**BAB II**

**LANDASAN TEORI**

**2.1. Konsep Dasar Web**

Menurut Arief (2011:7) “*Web* adalah salah satu aplikasi yang berisikan dokumen-dokumen multimedia (teks, gambar, suara, animasi, video) yang didalamnya yang menggunakan *protocol* HTTP (*Hyper Text Transfer Protocol*) dan untuk mengaksesnya menggunakan perangkat lunak yang disebut *browser*”.

**A.** ***Internet***

1. Definisi *Internet*

Menurut Setiawan (2012:1) “*internet* adalah kumpulan komputer dan jaringan lokal yang melakukan aktivitas komunikasi satu sama lain berbasis protokol TCP/IP dalam area yang tidak terbatas”.

Istilah-istilah yang umum di kenal di *internet* antara lain:

1. WWW(*World Wide Web*).

*World Wide Web* merupakan salah satu layanan yang didapat oleh pemakai komputer yang terhubung internet.

1. *Browser.*

*Browser* adalah *software* yang digunakan untuk menampilkan informasi dari *server web*.

1. *Universal Resource Locator*(URL).

*Universal Resource Locator* adalah konsep nama file standar yang diperluas dengan jaringannya.

1. *Hypertext Markup Language* (HTML).

Menurut Saputra (2012:1) ”HTML adalah bahasa paling dasar dan penting yang digunakan untuk menampilkan dan mengelola tampilan pada halaman *website*”.

1. *Domain Name Service*(DNS).

Menurut Hidayat (2010:9) ”*Domain* adalah alamat unik yang digunakan untuk mengidentifikasi sebuah *website*, atau dengan kata *domain* adalah alamat yang digunakan untuk mencari dan menentukan sebuah *website* pada dunia *internet*, sedangkan sistem pengalamatan berbasis *domain* di kenal sebagai *Domain Name Service(*DNS*)”.*

2. *Website*

Menurut Arief (2011:8) “*Website* adalah kumpulan dari halaman *web* yang sudah dipublikasikan di jaringan *internet* dan memiliki *domain*/URL (*Uniform Resource Locator*) yang dapat diakses semua pengguna *internet* dengan cara mengetikan alamatnya”. Jenis-jenis *web* menurut Arief (2011:8) dibagi menjadi 2 jenis yaitu:

1. *Web statis*

Adalah *web* yang isinya tidak berubah-ubah, maksudnya adalah isi dari dokumen tersebut tidak dapat diubah secara cepat dan mudah. Contohnya *web* kumpulan produk animasi.

*b. Web dinamis*

Adalah *web* yang isinya dapat berubah-ubah setiap saat. Contohnya *web* organisasi dan *web* berita.

3. Aplikasi Berbasis *Web*

a. *Web Browser*

Menurut Sutisna (2007:5) “*Web Browser* adalah *software* untuk menampilkan halaman *website*”. Cara kerja *browser* adalah menerjemahkan kode program HTML ke dalam bentuk visual sesuai dengan apa yang dirancang oleh pembuat *website*. Beberapa *browser* yang terkenal diantaranya *Internet Explorer, Mozilla Firefox, Opera* dan lain-lain.

b. *Web Server*

Menurut Sutisna (2007:5) “*Web Server* adalah halaman-halaman *website* yang diakses oleh *user* melalui *browser* disimpan pada *web server*”*.* Untuk itu, diperlukan program khusus agar *website* yang kita buat dapat diterima dengan baik oleh *client.* Ada beberapa *software* yang bisa digunakan baik gratis (*Open Source*) maupun yang komersial, diantaranya: *Nestcape Server, Microsoft IIS, Xitami, Webstar* dan lain-lain.

1. **Bahasa Pemrograman**

1*. Adobe Dreamweaver CS3*

 Menurut Wahana Komputer (2013:2) “*Adobe Dreamweaver* merupakan *software* aplikasi yang digunakan sebagai HTML editor profesional untuk mendesain *web* secara visual.Aplikasi ini juga yang biasa dikenal dengan istilah WYSIWYG (*What You See Is What You Get*), yang intinya adalah bahwa anda tidak harus berurusan dengan *tag-tag* HTML untuk membuat sebuah situs”.

