BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terdahulu

Penelitian yang dilakukan oleh Diah Ayu Puspitaningrum dan Ajib Susanto yang berjudul "**Implementasi Algoritma Vigenere, Caesar, Dan Affine Cipher Pada Database Sistem Inventori Toko Wiwin Elektronik Grobogan**", penelitian tersebut telah menghasilkan sebuah aplikasi sistem *inventory* yang mengutamakan keamanan data. Kelebihan dari sistem ini adalah menggunakan tiga macam algoritma sehingga pertahanan keamanan dalam database tersebut akan menjadi lebih ketat. Didalam sistem ini juga menggunakan 4 buah kunci sehingga sulit untuk dipecahkan.

Penelitian yang dilakukan oleh Hendra Agusvianto yang berjudul "**Sistem Informasi Inventori Gudang Untuk Mengontrol Persediaan Barang Pada Gudang Studi Kasus: PT. Alaisys Sidoarjo**", penelitian diatas menghasilkan sebuah aplikasi sistem *inventory* gudang yang dapat mengontrol persediaan barang pada gudang. Kelebihan dari sistem ini yaitu membuat pihak kantor pusat gudang mengetahui data barang dengan tepat dan efisien.

Penelitian yang dilakukan oleh Putu H. Arjana, Tri Puji Rahayu, Yakub, dan Hariyanto "**Implementasi Enkripsi Data Dengan Algoritma Vigenere Cipher**", penelitian ini menghasilkan sistem keamanan yang dapat mengamankan data perusahaan khususnya data pelanggan. Kelebihan sistem ini yaitu dapat

mengamankan identitas pelanggan termasuk harga dan *discount* yang diberikan untuk pelanggan tersebut.

Penelitian yang dilakukan oleh Sasono Wibowo, Florentina Esti Nilawati, dan Suharnawi "**Implementasi Enkripsi Dekripsi Algoritma Affine Cipher Berbasis Android**", penelitian tersebut menghasilkan sistem keamanan dalam bidang komunikasi berbasis *Android*. Kelebihan sistem ini adalah pesan yang dikirimkan akan terenkripsi sehingga komunikasi yang bersifat privasi tidak akan diketahui oleh orang lain.

Berdasarkan hasil penelitian yang terdahulu, maka dibuatlah kesimpulan untuk merancang sebuah sistem keamanan untuk suatu sistem pendataan gudang. Sehingga pada penulisan skripsi ini dibuatlah sebuah judul **"Rancang Bangun Keamanan Ssitem Pendataan Gudang Berbasis *Client Server* Menggunakan Algoritma Atbash, Vigenere, dan Affine Cipher".** Berdasarkan judul tersebut nantinya akan dihasilkan sebuah aplikasi untuk menginput data barang masuk berbasis *client server* yang memiliki keamanan.

II.2. Aplikasi

Pengertian aplikasi adalah penggunaan atau penerapan suatu konsep yang menjadi pokok pembahasan. Aplikasi dapat diartikan juga sebagai program komputer yang di buat untuk menolong manusia dalam melaksanakan tugas tertentu. Aplikasi *software* yang dirancang untuk penggunaan praktisi khusus, klasifikasi luas ini dapat dibagi menjadi 2 (dua) yaitu:

1. Aplikasi *software* spesialis, program dengan dokumentasi tergabung yang dirancang untuk menjalankan tugas tertentu.

2. Aplikasi paket, suatu program dengan dokumentasi tergabung yang dirancang untuk jenis masalah tertentu. (Rudi Hartono;2015).

II.3. Rancang Bangun

Rancang merupakan serangkaian prosedur untuk menerjemahkan hasil analisa dari sebuah sistem kedalam bahasa pemrograman untuk mendeskripsikan dengan detail bagaimana komponen-komponen sistem diimplementasikan (Pressman, 2002). Rancangan sistem adalah penentuan proses dan data yang diperlukan oleh sistem baru (McLeod, 2002). Perancangan adalah kegiatan yang memiliki tujuan untuk mendesain sistem baru yang dapat menyelesaikan masalah-masalah yang dihadapi perusahaan yang diperoleh dari pemilihan alternatif sistem yang terbaik (Ladjamudin, 2005). Sedangkan pengertian bangun atau pembangunan sistem adalah kegiatan menciptakan sistem baru maupun mengganti atau memperbaiki sistem yang telah ada baik secara keseluruhan maupun sebagian (Pressman, 2002). Bangun sistem adalah membangun sistem informasi dan komponen yang didasarkan pada spesifikasi desain (Whitten et al, 2004).

