

BAB II

TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Terdapat beberapa penelitian yang sudah dilakukan, diantaranya penelitian oleh Rachman Arief tahun 2018. Sistem ini bertujuan untuk membantu kinerja karyawan atau staff yang bersangkutan dalam pengelolaan data pembayaran administrasi, perizinan santri serta untuk memberikan informasi kepada orang tua atau wali santri mengenai laporan dan bukti kuitansi pembayaran administrasi sekolah setiap bulannya. Pengembangan sistem menggunakan metode *waterfall* dengan Bahasa pemrograman *PHP* dan *MYSQL* sebagai *database*. Hasil penelitian yang telah dilakukan adalah aplikasi pembayaran dan perizinan santri berbasis *website* yang dapat memudahkan santri dan petugas dikembangkan dengan *website* yang dapat diakses dengan menggunakan *browser* sehingga memberikan kemudahan akses santri dalam mengurus pembayaran dan perizinan di pondok pesantren[3].

Penelitian lainnya ditulis oleh Rizki Wahyudi dan Krisna Rhinaldi pada tahun 2018. Bertujuan untuk memudahkan bendahara dalam pengelolaan data dalam proses pembayaran administrasi santri yang terintegrasi dengan *SMS Gateway*. Dikembangkan dengan metode *waterfall*, sistem ini berbasis desktop dengan proses pengujian menggunakan pendekatan *alpha testing* (merupakan bentuk dari pengujian untuk sisi pengembangan untuk fungsional dari sistem menggunakan metode *black box*) dan *beta testing* (merupakan pengujian untuk penggunaan akhir dan kelayakan hasil dari aplikasi yang telah dibuat). Hasil pengujian menggunakan *beta testing* menunjukkan presentasi 95,5%. Hasil penelitian adalah aplikasi pembayaran administrasi desktop dapat diimplementasikan pada pondok pesantren Darussalam[4].

Penelitian lainnya yang juga digunakan sebagai referensi yang ditulis oleh Muhamad Sabar, Agus Heryanto, Fuji Lestari tahun 2019. Sistem ini bertujuan untuk memudahkan orang tua maupun wali santri dalam memperoleh informasi santri selama di pondok dan sebagai sarana monitoring orang tua terhadap kegiatan anaknya di pondok pesantren. Dikembangkan berbasis android dengan metode *waterfall* dibangun dengan konsep dasar pemrograman *PHP*, *MYSQL*, Basis Data, *Web Server*, *OOP (Object Oriented Programming)*, pengujian dilakukan dengan unit testing dimana pengujian fungsionalitas sistem dilakukan

pada setiap unit setelah seluruh sistem telah diintegrasikan dan diuji untuk mengecek setiap *error* program, kegagalan dalam menampilkan perintah *user* maupun kesalahan tampilan program. Hasil dari penelitian ini Sistem informasi yang membantu orang tua santri maupun wali santri dalam memantau hasil pembelajaran anaknya selama di pondok pesantren dan membantu memudahkan pondok pesantren dalam manajemen penyampaian informasi dan pencatatan data santri[5].

Penelitian lainnya ditulis oleh Muhammad Zikri Al Fajri tahun 2021. Bertujuan untuk memudahkan santri dalam proses pembayaran administrasi dan membantu pihak yayasan pengurus pondok dalam menyebarkan informasi pembayaran administrasi yang dilakukan oleh santri di Pondok Pesantren Qiroatul Qur'an Bungo. Metode yang digunakan *waterfall* dan *OOAD (object oriented analisis design)* yang lebih menekankan pada pendekatan objek serta menggunakan tools *UML (Usecase, Activity Diagram dan Class Diagram)*. Hasilnya yaitu sistem informasi administrasi pembayaran santri yang dapat difungsikan dan dikelola oleh pondok pesantren[6].

Dapat disimpulkan dari hasil penelitian yang telah ada sebelumnya, pada sistem yang akan saya kembangkan berfokus pada sistem administrasi registrasi ulang santri baru, pembayaran perbulan, heregistrasi yang nantinya sistem dapat digunakan santri dan orang tua maupun wali santri dalam mengecek pembayaran yang telah dilakukan. Santri dan wali santri dapat memantau kekurangan pembayaran perbulan, pembayaran registrasi ulang serta heregistrasi. Perbedaan sistem yang saya kembangkan dengan sistem yang telah ada sebelumnya terdapat pengelolaan untuk pendapatan dan pengeluaran pondok pesantren sehingga mempermudah pengurus dalam melaporkan total dari pembayaran administrasi. Sistem berbasis *website*, dikembangkan menggunakan bahasa pemrograman *PHP* serta *MYSQL* sebagai *database* dengan metode pengembangan sistem *waterfall*.

