BAB 2

LANDASAN TEORI

2.1. Perangkat Lunak

Menurut Pressman (2002, p10), perangkat lunak dapat diartikan dalam beberapa bentuk definisi, antara lain:

1. Perintah (program komputer) yang bila dieksekusi memberikan fungsi dan unjuk kerja seperti yang diinginkan.
2. Struktur data yang memungkinkan program memanipulasi informasi secara proporsional.
3. Dokumen yang menggambarkan operasi dan kegunaan program.

2.1.1. Pengertian Rekayasa Perangkat Lunak

Fritz Bauer [NAU69] mengusulkan definisi rekayasa perangkat lunak yang menjadi dasar dari seluruh diskusi pengertian rekayasa perangkat lunak, yaitu: pengembangan dan penggunaan prinsip pengembangan suara untuk memperoleh perangkat lunak secara ekonomis yang handal dan bekerja secara efisien pada mesin nyata (Pressman, 2002, p27-28).

Menurut Pressman (2002, p28), definisi rekayasa perangkat lunak telah dikembangkan oleh IEEE [IEEE93] menjadi lebih komprehensif, yaitu sebagai berikut:

1. Aplikasi dari sebuah pendekatan kuantifiabel, disiplin, dan sistematis kepada pengembangan, operasi, dan pemeliharaan perangkat lunak, yaitu aplikasi dari rekayasa perangkat lunak.
2. Studi tentang pendekatan-pendekatan seperti pada aplikasi tersebut.

2.1.2. Karakteristik Perangkat Lunak

Menurut Pressman (2002, p11-14), perangkat lunak yang lebih merupakan elemen logika memiliki ciri berbeda dari perangkat keras, yaitu sebagai berikut:

1. Perangkat lunak dibangun dan dikembangkan, tidak dibuat dalam bentuk klasik. Meskipun banyak kesamaan antara perangkat keras dan perangkat lunak, aktivitas keduanya secara mendasar sangat berbeda. Kualitas yang tinggi dicapai melalui perancangan yang baik, tetapi di dalam fase pembuatan perangkat keras, selalu saja ditemukan masalah kualitas yang tidak mudah untuk disesuaikan dengan perangkat lunak.
2. Perangkat lunak tidak rentan terhadap pengaruh lingkungan yang merusak yang menyebabkan perangkat keras menjadi usang, tetapi perangkat lunak bisa memburuk dari segi kualitas karena perubahan yang dilakukan untuk mengatasi kegagalan. Sedangkan perangkat keras mulai menjadi usang setelah terkena pengaruh lingkungan seperti penumpukan debu, getaran, ketidakhati-hatian, suhu tinggi, dan yang lainnya.
3. Sebagian besar perangkat lunak dibuat secara *custom-built*, serta tidak dapat dirakit dari komponen yang sudah ada. Maksudnya adalah perangkat lunak tidak hanya dibuat menggunakan satu metode saja, tetapi bisa menggunakan 2 metode atau lebih.

   Ada beberapa teknik fundamental pengembangan perangkat lunak yang dapat membantu meningkatkan kualitas dan efektivitas perangkat lunak, yaitu sebagai berikut:

a. Penggunaan aset yang bisa digunakan kembali (*reusable*), sebagaimana telah banyak diketahui, dengan menggunakan komponen yang dapat digunakan kembali akan menghemat banyak biaya. *Software engineer* tidak perlu membangun semuanya dari awal. Cukup menggunakan komponen yang sudah siap digunakan untuk menghemat waktu dan biaya.

b. Penggunaan bahasa pemrograman yang bersifat umum. Bahasa pemrograman seperti C++, Java, C#, dan lain sebagainya menyediakan kemampuan umum untuk mengembangkan perangkat lunak dengan berbagai variasi. Model bahasa pemrograman ini, biasanya memerlukan pustaka khusus atau bahkan *framework* untuk menyelesaikan permasalahan yang kompleks.

c. Penggunaan bahasa pemrograman yang bersifat khusus. Meskipun fungsionalitas tipe bahasa pemrograman ini dapat dijalankan oleh *general purpose programming language*, seringkali pada situasi tertentu lebih mudah menggunakan *special purpose programming language*, seperti *Structured Query Language* (SQL) untuk memanipulasi basis data. *General purpose programming language* merupakan bahasa pemrograman untuk keperluan berbagai macam hal pada komputer yang digunakan. Contoh bahasa pemrograman ini diantaranya adalah Pascal, Assembly, Java, dan PHP. Sedangkan *special purpose programming language* merupakan bahasa pemrograman yang mengkhususkan

untuk membuat satu macam hal saja, contohnya SQL untuk basis data.

d. Pemodelan dengan notasi khusus secara ekspresif mampu meningkatkan kualitas program juga sekaligus meningkatkan efisiensi program. Notasi diagram seperti *Unified Modelling Language* (UML), atau diagram basis data seperti *Entity Relationship Diagram* (ERD) mampu meningkatkan kualitas perangkat lunak dengan biaya yang lebih rendah karena mempunyai tingkat abstraksi yang tinggi.

2.1.3. Tahap Perancangan Perangkat Lunak

Membangun suatu perangkat lunak adalah pekerjaan yang besar dan sangat kompleks, karena itu diperlukan tahap-tahap yang dapat mengorganisasikan pekerjaan agar menjadi lebih mudah untuk dikontrol dan diketahui perkembangannya. Pada umumnya tahap-tahap itu adalah:

1. Investigasi Sistem (*System Investigation*)

   Pada tahap ini pihak pengembang perangkat lunak melakukan perhitungan awal proyek berdasarkan kebutuhan pengguna awal yang dimiliki, setelah itu pengembang akan memutuskan apakah perangkat lunak layak untuk dikerjakan atau tidak.

2. Analisis Sistem (*System Analysis*)

   Pada tahap ini pihak pengembang akan melakukan analisis atas apa saja yang dibutuhkan perangkat lunak ini, baik dari segi perangkat lunak, perangkat keras, maupun pengguna.

3. Desain Sistem (*System Design*)

   Desain sistem adalah tahap dimana mekanisme dari aplikasi perangkat lunak mulai dibentuk.

4. Pemrograman (*Programming*)

   Pemrograman adalah tahap dimana konsep dan desain yang telah dibuat pada tahap-tahap sebelumnya diterjemahkan ke dalam suatu bentuk bahasa pemrograman.

5. Pengujian (*Testing*)

   Pada tahap ini dilakukan pengecekan terhadap kesalahan yang mungkin terjadi pada aplikasi.

6. Implementasi (*Implementation*)

   Implemetasi adalah tahap dimana aplikasi telah selesai dan telah menjadi suatu sistem yang berjalan. Kegiatan yang dilakukan pada tahap ini adalah mengintegrasikan sistem dengan lingkungan dimana sistem akan berjalan.