2*. Javascipt*

Menurut Brooks (2008:3) “*Javascript* adalah bahasa pemrograman yang diinterpretasikan bukan dikompile, diadopsi dari bahasa *C atau C++* yang dikembangkan menjadi bahasa pemrograman *web client-side*”. *Javascript* didesain untuk bekerja sama dengan HTML membuat *web page* yang interaktif.

*Javascript* merupakan bahasa pemrograman yang cukup mudah dikuasai dan memiliki banyak fungsi yang dapat digunakan untuk meningkatkan efek *visual* dari halaman *web*. Kode dari *javascript* harus diapit oleh *tag*, diawali dengan *tag <script language=”javascript”>* dan diakhiri dengan *tag </script>*.

3. CSS (*Cascading Style Sheet*)

Menurut Juju (2008:95) “*CSS* adalah bahasa *stylesheet* yang digunakan untuk mengatur tampilan dokumen. *CSS* memungkinkan kita untuk menampilkan halaman yang sama dengan format yang berbeda”.

Dengan *CSS*, tampilan *website* akan lebih cantik dan konsisten. Ada dua cara untuk menuliskan kode *CSS.* Pertama secara *internal,* yaitu menuliskan langsung diantara *tag* HTML/XHTML. Kedua secara *eksternal*, yaitu kode *CSS* disimpan dalam *file* yang terpisah kemudian dipanggil saat halaman *web* dibuka. *CSS* sendiri merupakan sebuah teknologi *internet* yang direkomendasikan oleh *World Wide Web Consurtium* (W3C) dan diperkenalkan pada tahun 1996.

4. PHP *(Personal Home Page)*

Menurut Arief (2011:43) “PHP (*Hypertext Proprocessor*) adalah Bahasa *server-side scripting* yang menyatu dengan HTML untuk membuat halaman *web* yang dinamis”. Karena PHP merupakan *server-side scripting* maka sintaks dan perintah-perintah PHP akan dieksekusi di *server* kemudian hasilnya dikirimkan ke *browser* dalam format HTML. Dengan demikian kode program yang dituliskan dalam PHP tidak akan terlihat oleh *user* sehingga keamanan halaman *web* lebih terjamin. PHP dirancang untuk membentuk *web* yang dinamis, yaitu halaman *web* yang dapat membentuk suatu tampilan berdasarkan perintah terkini, seperti menampilkan isi basis data ke halaman *web.*

5. HTML (*Hypertext Markup Languange)*

Menurut Sutisna (2007:6) “HTML singkatan dari (*Hypertext Markup Languange)* ini adalah Bahasa *web* yang wajib bagi *web master”.* HTML terdiri dari *tag-tag* yang dapat dibaca dan diterjemahkan oleh *browser.* Menghafal *tag* HTML bisa menguras *energy*. Oleh karena itu, editor WYSIWYG seperti *Dreamweaver* dapat membantu meringankan tugas *web maste.*

**C. Basis Data (*Database)***

1. Definisi Basis Data

Menurut Kadir (2013:136) “*database* merupakan suatu bentuk pengelolaan data yang ditujukan agar pengaksesan terhadap data dapat dilakukan dengan mudah”. Sistem ysng ditujukan untuk menangani *database* biasa disebut DBMS (*database menegement system*), Dengan menggunakan DBMS, pemakai dapat melakukan hal- hal dengan mudah seperti menambahkan data, menghapus data, mengubah data, mencari data, menampilakan data dengan kriteria tertentu ataupun pengurutkan data.

Berdasarkan definisi tersebut maka suatu basis data mempunyai beberapa kriteria penting yang harus dipenuhi, yaitu:

a. Beriorientasi pada data (*data oriented*) dan bukan berorientasi pada program (*program oriented*) yang akan menggunakannya.

b. Data yang terdapat dalam basis data dapat berkembang dengan mudah, baik *volume* maupun strukturnya.

c. Data yang ada dapat memnuhi kebutuhan sistem-sistem baru secara mudah.

d. Data dapat digunkannya dengan cara yang berbeda-beda.

e. Kerangkapan data (*data redundancy*) minimal.

