BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Penelitian sebelumnya yang telah dilakukan oleh Mohammad Ahmadar, Perwito dan, Candra Taufik dengan judul "Perancangan Sistem Informasi Penjualan Berbasis *Web* Pada Rahayu Photo Copy Dengan *Database* MySQL" Penelitian ini bertujuan untuk membangun sebuah sistem informasi penjualan pada Rahayu Photo Copy berbasis *web*. Sistem ini dikembangkan menggunakan bahasa pemrograman PHP dengan menggunakan MySQL sebagai *database*. Metode pengumpulan data yang dilakukan yaitu dengan cara wawancara, observasi, dan ditambah dengan studi pustaka yang memiliki kaitan dengan masalah yang dihadapi. Sedangkan metode penelitian yang digunakan adalah penelitian kualitatif yang bersifat deskriptif dengan sitem pengembangan model Waterfall[2].

Kemudian, penelitian lainnya yang telah dilakukan oleh T.Bayu Kurniawan, dan Syarifuddin dengan judul "Perancangan Sistem Aplikasi Pemesanan Makanan Dan Minuman Pada Cafetaria No Caffe Di Tanjung Balai Karimun Menggunakan Bahasa Pemrograman PHP Dan MYSQL" Sistem Pemesanan Makanan dan Minuman dengan pemodelan perangkat lunak yang di gunakan adalah UML dan metode pengembangan yang digunakan adalah Prototype. Dengan adanya Sistem Pemesanan Makanan 8 dan Minuman, Cafetaria No Caffe dapat memperluas jaringan usaha dan memperbesar peluang untuk mendapatkan pelanggan, dapat mengurangi kebutuhan akan modal kerja tanpa harus membuka cabang baru, sehingga biaya tidak tumbuh secara proporsional dengan pertumbuhan bisnis[3].

Selanjutnya, Penelitian lainnya yang telah dilakukan oleh Syahirun Alam dan Rusdi dengan judul "Sistem Informasi *Coffeshop* Pada A Lot Of Caffe Berbasis *Web*" Sistem Pemesanan Makanan dan Minuman dengan pemodelan perangkat lunak yang di gunakan adalah UML. Dan metode atau jenis penelitian yang digunakan berawal dari penelitian kepustakaan, penelitian lapangan, studi literatur, pengumpulan data, analisis, perancangan, pengujian, implementasi, sampai dengan tahap penyelesaian[4].

Berdasarkan pada penelitian yang telah dilakukan tersebut diatas, dapat disimpulkan bahwa sistem yang akan penulis kembangkan akan berfokus pada membantu mempermudah dalam proses melihat ketersediaan menu, mencatat pesanan, mengirim struk pesanan. Sistem ini akan berbasis

website dan memanfaatkan dengan *framework Code*Igniter 3, menggunakan bahasa pemrograman PHP, *database* menggunakan MySQL, dan metode pengembangan Waterfall. Dan yang membedakan sistem ini dengan penelitian sebelumnya, penulis ingin mengembangkan sistem dengan memanfaatkan *QR Code* dimana nantinya *user* (konsumen atau pembeli) dapat melakukan *self-order* atau pemesanan secara mandiri sehingga, kasir tidak perlu meng*input*kan ulang pesanan yang diminta oleh konsumen atau pembeli, melainkan hanya konfirmasi pesanan, kemudian mengirim struk pesanan.

2.2 Landasan Teori

2.2.1 Sistem Informasi

Sistem informasi memiliki tujuan utama untuk menyimpan, memproses, dan mengkomunikasikan informasi dengan efisien. Dengan bantuan teknologi dan perangkat yang relevan, sistem ini mengubah data menjadi informasi yang berarti dan bermanfaat. Fungsinya tidak terbatas hanya pada penyimpanan data, tetapi juga mencakup proses pengolahan data yang lebih kompleks untuk menghasilkan informasi yang bernilai. Keberadaan sistem informasi menjadi kunci penting dalam berbagai sektor, seperti bisnis, pendidikan, kesehatan, dan pemerintahan. Dalam konteks masing-masing sektor, sistem informasi membantu dalam mengelola transaksi, mengoptimalkan operasional, memantau perkembangan siswa, menyimpan catatan medis, meningkatkan layanan publik, dan menghadapi tantangan masa depan yang berkelanjutan[5].

