BAB II
LANDASAN TEORI

2.1 Toko Kelontong

Toko kelontong *(Convenience Store)* adalah toko kecil yang menjual lini terbatas barang-barang kebutuhan sehari-hari dengan tingkat perputaran tinggi (Kotler, 2016). Pelayanan toko kelontong masih bersifat tradisional, sehinggan pembeli tidak dapat memilih produk sendiri. Posisi toko kelontong di dunia perdagangan berada di bawah minimarket jika dipandang dari sisi kenyamanan, pelayanan dan kelengkapan produk yang dijual. Terdapat beberapa keunggulan dan kelemahaan dari Warung kelontong sebagai berikut:

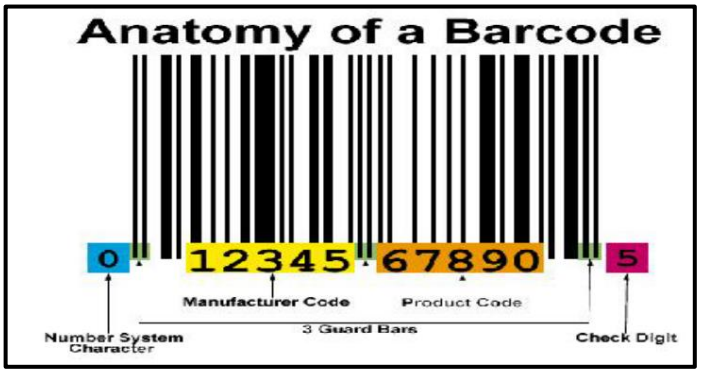
1. Keunggulan warung kelontong
   a. Bersahabat terhadap pembeli
   b. Harga barang bisa ditawar
   c. Bisa beli eceran
   d. Dapat memenuhi pesan untuk pelanggan
   e. Bisa berutang atau dibayar kemudian
2. Kelemahaan Warung Kelontong
   a. Bentuk warung tidak menarik
   b. Tata letak barang di dalam warung tidak diatur dengan nyaman dan efesien
   c. Tidak selalu memperhatikan dengan keyamanan dan kebersihan
   d. Kurangnya penerangan lampu

e. Barang tidak lengkap

f. Kekurangan modal

2.2 *Barcode*

Barcode pada dasarnya adalah susunan garis vertikal hitam dan putih dengan ketebalan yang berbeda, sangat sederhana tetapi sangat berguna. Dengan kegunaan untuk menyimpan data-data spesifik misalnya kode produksi, tanggal kadaluwarsa, nomor identitas dengan mudah dan murah, walaupun teknologi semacam itu terus berkembang dengan ditemukannya media magnetik, RFID, electronic tags, serial EEPROM (seperti pada *smart card*), *barcode* terus bertahan dan masih memiliki kelebihan-kelebihan tertentu yaitu mudah dan murah, sebab media yang digunakan adalah kertas dan tinta, sedangkan untuk membaca *barcode* ada begitu banyak pilihan di pasaran dengan harga yang relatif murah. Alat yang digunakan untuk membaca *barcode* adalah *barcode scanner*.Penggunaan *barcode scanner* sangat mudah sehingga pengguna (operator) hanya memerlukan sedikit latihan. *Barcode scanner* dapat membaca informasi/data dengan kecepatan yang jauh lebih tinggi dari pada mengetikkan data dan *barcode* scanner memiliki tingkat ketelitian yang lebih tinggi. Bentuk *barcode* dapat dilihat pada gambar berikut:



Gambar 2. 1 Spesifikasi *Barcode*

Keterangan gambar *barcode*:

a. *Number System*

Karakter Angka ini merupakan sebuah bilangan *barcode* UPC yang mengkarakteristikkan jenis-jenis khusus pada *barcode*. Di dalam *barcode* UPC, NSC ini biasanya terletak disebelah kiri *barcode*. Kode kode yang tertera adalah sebagai berikut: 0-*standard* UPC *number* 1-*reserved* 2-*random weight items like fruits,vegetables,and meats* 3-*Pharmaceuticals*. 4-*in-store code for retailers* 5-*Coupons* 6-*Standard* UPC *number* 7-*Standard* UPC *number* 8-*reserved* 9-*reserved*.

b. *Guard Bars*

Ada tiga *guard bars* yang ditempatkan diawal, ditengah, dan akhir *barcode*. *Guards bars* bagian awal dan akhir di *encode* kan sebagai *"bar-space-bar"*.

