sistem operasi dapat menanganinya) hingga 8 juta *terabytes* (TB).
- ***Customizable***. Lisensi GPL *open source* mengijinkan pemrogram untuk memodifikasi *software* MySQL untuk mencocokkannya dengan lingkungan tertentu.

.

2.10.2 Perintah MySQL

Menurut Achmad Solichin (2010), beberapa perintah sql yang didukung oleh mysql adalah :

- *SELECT*

  Untuk melihat data dari satu atau beberapa tabel.

select kolom-kolom

from nama-tabel

- *INSERT INTO*

  Untuk mengisi data pada suatu tabel atau menambah *record* pada tabel.

  insert into nama-tabel (kolom1, kolom2,…)

  values (nilai1, nilai2,…);

- *DISTINCT*

  Untuk menghilangkan *record–record* yang sama.

  select distinct kolom from nama-tabel;

- *SELECT **

  Untuk melihat isi kolom suatu tabel.

  *select * from nama-tabel;*

- *WHERE*

  Untuk menyaring/ membatasi hasil *query* sehingga *record* yang dikeluarkan hanya *record* yang sesuai kriteria yang diinginkan.

  select kolom, kolom ,..from nama-tabel

  where criteria;

- *BETWEEN*

Untuk membatasi suatu kolom berada pada suatu baris nilai tertentu.

select kolom, kolom,… from nama-tabel

where criteria(salah satu kolom sebagai parameter) between .. and..;

- *LIKE*

  Untuk mencari data yang memiliki pola tertentu.

  Select kolom, kolom,..from nama-tabel

  where criteria(salah satu kolom) like '%win%';

- *ORDER BY*

  Untuk mensortir data atau hasil *query*

  select kolom, kolom from nama-tabel

  order by kolom;

- *DESC*

  Untuk mensortir data dengan urutan terbalik

  select kolom, kolom from nama-tabel

  order by kolom desc;

- *DELETE*

  Untuk menghapus *record*

  delete from nama-tabel;

Dengan kriteria tertentu,

delete from nama-tabel where criteria;

- *UPDATE*

  Untuk memodifikasi *record* nilai kolom secara keseluruhan.

  update kolom set (kolom sbg parameter)= 100;

  Untuk memodifikasi nilai kolom dari suatu *record*

  update nama-tabel

  set nama-kolom1=nilai-baru1,

  nama-kolom2=nilai- baru2,…

  where criteria;

Dalam penelitian ini penulis menggunakan database MySQL untuk menyimpan data dan menggunakan perintah-perintah sql untuk proses pengolahan data di database. Alasan penulis menggunakan MySQL sebagai sarana penyimpanan data karena MySQL aman, murah dan mudah digunakan.

2.11 *White Box Testing* Dan *Black Box Testing*

Menurut Roger S Pressman dalam bukunya Rekayasa Perangkat Lunak : Cara pengetesan / pengujian sebuah *software* dapat dibagi menjadi 2 cara yaitu *White Box Testing* dan *Blackbox Testing*. Pengertian dari kedua metode tersebut adalah :

2.11.1 *Whitebox Testing*

White Box Testing merupakan cara pengujian dengan melihat ke dalam modul untuk meneliti kode-kode program yang ada, dan menganalisis apakah ada kesalahan atau tidak. Jika ada modul yang menghasilkan *output* yang tidak sesuai dengan proses bisnis yang dilakukan, maka baris-baris program, variabel, dan parameter yang terlibat pada unit tersebut akan dicek satu persatu dan diperbaiki, kemudian di-*compile* ulang.

Dengan menggunakan *white box* akan menguji semua keputusan *logical*, seluruh *Loop* yang sesuai dengan batasannya dan menguji seluruh struktur data *internal* yang menjamin validitasi.

Kelebihan *White Box Testing*

- Kesalahan Logika

  Digunakan pada sintaks '***if***' dan pengulangan. Dimana *White Box Testing* akan mendeteksi kondisi-kondisi yang tidak sesuai dan mendeteksi kapan proses pengulangan akan berhenti.

- Ketidaksesuaian asumsi

  Menampilkan asumsi yang tidak sesuai dengan kenyataan, untuk di analisa dan diperbaiki.

- Kesalahan ketik

  Mendeteksi bahasa pemrograman yang bersifat *case sensitive*.

Kelemahan *White Box Testing*

Untuk perangkat lunak yang tergolong besar, *White Box Testing* dianggap sebagai strategi yang tergolong boros, karena akan melibatkan sumber daya yang besar untuk melakukannya.

2.11.2 *Blackbox Testing*

Blackbox testing adalah metode pengujian perangkat lunak yang tes fungsionalitas dari aplikasi yang bertentangan dengan struktur internal atau kerja (lihat pengujian *white-box*). Pengetahuan khusus dari kode aplikasi / struktur internal dan pengetahuan pemrograman pada umumnya tidak diperlukan. Uji kasus dibangun di sekitar spesifikasi dan persyaratan, yakni, aplikasi apa yang seharusnya dilakukan.

Menggunakan deskripsi *eksternal* perangkat lunak, termasuk spesifikasi, persyaratan, dan desain untuk menurunkan uji kasus. Tes ini dapat menjadi fungsional atau non-fungsional, meskipun biasanya fungsional. Perancang uji memilih *input* yang *valid* dan tidak *valid* dan menentukan output yang benar. Tidak ada pengetahuan tentang struktur internal benda uji itu.

Metode uji dapat diterapkan pada semua tingkat pengujian perangkat lunak: unit, integrasi, fungsional, sistem dan penerimaan. Ini biasanya terdiri dari kebanyakan jika tidak semua pengujian pada tingkat yang lebih tinggi, tetapi juga bisa mendominasi unit *testing* juga.

Metode ujicoba *blackbox* memfokuskan pada keperluan fungsional dari *software*. Karna itu ujicoba *blackbox* memungkinkan pengembang software untuk membuat himpunan kondisi input yang akan melatih seluruh syarat-syarat fungsional suatu program. Ujicoba *blackbox* bukan merupakan alternatif dari ujicoba *whitebox*, tetapi merupakan pendekatan yang melengkapi untuk menemukan kesalahan lainnya, selain menggunakan metode *whitebox.* Ujicoba *blackbox* berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya :

1. Fungsi-fungsi yang salah atau hilang.
2. Kesalahan interface.
3. Kesalahan dalam struktur data atau akses database eksternal.
4. Kesalahan performa.
5. Kesalahan inisialisasi dan terminasi.

2.11.3 Perbedaan White Box & Black Box

1. Metode *White box* (Struktural)
    - Dilakukan oleh penguji yang mengetahui tentang QA.
    - Melakukan testing pada software/program aplikasi menyangkut security dan performance program tersebut (meliputi tes code, desain implementasi, security, data flow, software failure).
    - Dilakukan seiring dengan tahapan pengembangan software atau pada tahap testing.

2. Metode *BlackBox* (Fungsional)

   - Dilakukan oleh penguji Independent.
   - Melakukan pengujian berdasarkan apa yang dilihat, hanya fokus terhadap fungsionalitas dan output. Pengujian lebih ditujukan pada desain software sesuai standar dan reaksi apabila terdapat celah-celah bug/vulnerabilitas pada program aplikasi tersebut setelah dilakukan white box testing.
   - Dilakukan setelah white box testing