2. MYSQL

Menurut Kadir (2013:15), “*MySql* adalah sebuah database server yang berfungsi untuk menangani *database*”. Dengan menggunakan *MySql*, kita bisa menyimpan data dan kemudian data bisa diakses dengan cara mudah dan cepat. Selain itu, *MySql* tergolong sebagai *database* relasional karena pada model ini data dinyatakan dalam bentuk dua dimensi yang secara khusus dinamakan tabel. Tabel tersusun atas baris dan kolom.

3. PhpMyAdmin

Menurut Sadeli (2013:10) “*PhpMyAdmin* adalah sebuah *software* yang berbentuk seperti halaman situs yang terdapat pada *web server*”. Fungsi dari halaman ini adalah sebagai pengendali *database MySQL* sehingga pengguna *MySQL* tidak perlu repot untuk menggunakan perintah-perintah *SQL*. Karena dengan adanya halaman ini semua hal tersebut dapat dilakukan hanya dengan meng-klik menu fungsi yang ada pada halaman *PhpMyAdmin*.

4. XAMPP

Menurut Utomo (2013:7) “*Xampp* merupakan *bundle* paket instan yang terdiri dari *Apache* (*Server), MySQL* (*database*), dan PHP”. Paket instan ini banyak digunakan oleh para *programer* tidak perlu mengkonfigurasi dari awal melainkan tinggal pakai.

5. Navicat

Menurut Suprianto (2010:21) “*Navicat* adalah *software* yang berfungsi memudahkan kita saat menciptakan (DDL, *data definition language*) dan memanipulasi (DML, *data manipulation language*) *database MySQL*, seperti menciptakan *table*, menghapus *table*, memasukkan baris, menghapus baris, mengubah baris, menampilkan data, menciptakan *user*, dan lain sebagainya”.

Dengan *navicat*, anda tidak harus hebat dalam pengusaan *script SQL* karena *navicat* menyediakan sarana untuk memanipulasi *database* dengan mudah dan ramah (*user friendly*). Sebagai contoh, untuk memasukan baris data, perintah *SQL* yang digunakan adalah pernyataan *INSERT***.** Dengan *navicat*, anda dapat memasukan baris data melalui *grid* layaknya *spread sheet Excel*

**D. Metode Pengembangan Perangkat Lunak**

Pengembangan *software waterfall* model pertama kali diperkenalkan oleh Winston Royce tahun 1970. *Waterfall* model merupakan model klasik yang sederhana dengan aliran sistem yang linier. *Output* dari setiap tahap merupakan *input* bagi tahap berikutnya. Model ini telah diperoleh dari proses rekayasa lainnya dan menawarkan cara pembuatan rekayasa perangkat lunak secara lebih nyata. Model ini melibatkan tim SQA (*Software Quantity Assurance*) dengan 5 tahapan, dimana setiap tahapan selalu dilakukan verifikasi atau testing.

Menurut Pressman (2010:46) “model *waterfall* adalah model klasik yang bersifat sistematis, berurutan dalam membangun *software*”. Secara umum tahapan pada model *waterfall* dapat dilihat pada gambar berikut:

Sumber: Pressman (2010:46)

**Communication**

project initiation

requirements gathering

**Planning**

estimating

scheduling

tracking

**Modeling**

analysis

design

**Construction**

code

test

**Deployment**

delivery

support

**Gambar II.2**

**Pemodelan Waterfall**

Berikut ini adalah fase-fase dalam pemodelan *waterfall*, diantaranya:

1. Analisa Kebutuhan *Software*

Mengumpulkan data secara lengkap kemudian dianalisis dan didefinisikan kebutuhan yang harus dipenuhi oleh *software* yang akan dibangun. Hal ini sangat penting mengingat *software* harus dapat berinteraksi dengan elemen-elemen yang lain seperti *hardware, database* dan lain-lainnya.

**Beberapa macam** kebutuhan adalah sebagai berikut:

1. User Requirement**(Kebutuhan Pengguna)**

Pernyataan tentang layanan yang disediakan sistem dan tentang batasan-batasan operasionalnya. Dilengkapi dengan gambar/diagram yang dapat dimengerti dengan mudah.