c. *Manufacturer code*

Kode perusahaan ini ada lima digit bilangan yang secara khusus menentukan manufaktur suatu produk. Kode perusahaan/manufaktur ini dilindungi dan ditetapkan oleh *Uniform Code Council*.

d. *Product Code*

Kode Produk ini terdiri dari 5 digit bilangan yang ditetapkan oleh perusahaan/manufaktur untuk setiap produk yang dihasilkanya. Setiap produk yang berbeda dan setiap ukuran yang berbeda memiliki kode produk yang unik.

e. Check Digit

Disebut sebagai *digit shelf check*. *Check digit* ini terletak dibagian luar sebelah kanan *barcode*. Check digit ini merupakan suatu *olds Programmer's trick* untuk memvalidasi digit-digit lainya yang dibaca secara teliti. Selanjutnya, masing-masing batang pada *barcode* memiliki ketebalan yang berbeda. Ketebalan inilah yang akan diterjemahkan pada suatu nilai. Demikian, karena ketebalan batang *barcode* menentukan waktu lintasan bagi titik sinar pembaca yang dipancarkan oleh alat pembaca. Oleh sebab itu, batang-batang *barcode* harus dibuat demikian sehingga memilki kontras yang tinggi terhadap celah antara yang menentukan cahaya. Sisi- sisi batang *barcode* harus tegak dan lurus, serta tidak ada lubang atau noda titik tengah permukaannya. Sementara itu, ukuran titik sinar pembaca juga tidak boleh melebihi celah antara batang *barcode*. Saat ini, ukuran titik sinar yang umum digunakan adalah 4 kali titik yang dihasilkan printer pada resolusi 300dpi.

Barcode memiliki 2 tipe yaitu:

1) *Barcode* 1 Dimensi

Barcode satu dimensi biasanya dinamakan *linear bar codes* (kode berbentuk baris). *Barcode* ini dinamakan satu dimensi atau ada yang menyebut *linear bar codes* karena kodenya hanya terdiri dari baris-baris.

2) *Barcode* 2 Dimensi

Adalah *barcode* yang dikembangkan lebih dari sepuluh tahun lalu, tetapi baru sekarang ini mulai populer. *Barcode* dua dimensi ini memiliki beberapa keuntungan dibandingkan linear *bar codes* (*barcode* satu dimensi) yaitu, dengan menggunakan *barcode* dua dimensi, informasi atau data yang besar dapat disimpan di dalam suatu ruang (*space*) yang lebih kecil.

2.3 *Android*

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi (Nazruddin, Safaat, 2014). *Android* menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya, *Google Inc*. membeli *Android Inc*. yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel/*smartphone.* Kemudian untuk mengembangkan *android*, dibentuklah *Open Handset Alliance,* konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk *Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.*

Pada saat perilisan *Android*, 5 November 2007, *Android* bersama *Open Handset Alliance* menyatakan mendukung pengembangan *open source* pada perangkat mobile. Di lain pihak, Google merilis kode-kode *Android* dibawah lisensi Apache, sebuah lisensi perangkat lunak dan open *platform* perangkat seluler. Di dunia ini terdapat dua jenis distributor *system* operasi *Android*. Pertama yang mendapat dukungan penuh dari *Google* atau *Google Mail Services (GMS)* dan kedua adalah yang benar-benar bebas distribusinya

tanpa dukungan langsung Google atau dikenal sebagai *Open Handset Distribution* (OHD).

Sekitar September 2007 Google mengenalkan Nexus One, salah satu jenis *smartphone* yang menggunakan *Android* sebagai *sistem* operasinya. Telepon seluler ini diproduksi oleh HTC Corporation dan tersedia di pasaran pada 5 Januari 2010. Pada 9 Desember 2008, diumumkan anggota baru yang bergabung dalam program kerja *Android ARM Holdings, Atheros Communications, diproduksi oleh asustek Computer Inc, Garmin Ltd, Softbank, Sony Ericsson, Toshiba Corp, dan Vodafone Group Plc.* Seiring pembentukan *Open Handset Alliance, OHA* mengumumkan produk perdana mereka, *Android*, perangkat mobile yang merupakan modifikasi kernel *Linux 2.6*. Sejak *Android* dirilis telah dilakukan berbagai pembaruan berupa perbaikan bug dan penambahan fitur baru.