2.2 Landasan Teori

2.2.1 Sistem Informasi

Sistem adalah kumpulan orang yang bekerja sama dengan adanya ketentuan atau aturan yang terstruktur dan sistematis sehingga membentuk satu kesatuan melaksanakan untuk mencapai tujuan. Sistem memiliki beberapa karakteristik atau sifat yang terdiri dari komponen sistem, batasan sistem, lingkungan luar sistem, penghubung sistem, input sistem, output sistem, pengolahan sistem dan sasaran sistem[7].

Sistem Informasi merupakan suatu sistem yang dibuat oleh seseorang yang terdiri dari unsur-unsur dalam organisasi untuk mencapai suatu tujuan yaitu menyajikan informasi. Sistem informasi di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi perusahaan, mendukung operasi manajemen, bersifat manajerial, dan kegiatan strategi dari organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan[8].

2.2.2 Administrasi

Administrasi adalah suatu proses penyelenggaraan serta pengurusan dari kegiatan dengan tujuan menyediakan keterangan bagi pihak yang membutuhkan data dari informasi serta mempermudah dalam memperoleh beragam informasi yang dibutuhkan secara keseluruhan. Menurut KBBI administrasi adalah usaha yang dilakukan meliputi penetapan tujuan serta penetapan langkah dari penyelenggaraan pembinaan organisasi[9].

Fungsi administrasi yaitu :

1. Perencanaan / *Planning* adalah penyusunan data dalam kegiatan administrasi. Kegiatan tersebut meliputi pengumpulan data, pengolahan data dan penyusunan perencanaan.
2. Pengorganisasian / *Organizing* adalah aktivitas dalam membuat hubungan kerja yang baik antara orang-orang sehingga mewujudkan kesatuan usaha dalam mencapai tujuan yang telah disusun.
3. Pembagian Tugas / *Staffing* adalah manajemen sumber daya manusia dalam suatu organisasi mulai dari membukakan lowongan tenaga kerja sampai dengan pengembangan *skill* tenaga kerja sehingga memberi daya guna yang optimal dalam organisasi.
4. Pengarahan / *Directing* adalah usaha dalam memberi arahan maupun bimbingan terhadap tugas, saran, perintah agar dapat terlaksana kinerja yang diharapkan sesuai dengan apa yang telah ditetapkan.

5. *Kordinasi / Coordinating* adalah kegiatan yang bertujuan untuk mengurangi terjadinya resiko kecelakaan, kesalahan pengoprasian, ketidaksesuaian dari kegiatan yang dilakukan dengan cara mengatur pekerjaan bawahan sesuai keahliannya sehingga terdapat kerjasama yang terarah dalam untuk mencapai tujuan yang direncanakan.
6. *Reporting* adalah penyampaian dari *progress* hasil atau laporan dari kegiatan kepada atasan dalam bentuk lisan maupun tulisan, sehingga atasan memperoleh gambaran tentang kinerja tugas yang telah diselesaikan.
7. *Budgeting* adalah kegiatan mengelola hasil dari rencana yang telah disusun untuk diwujudkan dengan adanya pembiayaan keuangan atau menggunakan anggaran.

2.2.3 Pondok Pesantren

Pondok Pesantren merupakan lembaga pendidikan dan pusat pengajaran agama islam dimana para siswa atau santri akan tinggal bersama di sebuah asrama yang disediakan oleh pondok pesantren bertujuan untuk belajar dibawah bimbingan guru atau kiai. Pesantren bertujuan untuk mempelajari ilmu agama islam dengan menekankan pada pembentukan nilai-nilai keislaman dengan menitik beratkan pada pembentukan akhlak yang baik[10].

Metode pengajaran di pondok pesantren memiliki beberapa tingkatan dalam pembelajaran yang disesuaikan dengan pemahaman santri. Pemahaman tersebut disesuaikan dengan usia santri saat memulai mengaji atau memulai belajar. Tingkatan tersebut diantaranya Ula (setara dengan Sekolah Dasar), Wusta (setara dengan Sekolah Menengah Pertama), Ulya (setara dengan Sekolah Menengah Atas). Dari setiap tingkatan tersebut memiliki metode pengajaran yang berbeda.

