BAB II

LANDASAN TEORI

2.1 *Database*

Menurut Anhar (2010),"*Database* adalah sekumpulan tabel-tabel yang berisi data dan merupakan kumpulan dari *field* atau kolom. Struktur *file* yang menyusun sebuah database adalah *data record* dan *field*."

Menurut Raharjo (2011:3),"*Database* adalah kumpulan data yang terintegrasi dan diatur sedemikian rupa sehingga data tersebut dapat dimanipulasi, diambil, dan dicari secara cepat."

Menurut Connolly (2010, p65),"*Database* adalah suatu kumpulan data yang saling berhubungan secara logis dan penjelasan tentang data yang terhubung tersebut dirancang sedemikian rupa sehingga memberika informasi yang diperlukan oleh organisasi"

Jadi dapat disimpulkan bahwa *database* adalah sekumpulan dari data-data yang saling berhubungan satu sama lainnya membentuk sekumpulan *field*, dan sekumpulan *field* membentuk sekumpulan *record*, sekumpulan *record* membentuk sekumpulan *table*, dan kumpulan *table* membentuk sebuah *database* yang dapat digunakan untuk mencari sebuah data-data dengan cepat.

2.2 *Website*

Menurut Hidayat (2010:2) " *website* atau situs dapat diartikan sebagai kumpulan halaman-halaman yang digunakan untuk menampilkan informasi teks, gambar diam atau gerak, animasi, suara, dan atau gabungan dari semuanya, baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait, yang masing-masing dihubungkan dengan jaringan-jaringan halaman."

Menurut Kustiyahningsih (2011:113) "*web* adalah layanan yang didapat oleh pemakai komputer yang terhubung ke internet." Dapat disimpulkan bahwa *website*adalah sebuah tempat di internet yang terdiri dari kumpulan-kumpulan halaman yang setiap halamannya saling keterkaitan satu sama lainnya.

Menurut Hidayat (2010:3), perkembangan *website* sangatlah pesat, berikut ini adalah jenis-jenis website :

1. *Website* berdasarkan sifat atau *style*,
    a. *Website* dinamis

        Merupakan sebuah *website* yang menyediakan konten atau isi yang selalu berubah-ubah. Bahasa pemrograman yang digunakan biasanya adalah PHP, ASP, .NET, dan memanfaatkan *database* MySql.
    b. *Website* statis

        Merupakan sebuah *website* yang menyediakan konten atau isi yang jarang berubah.Bahasa pemrograman yang digunakan biasanya HTML dan belum menggunakan teknologi *database*.

2. *Website* berdasarkan fungsinya
    a. *Personal Website*

        Website yang biasanya berisikan informasi-informasi dari perseorangan.
    b. *Commercial Website*

        Website yang biasanya dimiliki oleh seseorang atau sebuah organisasi yang bertujuan untuk bisnis.
    c. *Government Website*

        Website yang dimiliki oleh instansi pemerintahan yang bertujuan untuk memberikan pelayanan kepada masyarakatnya.

d. *Non-Profit Organization Website*

Website yang biasanya dimiliki oleh seseorang atau sebuah organisasi yang tidak memiliki tujuan untuk berbisnis atau mencari keuntungan.

3. *Website* dari segi pemrogramannya

a. *Server Side*

Merupakan *website* yang menggunakan Bahasa pemrograman yang ketergantungan dengan ketersediaannya *server* seperti PHP, ASP, .NET dan lainnya.Jika server tidak tersedia, maka Bahasa pemrograman tersebut tidak dapat berjalan sebagaimana semestinya berjalan di *server*.

b. *Client Side*

Website yang tidak membutuhkan server dalam menjalankan programnya, cukup diakses melalui *browser* saja.

2.3 *System Development Life Cycle*

System Development Life Cycle (SDLC) merupakan tahapan-tahapan pengerjaan yang dilakukan oleh seorang *system analyst* dan *programmer* dalam membangun sebuah sistem informasi.

Menurut Kendall (2006) *System Development Life Cycle* (SDLC) adalah pendekatan bertahap untuk melakukan analisa dan membangun rancangan sistem dengan menggunakan siklus yang spesifik terhadap kegiatan pengguna.