2.2.2 Sistem Informasi Pemesanan

sistem informasi pemesanan adalah suatu rangkaian prosedur dan komponen organisasi yang didesain untuk membantu pengambilan keputusan dan pengendalian dalam aktivitas pemesanan dan penjualan barang atau jasa. Sistem ini berfungsi untuk mempermudah proses pemesanan, mengoordinasikan interaksi antara konsumen dan pihak penjual, serta memberikan informasi yang relevan kepada pihak yang berwenang dalam organisasi untuk mengoptimalkan efisiensi, pengelolaan persediaan, dan pengambilan keputusan strategis terkait penjualan. Keseluruhan, sistem informasi penjualan berperan penting dalam meningkatkan pengalaman pelanggan, mengurangi potensi kesalahan, dan menciptakan hubungan yang lebih lancar antara pelanggan dan penyedia barang atau jasa[6].

2.2.3 Rekayasa Perangkat Lunak

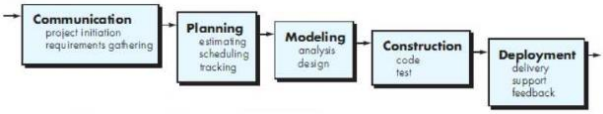
Perangkat lunak merupakan sebuah aplikasi komputer yang melibatkan panduan tertulis seperti model desain, dokumen kebutuhan, serta petunjuk penggunaan. Sebuah program komputer hanya dianggap sebagai perangkat lunak ketika disertai dengan dokumentasinya. Rekayasa perangkat lunak, atau yang dikenal sebagai Software Engineering, adalah proses pembuatan perangkat lunak yang mengikuti prinsip-prinsip rekayasa. Tujuannya adalah untuk menghasilkan perangkat lunak yang memiliki nilai ekonomi, dapat diandalkan, dan beroperasi secara efisien dengan menggunakan mesin[7].

Proses rekayasa perangkat lunak dilakukan selama pembangunan perangkat lunak. Dalam proses ini, terdapat tahapan-tahapan yang dimulai dengan analisis, desain, implementasi, dan pengujian. Siklus ini dapat diulang beberapa kali hingga perangkat lunak memenuhi persyaratan dan kebutuhan dari pelanggan atau pengguna[7].

1. *Software Development Life Cycle*

Pada proses pengembangan sistem ini, penulis menggunakan metode Software Development Life Cycle (SDLC) yaitu proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model dan metodologi yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya. SDLC memiliki beberapa model dalam penerapan tahapan prosesnya, seperti Waterfall[7]. Model waterfall adalah model klasik yang bersifat sistematis, berurutan dalam membangun software. Nama model ini sebenarnya adalah "*Linear Sequential Model*". Model ini sering disebut juga dengan "*classic life cycle*" atau metode *waterfall*[8].

Model ini termasuk ke dalam model generic pada rekayasa perangkat lunak dan pertama kali diperkenalkan oleh Winston Royce sekitar tahun 1970 sehingga sering dianggap kuno, tetapi merupakan model yang paling banyak dipakai dalam Software Engineering (SE). Model ini melakukan pendekatan secara sistematis dan berurutan. Disebut dengan waterfall karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan. Metode waterfall dapat dilhat pada gambar 2.1[9].



Gambar 2. 1 *Model Waterfall*

Fase-fase dalam Waterfall Model menurut referensi Pressman (2015:17):

1. Communication

Pada tahap ini dilakukan komunikasi dengan pihak tempat studi kasus demi memahami dan mencapai tujuan yang ingin dicapai, dan menganalisis terhadap apa yang menjadi permasalahan. Wawancara dilakukan untuk mengumpulkan data mengenai kebutuhan awal sistem. Data rekomendasi yang dihasilkan dalam sistem ini dilengkapi dengan informasi seputar *Coffeshop* seperti daftar menu, omset, dan lain-lain. Studi literatur dilaksanakan untuk mengumpulkan hasil riset dan informasi lain yang bersangkutan dengan pengembangan produk yang direncanakan[9].

2. Planning

Setelah kebutuhan untuk pengembangan diketahui, maka akan dilakukan perancangan sistem. Tahap ini menjelaskan tentang estimasi tugas-tugas teknis yang akan dilakukan, resiko - resiko yang dapat terjadi, sumber daya yang diperlukan dalam membuat sistem, produk kerja yang ingin dihasilkan, penjadwalan kerja yang akan dilaksanakan, dan tracking proses pengerjaan sistem[9].