2.2.4 Rekayasa Perangkat Lunak

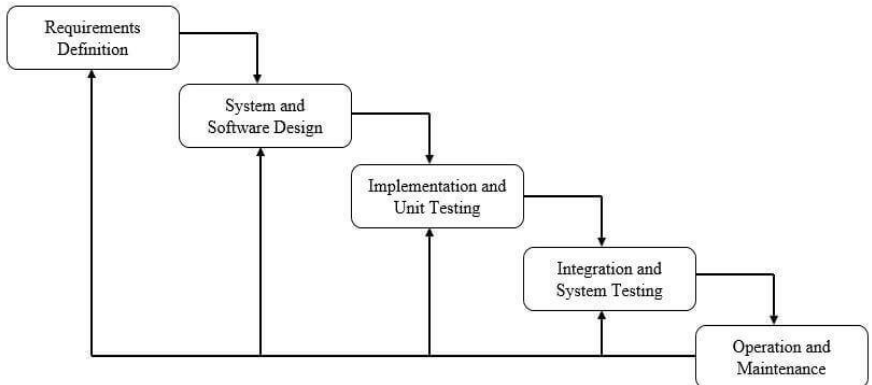
Rekayasa Perangkat Lunak merupakan ilmu yang membahas mengenai aspek produksi perangkat lunak, mulai dari tahap pertamakali pengembangan sistem yaitu tentang analisa kebutuhan pengguna, menentukan spesifikasi dari kebutuhan pengguna, design, menulis kode program atau *coding*, pengujian sampai pemeliharaan sistem setelah digunakan. Rekayasa Perangkat Lunak bukan hanya menjelaskan relasi dengan cara pembuatan program komputer namun semua hal yang berkaitan dengan proses produksi seperti manajemen proyek, anggaran biaya, sumber daya manusia, perencanaan kinerja, penjadwalan, sampai

dengan mengajarkan cara pengoperasian sistem kepada pengguna merupakan bagian dari Rekayasa Perangkat Lunak.

A. Metode Pengembangan Sistem

Metode penelitian yang digunakan merupakan metode *waterfall*. *Waterfall* merupakan model pengembangan perangkat lunak dengan metode yang disusun secara sistematis dari satu tahap ke tahap

selanjutnya berurutan seperti air terjun. Metode *waterfall* menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut. Model *waterfall* terdiri dari 5 tahapan, yaitu Analisa Kebutuhan, System and Desain Program, Penulisan Kode Program, Pengujian Program, Penerapan Program dan Pemeliharaan. Model *waterfall* ditunjukkan dapat dilihat pada Gambar 2.1



Gambar 2.1 Pemodelan *Waterfall*

Penjelasan tahapan-tahapan dari metode *waterfall*[11]:

1. *Requirements Definition*

Langkah ini merupakan analisa terhadap kebutuhan sistem. Pengumpulan data dilakukan dengan sebuah penelitian, wawancara atau *study literatur* dari berbagai sumber. Pada tahap ini informasi akan digali sebanyak banyaknya dari narasumber sehingga akan tercipta sebuah sistem komputer yang bisa melakukan tugas-tugas yang diinginkan oleh narasumber tersebut. Tahapan ini akan menghasilkan dokumen *user requirment* atau sebagai data yang berkaitan dengan keinginan pengguna akan

sistem ini dibuat. Dokumen inilah yang menjadi pedoman untuk sistem analisis dalam pengkodean kedalam bahasa pemrograman.

2. *System and Software Design*

Pada Tahap ini akan dibuat suatu kerangka dari sistem berdasarkan persyaratan-persyaratan yang telah dilalui. Selain itu proses selanjutnya akan dilakukan identifikasi dan penggambaran terhadap abstraksi dasar sistem perangkat lunak beserta hubungan-hubungannya.

3. *Implementation and Unit Testing*

Coding merupakan penerjemah design dalam bahasa yang bisa dikenali oleh komputer. Tahapan inilah yang merupakan tahapan secara nyata dalam mengerjakan suatu sistem, sistem dapat diakses dengan *browser*. Setelah pengkodean selesai maka akan dilakukan pengujian terhadap sistem yang telah dibuat. Apakah sudah memenuhi verifikasi program sesuai dengan spesifikasinya.