Menurut Jogiyanto (2003), *System Development Life Cycle* (SDLC) memiliki empat tahapan yaitu:

1. *SystemAnalyst*

   System Analyst adalah orang yang di didik khusus untuk mengembangkan sebuah *system* secara professional dikarenakan penggunaan metode *System Development Life Cycle* (SDLC) sangatlah kompleks. Berikut adalah kegiatan-kegiatan yang dilakukan oleh *system analyst*:

   a. Studi Pendahuluan

      Kegiatan awal dari analisi sistem adalah studi awal atau studi pendahuluan tentang jenis, ruang lingkup dan pemahaman awal dari proyek sistem teknologi informasi.Studi pendahuluan ini menghasilkan sistem secara awal, perkiraan biaya yang dibutuhkan dan waktu yang diperlukan.

   b. Studi Kelayakan

      Studi kelayakan merupakan suatu tinjauan sekilas pada faktor-faktor utama yang akan mempengaruhi kemampuan sistem untuk mencapai tujuan yang diinginkan.

   c. Identifikasi Permasalahan dan Kebutuhan Informasi Pemakai

      Mengidentifikasi masalah dilakukan dengan penyebab masalahnya.Penyebab masalahnya merupakan sumber dari permasalahan yang harus diperbaiki. Selanjutnya memahami sistem yang ada untuk mendapatkan data dan menganalisis permasalahannya

   d. Menganalisis Hasil Penelitian

      Menganalisis hasil penelitian adalah menemukan penyebab permasalahan sistem yang tidak berfungsi sehingga dapat cepat digantikan dengan sistem yang baru.

2. *SystemDesign*

   Tahapan *System Design* mempunyai dua tujuan yaitu :

   a. Perancangan sistem secara umum adalah memberikan gambaran umum kepada pemakai sistem tentang sistem teknologi informasi yang baru. Perancangan sistem secara umum lebih diarahkan kepada pemakai sistem untuk menyetujuinya ke perancangan sistem selanjutnya.
   b. Perancangan sistem terperinci dimaksudkan untuk meggambarkan bentuk secara fisik dari komponen-komponen sistem teknologi informasi yang ingin dibangun.

3. *SystemImplementation*

   Implementasi sistem juga merupakan proses mengganti atau meninggalkan sistem yang lama dengan mengganti sistem yang baru. Untuk menggantikan sistem yang lama ke sistem yang baru diperlukan suatu pendekatan atau strategi supaya berhasil.

4. *SystemOperationandMaintenance*

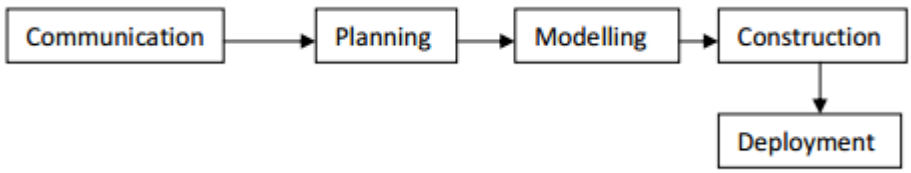
   Setelah sistem diiplementasi dengan berhasil, sistem akan dioperasikan dan di rawat. Sistem perlu dirawat karena beberapa hal, yaitu

   a. Sistem mengandung kesalahan yang belum diperbaiki,sehingga kesalahan sistem perlu diperbaiki.
   b. Sistem mengalami perubahan karena peemintaan baru dari pemakaian sistem.
   c. Sistem mengalami perubahan karena perubahan lingkungan luar.

Biaya perawatan sistem sering diabaikan karena biaya perawatan sistem merupakan biaya yang cukup besar.jadi sebisa mungkin kita harus merawatnya denga teliti agar suatu sistem dapat bertahan dengan lama.

2.4 *SDLC Waterfall Model*

Menurut Roger S Pressman (2010, p39) " *Waterfall Model* juga dapat disebut sebagi *Classic Life Cycle*. Menunjukan sebuah pendekatan sistematis untuk pengembangan perangkat lunak.Diawali dengan *communication, planning, modelling, construction,* dan *deployment*".