3. Modelling

Tahapan ini adalah tahap perancangan dan permodelan arsitektur sistem yang berfokus pada perancangan struktur data, arsitektur software, tampilan *interface*, dan algoritma program. Tujuannya untuk lebih memahami gambaran besar dari apa yang akan dikerjakan. Pada tahap ini, desain aplikasi yang telah dibuat diimplementasikan menjadi sebuah program. Pengembangan aplikasi ini menggunakan basis *web* dan diintegrasikan dengan *database* agar aplikasi bersifat dinamis[9].

4. Construction

Tahapan *Construction* ini merupakan proses penerjemahan bentuk desain menjadi kode atau bentuk/bahasa yang dapat dibaca oleh mesin. Dalam penelitian ini dilakukan pengujian. Sistem dilakukan verifikasi dan pengujian apakah sistem sepenuhnya atau sebagian memenuhi

persyaratan sistem. Metode pengujian yang digunakan adalah *blackbox* yaitu dengan cara meninjau *input* dan *output* sistem tanpa mengetahui programnya. Setelah ditemukan kekurangan maka dilakukan perbaikan. Apabila sistem telah memenuhi kriteria dan tidak perlu direvisi maka sistem siap untuk diujicobakan kepada *user*[9].

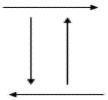
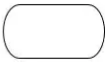
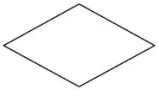
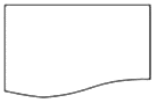
5. Deployment

Tahapan Deployment merupakan tahapan implementasi sistem ke klien, pemeliharaan sistem secara berkala, perbaikan sistem, evaluasi sistem, dan pengembangan sistem berdasarkan umpan balik yang diberikan agar sistem dapat tetap berjalan dan berkembang sesuai dengan fungsinya. Ini adalah tahap akhir dari metode waterfall. Perangkat lunak yang sudah selesai dibuat dijalankan serta dilakukan pemeliharaan. Pemeliharaan termasuk 12 dalam memperbaiki kesalahan yang tidak ditemukan pada langkah sebelumnya[9].

2. Flowchart

Flowchart merupakan representasi visual dari algoritma atau langkah-langkah dalam sebuah program atau sistem. Flowchart menggunakan bagan-bagan arus kerja yang menggambarkan secara detail langkah-langkah prosedur dalam menyelesaikan suatu masalah. Selain itu, flowchart juga menunjukkan hubungan antara berbagai proses dalam program. Keberadaan flowchart membantu analis dan programmer dalam memecah masalah menjadi segmen-segmen yang lebih kecil, sehingga memudahkan pemahaman dan analisis proses secara terperinci. Flowchart juga membantu dalam mengidentifikasi alternatif-alternatif lain dalam operasional sistem atau program yang sedang dikembangkan, sehingga dapat membantu dalam pengambilan keputusan. Dengan adanya flowchart, proses pengembangan dan analisis program menjadi lebih efisien dan terstruktur[10]. Simbol-simbol flowchart yang dipakai sebagai alat bantu menggambarkan proses di dalam program dapat dilihat pada Tabel 2. 3[11].

Tabel 2. 1 Simbol *Flowchart*

No.	Simbol	Nama simbol	Keterangan
1		*Flow*	Menghubungkan antara simbol satu dengan simbol yang lain atau menyatakan jalannya arus kerja dalam suatu proses. Simbol arus ini sering disebut juga dengan *connecting line*.
2		*Terminal*	Menyatakan permulaan atau akhir suatu program.
3		*Proses*	Proses perhitungan atau proses pengolahan data yang dilakukan oleh komputer.
4		*Manual Operation*	Menyatakan suatu tindakan (proses) yang tidak dilakukan oleh komputer (manual).
5		*Decision*	Keputusan dalan program yang menunjukan suatu kondisi tertentu yang akan menghasilkan data kemungkinan jawaban, ya atau tidak.
6		*Input - Output*	Menyatakan proses *input* dan *output* data yang diproses atau informasi tanpa tergantung dengan jenis peralatannya.
7		*Document*	*Input* atau *Output* dalam format yang di cetak.