Pada masa saat ini kebanyakan vendor-vendor *smartphone* sudah memproduksi *smartphone* berbasis *Android*, vendor-vendor itu antara lain *HTC, Motorola, Samsung, LG, Acer, Philips, T-Mobile, Nexian, IMO, Asus* dan masih banyak lagi vendor *smartphone* didunia yang memproduksi *Android*, hal ini karena *Android* itu adalah *system operasi* yang *Open Source* sehingga bebas didistribusikan dan dipakai oleh vendor manapun.

Tidak hanya menjadi system operasi di *smartphone*, saat ini *Android* menjadi pesaing utama dari Apple pada system operasi Tablet PC. Pesatnya pertumbuhan *Android* selain faktor yang disebutkan diatas adalah karena *Android* itu sendiri adalah *platform* yang sangat lengkap baik itu system operasinya, aplikasi dan Tool Pengembangan, Market aplikasi *Android* serta

dukungan yang sangat tinggi dari komunitas *Open Source* di dunia, sehingga *Android* terus berkembang pesat baik dari segi teknologi maupun dari segi jumlah *device* yang ada di dunia.

Android dipuji sebagai "*platform mobile* pertama yang lengkap, terbuka, dan bebas" (Nazruddin, Safaat, 2014).

a. Lengkap (*Complete Platform*): para desainer dapat melakukan pendekatan yang komprehensif ketika mereka sedang mengembangkan *platform Android*. *Android* merupakan sistem operasi yang aman dan banyak menyediakan *tools* dalam membangun *software* dan memungkinkan untuk peluang pengembangan aplikasi.
b. Terbuka (*Open Source Platform*): *platform Android* disediakan melalui lisensi *Open Source*. Pengembang dapat dengan bebas untuk mengembangkan aplikasi. *Android* sendiri menggunakan Linux Kernel 2.6.
c. *Free* (*Free Platform*): *Android* adalah *platform*/aplikasi yang bebas untuk *develop*. Tidak ada lisensi atau biaya *royalty* untuk dikembangkan pada *platform Android*. Tidak ada biaya keanggotaan diperlukan. Aplikasi untuk *Android* dapat didistribusikan dan diperdagangkan dalam bentuk apapun.

Android merupakan generasi baru *platform mobile*, *platform* yang memberikan pengembangan untuk melakukan pengembangan sesuai dengan yang diharapkannya. Sistem operasi yang mendasari *Android* dilisensikan di bawah *GNU, General Public Lisensi Versi 2 (GPLv2)*, yang sering dikenal dengan istilah "*copyleft*" lisensi dimana setiap perbaikan pihak ketiga harus

terus jatuh di bawah terms. *Android* didistribusikan di bawah lisensi *Apache Software* (ASL/Apache2), yang memungkinkan untuk distribusi kedua dan seterusnya. Komersialisasi pengembang (produsen *handset* khususnya) dapat memilih untuk meningkatkan *platform* tanpa harus memberikan perbaikan mereka kepada masyarakat *Open Source.* Sebaliknya, pengembang dapat keuntungan dari perangkat tambahan seperti perbaikan dan mendistribusikan ulang pekerjaan mereka dibawah lisensi apapun yang mereka inginkan.

2.4 *The Dalvik Virtual Machine* (DVM)

Salah satu elemen kunci dari *Android* adalah *Dalvik Virtual Machine* (DVM). *Android* berjalan di dalam *Dalvik Virtual Machine* (DVM) bukan di *Java Virtual Machine* (JVM), sebenarnya banyak persamaan dengan *Java Virtual Machine* (JVM) seperti *Java* ME(*Java Mobile Edition*), tetapi *Android* menggunakan *Virtual Machine* sendiri yang dikustomisasi dan dirancang untuk memastikan bahwa beberapa *feature-feature* berjalan lebih efisien pada perangkat *mobile*.

Dalvik Virtual Machine (DVM) adalah "*register bases"* sementara *Java Virtual Machine* (JVM) adalah "*stack based", DVM* didesain dan ditulis oleh Dan Bornsten dan beberapa engineers Google lainnya. Jadi bisa kita katakana "*Dalvik equals(Java) == False". Dalvik Virtual Machine* menggunakan kernel Linux untuk menangani fungsionalitas tingkat rendah termasuk keamanan, *threading,* dan proses serta manajemen memori. Ini memungkinkan kita untuk menulis Aplikasi C/C+ sama halnya seperti OS Linux kebanyakan. Meskipun dalam kenyataannya kita harus banyak

memahami Arsitektur dalam proses *system* dari kernel linux yang digunakan dalam *Android* tersebut.