Dengan demikian pengertian rancang bangun merupakan kegiatan menerjemahkan hasil analisa ke dalam bentuk paket perangkat lunak kemudian menciptakan sistem tersebut ataupun memperbaiki sistem yang sudah ada (Indah Permata Sari, 2013).

II.4. Keamanan Sistem Informasi

Masalah keamanan merupakan salah satu aspek penting dari sebuah sistem informasi. Sayang sekali masalah keamanan ini sering kali kurang mendapat perhatian dari para pemilik dan pengelola sistem informasi. Sering kali masalah keamanan berada di urutan kedua, atau bahkan diurutan terakhir dalam

daftar hal-hal yang dianggap penting. Apa bila menggangu performansi dari sistem, biasanya sistem keamanan akan dikurangi atau ditiadakan.

Informasi saat ini sudah menjadi sebuah komoditi yang sangat penting. Kemampuan untuk mengakses dan menyediakan informasi secara cepat dan akurat menjadi sangat esensial bagi sebuah organisasi, baik yang berupa organisasi komersial (perusahaan), perguruan tinggi, lembaga pemerintahan, maupun individual. Hal ini dimungkinkan dengan perkembangan pesat di bidang teknologi komputer dan telekomunikasi.

Sangat pentingnya nilai sebuah informasi menyebabkan sering kali informasi diinginkan hanya boleh diakses oleh orang-orang tertentu. Jatuhnya informasi ke tangan pihak lain (misalnya pihak lawan bisnis) dapat menimbulkan kerugian bagi pemilik informasi. Sebagai contoh, banyak informasi dalam sebuah perusahaan yang hanya diperbolehkan diketahui oleh orang-orang tertentu di dalam perusahaan tersebut, seperti misalnya informasi tentang produk yang sedang dalam development, algoritma-algoritma dan teknik-teknik yang digunakan untuk menghasilkan produk tersebut. Untuk itu keamanan dari sistem informasi yang digunakan harus terjamin dalam batas yang dapat diterima.

Menurut G. J. Simons, keamanan informasi adalah bagaimana kita dapat mencegah penipuan (*cheating*) atau paling tidak, mendeteksi adanya penipuan di sebuah sistem yang berbasis informasi, dimana informasinya sendiri tidak memiliki arti fisik.

Selain itu keamanan sistem informasi bisa diartikan sebagai kebijakan, prosedur, dan pengukuran teknis yang digunakan untuk mencegah akses yang tidak sah, perubahan program, pencurian, atau kerusakan fisik terhadap sistem

informasi. Sistem pengamanan terhadap teknologi informasi dapat ditingkatkan dengan menggunakan teknik-teknik dan peralatan-peralatan untuk mengamankan perangkat keras dan lunak komputer, jaringan komunikasi, dan data (Christina Septi, 2012).