2.2.4 UML (*Unified Modeling Language*)

UML merupakan salah satu alat bantu yang digunakan sebagai standarisasi bahasa pemodelan dalam pembangunan sistem yang

dibangun dengan menggunakan teknik pemrograman berorientasi objek. Suatu sistem perangkat lunak membutuhkan sebuah pemodelan visual untuk menspesifikasikan, menggambarkan, membangun dan mendokumentasikan perangkat lunak tersebut. Hal tersebut dapat dilakukan menggunakan UML, karena merupakan bahasa visual sebagai pemodelan dan komunikasi terkait sebuah sistem dengan menggunakan *Diagram* dan teks pendukung lainya[7]. Macam dari UML antara lain:

1. *Use Case Diagram*

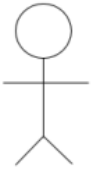
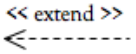
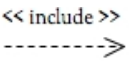
Use case Diagram adalah representasi visual yang menggambarkan hal-hal yang dapat dilakukan oleh aktor dalam menyelesaikan sebuah pekerjaan. Diagram ini berfungsi sebagai pemodelan perilaku sistem informasi yang akan dibuat. Use Case Diagram mendeskripsikan interaksi antara satu atau lebih aktor dengan sistem informasi yang sedang dikembangkan. Secara kasar, Diagram use case digunakan untuk mengidentifikasi fungsi-fungsi yang ada di dalam sebuah sistem dan mengetahui siapa saja yang berhak menggunakan fungsi-fungsi tersebut[12].

Dalam *Diagram* ini, aktor direpresentasikan dengan simbol manusia atau entitas luar lainnya, sementara *use case* menunjukkan tugas-tugas yang dapat dilakukan oleh aktor untuk mencapai tujuan tertentu dengan dukungan dari sistem informasi. *Use Case Diagram* merupakan alat komunikasi yang efektif antara tim pengembang dan pemangku kepentingan, membantu memahami interaksi pengguna dengan sistem,serta memberikan pandangan yang lebih jelas tentang fitur dan fungsionalitas sistem yang akan dibangun[12].

Berikut merupakan simbol-simbol pada *Use Case Diagram* yang terdapat pada tabel[12]:

Tabel 2. 2 Simbol *Use Case Diagram*

No.	Simbol	Nama Simbol	Keterangan
1		*Use case*	*Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor. Dinyatakan menggunakan kata kerja di awal frase nama *use case*

2		*Actor*	Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri. Walaupun simbol dari actor adalah orang,tetapi actor belum tentu merupakan orang, biasanya actor dinyatakan menggunakan kata benda di awal frase nama actor
3		*Association*	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau memiliki interaksi dengan aktor
4		*Extend*	Relasi tambahan ke sebuah *use case* dimana yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan.
5		*Generalization*	Hubungan generalisasi dan spesialisasi antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
6		*Include*	Relasi tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini.

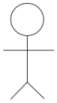
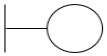
2. *Sequence Diagram*

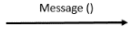
Sequence Diagram merupakan sebuah alat yang populer dalam pengembangan sistem informasi dengan pendekatan object-oriented yang digunakan untuk menampilkan interaksi antara objek-objek. Diagram ini berfungsi sebagai alat untuk menggambarkan interaksi dan urutan pesan

yang terjadi antara objek-objek dalam sistem, sehingga membantu dalam pemahaman dan analisis interaksi yang kompleks di dalam pengembangan sistem informasi[12].

Sequence Diagram merupakan jenis *Diagram* pada pemodelan UML (Unified Modeling Language) yang digunakan untuk menggambarkan interaksi antara sejumlah objek dalam urutan waktu. *Diagram* ini menunjukkan bagaimana objek-objek berkomunikasi satu sama lain melalui pertukaran pesan dan memperlihatkan urutan kronologis dari pesan-pesan tersebut. Dengan bantuan *sequence Diagram*, pengembang dan pemangku kepentingan dapat memvisualisasikan interaksi yang terjadi di dalam sistem secara lebih jelas dan detail, sehingga membantu dalam memahami alur logika danproses yang terjadi dalam sebuah sistem. Berikut beberapa simbol yang ada pada *sequence Diagram* dapat dilihat pada Tabel 2.2[13]:

Tabel 2. 3 Simbol *Sequence Diagram*

No.	Simbol	Nama Simbol	Keterangan
1		*Actor*	Aktor, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri.
2		*Boundary*	Menggambarkan sebuah *form*.
3		*Control Class*	Menghubungkan *boundary* dengan Tabel.
4		*Entity Class*	Menggambarkan hubungan kegiatan yang akan dilakukan.
5		*Lifeline*	*Lifeline, object entity,* antarmuka yang saling berinteraksi. Menyatakan kehidupan suatu objek.