Semua *hardware* yang berbasi *Android* dijalankan menggunakan *Virtual Machine* untuk eksekusi aplikasi, pengembang tidak perlu khawatir tentang implementasi perangkat keras tertentu. *Dalvik Virtual Machine* mengeksekusi *executable file*, sebuah format yang dioptimalkan untuk memastikan memori yang digunakan sangat kecil. *The Executable file* diciptakan dengan mengubah kelas bahasa java dan dikompilasi menggunakan *tools* yang disediakan dalam SDK *Android*.

2.5 *Android* SDK (*Software Development Kit*)

Android SDK tools API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform Android* menggunakan bahasa pemrograman Java. *Android* merupakan subset perangkat lunak untuk ponsel yang meliputi system operasi, *middleware* dan aplikasi kunci yang di rilis oleh Google. Saat ini disediakan *Android SDK* (*Software Development Kit*) sebagai alat bantu dan *API* untuk mulai mengembangkan aplikasi pada *platform* menggunakan Bahasa pemrograman *Java*.

Sebagai aplikasi-netral, *Android* memberi kita kesempatan untuk membuat aplikasi yang kita butuhkan yang bukan merupakan aplikasi bawaan *Handphone/Smartphone.* Beberapa fitur-fitur *Android* yang paling penting adalah:

a. *Framework* Aplikasi yang mendukung penggantian komponen dan *reusable.*

b. Mesin *Virtual Dalvik* dioptimalkan untuk perangkat *mobile*

c. *Integrated Browser* berdasarkan *engine open source* WebKit

d. *Grafis* yang dioptimalkan dan dudukung oleh libraries grafis 2D, grafis 3D berdasarkan spesifikasi *opengl ES 1,0(Opsional akselerasi hardware)*

e. *SQLite* untuk penyimpanan data

f. Media Support yang mendukung audio, video, dan gambar *(MPEG4, H.164, MP3, AAC, AMR, JPG, PNG, GIF), GSM Telephony* (tergantumg *hardare*)

g. *Bluetooth, EDGE, 3G, dan WiFi* (tergantung *hardware*)

h. Kamera, *GPS*, Kompas, dan *accelerometer* (tergantung *hardware*)

i. Lingkungan *Development* yang lengkap dan kaya termasuk perangkat emulator, *tools* untuk *debugging*, profil dan kinerja memori, dan plugin.

2.6 *Ionic Framework*

Ionic adalah sebuah *framework* untuk *user interface* berbasis *JavaScript* dan CSS yang dikembangkan secara *open source* (Anditya, & Ilhami, M. 2015). *Framework* ini dikembangkan dengan tujuan mempermudah *developer* untuk membuat *user interface* yang optimal untuk digunakan pada device yang memiliki *input* berupa layar sentuh. Dengan *Ionic* maka developer tidak perlu membuat *user interface* yang sesuai dengan perangkat berbasis layar sentuh melainkan developer tinggal memakai *user interface* yang telah disediakan. *Ionic* juga menyediakan *command line tool* yang dapat dipergunakan untuk melakukan *scaffolding* aplikasi baru dan juga menyediakan konversi sebuah *project* aplikasi berbasis web menjadi aplikasi berbasis *mobile* dengan menggunakan Cordova.

Struktur aplikasi *Ionic Framework* menganut konsep MVC (*Model-ViewController*), dimana *Model* adalah komponen yang khusus terkait dengan manipulasi *database*, *View* adalah komponen yang khusus menampilkan data maupun konten dalam format HTML yang siap disajikan ke pengguna, dan *Controller* adalah jembatan antara manipulasi *database*/konten *Model* ke *View*.