7. Operasi (*Operation*)

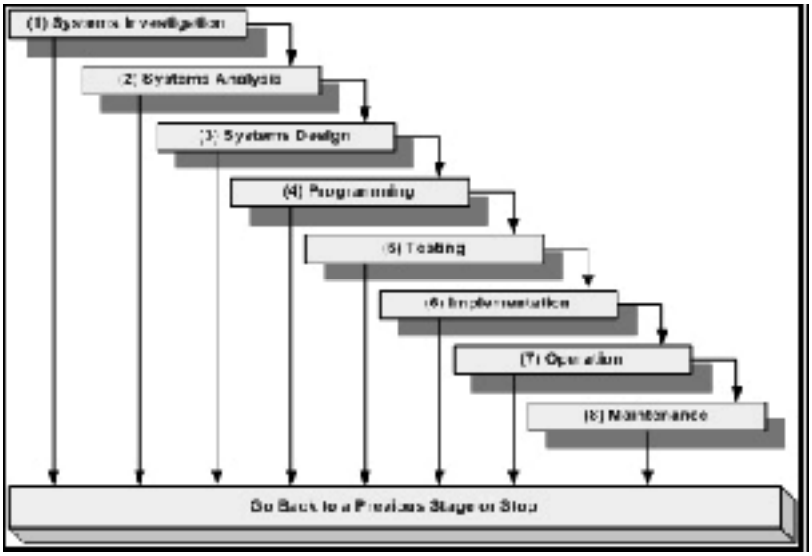
   Operasi adalah tahap dimana aplikasi telah selesai diimplementasi dan diintegrasikan dengan lingkungan dimana sistem siap dioperasikan.

8. Perawatan (*Maintenance*)

   Kegiatan pada tahap akhir ini adalah menjaga agar aplikasi tetap dapat berjalan dengan baik.

Tahap-tahap tersebut sering juga disebut sebagai daur hidup perangkat lunak (*software life-cycle*). Tahap-tahap di atas merupakan bagian dari model

pengembangan perangkat lunak seperti *waterfall, incremental,* dan *prototyping*. Di bawah ini adalah gambar dari model *software life-cycle*:



Gambar 2.1 - Model *software life-cycle* (Turban, Rainer, Potter, 2001, p477)

2.1.4. Metodologi yang Digunakan

Pada penelitian ini, penulis menggunakan *evolutionary development approach*, yang di dalamnya terdapat metode *prototyping*. *Evolutionary development approach* adalah proses secara berulang. Pendekatan tersebut mempunyai karakteristik yang memungkinkan pengembang perangkat lunak untuk mengembangkan aplikasinya menjadi versi yang lebih komplit.

Metode *prototyping* digunakan saat klien sulit untuk memberikan kebutuhan pengguna (*user requirement*). Selain itu, pengembang aplikasi merasa tidak yakin dengan tingkat efisiensi dari sebuah algoritma, tingkat adaptasi dari sistem operasi, atau tampilan GUI (*Graphical User Interface*) yang akan dilihat oleh pengguna. Oleh sebab itu, maka metode *prototyping* memberikan pendekatan yang terbaik (Pressman, p83).

Prototyping dimulai dengan komunikasi antara pengembang aplikasi dan klien untuk menentukan tujuan dari perangkat lunak dan identifikasi segala kebutuhan yang diketahui. *Prototyping* dirancang secara cepat antara model dan desain. Desain yang cepat difokuskan untuk representasi dari segala aspek perangkat lunak yang akan dilihat oleh klien/pengguna (GUI), dan ini mengharuskan untuk pembuatan *prototype*. *Prototype* diluncurkan dan dievaluasi oleh klien/pengguna. Umpan balik (*feedback*) dibutuhkan untuk membenahi kebutuhan atas perangkat lunak. Proses perulangan yang terjadi pada *prototype* digunakan untuk memuaskan kebutuhan klien, dan pada saat yang sama menjadikan pengembang aplikasi untuk lebih memahami apa yang harus dilakukan selanjutnya.

Langkah-langkah dari *prototyping*:

1. Identifikasi kebutuhan dasar, yaitu dengan cara menentukan kebutuhan dasar yang mencakup *input* dan *output* atas informasi yang diinginkan. Kebutuhan detail seperti keamanan dapat diabaikan terlebih dahulu.
2. Mengembangkan *prototype* awal yang biasanya hanya untuk tampilan layar (GUI).
3. Klien, termasuk pengguna, menguji *prototype* dan memberikan *review* berupa umpan balik sebagai tambahan atau perubahan.
4. Melakukan revisi dan peningkatan *prototype* yang didapat dari umpan balik. Dengan umpan balik yang didapat, spesifikasi atas aplikasi dan *prototype* dapat ditingkatkan. Jika perubahan dilakukan, maka ulangi kembali langkah ke-3 dan ke-4.

Dimensi dari *prototype* terbagi 2, yaitu:

1. *Prototype* secara horizontal.

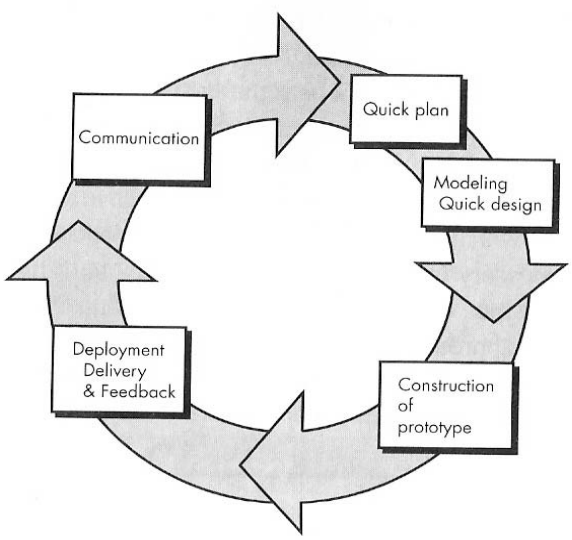
   Contoh yang paling sering dijumpai adalah sebuah *prototype* tampilan layar. Bentuk ini menggambarkan secara luas atas keseluruhan sistem dan subsistem, lebih fokus ke tampilan layar daripada fungsionalitas sistem, contohnya akses basis data. Bentuk ini cocok digunakan untuk:

   - Konfirmasi dari kebutuhan tampilan layar dan jangkauan sistem.
   - Versi demo dari sistem untuk memperoleh kebutuhan bisnis.
   - Membuat estimasi dari waktu pengembangan, biaya, dan hasil.

2. *Prototype* secara vertikal.

   Prototype ini lebih komplit dari sebuah fungionalitas atau subsistem. Bentuk ini berguna untuk mendapatkan detail dari kebutuhan untuk fungsi yang diberikan. Keuntungannya adalah:

   - Perbaikan desain basis data.
   - Mendapatkan informasi dari banyaknya data dan kebutuhan sistem untuk menentukan besarnya jaringan dan performa.
   - Mengklarifikasi kebutuhan secara kompleks dengan memecah fungsionalitas sistem.



Gambar 2.2 – Model *prototyping* (Pressman, p84)

2.2. Sistem Basis Data

2.2.1. Pengertian Sistem Basis Data

Menurut Gerald V. Post (Gerald, 2005, p2), sistem basis data adalah koleksi banyak data yang tersimpan dalam format yang telah distandarisasi dan dirancang untuk dapat digunakan oleh banyak pengguna.

Sedangkan menurut Thomas Connolly, (Connolly, 2002, p14), basis data merupakan sebuah kumpulan dari data yang berhubungan secara logis yang dapat digunakan secara bersama-sama, dan sebuah deskripsi dari data tersebut dirancang untuk mendapatkan informasi yang diperlukan dari sebuah organisasi.