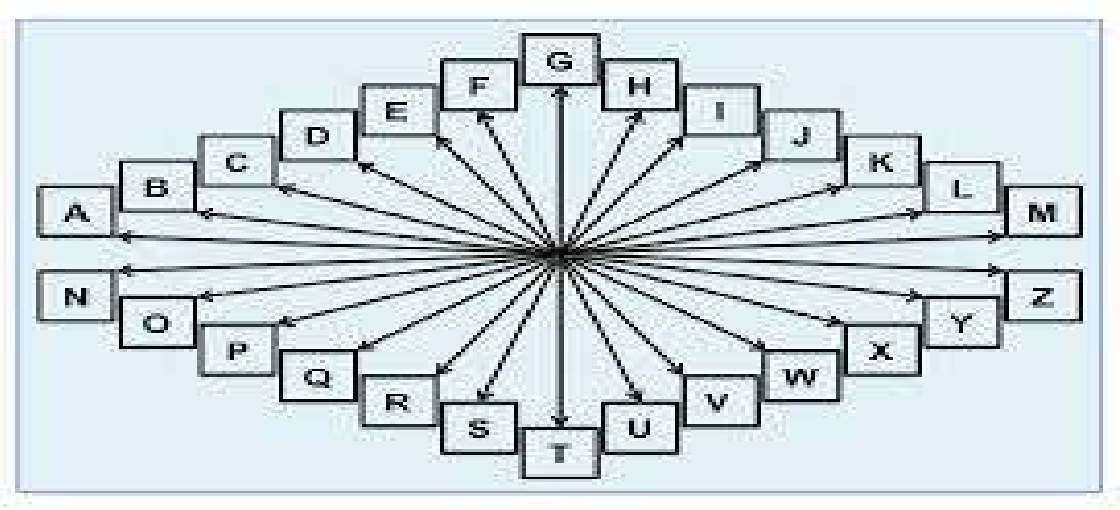
II.5. Kriptografi

Menurut Andi Riski Alvianto (2015), Kriptografi berasal dari bahasa Yunani, *"kryptós"* yang berarti tersembunyi dan *"gráphein"* yang berarti tulisan. Sehingga kata kriptografi dapat diartikan berupa frase "tulisan tersembunyi". Kriptografi adalah ilmu yang digunakan untuk mengubah data agar tidak dapat dipahami arti dan maknanya oleh orang lain dan untuk memberi keamanan pada suatu data agar tidak di modifikasi tanpa izin.

Adapun metode-metode kriptografi yang akan digunakan pada skripsi ini adalah sebagai berikut.

II.5.1. *Atbash Cipher*

Atbash Cipher merupakan suatu teknik enkripsi, dimana huruf alphabet disubstitusi dengan kebalikannya. Maksud dari kebalikannya adalah huruf awal diganti dengan huruf terakhir, huruf kedua diganti dengan sebelum terakhir, dan seterusnya. Dapat dilihat seperti gambar *Enkripsi Atbash* II.1.



Gambar II.1. Contoh Gambaran Enkripsi *Atbash*

Sumber : Geniones (2013), "Atbash Cipher"

Contoh lebih jelasnya:

Plain : ABCDEFGHIJKLMNOPQRSTUVWXYZ

Cipher : ZYXWVUTSRQPONMLKJIHGFEDCBA

II.5.2. *Vigenere Cipher*

Sandi *Vigenere* adalah metode menyandikan teks alfabet dengan menggunakan deretan sandi *Caesar* berdasarkan huruf-huruf pada kata kunci. Sandi *Vigenère* merupakan bentuk sederhana dari sandi substitusi polialfabetik. Kelebihan sandi ini dibanding sandi *Caesar* dan sandi monoalfabetik lainnya adalah sandi ini tidak begitu rentan terhadap metode pemecahan sandi yang disebut analisis frekuensi. Sandi *Vigenère* sebenarnya merupakan pengembangan dari sandi *Caesar*. Pada sandi *Caesar*, setiap huruf teks yang disandikan digantikan dengan huruf lain yang memiliki perbedaan tertentu pada urutan alfabet. Teknik substitusi *Vigenere* dengan menggunakan angka dilakukan dengan

menukarkan huruf dengan angka, hampir sama dengan kode geser. Contohnya seperti berikut:

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Gambar II.2. Contoh Tabel Substitusi Kriptografi *Vigenere Cipher*

Sumber : Rahman Hidayat (2013),"Algoritma Kriptografi Klasik: Vigenere Cipher"

Rumus enkripsi *Vigenere Cipher* :

Ci =(Pi+Ki) mod 26 ..... (1)

kalau jumlah di bawah 26

atau

Ci =(Pi+Ki)-26 ..... (2)

kalau hasil jumlah di atas 26.

Rumus dekripsi *Vigenere Cipher* :

Pi=(Ci-Ki) mod 26 ..... (3)

kalau hasilnya positif

atau

Pi=(Ci-Ki)+26 ..... (4)

kalau hasil pengurangannya minus.

Ada metode lain juga untuk melakukan proses enkripsi dengan metode *Vigenere Cipher* yaitu menggunakan *Tabula Recta* (disebut juga bujur sangkar *Vigenere*) seperti gambar berikut:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
a	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
b	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
c	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
d	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
e	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
f	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
g	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
h	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
i	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
j	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
k	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
l	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
m	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
n	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
o	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
p	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
r	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
s	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
t	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
u	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
v	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
w	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
x	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Gambar II.3. Tabel Enkripsi Kriptografi *Vigenere Cipher (Tabula Recta)*

Sumber : Rahman Hidayat (2013),"Algoritma Kriptografi Klasik: Vigenere Cipher"