Ionic menyediakan semua fungsi pada *development* SDK untuk pembuatan aplikasi *native mobile*. Pengguna dapat membangun aplikasi untuk Android maupun iOS, dan dapat disebarkan melalui Cordova. *Ionic* meliputi *mobile component*, *typhography*, paradigma interaktif, dan basis tema *extensible* (Nazruddin, Safaat. 2014). Ionic menyediakan komponen *custom* dan metode untuk berinteraksi menggunakan Angular. Salah satu komponen tersebut dapat memungkinkan *user* untuk menelusuri daftar ribuan item tanpa mempengaruhi kinerja. Komponen lainnya adalah *scroll-view*, menciptakan *container* yang *scrollable* dimana *user* dapat berinteraksi menggunakan *native-influenced delegate system*. Selain SDK, Ionic juga menyediakan layanan yang dapat digunakan developer untuk mengaktifkan fitur, seperti *push notifications*, A/B *testing*, *analytics*, *code deploys*, dan *automated builds*.

Ionic juga menyediakan *Command-Line Interface* (CLI) yang *powerful*, sehingga para developer dapat memulai dan membuat proyek dengan perintah sederhana. CLI juga memungkinkan pengembang untuk menambahkan *plugin Cordova* dan paket *front-end* tambahan, mengaktifkan *push notifications, generate app Icons* dan *Splash screens*, serta membangun *binary native*.

2.7 *Framework*

Framework secara sederhana dapat diartikan kumpulan dari fungsi-fungsi/prosedur-prosedur dan *class-class* untuk tujuan tertentu yang sudah siap digunakan sehingga bisa lebih mempermudah dan mempercepat pekerjaan seorang programer, tanpa harus membuat fungsi atau *class* dari awal (Raharjo, Budi. 2015). Ada beberapa alasan mengapa menggunakan *Framework*:

1. Mempercepat dan mempermudah pembangunan sebuah aplikasi web.
2. Relatif memudahkan dalam proses maintenance karena sudah ada pola tertentu dalam sebuah *framework* (dengan syarat programmer mengikuti pola standar yang ada).
3. Umumnya *framework* menyediakan fasilitas-fasilitas yang umum dipakai sehingga kita tidak perlu membangun dari awal (misalnya validasi, ORM, pagination, multiple database, scaffolding, pengaturan session, error handling, dll.
4. Lebih bebas dalam pengembangan jika dibandingkan CMS.

Framework menganut konsep *Model View Controller* (MVC) merupakan suatu konsep yang cukup populer dalam pembangunan aplikasi web. Berawal pada bahasa pemrograman Small Talk, MVC memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, *user interface*, dan bagian yang menjadi kontrol aplikasi. Terdapat 3 jenis komponen yang membangun suatu MVC *pattern* dalam suatu aplikasi yaitu :

1. *View*, merupakan bagian yang menangani *presentation logic*. Pada suatu aplikasi web bagian ini biasanya berupa file template HTML, yang diatur oleh *controller*. View berfungsi untuk menerima dan merepresentasikan data kepada *user*. Bagian ini tidak memiliki akses langsung terhadap bagian *model*.
2. *Model*, biasanya berhubungan langsung dengan database untuk memanipulasi data (*insert, update, delete, search*), menangani validasi dari bagian *controller*, namun tidak dapat berhubungan langsung dengan bagian *view*.
3. *Controller*, merupakan bagian yang mengatur hubungan antara bagian *model* dan bagian *view*, *controller* berfungsi untuk menerima request dan data dari user kemudian menentukan apa yang akan diproses oleh aplikasi.

Dengan menggunakan prinsip MVC suatu aplikasi dapat dikembangkan sesuai dengan kemampuan developernya, yaitu *programmer* yang menangani bagian model dan *controller*, sedangkan *designer* yang menangani bagian *view*, sehingga penggunaan arsitektur MVC dapat meningkatkan *maintanability* dan organisasi kode. Walaupun demikian dibutuhkan komunikasi yang baik antara *programmer* dan *designer* dalam menangani variabel-variabel yang akan ditampilkan.

2.8 *MySQL*

MySQL adalah *Relational Database Management System* (*RDBMS*) yang didistribusikan secara gratis dibawah lisensi *GPL (General Public*

License). Dimana setiap orang bebas untuk menggunakan *MySQL*, namun tidak boleh dijadikan produk turunan yang bersifat *closed source* atau komersial. *MySQL* sebenarnya merupakan turunan salah satu konsep utama dalam *database* sejak lama, yaitu *SQL (Structured Query Language). SQL* adalah sebuah konsep pengoperasian *database*, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis (Kadir, Abdul. 2014). Keandalan suatu sistem *database* (*DBMS*) dapat diketahui dari cara kerja *optimizer*-nya dalam melakukan proses perintah-perintah *SQL*, yang dibuat oleh *user* maupun program-program aplikasinya. *MySQL* biasanya digunakan atau di-*install* bersamaan dengan *XAMPP* sehingga untuk melihat isi tabel bisa menggunakan *PHPmyAdmin*.