Basis data juga didefinisikan sebagai *self-describing collection of integrated records*, artinya struktur dari basis data (*metadata*) disimpan di dalam sistem basis data, bukan di dalam aplikasi program. Selain itu, DBMS (*Database Management System*)-lah yang mengintegrasikan data yang dibutuhkan, bukan

programmer. Deskripsi data dikenal dengan katalog sistem (kamus data atau *metadata* – data tentang data).

Tujuan dari sistem basis data secara keseluruhan adalah untuk melakukan perawatan informasi dan menyajikan kapan saja dibutuhkan.

Menurut Connolly (2002, p8), sistem basis data terdiri dari:

a. *Field*, merupakan unit terkecil dalam suatu *record* yang merepresentasikan karakteristik dari sebuah objek.

b. *Record*, merupakan data yang berhubungan secara logis dari satu *field* atau lebih.

c. *File*, merupakan kumpulan dari *record* yang membentuk suatu kesatuan data.

Selain itu, beberapa objek basis data (Connolly, 2002, p15) antara lain:

- *Entity*

  Entity adalah sebuah objek (berupa orang, tempat, benda, atau *event*) yang berkaitan dengan informasi untuk direpresentasikan dalam basis data.

- *Attribute*

  Attribute adalah properti yang mendeskripsikan setiap karakteristik objek yang akan kita masukkan dalam *record*.

- *Relationship*

  Relationship adalah hubungan atau kaitan antar *entity*.

Beberapa *relational keys* dalam basis data (Connolly, 2002, p78-79):

- *Super Key*

  Super key adalah sebuah set atribut, yang secara unik mengidentifikasi setiap baris elemen dalam satu relasi.

- *Candidate Key*

  Candidate key yaitu *super key* yang pantas untuk dijadikan kandidat *primary key* pada suatu relasi.

- *Primary Key*

  Primary key adalah *candidate key* yang dipilih untuk menghubungkan dan mengidentifikasikan *record* yang unik dalam sebuah relasi. *Primary key* sebagai tanda pengenal dalam suatu *field*.

- *Foreign Key*

  Foreign key adalah sebuah atribut, yang dapat sesuai dengan c*andidate key* dari beberapa relasi yang sama.

2.2.2. Komponen Basis Data

Komponen-komponen basis data (Connolly, 2002, p18-20) antara lain:

1. Perangkat Keras

Perangkat keras komputer berfungsi untuk memproses dan mengelola data. Perangkat keras dapat berupa sebuah personal komputer, sebuah *mainframe*, sampai jaringan banyak komputer (*multi-computer*).

2. Perangkat Lunak

   Perangkat lunak adalah komponen yang menghubungkan fisik basis data aplikasi program yang digunakan pengguna.

3. Data

   Hal terpenting dalam komponen basis data. Data berperan sebagai jembatan antara komponen mesin dan komponen manusia. Basis data mengandung data operasional serta *metadata* (data tentang data).

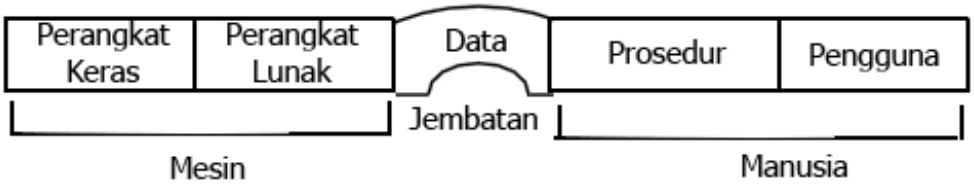
4. Prosedur

   Prosedur adalah suatu instruksi atau aturan yang digunakan dalam membuat desain dan penggunaan dari basis data.

5. Pengguna

   Pengguna adalah komponen akhir/pengguna dari basis data yang terlibat di dalam sistem. Jenis-jenis pengguna basis data:

   - Admin basis data (*Database administrator*).
   - Perancang basis data (*Database designer*).
   - Pengembang aplikasi (*Application developers*).
   - Pemakai (*End user*).



Gambar 2.3 - Hubungan komponen basis data (Connolly, 2002)

2.2.3. MySQL

MySQL adalah basis data *multiuser* yang menggunakan bahasa SQL. MySQL merupakan perangkat lunak yang bersifat *open source*. SQL adalah singkatan dari *Structured Query Language*, merupakan bahasa standar untuk pengolahan basis data. SQL mulai dikembangkan pada akhir tahun 1970-an di laboratorium IBM (Bimo Sunarfrihantono, ST, 2002, p65).

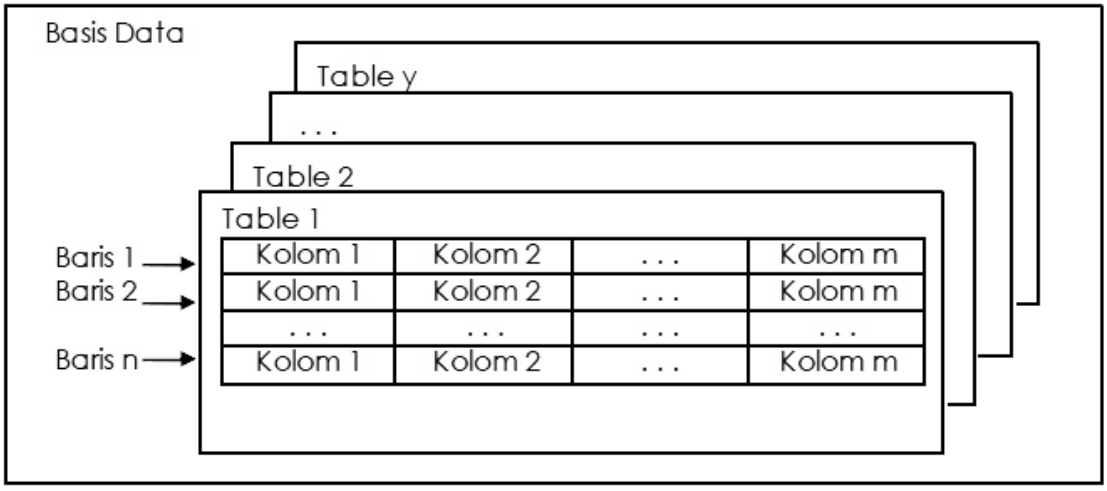
MySQL mampu menangani data yang cukup besar. Perusahaan yang mengembangkan MySQL yaitu TcX, mengaku mampu menyimpan data lebih dari 40 basis data, 10.000 tabel, dan sekitar 7 juta baris. Informasi selengkapnya tentang MySQL dapat dilihat di *www.mysql.com*.

Pemakaian basis data MySQL yang dimaksud adalah pengembang aplikasi basis data yang ingin menggunakan MySQL sebagai basis datanya atau *back end* dari aplikasi yang dibangun. Sedangkan MySQL mempunyai kelebihan dapat diakses oleh banyak bahasa pemrograman. MySQL merupakan perangkat lunak server basis data yang ideal untuk data segala ukuran dengan kemampuan mempunyai kecepatan yang sangat tinggi dalam pemrosesan data, *multi-threaded, multi-user* dan *query*. Ukuran basis data MySQL lebih kecil bila dibandingkan *file* basis data yang lain.

Beberapa pertimbangan *programmer* memilih MySQL sebagai *back end* dalam mengolah basis data yaitu kecepatan, mudah digunakan, *open source*, kapabilitas, keamanan, dan lintas *platform*.