1. System Requirement**(Kebutuhan Sistem)**

Sekumpulan layanan/kemampuan sistem dan batasan-batasannya yang ditulis secara detil. Ini bisa berlaku sebagai kontrak antara klien dan pembangun.

1. ***Software Requirement Specification*(Spesifikasi Kebutuhan Perangkat Lunak).**

Gambaran abstrak dari rancangan perangkat lunak yang menjadi dasar bagi perancangan dan implementasi yang lebih detil.

1. Desain

Proses pencarian kebutuhan diintensikan dan difokuskan pada *software* untuk mengetahui sifat dari program yang akan dibuat, maka para *software engineer* harus mengerti tentang informasi dari *software*, misalnya fungsi yang dibutuhkan *user interface*. Dari dua aktivitas tersebut (pencarian kebutuhan sistem dan *software*) harus didokumentasikan dan ditunjukkan kepada *user*.

Proses *software design* untuk mengubah kebutuhan-kebutuhan di atas menjadi representasi ke dalam bentuk “*blueprint*” *software* sebelum *coding* dimulai. Desain harus dapat mengimplementasikan kebutuhan yang telah disebutkan pada tahap sebelumnya. Seperti dua aktivitas sebelumnya, maka proses ini juga harus didokumentasikan sebagai konfigurasi dari *software*.

1. *Code Generation*

Desain program diterjemahkan ke dalam kode-kode dengan menggunakan bahasa pemrograman yang sudah ditentukan. Program yang dibangun langsung diuji baik secara *unit*.

1. *Testing*

Untuk dapat mengerti oleh mesin, dalam hal ini adalah komputer, maka desain tadi harus diubah bentuknya menjadi bentuk yang dapat dimengerti oleh mesin, yaitu kedalam bahasa pemrograman melalui proses *coding*. Tahap ini merupakan implementasi dari tahap *design* yang secara teknis nantinya dikerjakan oleh *programmer*. Penyatuan unit-unit program kemudian diuji secara keseluruhan (*system testing*).

**Metode pengetesan dilakukan dengan cara sebagai berikut:**

1. Black box testing

Black box testing memperlakukan pengujian perangkat lunak sebagai “kotak hitam” tanpa pengetahuan tentang pelaksanaan internal.

1. White box testing

White box testing adalah ketika penguji memiliki akses ke struktur data internal dan algoritma termasuk source code.

1. *Support*

Sesuatu yang dibuat harus diujicobakan. Demikian juga dengan *software* Semua fungsi-fungsi *software* harus diujicobakan, agar *software* bebas dari *error*, dan hasilnya harus benar-benar sesuai dengan kebutuhan yang sudah didefinisikan sebelumnya.

Pemeliharaan suatu *software* diperlukan, termasuk di dalamnya adalah pengembangan, karena *software* yang dibuat tidak selamanya hanya seperti itu. Ketika dijalankan mungkin saja masih ada *error* kecil yang tidak ditemukan sebelumnya, atau ada penambahan fitur-fitur yang belum ada pada *software* tersebut. Pengembangan diperlukan ketika adanya perubahan dari eksternal perusahaan seperti ketika ada pergantian sistem operasi, atau perangkat lainnya.

Kelebihan dari model ini adalah selain karena pengaplikasian menggunakan model ini mudah, kelebihan dari model ini adalah ketika semua kebutuhan sistem dapat didefinisikan secara utuh, eksplisit, dan benar di awal proyek, maka *software engineering* (SE) dapat berjalan dengan baik dan tanpa masalah. Meskipun seringkali kebutuhan sistem tidak dapat didefinisikan se-eksplisit yang diinginkan dalam hal uang (lebih murah), usaha, dan waktu yang terbuang lebih sedikit jika dibandingkan problem yang muncul pada tahap-tahap selanjutnya.

Kekurangan yang utama dari model ini adalah kesulitan dalam mengakomodasi perubahan setelah proses dijalani. Fase sebelumnya harus lengkap dan selesai sebelum mengerjakan fase berikutnya.