4. *Integration and System Testing*

Unit individu program akan digabung dan diuji dalam sebuah sistem lengkap untuk mengetahui apakah kebutuhan perangkat lunak sudah terpenuhi atau belum. Setelah melakukan tahapan ini maka sistem yang sudah jadi akan digunakan oleh user.

5. *Operation and Maintenance*

Perangkat lunak yang sudah dikembangkan akan disampaikan kepada pengguna program dan akan dievaluasi sehingga akan timbul perubahan. Perubahan terjadi karena mengalami *error* program karena perangkat lunak harus menyesuaikan dengan lingkungan (perangkat tambahan, *update* program atau dengan sistem operasi yang lebih mutakhir) atau karena pengguna memerlukan pembaharuan fungsional dari sistem yang dibuat.

B. Metode Pengujian Sistem

Metode pengujian sistem yang digunakan adalah dengan metode *black box testing*. Metode ini merupakan suatu pengujian sistem yang berfokus pada spesifikasi fungsional dari perangkat lunak, penguji dapat menjabarkan hasil dari kumpulan pengecekan kondisi sistem dan melakukan pengujian program khususnya pada spesifikasi fungsional program yang berjalan apakah sesuai harapan atau terjadi suatu *error* tampilan program[12].

Ciri-ciri *black box testing* :

1. *Black box testing* pengecekan dilakukan khususnya pada kebutuhan fungsional *software*, berdasarkan pada spesifikasi kebutuhan pengguna dari *software* yang telah dibuat.
2. *Black box testing* pada saat melakukan pengujian tanpa dibekali pengetahuan secara detail mengenai struktur internal dari sistem atau komponen yang dites.
3. *Black box testing* merupakan pendekatan pelengkap dalam mencakup kesalahan program.

Kategori *error* yang akan diketahui melalui *black box testing* :

1. Fungsi-fungsi yang salah, tidak sesuai atau hilang,
2. Inisialisasi dan kesalahan terminal,
3. Kesalahan kinerja program yang ditampilkan,
4. Kesalahan tampilan program atau *interface*,
5. Kesalahan dalam akses database atau struktur data eksternal.

2.2.5 Pemograman Berorientasi Objek

Metodologi berorientasi objek adalah sebagai *Metodologi* berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan proyek yang berisi data dan operasi. *Metodologi* berorientasi objek merupakan suatu cara bagaimana sistem perangkat lunak dibangun melalui pendekatan objek secara sistematis.

Sistem berorientasi objek merupakan sebuah sistem yang dibangun dengan berdasarkan metode berorientasi objek adalah sebuah sistem yang komponennya dibungkus menjadi kelompok data dan fungsi. Setiap komponen dalam sistem tersebut dapat mewarisi atribut, sifat, komponen lainnya, dan dapat berinteraksi satu sama lain.

Berikut ini adalah beberapa konsep dasar yang harus dipahami tentang metodologi berorientasi objek:

1. Kelas (*class*)
Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama yang mungkin lahir atau diciptakan dan kelas tersebut. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (*relationship*) dan arti. Suatu kelas dapat diturunkan dan kelas yang lain, dimana atribut dan kelas semula dapat diwariskan ke kelas yang baru.

2. **Objek (*object*)**
Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda, manusia, suatu organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.
3. **Metode (*method*)**
Operasi atau metode atau *method* pada sebuah kelas hampir sama dengan fungsi atau prosedur pada metodologi struktural. Sebuah kelas boleh memiliki lebih dari satu metode atau operasi. Metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri. Operasi atau metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan oleh objek.
4. **Atribut (*attribute*)**
Atribut dari sebuah kelas adalah variabel global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut dipunyai secara individual oleh sebuah objek, misalnya berat, jenis, nama dan sebagainya. Atribut sebaliknya bersifat privat untuk menjaga konsep enkapsulasi.
5. **Abstraksi (*abstract*)**
Prinsip untuk mempresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan.
6. **Enkapsulasi (*encapsulation*)**
Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya.
7. **Pewarisan (*inheritance*)**
Mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain sebagai bagian dan dirinya.
8. **Antarmuka (*interface*)**
Antarmuka atau *interface* sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah *interface* dapat diimplementasikan oleh kelas lain.