2.9 *Cascading Style Sheets (CSS)*

Cascading Style Sheets (CSS) adalah suatu bahasa *stylesheet* yang digunakan untuk mengatur tampilan suatu dokumen yang ditulis dalam bahasa *markup* (Kadir, Abdul. 2014). Penggunaan yang paling umum dari CSS adalah untuk memformat halaman *web* yang ditulis dengan *HTML* dan *XHTML*. Walaupun demikian, bahasanya sendiri dapat dipergunakan untuk semua jenis dokumen *XML* termasuk *SVG dan XUL*. *Spesifikasi CSS* diatur oleh *World Wide Web Consortium (W3C).*

CSS digunakan oleh penulis maupun pembaca halaman *web* untuk menentukan warna, jenis huruf, tata letak, dan berbagai aspek tampilan dokumen. *CSS* digunakan terutama untuk memisahkan antara isi dokumen (yang ditulis dengan *HTML* atau *bahasa markup* lainnya) dengan presentasi

dokumen (yang ditulis dengan *CSS*). Pemisahan ini dapat meningkatkan aksesibilitas isi, memberikan lebih banyak keleluasaan dan kontrol terhadap tampilan, dan mengurangi kompleksitas serta pengulangan pada stuktur isi.

2.10 *Node Java Script (Node JS)*

Nodejs merupakan salah satu piranti pengembang yang dapat digunakan untuk membuat aplikasi berbasis *Cloud* (Handayani. 2015). *Node.js* dikembangkan dari *engine JavaScript* yang dibuat oleh Google untuk *browser* Chrome ditambah dengan *library* serta beberapa pustaka lainnya.Node.js menggunakan *JavaScript* sebagai bahasan pemrograman dan *event-driven*, *non-blocking* I/O (*asynchronous*) model yang membuatnya ringan dan efisien. Node.js memiliki fitur *built-in HTTP server library* yang menjadikannya mampu menjadi sebuah web server tanpa bantuan software lainnya seperti Apache dan Nginx. Pada dasarnya, Node.js adalah sebuah *runtime environment* dan *script library*. Sebuah *runtime environment* adalah sebuah software yang berfungsi untuk mengeksekusi, menjalankan dan mengimplementasikan fungsi-fungsi serta cara kerja inti dari suatu bahasa pemprograman.Sedangkan *script library* adalah kumpulan, kompilasi atau bank data berisi *skript*/kode-kode pemrograman.

2.11 *Unified Modelling Language*

Unified Modeling Language (*UML*) merupakan sistem arsitektur yang bekerja dalam *OOAD* (*Object-Oriented Analysis/Design*) dengan satu

bahasa yang konsisten untuk menentukan, visualisasi, mengkontruksi, dan mendokumentasikan *artifact* (sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa *software*, dapat berupa model, deskripsi, atau *software*) yang terdapat dalam sistem *software* . *UML* merupakan bahasa pemodelan yang paling sukses dari tiga metode *OOP* yang telah ada sebelumnya, yaitu *Booch, OMT* (*Object Modeling Technique*), dan *OOSE* (*Object-Oriented Software Engineering*). *UML* merupakan kesatuan dari ketiga pemodelan tersebut dan ditambah kemampuan lebih karena mengandung metode tambahan untuk mengatasi masalah pemodelan yang tidak dapat ditangani ketiga metode tersebut. *UML* dikeluarkan oleh *OMG(Object Management Group, Inc)* yaitu organisasi internasional yang dibentuk pada 1989, terdiri dari perusahaan sistem informasi, *software developer*, dan para *user* sistem komputer.

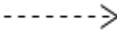
Dengan adanya *UML*, diharapkan dapat mengurangi kekacauan dalam bahasa pemodelan yang selama ini terjadi dalam lingkungan industri. *UML* diharapkan juga dapat menjawab masalah penotasian dan mekanisme tukar menukar model yang terjadi selama ini (Nugroho, Adi, 2014)**.** Tujuan *UML* diantaranya adalah:

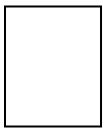
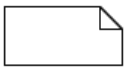
1. Memberikan model yang siap pakai, bahasa pemodelan *visual* yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
2. Memberikan bahasa pemodelan yang bebas dari berbagai Bahasa pemrograman dan proses rekayasa.
3. Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan

2.11.1 *Use Case Diagram*

Use case diagram adalah rangkaian/uraian sekelompok yang saling terkait dan membentuk sistem secara teratur yang dilakukan atau diawasi oleh sebuah aktor. *Use Case* Diagram merupakan rangkaian tindakan yang dilakukan oleh sistem, aktor mewakili *user* atau sistem lain yang berinteraksi dengan sistem yang dimodelkan .