Masalah dengan *waterfall*:

1. Perubahan sulit dilakukan karena sifatnya kaku.
2. Karena sifatnya kaku, model ini cocok ketika kebutuhan dikumpulkan secara lengkap sehingga perubahan bias ditekan sekecil mungkin. Tapi pada kenyataannya jarang sekali konsumen/pengguna yang bias memberikan kebutuhan secara lengkap, perubahan kebutuhan adalah sesuatu yang wajar terjadi.
3. *Waterfall* pada umumnya digunakan untuk rekayasa sistem yang besar yaitu dengan proyek yang dikerjakan di beberapa tempat berbeda, dan dibagi menjadi beberapa bagian sub-proyek.

**2.2. Teori Pendukung**

Dalam teori pendukung ini, berisi tentang Struktur Navigasi, *ERD* (*Entity Relationship Diagram*), *LRS* (*Logical Record Structure*), dan Pengujian *Black Box Testing.*

**A. Struktur Navigasi**

Menurut Suyanto (2008:62) “Struktur navigasi merupakan rancangan hubungan dan rantai kerja dari beberapa area yang berbeda dan dapat membantu mengorganisasikan seluruh elemen pembuatan *website*.”

Ada empat macam bentuk dasar yaitu :

1. *Linear*

 Pengguna akan melakukan navigasi secara berurutan dari *frame* atau *byte* informasi yang satu ke yang lainnya.



Sumber : Binanto (2010:269)

**Gambar II.3**

**Struktur Navigasi *Linear***

2. Hirarki

 Struktur dasar ini disebut juga struktur “linier dengan percabangan-percabangan” karena pengguna melakukan navigasi disepanjang cabang pohon struktur yang terbentuk oleh logika isi.



Sumber : Binanto (2010:269)

**Gambar II.4**

**Struktur Navigasi Hirarki (*Hierarchical*)**

3. *Non Linear*

 Pengguna akan melakukan navigasi dengan jelas melalui isi proyek dengan tidak terikat dengan jalur yang sudah ditentukan sebelumnya.



Sumber : Binanto (2010:269)

**Gambar II.5**

**Struktur Navigasi *Non Linear***

4. Campuran (*Composite*)

 Pengguna akan melakukan navigasi dengan bebas (secara *non-linear*), tetapi terkadang dibatasi prestasi linear film atau informasi penting dan atau pada data yang paling terorganisasi secara logis pada suatu hirarki.



Sumber : Binanto (2010:270)

**Gambar II.6**

**Struktur Navigasi Campuran (*Composite*)**

**B. *Entity Relationship Diagram* (ERD)**

1. Definisi ERD

Menurut Utami (2012:18) “ERD adalah suatu diagram untuk menggambarkan desain konseptual dari model konseptual suatu basis data relasional”. ERD juga merupakan gambaran yang menghubungkan antara objek suatu dengan objek yang lain dalam dunia nyata. Bisa dikatakan bahwa bahan yang akan digunakan untuk membuat ERD adalah dari objek di dunia nyata.

Secara umum ERD terdiri dari 3 komponen, yakin:

 a. Entitas (Entity)

 b. Atribut (Attribute)

 c. Relasi (Relationship)

a. Entitas (Entity)

 Entitas merupakan suatu “objek nyata” yang mampu di bedakan dengan objek yang lain. Objek tersebut dapat berupa orang benda ataupun hal yang lainnya. Penggambaran entitas dalam ERD.



Sumber : Utami (2012:19)

**Gambar II.7**

**Entitas**

1) Peran

entitas berupa peran contoh adalah mahasiswa, dosen, karyawan, dan lain-lain.

2) Kejadian

Entitas berupa kejadian contohnya adalah mengajar, ambil\_mata\_kuliah dan lain-lain.

3) Abstrak

Entitas berupa abstrak contohnya adalah kurikulum, mata\_kuliah, pekerjaan dan lain-lain.

4) Fisik

Entitas berupa tempat contohnya adalah ruang kelas.

b. Atribut

 Atribut merupakan semua informasi yang berkaitan dengan entitas. Di dalam dunia pemrograman, atribut adalah *property* dari suatu objek. Sebagai contoh jika entitas adalah manusia atau orang maka atributnya adalah rambut, mata, hidung, tangan, kaki, dan lain-lain. Atribut di gmbarkan dengan suatu lingkaran dengan nama atribut ditulis di tengahnya.