6		*Message*	Spesifikasi dari komunikasi antar *objek* yang memuat informasi-informasi tentang aktivitas yang terjadi.

3. *Pengujian Black box*

Pengujian *black box*, juga dikenal sebagai behavioral testing, merupakan metode pengujian perangkat lunak yang berfokus pada fungsionalitas perangkat lunak tanpa memeriksa atau menguji kode program secara langsung. Dalam metode pengujian *black box*, penekanan diberikan pada pengetahuan tentang persyaratan fungsional perangkat lunak dan bukan pada detail implementasi kode. Tujuan dari pengujian *black box* adalah untuk memastikan apakah fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi dan persyaratan yang telah ditentukan. Dengan demikian, pengujian *black box* bertujuan untuk memverifikasi apakah perangkat lunak berperilaku sesuai dengan harapan dan spesifikasi yang telah ditetapkan, tanpa memerhatikan bagaimana kode programnya diimplementasikan[14].

Pengujian menggunakan *Black box* memiliki beberapa teknik antara lain*, Equivalence Partitions, Boundary Value Analysis, Comparison Testing, Sample Testing, Robustness Testing, Behavior Testing, Performance Testing, Requirement Testing, Endurance Testing dan Cause–Effect Relationship Testing.* Salah satu teknik yang mudah digunakan adalah teknik *Boundary Value Analysis*. Cara kerja Teknik ini yaitu dengan menentukan batasan-batasan yang akan di*input*kan atau dilakukan pada aplikasi, selanjutnya hasil keluaran atau respon pada aplikasi akan diamati apakah sesuai dengan yang diharapkan. Estimasi banyaknya data uji dapat dihitung melalui banyaknya *field* data *entry* yang akan diuji[15].

Pengujian *black box* berusaha menemukan kesalahan dalam kategori sebagai berikut[15]:

1. Fungsi-fungsi yang tidak benar atau hilang.
2. Kesalahan *interface*.
3. Kesalahan dalam struktur data atau akses *database* eksternal.
4. Kesalahan kinerja.
5. Inisialisasi dan kesalahan terminasi.

Tidak seperti pengujian *white box* yang dilakukan pada saat awal proses pengujian, pengujian *black box* cenderung diaplikasikan selama

tahap akhir pengujian. Karena pengujian *black box* memperhatikan struktur *control* maka perhatikan berfokus pada domain informasi.

2.2.5 Rekayasa *Web*

Rekayasa Web adalah proses yang menggunakan pendekatan terorganisir, metodologi yang jelas, dan evaluasi yang terukur untuk membangun, mengoperasikan, dan menjaga aplikasi yang berbasis Web. Ini merupakan cabang khusus dari rekayasa perangkat lunak yang memberikan kerangka kerja untuk perancangan, pengembangan, perawatan, dan pengelolaan aplikasi web. Keberadaan disiplin rekayasa Web menunjukkan fokus pada keberhasilan pembuatan dan pengelolaan aplikasi dan sistem berbasis web. Melalui rekayasa web, pengembang dapat menjaga kendali, mengurangi risiko, meningkatkan kualitas, dan memastikan skalabilitas dari aplikasi web yang dapat dijaga dan ditingkatkan seiring waktu. Tujuan utamanya adalah mengatasi kompleksitas dan variasi dalam pengembangan aplikasi web dengan sukses[16].

2.2.6 Pemrograman Berorientasi Objek

Pemrograman Berorientasi Objek merupakan pendekatan dalam pengembangan perangkat lunak di mana sistem diatur sebagai kumpulan objek yang mengandung data dan operasi yang diterapkan pada objek tersebut. Pendekatan berorientasi objek adalah metode terstruktur untuk membangun sistem perangkat lunak melalui pendekatan yang terfokus pada objek. Pendekatan ini didasarkan pada prinsip pengelolaan kompleksitas. Metode berorientasi objek mencakup serangkaian langkah, mulai dari analisis berorientasi objek, desain berorientasi objek, pemrograman berorientasi objek, hingga pengujian berorientasi objek.[7]

Sistem berorientasi objek merujuk pada sebuah sistem yang dibangun dengan menggunakan metode berorientasi objek, di mana komponen-komponennya dikemas menjadi kelompok data dan fungsi terkait. Setiap komponen dalam sistem ini memiliki kemampuan untuk mewarisi atribut, sifat, dan komponen lainnya dari komponen yang lebih tinggi. Berikut beberapa prinsip dasar yang terkait dengan metode berorientasi objek[7]:

1. Kelas (class)

Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan bentuk struktur pada kode program yang menggunakan metodologi berorientasi objek[7].