Gambar 2.1 Tahapan pada Waterfall Model

(Sumber: Software Engineering, A Practicioner Approach 7th – Pressman p39)

1. *Communication*

   Sebelum melakukan pekerjaan teknis, penting bagi para developer untuk berkomunikasi dan berkolaborasi secara berkala dengan client yang bertujuan untuk mengumpulkan segala persyaratan dan fitur-fitur yang ingin dibangun.

2. *Planning*

   Pada tahap ini para developer mulai memikirkan tugas-tugas apa saja yang harus dilakukan, resiko yang mungkin terjadi, biaya yang dikeluarkan, sumber daya yang digunakan dalam proses pengerjaan.

3. *Modelling*

   Seorang yang ingin membangun sesuatu selalu membutuhkan sebuah model atau kerangka untuk menjadi acuan dalam membangun, sama

dengan *software*, para developer perlu membuat sebuah model atau sketsa yang menggambarkan sebuah *software* yang ingin dibangun berikut dengan proses-proses yang berjalan.

4. *Construction*

   Pada tahap ini para developer mulai melakukan koding baik secara manual atau otomatis.Apabila sudah selesai harus langsung diadakan *testing* untuk mengetahui kesalahan-kesalahan yang masih terjadi dan memperbaikinya.

5. *Deployment*

   Piranti lunak (*software*) sudah siap untuk dikirim ke client dan client memberikan *feedback* terhadap piranti lunak (*software*) yang diterimanya kepada developer untuk menjadi evaluasi.

2.5 *Unified Modelling Language*

Menurut Pressman (2010,p841), *Unified Modelling Language* (UML) adalah sebuah bahasa standar untuk menulis perangkat lunak dalam bentuk gambar. Menurut Widodo (2011:6), "UML adalah Bahasa pemodelan standar yang memiliki sintak dan semantik"

Menurut Henderi (2008:6) langkah-langkah penggunaan *Unified Modelling Language* (UML) sebagai berikut :

1. Buat daftar *Business Process* dari level tertinggi untuk mendefinisikan aktifitas dan proses yang mungkin terjadi.
2. Petakan *Use Case* untuk setiap *Business Process* untuk mendefinisikan dengan tepat fungsional yang harus disediakan oleh sistem, kemudian perhalus *Use Case Diagram* dengan melengkapinya dengan *Requirement*, *Constraint*, dan catatan lain-lain.
3. Buatlah *Deployment Diagram* secara kasar untuk mendefinisikan arsitektur fisik sistem.

4. Definisikan *Requirement* dan *Non-Functional*, *Security*, dan sebagainya yang juga harus disediakan oleh sistem.
5. Berdasarkan *Use Case Diagram*, mulai buat *Activity Diagram*.
6. Definisikan objek-objek level atas *Package* atau *Domain* dan buatlah *Sequence* dan atau *Collaboration* untuk setiap alur pekerjaan. Jika sebuah *Use Case* memiliki kemungkinan alur normal dan *Error*, buat lagi satu *Diagram* untuk masing-masing alur.
7. Buatlah rancangan *User Interface Model* yang menyediakan antar muka bagi pengguna untuk menjalankan skenario *Use Case*.
8. Berdasarkan model-model yang sudah ada, buatlah *Class Diagram*. Setiap *Package* atau *Domain* dipecah menjadi hirarki *Class* lengkap dengan atribut dan metodenya. Akan lebih baik jika untuk setiap *Class* dibuat *Unit Test* untuk menguji fungsionalitas *Class* dan interaksi dengan *Class* lain.
9. Setelah *Class Diagram* dibuat, kita dapat melihat kemungkinan pengelompokan *Class* menjadi komponen-komponen karena itu dibuatlah *Component Diagram* pada tahap ini. Juga definisikan *Test Integration* untuk setiap komponen meyakinkan ia berfungsi dengan baik.
10. Perhalus *Deployment* diagram yang sudah dibuat. Detilkan kemampuan dan *Requirement Software*, sistem operasi, jaringan dan sebagainya. Petakan komponen ke dalam *Node*.
11. Mulai membangun sistem. Ada dua pendekatan yang dapat digunakan yaitu :

a. Pendekatan *Use Case* dengan meng-*assign* setiap *Use Case* kepada tim pengembang tertentu untuk mengembangkan unit kode yang lengkap dengan *Test*.
b. Pendekatan komponen yaitu meng-*assign* setiap komponen kepada tim pengembang tertentu.