Pada pemrograman berorientasi objek, *UML* digunakan untuk pemodelan sistem. *UML* adalah bahasa untuk menspesifikasi, memvisualisasi, membangun dan mendokumentasikan *artifacts* (bagian dari informasi yang digunakan atau dihasilkan oleh proses pembuatan perangkat lunak, *artifact* tersebut dapat berupa model, deskripsi atau perangkat lunak) dari sistem perangkat lunak, seperti pada pemodelan bisnis dan sistem non perangkat lunak lainnya.






A. *Use Case Diagram*





Use Case adalah deskripsi fungsi dari sebuah sistem dari sudut pandang pengguna. *Use Case* bekerja dengan cara mendeskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita yang dituangkan dalam bentuk bagan tentang bagaimana sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem tersebut *scenario* sedangkan pengguna disebut *actor*. *Actor* adalah sebuah peran yang biasanya dimainkan oleh pengguna melalui interaksi yang dilakukan dengan sistem. Definisi lain *use case* adalah abstraksi dari interaksi antara sistem dan *actor*. *Use case* dibuat berdasarkan keperluan *actor*.

Berdasarkan definisi diatas maka dapat disimpulkan bahwa *Use Case* adalah kontruks untuk menjelaskan ciri-ciri bagaimana sistem akan terlihat dimata pengguna yang terdiri dari sekumpulan *scenario* dan *actor*. Sedangkan *use case* diagram menyediakan ruang komunikasi diantara analis dan pengguna serta analis dan klien. Simbol *Use Case* dapat dilihat pada Table 2.1.

Tabel 2.1 Simbol dan Fungsi *Use Case Diagram*

No.	Simbol	Keterangan
1.	 <p data-bbox="367 1281 423 1305"><i>Actor</i></p>	Merupakan simbol yang digunakan untuk menjelaskan spesifikasi himpunan peran yang pengguna mainkan ketika berinteraksi dengan <i>use case</i> .

2.	 <i>Dependency</i>	Merupakan simbol yang berhubungan antara perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri(<i>independent</i>).
3.	 <i>Generalization</i>	Merupakan simbol yang berhubungan antara objek anak (<i>descendent</i>) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk.
4.	 <i>Include</i>	Merupakan simbol yang men-spesifikasikan bahwa <i>use case</i> sumber secara eksplisit.
5.	 <i>Extend</i>	Merupakan simbol yang menjelaskan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan.
6.	 <i>Association</i>	Merupakan simbol yang menghubungkan antara objek satu dengan objek lainnya

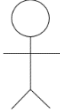

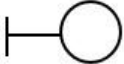


7.	 <i>System</i>	Merupakan simbol yang men-spesifikasikan paket yang menampilkan sistem secara terbatas
8.	 <i>Use Case</i>	Merupakan simbol yang berisi tentang deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu <i>actor</i> .
9.	 <i>Collaboration</i>	Merupakan simbol yang berisi tentang Interaksi aturan-aturan dan elemen lain yang bekerjasama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10.	 <i>Note</i>	Merupakan simbol yang berisi tentang Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi


B. *Sequence Diagram*

Sequence Diagram digunakan untuk menggambarkan perilaku pada sebuah *scenario*. Diagram ini menunjukkan sejumlah obyek dan message yang diletakan antara obyek-obyek didalam *use case*.

Komponen utama *sequence diagram* yaitu objek yang dituliskan dengan kotak segiempat bernama. *Message* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan *progress vertical*. *Sequence Diagram* menambahkan dimensi waktu pada interaksi diantara obyek. Simbol-simbol yang dipakai dalam pembuatan *sequence diagram* dapat dilihat pada Berikut simbol-simbol yang ada pada diagram *sequence*.

Tabel 2.2 Simbol dan Fungsi *Sequence Diagram*

No.	Simbol	Nama Simbol dan Keterangan
1.	 <i>Actor</i>	Deskripsi dari urutan aksi-aksi yang menghasilkan suatu hasil yang ditampilkan oleh sistem.
2.	 <i>Entity Class</i>	Menyimpan data atau informasi berupa objek model pada sistem.
3.	 <i>Boundary Class</i>	Menggambarkan sebuah gambaran dari form atau daerah yang membatasi suatu sistem dengan sistem lain dengan lingkungan luar.
4.	 <i>Control Class</i>	Menggambarkan hubungan antara boundary dengan tabel.
5.	 <i>A Focus of Control & Life Line</i>	Menggambarkan tempat mulai dan berakhirnya message.