II.5.3. *Affine Cipher*

Metode *Affine Cipher* adalah perluasan dari metode *Caesar Cipher*, yang mengalikan plainteks dengan sebuah nilai *P* dan menambahkannya dengan sebuah pergeseran *b* menghasilkan cipherteks *C* dinyatakan dengan fungsi kongruen:

$$C \equiv m\,P + b \pmod{n} \quad \ldots\ldots \quad (5)$$

Yang mana *n* adalah ukuran alphabet, *m* adalah bilangan bulat yang harus relatif prima dengan *n* (jika tidak relatif prima, maka dekripsi tidak bisa dilakukan) dan *b* adalah jumlah pergeseran (*Caesar Cipher* adalah bentuk khusus dari *Affine Cipher* dengan *m*=1). Untuk melakukan deskripsi, persamaan (2.3) harus dipecahkan untuk memperoleh *P*. Solusi kekongruenan tersebut hanya ada jika inver *m (mod n)*, dinyatakan dengan m^{-1}. Jika m^{-1} ada maka dekripsi dilakukan dengan persamaan sebagai berikut: (Munir, 2006)

$$P \equiv m^{-1}(C - b) \pmod{n} \quad \ldots\ldots \quad (6)$$

Contoh:

Misalkan plainteks

G I L D A

Yang ekuivalen dengan:

6 8 11 3 0 (dengan memisalkan ‘A’ = 0, ‘B’ = 1 dst)

Dienkripsi dengan *Affine Cipher* dengan mengambil *m* = 7 (karena 7 relatif prima dengan 26) dan *b* = 10. Karena alphabet yang digunakan 26 huruf, maka *n* = 26.

Enkripsi *plaintext* dihitung dengan kekongruenan:

$$C \equiv 7P + 10 \pmod{26}$$

Perhitungannya adalah sebagai berikut:

$P_1 = 6$ → $C_1 \equiv 7 .6 + 10 \equiv 52 \equiv 0 \pmod{26}$ (huruf 'A')

$P_2 = 8$ → $C_2 \equiv 7 .8 + 10 \equiv 66 \equiv 14 \pmod{26}$ (huruf 'O')

$P_3 = 11$ → $C_3 \equiv 7 . 11 + 10 \equiv 87 \equiv 9 \pmod{26}$ (huruf 'I')

$P_4 = 3$ → $C_4 \equiv 7 .3 + 10 \equiv 31 \equiv 5 \pmod{26}$ (huruf 'F')

$P_5 = 0$ → $C_5 \equiv 7 .0 + 10 \equiv 10 \equiv 10 \pmod{26}$ (huruf 'K')

Ciphertext yang dihasilkan adalah:

AOIFK

Untuk melakukan dekripsi, pertama-tama dihitung $7^{-1} \pmod{26}$, yang dapat dihitung dengan memecahkan kekongruenan lanjar:

$7x \equiv 1 \pmod{26}$

Solusinya adalah $x \equiv 15 \pmod{26}$ sebab $7.15 = 105 \equiv 1 \pmod{26}$. Jadi, untuk dekripsi digunakan kekongruenan:

$P \equiv 15(C - 10) \pmod{26}$

Perhitungannya adalah sebagai berikut:

$C_1 = 0$ → $P_1 \equiv 15 . (0 - 10) = -150 \equiv \pmod{26}$ (huruf 'G')

$C_2 = 14$ → $P_2 \equiv 15 . (14 - 10) = 60 \equiv 8 \pmod{26}$ (huruf 'I')

$C_3 = 9$ $\rightarrow P_3 \equiv 15 . (9 - 10) = -15 \equiv 11 \pmod{26}$

(huruf 'L')

$C_4 = 5$ $\rightarrow P_4 \equiv 15 . (5 - 10) = -75 \equiv 3 \pmod{26}$

(huruf 'D')

$C_5 = 10$ $\rightarrow P_5 \equiv 15 . (10 - 10) = 0 \equiv 0 \pmod{26}$

(huruf 'A')

Plaintext yang dihasilkan adalah

G I L D A

II.6. *Client Server*

Client-server adalah suatu bentuk arsitektur, dimana *client* adalah perangkat yang menerima yang akan menampilkan dan menjalankan aplikasi (*software* komputer) dan *server* adalah perangkat yang menyediakan dan bertindak sebagai pengelola aplikasi, data, dan keamanannya. *Server* biasanya terhubung dengan *client* melalui kabel *UTP* dan sebuah kartu jaringan (*network card*). Kartu jaringan ini biasanya berupa kartu *PCI* atau *ISA*.