2. Objek (object)

Objek adalah abstraksi yang mewakili dunia nyata seperti benda, manusia, satuan organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak. Objek merupakan suatu entitas yang mampu menyimpan informasi dan mempunyai operasi yang dapat diterapkan atau dapat berpengaruh pada status objeknya[7].

3. Metode (method)

Metode pada sebuah kelas hamper sama dengan fungsi atau prosedur pada metodologi struktural. Metode berfungsi untuk memanipulasi objek itu sendiri. Metode merupakan fungsi atau transformasi yang dapat dilakukan terhadap objek atau dilakukan[7].

4. Atribut (attribute)

Atribut adalah variable global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek[7].

5. Abstraksi (*abstraction*)

Mepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan[7].

6. Enkapsulasi (*encapsulation*)

Pembungkusan atribut data dan layanan yang dipunyai objek untuk menyembunyikan implemetasi dan objek sehingga objek lain tidak mengetahui cara kerjanya[7].

7. Antarmuka (*interface*)

Antarmuka atau *interface* sangat mirip dengan kelas, tetapi tanpa atribut kelas dan memiliki motode yang dideklarasikan tanpa isi. Interface dapat diimplementasikan oleh kelas lain[7].

Dalam situasi di mana ada fungsi yang sama antara kelas induk dan kelas anak, konsep *Override* digunakan. *Override* memungkinkan fungsi pada kelas anak menimpa fungsi yang ada pada kelas induk. Fungsi yang didefinisikan sebagai final di kelas induk tidak dapat diubah ulang oleh kelas anak. Konsep "*final*" juga digunakan untuk membatasi perubahan pada metode tertentu oleh kelas anak[7].

Untuk menggambarkan interaksi kelas dan pengguna kelas dengan cara yang lebih konkret, dibentuk struktur yang berfungsi sebagai antarmuka atau *interface*. Penggunaan *interface* digunakan untuk mengelompokkan kelas ke dalam bentuk yang lebih terorganisir[7].

2.2.7 Basis Data

Sistem basis data adalah sebuah sistem yang terkomputerisasi yang tujuannya utamanya adalah untuk mengelola data yang telah diolah menjadi informasi, dan memberikan akses terhadap informasi ini pada saat dibutuhkan. Basis data digunakan sebagai wadah penyimpanan data agar dapat diakses dengan efisiensi dan kecepatan. Dalam konteks sistem informasi, basis data memiliki peran yang sangat penting karena berperan sebagai sumber informasi bagi pengguna.[7]

Sistem basis data memiliki kemampuan untuk menyediakan bahasa yang digunakan dalam merancang struktur basis data, yang dikenal sebagai DDL (*Data Definition Language*), serta bahasa yang digunakan untuk mengoperasikan atau mengubah data dalam basis data, yang disebut DML (*Data Manipulation Language*). DDL dan DML merupakan bagian integral dari bahasa data tunggal yang dikenal dengan sebutan SQL (*Structured Query Language*)[17].

A. DBMS (*Database Management System)*

Database Management System (DBMS) adalah sebuah sistem aplikasi yang berfungsi untuk menyimpan, mengelola, dan menampilkan data. DBMS memenuhi persyaratan dengan menyediakan fasilitas untuk mengelola akses data, menangani integritas data, serta melakukan pengelolaan dan pemulihan data dari backup. Dengan keberadaannya, DBMS mampu mengakomodasi berbagai macam pemakai yang memiliki kebutuhan akses data yang berbeda-beda. Sebagai sistem manajemen basis data, DBMS memiliki peran penting dalam mendukung efisiensi dan keamanan pengelolaan data dalam sebuah organisasi atau aplikasi[18].

Tujuan pengolahan data dalam database adalah agar dapat memperoleh atau menemukan kembali data yang ingin dicari dengan cepat mudah selain itu juga pengolahan data dan tujuan-tujuan yang lainya. Berikut tujuan database: kecepatan dan kemudahan, efisien ruang penyimpanan, keakuratan, ketersediaan, kelengkapan, keamanan, kebersamaan pemakai[19].