Tabel 2. 1 Tipe relasi pada *Use Case Diagram*

	GAMBAR	**NAMA**	**KETERANGAN**
		*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
		Dependency	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
		Generalization	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
		*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.

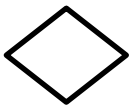
		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
		Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.
		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi

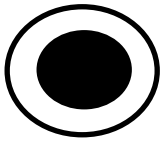
2.11.2 *Activity Diagram*

Activity Diagram atau Diagram Aktivitas adalah teknik untuk mendeskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. Diagram aktivitas mempunyai peran seperti

halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah diagram aktivitas bisa mendukung perilaku paralel sedangkan *flowchart* tidak bisa. Sebuah *activity diagram* memiliki simbol-simbol dengan kegunaannya dalam setiap aktivitas sistem seperti yang ditunjukkan pada Tabel 2.3.

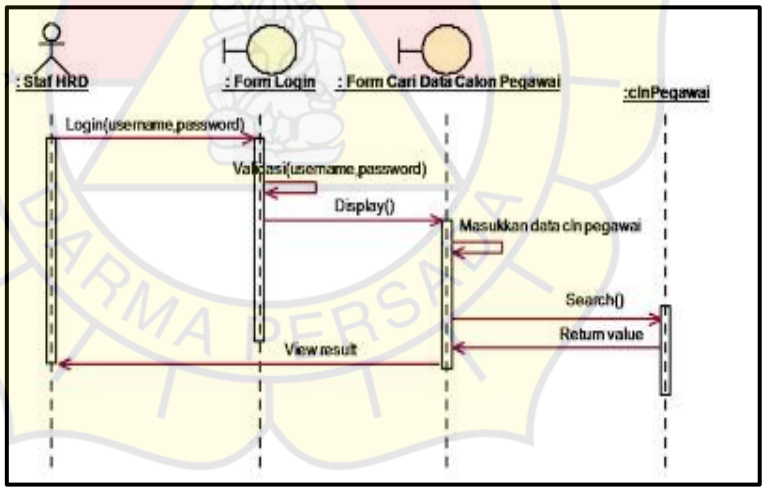
Tabel 2. 2 Simbol pada *Activity Diagram*

Simbol	**Penjelasan**
	*Initial State* Mempresentasikan dimulainya alur kerja suatu sistem dalam *activity diagram*.
	*Action State* Sebuah *state* yang menggambarkan eksekusi dari aksi *atomic*.
	*Transition Between Activities* Mengidentifikasikan bahwa suatu objek dari *state* pertama akan menampilkan aksi-aksi tertentu dan memasuki *state* kedua ketika peristiwa terjadi pergerakan dari aksi ke aksi lainnya.
	*Decision Point* Menentukan kapan alur dalam aktivitas menjadi bercabang.
	*Fork* Adanya percabangan paralel dari aktivitas

	*Final State* Mempresentasikan bahwa telah diakhirinya alur suatu sistem dalam *activity diagram.*

2.11.3 *Sequence Diagram*

Merupakan diagram yang menunjukkan aliran fungsionalitas dalam *use case*. *Sequence* adalah satu dari dua interaksi diagram yang mengilustrasikan objek-objek yang berhubungan dengan *use case* dan *message* atau pesan-pesannya. Komponen utama *sequence* diagram terdiri atas objek yang dituliskan dengan kotak segiempat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan progress *vertical.*