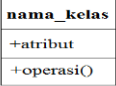
No.	Simbol	Nama Simbol dan Keterangan
6.		Menggambarkan pengiriman pesan

C. *Class Diagram*

Class Diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas. Sedangkan operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas[15]. Diagram kelas dibuat agar programmer membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron.

Class diagram membantu dalam visualisasi struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak. *Class diagram* memperlihatkan hubungan antar kelas dan penjelasan detail tiap – tiap kelas di dalam model desain (dalam *logical view*) dari suatu sistem. Selama proses analisis, class diagram memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. Selama proses analisis, *class diagram* memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. Selama tahap decían, *class diagram* berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat tabel 2.3.

Tabel 2.3 Simbol dan Fungsi *Class Diagram*

No.	Simbol	Nama	Keterangan
1.		Kelas	Kelas pada struktur sistem.

2.		Asosiasi/ <i>Association</i>	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan <i>multiplicity</i> .
3.		Asosiasi berarah/ <i>Directed association</i>	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain.
4.		Generalisasi	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum khusus).
5.		Kebergantungan <i>Dependency</i>	Relasi antarkelas dengan makna kebergantungan antarkelas.
6.		Agregasi / <i>Aggregation</i>	Relasi antarkelas dengan makna semua-bagian (<i>whole-part</i>).

2.2.6 Basis Data

Basis data atau database adalah kumpulan file-file yang mempunyai kaitan antara satu file dengan file lain sehingga membentuk satu bangun data untuk penginformasikan suatu perusahaan instansi, dalam bahasan tertentu.

MySQL adalah sebuah program *database* server yang mampu

menerima dan mengirimkan datanya dengan sangat cepat, multi user [16] serta menggunakan perintah standar *Structured Query Language (SQL)*. *MySQL* dapat berperan sebagai client sehingga sering disebut *database client / server* yang *open source* dengan kemampuan dapat berjalan baik di Operasi Sistem maupun dengan *Platform Windows* maupun *Linux*.


MySQL sebenarnya merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu *SQL (Structured Query Language)*. *SQL* adalah suatu bahasa (*language*) yang digunakan untuk mengakses data di dalam sebuah database relasional. *SQL* sering juga disebut dengan istilah query, dan bahasa *SQL* secara praktiknya digunakan sebagai bahasa standar untuk manajemen database relasional. Hingga saat ini hampir seluruh server database atau software database mengenal dan mengerti bahasa *SQL*.



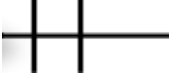
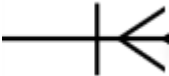
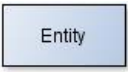
A. ERD (*Entity Relationship Diagram*)

Pengertian *ERD* adalah suatu rancangan atau bentuk hubungan suatu kegiatan di dalam sistem yang berkaitan langsung dan mempunyai fungsi di dalam proses tersebut. *ERD* adalah suatu pemodelan dari basis data relasional yang didasarkan atas persepsi di dalam dunia nyata, dunia ini senantiasa terdiri dari sekumpulan objek yang saling berhubungan antara satu dengan yang lainnya[16]. Suatu objek disebut *entity* dan hubungan yang dimilikinya disebut *relationship*. Suatu *entity* bersifat unik dan memiliki atribut sebagai pembeda dengan *entity* lainnya.

Berikut merupakan simbol-simbol dari ERD :

Tabel 2.4 Simbol *ERD (Entity Relationship Diagram)*

No	Keterangan
<p data-bbox="238 1058 445 1082">1. Relasi/Hubungan</p> 	<p data-bbox="507 1058 921 1201">Hubungan yang terjadi antara 1 entitas atau lebih yang tidak mempunyai fisik tetapi hanya sebagai konseptual. Dan untuk mengetahui jenis hubungan yang ada antara 2 file.</p>