Dalam teknologi informasi, *client-server* merujuk kepada cara mendistribusikan aplikasi ke pihak *client* dan pihak *server*. Dalam model *client-server*, sebuah aplikasi dibagi menjadi dua bagian yang terpisah (tetapi masih dalam sebuah kesatuan) yakni komponen *client* dan komponen *server*.

Komponen *client* dijalankan pada sebuah workstation. Pemakai *workstation* memasukkan data dengan menggunakan teknologi pemrosesan tertentu, kemudian mengirimkannya ke komponen *server*, umumnya berupa permintaan layanan tertentu yang dimiliki oleh *server*. Komponen *server* akan menerima permintaan layanan tersebut dan langsung memprosesnya serta

mengembalikan hasil pemrosesan kepada *client*. *Client* pun menerima informasi hasil pemrosesan data tadi dan menampilkannya kepada pemakai dengan menggunakan aplikasi yang digunakan oleh pemakai.

Sebuah contoh dari aplikasi *client-server* sederhana adalah aplikasi web yang didesain dengan menggunakan *Active Server Pages* (*ASP*). Skrip *ASP* akan dijalankan di dalam *web server* (*Apache* atau *Internet Information Services*), sementara skrip yang berjalan di pihak *client* akan dijalankan oleh *web browser* pada komputer *client* (*workstation*). *Client-server* merupakan penyelesaian masalah pada *software* yang menggunakan *database* sehingga setiap komputer tidak perlu diinstall *database*. Dengan metode *client-server database* dapat diinstal pada komputer *server* dan aplikasinya diinstal pada *client*. Komponen *client* juga sering disebut sebagai *front-end*, sementara komponen *server* disebut sebagai *back-end* (Ardianto, 2011).

II.7. *Website*

Menurut Betha Sidik (2002, h.1), *World Wide Web (WWW)* lebih dikenal dengan *web*, merupakan "salah satu layanan yang didapat oleh pemakai komputer yang terhubung ke internet".

Web pada awalnya adalah ruang informasi dalam internet, dengan menggunakan teknologi *hypertext*, pemakai dituntun untuk menemukan informasi dengan mengikuti *link* yang disediakan dalam dokumen *web* yang ditampilkan dalam *browser web*.

II.8. *PHP*

Menurut Bunafit Nugroho (2004:139,140) ada beberapa pengertian tentang *PHP*. Akan tetapi, kurang lebih *PHP* dapat kita ambil arti sebagai *PHP Hypertext Preprocesor*, ini merupakan bahasa yang hanya dapat berjalan pada *server* yang hasilnya dapat ditampilkan pada klien.

Interpreter PHP dalam mengeksekusi kode *PHP* pada sisi *server* (disebut *server-side*) berbeda dengan mesin maya *Java* yang mengeksekusi program pada sisi klien (*client-side*).

PHP merupakan bahasa standar yang digunakan dalam dunia *website*. *PHP* adalah bahasa program yang berbentuk *script* yang diletakkan di dalam *server web*. Jika kita lihat dari sejarah, mulanya *PHP* diciptakan dari ide Rasmus Lerdof yang membuat sebuah *script perl*. *Script* tersebut sebenarnya dimaksudkan untuk digunakan sebagai program untuk dirinya sendiri. Akan tetapi, kemudian dikembangkan lagi sehingga menjadi sebuah bahasa yang disebut "*Personal Home Page*". Inilah awal mula munculnya *PHP* sampai saat ini.

II.9. *HTML*

Menurut Bunafit Nugroho (2004, h.201), *HTML* (*Hypertext Markup Language*) adalah "sebuah bahasa *scripting* yang berguna untuk menuliskan halaman *web*.".

Pada web, *HTML* dijadikan sebagai Bahasa *Script* dasar yang berjalan bersama berbagai bahasa *scripting* pemrograman lainnya. Semua *tag-tag HTML* bersifat dinamis artinya kode *HTML* tidak dapat dijadikan sebagai *file executable* program. Hal tersebut disebabkan, *HTML* hanyalah sebuah bahasa *scripting* yang dapat berjalan apabila dijalankan di dalam *browser* (pengakses *web*). *Browser-*

browser yang mendukung *HTML* antara lain *Internet Explorer*, *Netscape Navigator*, *Operasi*, *Mozilla*, dan lain-lain.