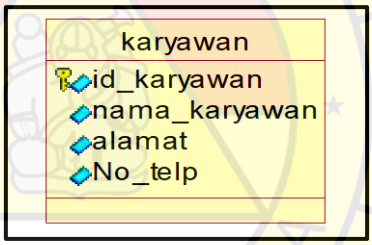
Gambar 2. 2 Contoh Sequence Diagram

2.11.4 *Class Diagram*

Diagram kelas atau *Class diagram* sangat membantu dalam visualisasi struktur kelas dari suatu sistem. Hal ini disebabkan karena *class* adalah deskripsi kelompok obyek-obyek dengan *property*,

operasi dan relasi yang sama. Disamping itu diagram kelas bisa memberikan pandangan global atas sebuah sistem. Hal tersebut tercermin dari *class-class* yang ada dan relasinya satu dengan lainnya. Itulah sebabnya diagram kelas menjadi diagram yang paling popular di *UML*. Komponen diagram kelas adalah :

1. Asosiasi adalah *class-class* yang berhubungan satu sama lain secara konseptual. yaitu menghubungkan dua kelas menjadi satu asosiasi.
2. Atribut adalah properti dari sebuah kelas. Atribut ini menjelaskan batas nilai yang mungkin ada pada obyek dari kelas. Sebuah kelas mungkin mempunyai nol atau lebih atribut.



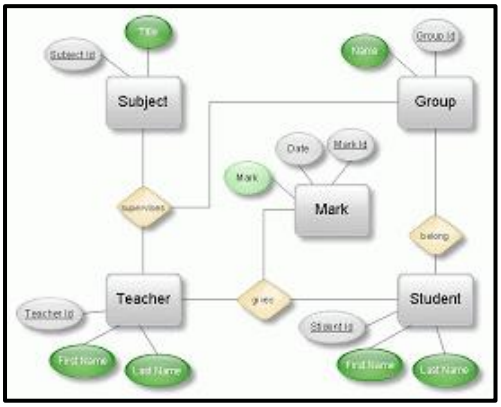
Gambar 2. 3 Contoh *Attribute Class Diagram*

Operasi adalah sesuatu yang bisa dilakukan oleh sebuah *class* atau yang anda (atau *class* yang lain) dapat lakukan untuk sebuah *class*. Sama halnya dengan atribut, kita bisa juga memberikan tambahan informasi untuk operasi dengan menambahkan parameter yang akan dilakukan oleh operasi dengan tanda kurung.

2.12 *Entity Relationship Diagram* (ERD)

Entity Relational Diagram adalah alat peraga atas desain database yang menjadi dasar sistem informasi yang tengah dikembangkan

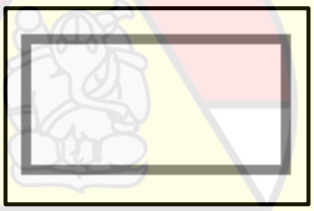
(Indrajani, 2014).



Gambar 2. 4 Contoh *Entity Relationship Diagram*

Simbol-simbol dalam *Entity Relationship Diagram* adalah sebagai berikut:

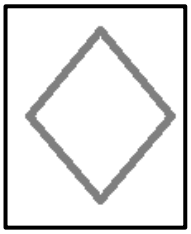
1. Entitas (*Entity*)



Gambar 2. 5 Simbol Entitas ERD

Entitas Ialah satu objek yang dapat dibedakan dengan objek lainnya. Entitas berfungsi untuk memberikan identitas pada entitas yang memiliki label dan nama. Entitas memiliki bentuk persegi panjang.

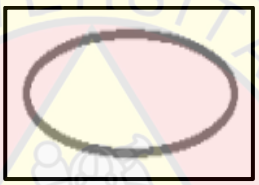
2. Relasi/Hubungan Antar Entitas (relationship)



Gambar 2. 6 Simbol Relasi ERD

Relasi ialah hubungan yang terjadi antara 1 entitas atau lebih yang tidak mempunyai fisik tetapi hanya sebagai konseptual. Dan berfungsi untuk mengetahui jenis hubungan yang ada antara 2 file. Relisi memiliki bentuk belah ketupat. Gambar 2.5 merupakan simbol relasi pada ERD.

3. Atribut



Gambar 2. 7 Simbol Atribut ERD

Atribut ialah karakteristik dari entitas atau relasi yang menyediakan penjelasan detil tentang entitas atau relasi tersebut. Dan berfungsi untuk memperjelas atribut yang dimiliki oleh sebuah entitas. Atribut memiliki bentuk lingkarang lebih tepatnya elips. Gambar 2.6 merupakan simbol atribut di dalam ERD.

4. Alur

Gambar 2. 8 Simbol Alur ERD

Alur memiliki fungsi untuk menghubungkan atribut dengan entitas dan entitas dengan relasi. Simbol alur berbentuk garis.