2.2.4. Basis Data, Tabel, Baris, dan Kolom

Dalam konteks bahasa SQL, pada umumnya informasi tersimpan dalam tabel-tabel yang secara logika merupakan struktur dua dimensi yang terdiri atas baris-baris data yang berada dalam satu atau lebih kolom. Baris pada tabel sering disebut sebagai *instance* dari data, sedangkan kolom sering disebut sebagai atribut atau *field*. Keseluruhan tabel itu dihimpun dalam satu kesatuan yang disebut basis data. MySQL dapat melakukan proses *query* dan menentukan informasi yang diinginkan. Nilai yang dikembalikan dari proses *query* berupa *recordset* atau hasil yang ditampilkan berupa baris-baris pada kolom, berdasarkan *query* tertentu pada tabel tertentu.



Gambar 2.4 - Deskripsi konsep basis data (Bimo Sunarfrihantono, ST, 2002, p66)

2.3. Multimedia

2.3.1. Pengertian Multimedia

Menurut Hofstetter (2001, p2) multimedia adalah penggunaan komputer untuk mempresentasikan dan menggabungkan teks, grafik, audio, dan video dengan hubungan dan perlengkapan yang memperbolehkan pengguna untuk menavigasi, berinteraksi, membuat, dan berkomunikasi.

Multimedia memiliki komponen utama, yaitu:

1. Komputer, untuk mengkoordinasi apa yang dilihat dan didengar, dan berinteraksi dengan pengguna.
2. Tautan (*link*), sebagai koneksi dengan informasi.
3. Alat navigasi, untuk menghubungkan antara pengguna dengan jaringan informasi yang ada.
4. Multimedia harus memiliki jalan untuk mendapatkan, memproses dan berkomunikasi dengan informasi dan ide yang tersedia.

2.3.2. Alasan Pentingnya Multimedia

Multimedia menjadi suatu kemampuan dasar yang penting dalam kehidupan di abad 21. Faktanya, multimedia merubah kebiasaan cara membaca. Jika biasanya membaca berupa presentasi teks yang tercetak dalam buku, maka multimedia menjadikan kegiatan membaca menjadi lebih dinamis, dengan menyampaikan kata-kata dalam dimensi baru. Dalam membawa arti yang dimaksud, kata-kata dalam multimedia menyajikan sarana agar pembaca dapat memperluas teks yang ada, untuk mengetahui lebih banyak tentang sebuah topik.

Tidak hanya dengan menyajikan teks yang lebih banyak, tetapi menambahkan dengan suara, gambar, musik, atau video.

2.3.3. Objek Dalam Multimedia

Menurut Fred T. Hofstetter dalam bukunya, Multimedia Literacy (2001, p16-27), ada lima objek dalam multimedia.

1. Teks

   Pada umumnya, multimedia pasti mempergunakan teks, karena teks merupakan sarana yang efektif untuk mengkomunikasikan ide dan menyajikan instruksi kepada user.

   Terdapat empat jenis teks, yaitu *printed text*, *scanned text*, *electronic text*, dan *hypertext*.

   a. *Printed Text*

      Merupakan teks yang tercetak pada kertas. Apabila kita akan mempergunakan *printed text* sebagai bagian dari dokumen multimedia, maka kita harus mentransformasikan teks tersebut ke dalam bentuk yang dapat dimengerti oleh mesin. Cara yang biasa dilakukan adalah dengan menyalin atau mengetik ulang teks tersebut ke sebuah *text processor* atau *text editor*, namun sangat melelahkan dan menghabiskan banyak waktu.

   b. *Scanned Text*

      Untuk mengkonversi *printed text* menjadi bentuk yang dimengerti oleh mesin, dalam hal ini adalah *scanned text*, kita dapat mempergunakan *scanner*.

c. *Electronic Text*

Merupakan bentuk teks yang dapat dibaca oleh komputer, dan ditransmisikan secara *online* ke jaringan.

d. *Hypertext*

Kata '*hyper*' berarti proses tautan, yang menjadikan multimedia interaktif. Kata *hypertext* yang ditemukan oleh Ted Nelson (1965) artinya teks yang telah dihubungkan dengan sebuah tautan. Ketika kita melihat sebuah *hypertext* lalu kita mengklik kata yang telah terhubung dengan tautan, maka komputer akan menuju ke objek yang berada dalam tautan tersebut. Tautan tersebut memberikan sebuah dimensi baru ke dalam teks, sehingga disebut *hyper*.

2. Grafik

Sering dikatakan bahwa sebuah gambar mengandung arti ribuan kata-kata. Grafik seringkali dimunculkan di samping teks, sebagai *pictorial framework* dari teks tersebut.

Ada berbagai bentuk grafik, antara lain:

a. *Bitmap*

Merupakan gambar yang tersimpan sebagai *set pixel* yang menunjukan barisan titik pada layar komputer. Untuk menampilkan gambar tersebut, komputer menempatkan tiap titik pada layar sebagai warna yang telah dispesifikasikan dalam *bitmap*.

b. Gambar Vektor

Gambar vektor tersimpan dalam bentuk persamaan matematika yang disebut algoritma, yang mendefinisikan kurva, garis, dan bentuk dalam sebuah gambar. Untuk gambar yang tidak mengandung banyak pergantian warna, vektor merupakan cara yang lebih efisien untuk menyimpan gambar, dibandingkan dengan *bitmap*.

Gambar vektor memiliki dua keuntungan dibandingkan dengan *bitmap*. Yang pertama, vektor dapat diubah skalanya, berarti kita dapat menambah atau mengurangi ukuran gambar tanpa kehilangan kualitasnya. Kedua, gambar vektor memiliki ukuran *file* yang lebih kecil dibandingkan *bitmap*, jadi lebih cepat diunduh di Internet.

c. *Clip Art*

Untuk menghemat waktu dalam pembuatan aplikasi multimedia, kita dapat menggunakan sebuah *library* yang berisi *clip art*. Ada beberapa kategori dari *clip art* termasuk foto, ikon, animasi, latar belakang dan tombol.

d. Gambar Digital

Gambar digital adalah gambar yang didapatkan dari sebuah *frame* dari rekaman kamera, *Video Cassette Recorder* (VCR), *Video Compact Disc* (VCD) atau *live video* lain yang diambil dan digunakan pada aplikasi multimedia.

e. *Hyperpictures*

Hyperpictures adalah sebuah gambar di mana bagian-bagian dapat digunakan sebagai objek sebagai pemicu objek lain atau *event-event* pada aplikasi multimedia.

3. Suara

Terdapat empat objek suara yang dapat dipergunakan dalam multimedia, yaitu: *waveform audio, MIDI soundtrack, CD audio,* dan *file* MP3.

a. *Waveform Audio*

Setiap suara memiliki *waveform* yang menjelaskan frekuensi, amplitudo, dan isi harmonik dari suara tersebut. *Waveform Audio Digitizers* adalah alat untuk merekam suara dengan menangkap *waveform* suara tersebut ribuan kali per detik. Rekaman ini tersimpan dalam komputer, biasanya memiliki ekstensi *file* .wav, yang berarti *waveform.*

b. MIDI

MIDI singkatan dari *Musical Instrument Digital Interface.* MIDI merekam informasi yang dibutuhkan oleh *chip* suara komputer, untuk memutar musik. *File* MIDI memiliki ekstensi .mid.

c. *Audio Compact Disc* (CD)

Audio CD dapat mencakup lebih dari 75 menit rekaman suara berkualitas tinggi. *Sampling rate* 44.100 *sample* per detik, yang berarti cukup cepat untuk merekam setiap suara yang terdengar oleh manusia.

d. MP3

MP3 adalah kepanjangan dari *MPEG Audio Layer 3.* Merupakan format *file* audio yang menggunakan *MPEG audio codec* untuk meng-*encode*

(kompresi) dan men-*decode* (dekompresi) rekaman musik MP3 dapat meng-*encode* sebuah *track audio CD* kedalam bentuk yang memiliki ukuran lebih kecil, sehingga membutuhkan lebih sedikit *bandwidth* untuk men-*transmit file* tersebut ke Internet.