<p>2. Atribut</p> 	<p>Atribut ialah karakteristik dari entitas atau relasi yang menyediakan penjelasan detil tentang entitas atau relasi tersebut. Berfungsi untuk memperjelas atribut yang dimiliki oleh sebuah entitas. Atribut memiliki bentuk lingkaran lebih tepatnya elips.</p>
<p>3. Alur</p> 	<p>Alur memiliki fungsi untuk menghubungkan atribut dengan entitas dan entitas dengan relasi. Dan berbentuk garis.</p>
<p>4. <i>One to One</i></p> 	<p><i>One to One</i>, digunakan untuk menghubungkan antar entitas dengan hubungan satu ke satu</p>
<p>5. <i>One to Many</i></p> 	<p><i>One to Many</i> atau <i>Many to One</i>, digunakan untuk menghubungkan antar entitas dengan hubungan satu ke banyak atau sebaliknya</p>
<p>6. Entitas (<i>Entity</i>)</p> 	<p>Entitas ialah suatu objek yang dapat dibedakan dengan objek lainnya. Entitas berfungsi untuk memberikan identitas pada entitas yang memiliki label dan nama. Entitas memiliki bentuk persegi panjang.</p>

Relasi antar table sebagai berikut:

1. Hubungan *One-to-One*, masing-masing tabel hanya terdapat satu data yang saling berhubungan.
2. Hubungan *One-to-Many*, berelasi dengan banyak *record* pada tabel yang lain.
3. Hubungan *Many-to-Many*, banyak *record* pada sebuah tabel berhubungan dengan banyak *record* pada tabel yang lain.

2.2.7 Flowchart

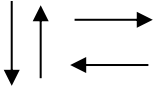
Flowchart adalah bagan (*chart*) yang menunjukkan alir (*flow*) di dalam prosedur sistem atau program secara logika[13]. Bagan alir (*flowchart*) digunakan terutama untuk alat bantu komunikasi dan untuk dokumentasi. Gambaran ini dinyatakan dengan simbol. Dengan demikian setiap simbol menggambarkan proses tertentu. Sedangkan hubungan antar proses digambarkan dengan garis penghubung. *Flowchart* ini merupakan langkah awal pembuatan program. Dengan adanya *flowchart* urutan poses kegiatan menjadi lebih jelas. Jika ada penambahan proses maka dapat dilakukan lebih mudah. Setelah *flowchart* selesai disusun, selanjutnya programmer menerjemahkannya ke bentuk program dengan bahasa pemrograman [9].

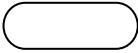

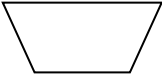
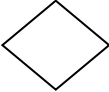


Simbol yang dipakai dalam *flowchart* dibagi menjadi 3 kelompok, yaitu :

1. *Flow direction symbols*
Digunakan untuk menghubungkan simbol satu dengan yang lain (*connecting line*).
2. *Processing symbols*
Menunjukkan jenis operasi pengolahan dalam suatu proses atau prosedur.
3. *Input / Output symbols*
Menunjukkan jenis peralatan yang digunakan sebagai media input atau output.

Flowchart disusun dengan simbol-simbol. Simbol ini dipakai sebagai alat bantu menggambarkan proses di dalam program. Simbol-simbol yang dipakai antara lain :

Tabel 2.5 Simbol dan Fungsi *Flowchart*

No	Simbol	Keterangan
1.	 <p style="text-align: center;">Flow Direction Symbol</p>	Yaitu simbol yang digunakan untuk menghubungkan antara simbol yang satu dengan simbol yang lain. Simbol ini disebut juga <i>connecting line</i> .

2.	 <p>Terminator Symbol</p>	Yaitu simbol yang bertujuan untuk permulaan (<i>start</i>) atau berakhirnya (<i>end</i>) suatu kegiatan.
3.	 <p>Processing Symbol</p>	Yaitu simbol yang menunjukkan pengolahan yang dilakukan oleh komputer.
4.	 <p>Manual Operation Symbol</p>	Yaitu simbol yang menunjukkan pengolahan data yang tidak dilakukan oleh komputer dan biasanya dilakukan oleh user.
5.	 <p>Decision Symbol</p>	Yaitu simbol untuk pemilihan keputusan dari proses berdasarkan kondisi yang ada.
6.	 <p>Input-Output Symbol</p>	Yaitu simbol yang menyatakan proses <i>input</i> dan <i>output</i> tanpa tergantung dari jenis peralatannya.
7.	 <p>Document Symbol</p>	Yaitu simbol yang menyatakan input berasal dari dokumen dalam bentuk kertas atau <i>output</i> dicetak ke kertas.