B. ERD (Entity-Relationship Diagram)

Entity-Relationship Diagram (ERD) merupakan salah satu teknik yang digunakan sebagai tahap dasar dalam pembuatan *database*. ERD merupakan salah satu teknik merancang *database* yang paling banyak digunakan. ERD merupakan representasi visual dari *Diagram* yang berbentuk notasi grafis untuk mendeskripsikan bagaimana entitas saling terkait antara satu dengan yang lainnya dalam *database*. Fungsi ERD adalah sebagai alat bantu dalam menganalisis pembuatan *database* dan memberikan gambaran bagaimana kerja *database* yang akan dibuat. Di dalam ERD terdapat 3 elemen dasar, yaitu entitas, atribut, dan relasi sebagai berikut[19]:

1. Entitas

Entitas merupakan objek yang akan menjadi perhatian dalam suatu *database*. Entitas dapat berupa manusia, tempat, benda, atau kondisi mengenai data yang dibutuhkan. Simbol dari entitas berbentuk persegi panjang.

2. Atribut

Atribut merupakan informasi yang terdapat dalam entitas. Sebuah entitas harus memiliki *primary key* sebagai ciri khas entitas dan atribut deskriptif. Atribut biasanya terletak dalam tabel entitas atau dapat juga terpisah dari tabel. Simbol dari atribut berbentuk elips.

3. Relasi

Relasi di dalam ERD merupakan hubungan antara dua atau lebih entitas. Simbol dari relasi berbentuk belah ketupat. Relasi yang dapat dimiliki oleh ERD ada beberapa macam, yaitu:

a. One to One

Satu anggota entitas dapat berelasi dengan satu anggota entitas lain

b. One to Many

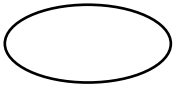
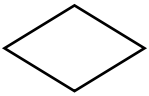
Satu anggota entitas dapat berelasi dengan beberapa anggota entitas lain

c. Many to Many

Beberapa anggota entitas dapat berelasi dengan beberapa anggota entitas lain

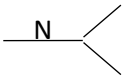
Simbol-simbol yang digunakan pada ERD dapat dilihat pada Tabel 2.4[19].

Tabel 2. 4 Simbol *Entity Relationship Diagram*

No.	Simbol	Nama Simbol	Keterangan
1		*Entity* / Entitas	Merupakan data inti yang akan disimpan pada tabel basis data. Penamaan entitas biasanya lebih mengarah pada kata benda dan belum merupakan nama tabel.
2		Atribut	*Field* atau kolom data yang butuh disimpan dalam suatu entitas.
3		*Multi value* / Atribut multi nilai	*Field* atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki nilai lebih dari satu.
4		Relasi	Relasi yang menghubungkan antar entitas biasanya diawali kata kerja.
5		*Link*	penghubung antara himpunan relasi dengan himpunan entitas dan himpunan entitas dengan atributnya

Simbol ERD juga memiliki kardinalitas untuk menunjukkan maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Kardinalitas relasi yang terjadi di antara dua himpunan entitas dapat berupa satu ke satu (*one to one*), satu ke banyak (*one to many*), dan banyak ke banyak (*many to many*). Simbol-simbol kardinalitas ERD dapat dilihat pada Tabel 2.5 berikut[19].

Tabel 2. 5 Simbol *Kardinalitas ERD*

No.	Simbol	Nama	Keterangan
1		*Association* / Asosiasi	Penghubung antar relasi dan entitas dimana di kedua ujungnya memiliki *multiplicity* kemungkinan jumlah pemakaian (kardinalitas).
2		Relasi Satu ke Satu (*One to One*)	Relasi yang menunjukkan bahwa setiap himpunan entitas berhubungan dengan tepat satu himpunan entitas lainnya.
3		Relasi Satu ke Banyak (*One to Many*)	Relasi yang menunjukkan bahwa hubungan antara entitas pertama dengan entitas kedua adalah satu banding banyak tetapi tidak sebaliknya.
4		Relasi Banyak ke Banyak (*Many to Many*)	Relasi yang menunjukkan bahwa setiap himpunan entitas boleh berhubungan dengan banyak himpunan entitas lainnya dan sebaliknya.