4. Video

Video menyajikan sumber yang kaya dan secara langsung untuk aplikasi multimedia. Terdapat empat tipe video dapat dipergunakan sebagai tautan objek pada aplikasi multimedia, yaitu:

a. *Live Video*

Live video menyajikan objek multimedia secara *real-time*, seperti *channel* televisi atau siaran yang ditangkap langsung oleh kamera.

b. *Videotape*

Videotape terbatas pada dua faktor utama, yaitu:

Pertama, *videotape* merupakan bentuk linear. Informasi yang dalam *videotape* direkam secara serial, dan untuk mengakses informasi tersebut, kita harus menunggu lama untuk memajukan atau memundurkan *videotape* untuk memperoleh informasi yang diinginkan.

Kedua, *videotape player* tidak dioperasikan dengan komputer. Artinya, kita harus secara manual menekan tombol *play*, *stop*, *fast-forward*, dan *rewind* dalam presentasi multimedia.

c. *Videodisc*

Terdapat dua format *videodisc*, yaitu *Constant Angular Velocity* (CAV) dan *Constant Linear Velocity* (CLV).

Disk CAV dapat menyimpan lebih dari 54.000 *frame* atau 30 menit video bergerak dengan *soundtrack* stereo. Sedangkan disk CLV dapat menyimpan lebih dari satu jam video pada setiap sisinya.

d. Video Digital

Video digital merupakan media penyimpanan video yang paling menjanjikan. Seperti *waveform audio*, *file* video digital tersimpan dalam *harddisk*, CD-ROM, atau DVD. Karena video digital, maka dapat disajikan di komputer melalui jaringan Internet. Banyak sekali istilah-istilah baru seperti format MPEG (*Moving Picture Expert Group*), 3GP (*Third Generation Partnership Project*), AVI (*Audio Video Interleave*)**,** WMA **(***Windows Media Audio*)**,** MOV (*Marconi-Osram Valve*), WMV (*Windows Media Video*)**,** 3G2 (*3GPP2 file format*), MP4 (salah satu format berkas pengodean suara dan gambar/video digital yang dikeluarkan oleh sebuah organisasi MPEG) yang semuanya menunjuk ke berbagai macam format *file* multimedia. *File* multimedia yang dimaksud adalah hasil konversi dari format multimedia analog (film dan musik yang menggunakan media pita analog), menjadi format yang sesuai dengan sistem operasi komputer.

5. Animasi

Dalam multimedia, animasi adalah penggunaan komputer untuk menciptakan pergerakan pada layar. Ada beberapa jenis animasi, antara lain:

a. *Frame Animation*

Frame animation membuat objek bergerak dengan cara menampilkan sejumlah seri gambar, yang disebut *frame*, yang ditampilkan di posisi berbeda pada layar.

b. *Vector Animation*

Vektor adalah sebuah garis yang memiliki awal, arah, dan panjang animasi vektor membuat objek bergerak dengan menentukan variabel dari tiga parameter tersebut yang mendefinisikan sebuah objek.

c. *Computational Animation*

Pada *computational*, untuk membuat objek bergerak adalah dengan merubah koordinat x dan y pada layar.

d. *Morphing*

Morphing merupakan proses transisi dari suatu bentuk ke bentuk yang lain dengan cara menampilkan sejumlah seri *frame*, yang membentuk pergerakan yang halus dari bentuk awal dan bertransformasi menjadi bentuk lain.

2.4 Interaksi Manusia dan Komputer

2.4.1 Pengertian Interaksi Manusia dan Komputer

Definisi Interaksi Manusia dan Komputer menurut Shneiderman (1998, p4) adalah disiplin ilmu yang berhubungan dengan perancangan, evaluasi, dan implementasi sistem komputer interaktif untuk digunakan oleh manusia, serta studi fenomena-fenomena besar yang berhubungan dengannya.

Sistem interaktif terus berkembang menjadi suatu hal yang penting seiring dengan perkembangan dalam penggunaan komputer. Dalam merancang suatu sistem yang interaktif, apabila hasil rancangannya baik maka pengguna dapat menggunakan sistem dengan lancar dan sesuai, serta pengguna dapat ikut berinteraksi dengan baik dalam penggunaannya. Oleh sebab itu, sistem yang baik biasanya merupakan suatu sistem yang mudah untuk digunakan atau bersifat *user-friendly*.

2.4.2 Delapan Aturan Emas

Delapan aturan emas mengenai antarmuka yang baik menurut Shneiderman (1998, p74-75) adalah:

1. Berusaha untuk konsisten (*strive for consistency*)

   Urutan tahap-tahap yang dilakukan harus konsisten, istilah-istilah yang identik harus digunakan pada *prompt*, menu, dan layar bantu, pewarnaan, layout, kapitalisasi, huruf, dan lainnya yang konsisten.

2. Memungkinkan *frequent user* menggunakan *shortcut* (*enable frequent users to use shortcuts*)

   Menyediakan tombol-tombol *shortcut* untuk aksi yang sering digunakan, pengguna mengharapkan disediakan *special keys*, perintah-perintah tersembunyi, dan fasilitas makro serta waktu respon yang singkat dan tampilan yang cepat.

3. Memberikan umpan balik yang informatif (*offer informative feedback*)

   Untuk setiap aksi yang dilakukan pengguna, harus disediakan umpan balik. Untuk aksi yang sering digunakan dan kecil, respon yang diberikan

sederhana, sedangkan untuk aksi yang jarang digunakan dan besar, respon yang diberikan harus lebih banyak dan rinci.

4. Merancang dialog yang memberikan keadaan akhir (*let the user know when they have completed a task*)

   Urutan aksi harus diatur dalam group dimana ada awal, tengah, dan akhir. Dengan adanya umpan balik dapat memberikan pilihan untuk menyiapkan grup aksi berikutnya.

5. Memberikan pencegahan kesalahan dan penanganan kesalahan sederhana (*offer simple error handling*)

   Desain sistem sedemikian rupa sehingga pengguna tidak melakukan kesalahan yang fatal. Misalnya adanya pilihan pada menu lebih baik dari mengisi *textbox* yang kosong.

6. Memungkinkan pembalikan aksi yang mudah (*permit easy undo*)

   Apabila memungkinkan, aksi harus bisa dibalik. Ciri ini mengurangi kegelisahan, karena pengguna tahu bahwa kesalahan dapat diperbaiki sehingga mendorong penjelajahan pilihan yang tidak biasa dipakai.

7. Mendukung pusat kendali internal (*provide a sense of user-control*)

   Operator yang berpengalaman menginginkan bahwa mereka bertanggung jawab terhadap sistem dan sistem merespon aksi yang diberikan, karena manusia yang memegang kontrol.