II.10. *MySQL*

Menurut Bunafit Nugroho (2004, h.133), "*MySQL* merupakan *database* yang paling digemari dikalangan programmer *web*, dengan alasan bahwa program ini merupakan *database* yang sangat kuat dan cukup stabil untuk digunakan sebagai media penyimpanan data".

II.11. Pengertian UML

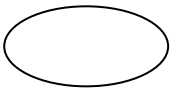
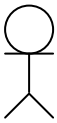
Menurut Sri Dharwiyanti dan Romi Satria Wahono (2003), UML adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak.

UML merupakan metode untuk merancang dan mengembangkan sebuah sistem juga merupakan alat untuk mendukung pengembangan sistem. Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

II.11.1. *Use Case Diagram*

Use case Diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut (Ade Hendini, 2016).

Tabel II.1. *Use Case* Diagram

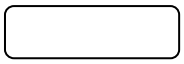
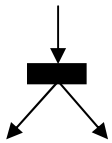
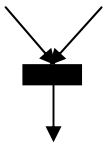
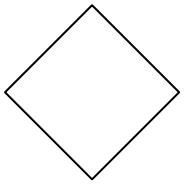
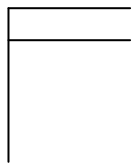
Gambar	Keterangan
	*Use Case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktir, yang dinyatakan dengan menggunakan kata kerja
	*Actor* atau Aktor adalah Abstraction dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifika sikan aktir, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *Use Case*, tetapi tidak memiliki kontrol terhadap *use case*
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengindikasikan bila aktor berinteraksi secara pasif dengan sistem
	Include, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program
	Extend, merupakan perluasan dari *use case* lain jika kondisi atau syarat

Sumber : Ade Hendini (2016), "Pemodelan UML Sistem Informasi Monitoring Penjualan Dan Stok Barang".

II.11.2. *Activity Diagram*

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis (Ade Hendini, 2016).

Tabel II.2. *Activity* Diagram

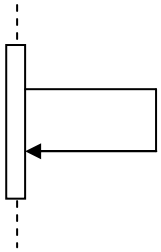
Gambar	Keterangan
	*Start Point*, diletakkan pada pojok kiri atas dan merupakan awal aktivitas
	*End Point*, akhir aktivitas
	*Activities*, menggambarkan suatu proses/kegiatan bisnis
	*Fork*/percabangan, digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabung kan dua kegiatan paralel menjadi satu
	*Join* (penggabungan) atau *rake*, digunakan untuk menunjukkan adanya dekomposisi
	*Decision* Points, menggambar kan pilihan untuk pengambilan keputusan, tru atau false
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa

Sumber : Ade Hendini (2016), "Pemodelan UML system Informasi Monitoring Penjualan Dan Stok Barang".

II.11.3. *Sequence Diagram*

Sequence diagram menggambarkan kelakuan obyek pada *use case* dengan mendeskripsikan waktu hidup obyek dan pesan yang dikirimkan dan diterima antar obyek (Ade Hendini, 2016).

Tabel II.3. *Sequence Diagram*

Gambar	Keterangan
	*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data
	*Boundary Class*, berisi kumpulan kelas yang menjadi interfaces atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan *form entry* dan *form* cetak
	*Control class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek
	Message, simbol mengirim pesan antar class
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri
	*Activation*, mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang lifeline terdapat activation

Sumber : Ade Hendini (2016), "Pemodelan UML system Informasi Monitoring Penjualan Dan Stok Barang".

II.11.4. *Class Diagram*

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas didalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. *Class* diagram juga menunjukan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint*

yang berhubungan dengan obyek yang dikoneksikan. *Class* diagram secara khas meliputi:Kelas (*Class*), *Relasi, Associations, Generalization* dan *Aggregation,* Atribut *(Attributes),* Operasi *(Operations/Method),* dan*Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar Kelas mempunyai keterangan yang disebut dengan *Multiplicity* atau kardinaliti (Ade Hendini, 2016).

Tabel II.4. *Multiplicity Class* Diagram

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimal 4

Sumber : Ade Hendini (2016), "Pemodelan UML system Informasi Monitoring Penjualan Dan Stok Barang".