Berikut ini adalah diagram UML menurut Henderi (2010:6) yaitu :

1. *Use Case Diagram*

   Use Case Diagram secara grafis menggambarkan, interaksi secara sistem, sistem eksternal dan pengguna. Dengan kata lain use case diagram secara grafis mendeskripsikan siapa yang akan menggunakan sistem dan dalam cara apa pengguna mengharapkan interaksi dengan sistem itu. *Use Case* secara naratif digunakan untuk secara tekstual menggambarkan sekuensi langkah-langkah dari tiap interaksi.
2. *Class Diagram*

   Menggambarkan struktur *Object* sistem.*Diagram* ini menunjukan *Class Diagram* yang menyusun sistem dan hubungan antar *Class Object* tersebut.
3. *Sequence Diagram*

   Secara grafis menggambarkan bagaimana *Object* berinteraksi satu sama lain melalui pesan pada sekuensi sebuah *Use Case* atau operasi.
4. *State Chart Diagram*

   Digunakan untuk memodelkan *Behavior Object Khusus* yang dinamis. Diagram ini mengilustrasikan siklus hidup objek berbagai keadaan yang dapat diasumsikan oleh objek dan *event-event* yang menyebabkan objek dari satu *state* ke *state* yang lain.

5. *Activity Diagram*

Secara grafis untuk menggambarkan rangkaian aliran aktifitas baik proses bisnis maupun *Use Case*, *Activity Diagram* dapat juga digunakan untuk memodelkan *action* yang akan dilakukan saat operasi dieksekusi, dan memodelkan hasil dari *action*.

2.6 *Entity Relationship Diagram*

Entity Relationship Diagram (ERD) merupakan sebuah model yang menjelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang memiliki hubungan antar relasi.

Menurut salah satu ahli, Brady dan Loonam (2010), *Entity Relationship Diagram* (ERD) merupakan teknik yang digunakan untuk memodelkan kebutuhan data dari suatu organisasi, biasanya *system analyst* dalam tahap analisis persyaratan proyek pengembangan system.

Komponen penyusun ERD adalah sebagai berikut :

Notasi	Keterangan
	**Entitas**, adalah suatu objek yang dapat diidentifikasi dalam lingkungan pemakai.
	**Relasi**, menunjukkan adanya hubungan di antara sejumlah entitas yang berbeda.
	**Atribut**, berfungsi mendeskripsikan karakter entitas (atribut yg berfungsi sebagai key diberi garis bawah)
	Garis, sebagai penghubung antara relasi dengan entitas, relasi dan entitas dengan atribut.

Entitas adalah objek dalam dunia nyata yang dapat dibedakan dengan objek lain. Contohnya mahasiswa, dosen.Entitas terdiri dari beberapa atribut. Contohnya dari entitas mahasiswa memiliki atribut NIM, nama, alamat, email, dll. Setiap entitas harus memiliki satu atribut unik yang disebut *primary key*.

Terdapat dua jenis atribut yaitu :

a. *Identifier* (*key*) digunakan untuk menentukan atribut dari sebuah entitas yang unik *(primary key*).
b. *Descriptor* (*nonkey attribute*) digunakan untuk menspesifikasikan karakteristik dari suatu entitas yang tidak unik.

Relasi adalah hubungan antar beberapa entitas.Contohnya hubungan entitas antara mahasiswa dengan mata kuliah, dimana satu mahasiswa dapat mengambil banyak mata kuliah tetapi satu mata kuliah dan dapat diambil oleh banyak mahasiswa saja.

Kardinalitas menyatakan jumlah himpunan relasi antar entitas. Pemetaan kardinalitas terdiri dari :

a. One-To-One

Sebuah entitas A berhubungan dengan entitas B paling banyak satu. Contohnya relasi antara pegawai dengan departemen dimana setiap pegawai hanya bekerja di satu departemen.

b. One-To-Many

Sebuah entitas A berhubungan dengan entitas B yang lebih dari satu. Contohnya relasi antara departemen dengan pegawai dimana satu departemen memiliki banyak pegawai.

c. Many-To-Many

Sebuah entitas A berhubungan dengan entitas B yang masing-masing memiliki hubungan yang lebih dari satu. Contohnya relasi

antara entitas mahasiswa dengan entitas mata kuliah dimana setiap mahasiswa dapat mengambil banyak mata kuliah dan setiap mata kuliahdapat diambil banyak mahasiswa.