8. Mengurangi beban ingatan jangka pendek (*reduce short-term memory load*)

   Batasan jangka pendek pemrosesan informasi pada manusia memerlukan tampilan yang sederhana, tampilan banyak halaman digabungkan,

frekuensi pergerakan jendela tampilan dikurangi, dan waktu pelatihan yang cukup diberikan untuk kode-kode, hafalan, dan urutan aksi-aksi.

2.5 Internet

2.5.1 Pengertian Internet

Secara harfiah, Internet (kependekan daripada perkataan '*inter-network*') ialah rangkaian komputer yang saling berhubungan membentuk jaringan komputer. Sedangkan dari segi ilmu pengetahuan, Internet merupakan sebuah perpustakaan besar yang didalamnya terdapat banyak sekali informasi atau data yang dapat berupa teks, grafik, audio maupun animasi, dan lain-lain dalam bentuk media elektronik. Orang bisa berkunjung ke perpustakaan tersebut kapan saja dan dimana saja. Dari segi komunikasi, Internet adalah sarana yang sangat efesien dan efektif untuk melakukan pertukaran informasi jarak jauh, maupun didalam lingkungan perkantoran.

2.5.2 Istilah dalam Internet

Beberapa hal yang terkait dengan Internet antara lain sebagai berikut:

1. *World Wide Web* (WWW)

   Sering disebut "*the web*"/"*W3*", merupakan sistem Internet yang memiliki fasilitas pencarian dan pemberian informasi yang cepat dengan menggunakan teknologi *hypertext*.

   Sebutan *world wide web* (*web* = jaring laba-laba) sangat tepat untuk menggambarkan struktur data pada jaringan Internet. Berbeda dengan misalnya susunan data logis berstruktur pohon yang dikenal dari *Disk*

Operating System (DOS), WWW memungkinkan penanganan atau akses yang jauh lebih fleksibel pada *file* yang di kelola.

Di WWW, struktur sumber daya Internet dapat dibandingan dengan jaringan laba-laba. Bila dilihat polanya, jaringan ini terdiri atas lingkaran-lingkaran berbagai ukuran yang berpusat pada titik tengah yang sama. Dari titik tengah ini terbentuk garis-garis penghubung yang tegak lurus pada lingkaran, sehingga terdapat titik simpul. Bila pada struktur pohon percabangan merupakan jalur hubungan, pada *web* semua garis merupakan penghubung setiap titik simpul yang mengandung data. Pemilihan di sini dilakukan dengan *item hypertext*. Pada titik simpul bisa terdapat sebuah komputer di Internet atau sebuah petunjuk untuk *file* tertentu pada sebuah komputer. Hal ini berarti, dengan memilih sebuah *item hypertext* diciptakan hubungan dengan sebuah komputer pada suatu tempat di dunia, dimana anda dapat melanjutkan perjalanan atau langsung ke sebuah *file* tertentu.

2. Situs (*website*)

Sebuah situs adalah sebutan bagi sekelompok halaman *web* (*webpage*), yang umumnya merupakan bagian dari suatu nama domain (*domain name*) atau *subdomain* di *World Wide Web* (WWW). Di Internet, WWW terdiri dari seluruh situs yang tersedia kepada publik. Halaman-halaman sebuah situs di akses dari sebuah *Uniform Resource Locator* (URL) yang menjadi akar (*root*), yang disebut *homepage* (halaman induk; sering diterjemahkan menjadi beranda atau halaman muka), dan biasanya disimpan dalam server yang sama.

Tidak semua situs dapat diakses dengan gratis. Beberapa situs memerlukan pembayaran agar dapat menjadi pelanggan, misalnya situs-situs layanan surat elektronik (*e-mail*), situs pornografi, dan lain-lain. Halaman-halaman dari situs akan bisa diakses melalui sebuah URL yang biasa disebut *homepage*. URL ini mengatur halaman-halaman situs untuk menjadi sebuah hirarki, meskipun tautan-tautan yang ada di halaman tersebut mengatur para pembaca dan memberitahu mereka susunan keseluruhan dan bagaimana arus informasi ini berjalan.

Penemu situs adalah Sir Timothy John "Tim" Berners-Lee, sedangkan situs yang tesambung dengan jaringan, pertama kali muncul pada tahun 1991. Maksud dari Tim ketika membuat situs adalah untuk mempermudah tukar-menukar dan memperbarui informasi kepada sesama peneliti di tempat dia bekerja. Pada tanggal 30 April 1993, CERN (tempat dimana Tim bekerja) menginformasikan bahwa WWW dapat digunakan secara gratis oleh semua orang.

3. Surat Elektronik (*e-mail*)

   Surat elektronik atau pos elektronik atau nama umumnya dalam bahasa Inggris, *e-mail* adalah sarana kirim-mengirim surat melalui jalur Internet.

   E-mail sudah mulai dipakai di tahun 1960-an. Pada saat itu Internet belum terbentuk, yang ada hanyalan kumpulan *mainframe* yang terbentuk sebagai jaringan. Mulai tahun 1980-an, *e-mail* sudah bisa

dinikmati oleh khalayak umum. Sekarang ini banyak perusahaan pos di berbagai negara menurun penghasilannya disebabkan masyarakat sudah tidak memakai jasa pos lagi. Untuk mengirim *e-mail* kita memerlukan suatu program *mail-client*, *e-mail* yang kita kirim akan melalui beberapa titik sebelum sampai di tujuan. Untuk lebih jelasnya lihat skema dibawah. Contoh yang dipakai adalah layanan *Simple Mail Transfer Protocol* (SMTP) dan *Post Office Protocol Version 3* (POP3).

Saya menulis *e-mail* → *e-mail client* (di komputer saya) → server SMTP penyedia *e-mail* saya → Internet → server POP3 penyedia *e-mail* penerima → *e-mail client* (di komputer si penerima) → *e-mail* dibaca si penerima.

4. *Hypertext Transfer Protocol* (HTTP)

   HTTP adalah suatu protokol utama yang digunakan dalam *World Wide Web* (WWW) yang menentukan aturan yang harus diikuti *web browser* dalam meminta dan mengambil suatu dokumen, kemudian *web server* akan menyampaikan dokumen yang diminta *web browser*. Protokol ini merupakan standar dalam mengakses *Hypertext Markup Language* (HTML).

5. *Hypertext Markup Language* (HTML)

   HTML merupakan sistem yang digunakan untuk menciptakan halaman dan dokumen yang disajikan pada *web*.

6. *Uniform Resouce Locators* (URL)

   URL merupakan cara standar untuk menampilkan informasi tentang jenis isi dan lokasi *file*, lokasi komputer di Internet, letak *file* di dalam komputer dan protokol yang digunakan untuk mengakses *file* itu. Internet sangat besar, merupakan interkoneksi, terdistribusi, tempat yang sangat tidak seragam dan URL menstandarkan keanekaragaman ini.

2.6 Aplikasi Berbasis *Web*

2.6.1 Fungsi Umum Aplikasi Berbasis *Web*

Aplikasi berbasis *web* adalah perangkat lunak yang dijalankan pada server dengan bantuan antarmuka yang dijalankan pada *browser* klien. Bila pada aplikasi berbasis *desktop*, perangkat lunak menjalankan fungsinya secara langsung pada masing-masing komputer, perangkat lunak berbasis *web* hanya mengirimkan hasil jadi proses yang dijalankan di komputer server pada komputer klien dengan bantuan *web browser*.