Sumber : Utami (2012:20)

**Gambar II.8**

**Atribut**

c. Relasi

Belah ketupat merupakan penggambaran hubungan (relasi) antar entitas atau sering disebut kerelasi. Ada dua macam penggambaran relasi, yakin relasi kuat dan relasi lemah. Relasi kuat biasanya untuk menghubungkan antar entitas kuat, sedangkan relasi lemah untuk menghubungkan antara entitas kuat dengan entitas lemah, penggambaran kerelasian.



Sumber : Utami (2012:24)

**Gambar II.9**

**Kerelasian**

Ada tiga macam relasi menurut derajatnya yaitu:

1) Urary

Relasi yang menghubungkan satu entitas.

2) Binary

Relasi yang menghubungkan dua entitas.

3) Ternary

Relasi yang menghubungkan lebih dari dua entitas. Pada kerelasian *binary* terdapat kardinalitas atau derajat hubungan antar entitas. Untuk menghubungkan entitas-kerelasian-entitas digunakan garis lurus.



Sumber : Utami (2012:24)

**Gambar II.10**

**Kerelasian antar entitas**

**C. *Logical Record Structure* (LRS)**

Menurut Simarmata (2007:115) “*Logical record structure* (LRS) adalah representasi dari struktur *record-record* pada table-tabel yang terbentuk dari hasil relasi antar himpunan entitas”. Menentukan kardinalitas, jumlah *table*, dan *Foreign Key* (FK).

Membentuk *Logical record structure* (LRS) berdasarkan *Entity Relationship Diagram* (ERD) :

a. Jika relasinya satu ke satu maka *foreign key* diletakkan pada salah satu dari dua entitas yang ada atau menyatukan kedua entitas tersebut.

b. Jika relasinya satu ke banyak maka *foreign key* diletakkan pada entitas *Many.*

c. Jika relasinya banyak ke banyak maka *file* disebut “*file connector”* yang berisi dua *foreign key* yang berasal dari kedua entitas.

**D. Pengujian Web**

Setiap aplikasi ataupun *website* setelah selesai dikerjakan atau dibuat, maka akan dilakukan sebuah pengujian pada fitur dan juga kegunaannya, apakah sudah sesuai dengan yang dibutuhkan atau belum. Pada dasarnya teknik pengujian ini terbagi menjadi 3 jenis, yakni *white box, grey box,* dan *black box testing.* Penulis sendiri menggunakan *black box testing* untuk metode pengujian dari *website* yang telah dibuat.

Menurut Sommerville (2011:735) “*Black box testing* adalah sebuah pendekatan untuk pengujian dimana *tester* tidak memiliki akses ke kode program dari sistem atau komponen-komponennya”.

*Black box testing* sendiri merupakan sebuah teknik pengujian perangkat lunak tanpa memiliki pengetahuan tentang bagaimana sistem dari sebuah aplikasi atau *website* tersebut bekerja, ketika melakukan *black box testing* maka *tester* akan berinteraksi dengan *user interface* sistem dengan memberikan sebuah data masukan dan memeriksa data keluaran tanpa mengetahui bagaimana dan dimana data tersebut di proses.

*Black box testing* memiliki beberapa keuntungan, diantaranya adalah:

1. Cocok dan efisien untuk segmen kode berskala besar
2. Tidak dibutuhkan akses ke kode program
3. Dengan jelas memisahkan *interface* untuk pengembang dengan *interface* untuk pengguna
4. *Tester* dapat melakukan pengujian aplikasi tanpa mengetahui tentang pembuatan aplikasi, Bahasa program atau sistem operasinya.

*Black box testing* memiliki kelemahan, yaitu:

1. *Tester* tidak pernah yakin apakah program tersebut benar-benar lulus uji.

2. Fungsi-fungsi yang salah atau hilang.

3. Kesalahan *interface.*

4. Kesalahan dalam struktur data atau akses *database* eksternal.