Berikut adalah tahapan dalam membuat ERD:

1. Tentukan entitas
2. Tentukan relasi
3. Gambar ERD sementara
4. Isi kardinalitas
5. Tentukan *primary key*
6. Gambar ERD berdasarkan *key*
7. Tentukan atribut
8. Petakan atribut
9. Gambar ERD dengan atribut

2.7 Normalisasi

Normalisasi adalah sebuah teknik analisis data dengan mengorganisasikan atribut-atribut data untuk membentuk entitas yang tidak *redundant*, stabil, dan fleksibel.

Menurut Abdul Kadir (2009:116), Normalisasi adalah suatu proses yang digunakan untuk menentukan kelompok-kelompok atribut-atribut dalam sebuah relasi sehingga diperoleh relasi yang berstrutur baik.

Dalam normalisasi terdapat sejumlah langkah yang penting yaitu:

1. Bentuk Tidak Normal (*Unnormalized Form / UNF*)

   Merupakan bentuk tidak normal dari kumpulan data yang di *record*, tidak harus mengikuti format tertentu, data dapat tidak lengkap atau terduplikasi.

2. Bentuk Normal Pertama (*First Normal Form / 1NF*)

   Pada 1NF, data-data yang bersifat berulang (*repeating*) harus dihilangkan agar menjadi data tunggal.

3. Bentuk Normal Kedua (*Second Normal Form / 2NF*)

   Pada 2NF, data-data sudah harus melewati tahap 1NF dan data-data tersebut tidak mengandung dependensi atau ketergantungan satu sama lain dan pada 2NF sudah harus ditentukan *primary key* dan *foreign key* nya masing-masing.

4. Bentuk Normal Ketiga (*Third Normal Form / 3NF*)

   Pada 3NF, data-data sudah harus melewati tahap 2NF dan pada 3NF pastikan *foreign key* suatu entitas tidak saling bergantung dengan dengan *foreign key* entitas lainnya.

2.8 PHP (*Hypertext Preprocessor*)

PHP pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995. Pada saat itu PHP masih bernama *Form Interpreted* (FI), yang berwujud sekumpulan skrip yang digunakan untuk mengolah data di *web* yang selanjutnya Rasmus Lerdorf merilis *source code* tersebut dengan nama PHP/FI menjadi *open source.*

Pada tahun 1997, sebuah perusahaan bernama Zend menulis ulang PHP/FI menjadi lebih mudah dimengerti, lebih mudah dan lebih cepat dengan menamakannya menjadi PHP *: Hypertext Preprosessing*.

Menurut Sibero (2012:49), "PHP (*Personal Home Page*) adalah pemrograman (*interpreter*) adalah proses penerjemahan baris sumber menjadi kode mesin yang dimengerti komputer secara langsung pada saat baris kode dijalankan".

2.9 Apache

Apache merupakan sebuah perangkat lunak (software) *open source* yang menjadi alternatif dari *server webSun Java System Web Server.* Menurut Bowen and Coar (2000,p2) Apache merupakan sebuah *server* yang cukup sederhana, orang-orang

pada saat meletakan *website* membutuhkan tampilan tertentu dan *bugs* yang perlu diperbaiki, jadi apache dilahirkan oleh *user* untuk *user*.

2.10 MySQL

MySQL adalah sebuah produk *database relational* terpopuler pada saat ini yang dapat berjalan di *platformWindows, Linux, Unixbased system* termasuk *Mac OS*. MySQL pertama kali dibuat dan dikembangkan di swedia, oleh David Axmark, Allan Larsson, dan Michael "Monty" Widenius.Mereka mengembangkan MySQL sejak tahun 1980-an. Pada saat ini versi MySQL sudah cukup stabil dan mencapai ke versi 5X dan sedang dikembangkan ke 6X.