Pada awal periode kemunculan Internet, situs masih berupa halaman-halaman statis berisi informasi yang tidak mengandung interaktifitas dengan pengunjung, sehingga tidak ada kebutuhan untuk memasukkan data secara dinamis. Semua pengunjung dianggap sama, baik itu pengunjung tamu atau klien penting atau bahkan administrator. Semua hanya mampu membaca halaman. Sekarang situs sama sekali berbeda dengan wajah halaman situs pada kemunculan pertama kalinya. Faktanya, sekarang hampir semua situs adalah aplikasi yang mempunyai fungsionalitas tinggi dan bergantung pada aliran

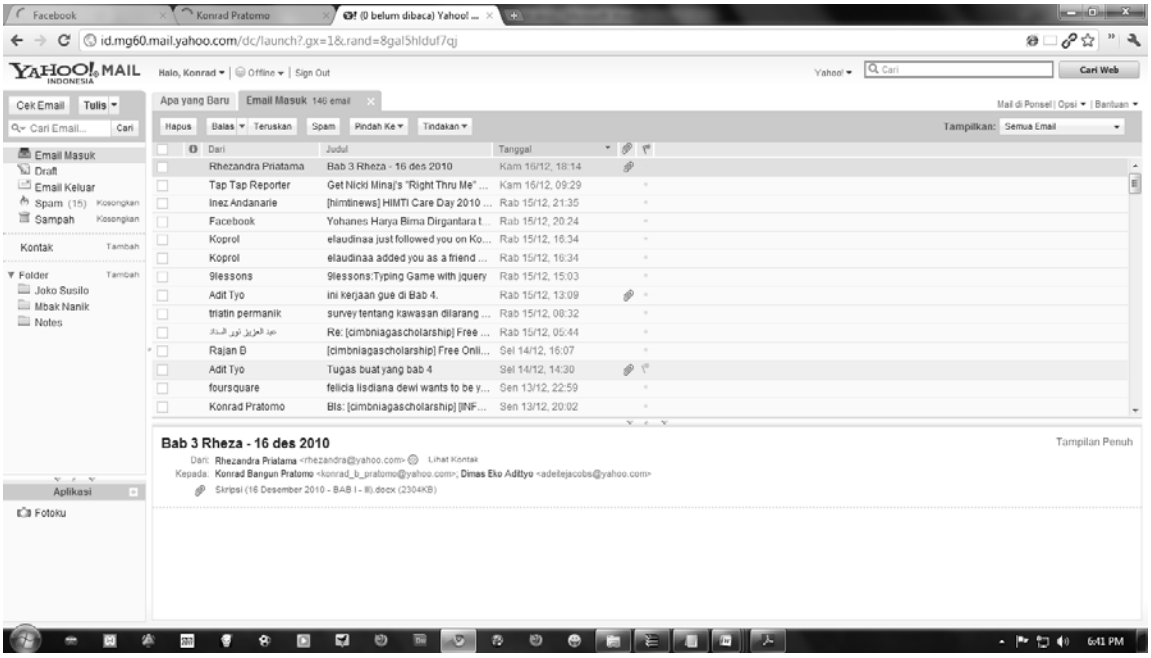
informasi dua arah. Sekarang situs telah mendukung registrasi, *login*, transaksi, dan konten yang dapat dibuat secara dinamis. Aplikasi *web* dibuat untuk menjalankan fungsi praktis yang sangat berguna dan dapat dijalankan secara *online*.

Beberapa contoh aplikasi berbasis *web* yang paling banyak digunakan sekarang adalah seperti:

- Aplikasi penjualan seperti Amazon (*http://www.amazon.com/*)
- Jejaring sosial seperti Facebook (*http://www.facebook.com/*)
- Perbankan seperti klikBCA (*http://www.klikbca.com/*)
- Mesin Pencari seperti Google (*http://www.google.com/*)
- Informasi interaktif seperti Wikipedia (*http://www.wikipedia.com/*)
- Surat elektronik seperti Yahoo! Mail (*http://mail.yahoo.com/*)
- *Web logs* seperti Wordpress (*http://wordpress.com/*)

Selain untuk keperluan jaringan Internet pubilk, aplikasi *web* juga banyak digunakan perusahaan secara internal untuk keperluan manajemen sumber daya dan informasi. Bahkan juga bisa digunakan untuk keperluan antarmuka perangkat keras dan perangkat lunak lain.

Banyak aplikasi berbasis *desktop* yang telah diintegrasikan dengan aplikasi *web*. Aplikasi bisnis seperti *Enterprise Resource Planning* (ERP), yang sebelumnya hanya bisa diakses dengan perangkat lunak berbasis *desktop*, sekarang dapat diakses menggunakan *web browser*.



Gambar 2.5 – *E-mail* sebagai salah satu bentuk aplikasi berbasis *web*

Semakin kompleksnya perangkat lunak berbasis *web* yang sekarang banyak dibutuhkan, perlu juga mempertimbangkan pemilihan arsitektur sistem yang digunakan. Pemilihan ini akan berakibat sangat besar pada *maintainability* dan *scalability* dari perangkat lunak yang kita kembangkan.

2.6.2 Keuntungan dan Tantangan Aplikasi Berbasis *Web*

Tidak sulit untuk melihat mengapa aplikasi berbasis *web* meningkat begitu pesat. Beberapa faktor teknis telah memicu perkembangan revolusi penggunaan Internet, di antaranya adalah:

1. *Hypertext Transfer Protocol* (HTTP), protokol komunikasi inti yang digunakan dalam mengakses *web* cukup ringan dan dapat bersifat *connectionless*, yaitu langsung terkoneksi tanpa harus melakukan otentifikasi digital.

2. Semua pengguna situs telah mempunyai *web browser* yang telah langsung ter-*install* di komputer mereka. Aplikasi *web* yang antarmuka penggunanya didistribusikan menggunakan *web browser*, sehingga pengguna tidak perlu memasang perangkat lunak independen sebagai syarat pemasangan aplikasi. Perangkat lunak hanya perlu di-*install* sekali pada server, dan langsung bisa dijalankan pada semua komputer klien karena secara langsung mereka telah memiliki *web browser* saat mereka memasang sistem operasi.
3. Saat ini *web browser* telah mempunyai fitur yang sangat mudah digunakan, selain itu juga antarmuka yang disuguhkan cukup kaya dan memuaskan. Antarmuka situs menggunakan navigasi standar dan kontrol masukan yang mudah dikenali oleh pengguna, sehingga pengguna tidak perlu mempelajari fungsi-fungsi khusus pada aplikasi tertentu.
4. Bahasa pemrograman yang digunakan untuk mengembangkan aplikasi berbasis *web*, relatif cukup mudah. Sudah banyak alat pengembangan yang dapat digunakan untuk mengembangkan perangkat lunak berbasis *web* secara mudah, bahkan oleh pengguna pemula sekalipun. Di samping itu, banyak juga alat pengembangan perangkat lunak berbasis *web* yang bersifat *open source* dan dapat digunakan siapa saja tanpa harus membayar royalti. Juga banyak contoh aplikasi berbasis *web* yang dapat digunakan dan dicontoh bahkan dapat diubah secara mudah karena bersifat *open source* yang disimpan dengan teknologi yang berbeda oleh pengembang yang berbeda. Sebab itulah sebisa mungkin aplikasi berbasis *web* mampu mengintegrasikan data yang dimiliki dengan data yang

mungkin didapat dari sumber lain, atau sebaliknya perangkat lunak tersebut harus menjamin data yang disimpan dapat diintegrasikan dengan sistem lain dengan perantara *middleware*.

5. Kecepatan pengembangan pada aplikasi situs. Perangkat lunak yang dirancangkan dengan baik dan dengan kualitas yang bagus adalah suatu keuntungan. Sebab itulah kecepatan waktu implementasi akan memberikan keuntungan dalam pengembangan, yaitu memperpanjang waktu perancangan dan *testing*.

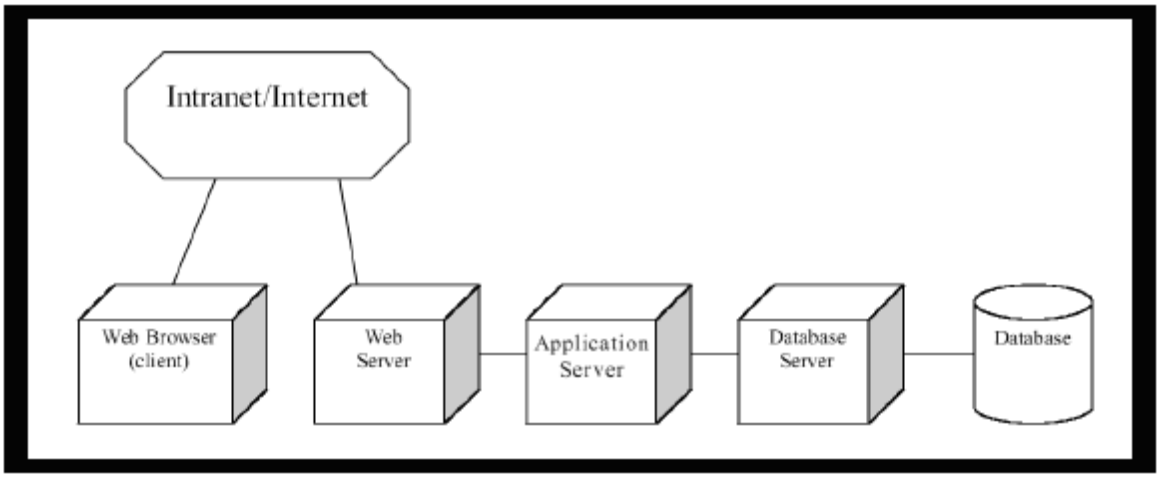
2.6.3 Arsitektur Aplikasi Berbasis *Web*

Secara umum ada tiga jenis aplikasi berbasis *web* berdasarkan skalanya, yaitu:

1. Yang pertama adalah situs skala kecil dengan hanya konten situs yang bersifat statis. Aplikasi situs model ini hanya menampilkan konten kepada pengunjung dan pengunjung sama sekali tidak memberikan masukkan apapun pada situs tersebut.
2. Yang kedua adalah situs skala menengah dengan hanya masukan minim dari pengunjung untuk memberikan umpan balik seperti buku tamu dan komentar.
3. Yang ketiga adalah situs skala besar dengan proses logika yang rumit seperti situs *e-commerce* dan aplikasi berbasis situs lain yang membutuhkan banyak tabel dari basis data.

Untuk menjalankan fungsinya, sebuah aplikasi berbasis *web* harus menjalankan serangkaian urutan pekerjaan yang melibatkan beberapa komponen

perangkat keras, sebagaimana ditampilkan pada gambar 2.6. Pertama pengguna berinteraksi menggunakan *web browser* dengan skrip pemrograman sisi klien seperti HTML dan JavaScript. Kemudian melalui jaringan Internet atau intranet permintaan yang dikirimkan klien diterima oleh *web server* untuk kemudian diolah oleh aplikasi server. Jika membutuhkan pengunduhan data dari basis data, aplikasi server memberikan perintah pada *Database Management System* (DBMS) untuk mengubah atau sekedar membaca data. Terakhir dari basis data proses berbalik kembali ke antarmuka pengguna yang menggunakan *web browser*.



Gambar 2.6 - Komponen aplikasi berbasis *web*

Pada aplikasi berbasis *web* klasik, pemrograman hanya ada pada sisi server, sedangkan pada aplikasi berbasis *web modern*, pemrograman ada pada dua sisi yaitu server dan klien. Pada sisi server pemrograman menggunakan bahasa pemrograman seperti PHP, ASP, JSP, ColdFusion, dan lain sebagainya. Sedangkan pada sisi klien pemrograman menggunakan JavaScript yang

dilengkapi dengan DOM HTML dan juga Applet (aplikasi kecil) yang berbasis Java.

Perkembangan teknologi pemrograman di sisi klien membawa aplikasi berbasis *web* ke tahap selanjutnya yaitu perlahan mengadopsi kemampuan pemrograman berbasis *desktop*. Dengan menggunakan Java Applet, aplikasi berbasis *web* mempunyai antarmuka yang mirip dengan aplikasi berbasis *desktop* dan bahkan sudah banyak permainan *online* yang dibuat dengan menggunakan Java Applet. JavaScript dan DOM HTML juga berkembang, dengan teknik pemrograman AJAX yang memungkinkan pengguna berkomunikasi dengan server tanpa harus me-*load* ulang halaman. Hasilnya sebagai implementasi dari *Rich Internet Application* (RIA), banyak aplikasi *desktop* yang diwujudkan dalam aplikasi berbasis *web* seperti Google Docs.

2.7 UML (*Unified Modelling Language*)

UML merupakan hasil pemikiran dari Grady Booch, James Rumbaugh, dan Ivar Jacobson. Belakangan sering dijuluki "*Three Amigos*", mereka bekerja di organisasi terpisah pada era 80-an dan awal 90-an, yang masing-masing merancang metodologinya sendiri untuk melakukan analisis dan desain yang berorientasi objek. Metodologi mereka mencapai keunggulan diantara kompetitor mereka. Pada pertengahan 90-an, saat mereka mulai saling membawakan ide-ide mereka satu sama lain, mereka juga memutuskan untuk mengembangkan pekerjaan mereka secara bersama-sama.

Pada tahun 1994, Rumbaugh masuk ke dalam *Rational Software Corporation* (RSC), dimana Booch telah bekerja disana. Setahun kemudian

Jacobson juga masuk ke (RSC). Versi *draft* dari UML mulai beredar dalam industri perangkat lunak dan umpan balik yang ada menghasilkan perubahan-perubahan yang substansial. Pada umumnya perusahaan-perusahaan banyak merasakan bahwa UML dapat menjalankan tujuan-tujuan strategis mereka. Atas dasar pertimbangan tersebut sebuah konsorsium UML pun dibangun. Anggota-anggota konsorsium ini termasuk DEC, Hewlett-Packard, Intellicorp, Microsoft, Oracle, Texas Instruments, Rational, dan yang lainnya. Pada tahun 1997, konsorsium ini mengeluarkan versi 1.0 dari UML dan dikoordinasikan dengan *Object Management Group* (OMG) sebagai tanggapan dari permintaan OMG untuk proposal dari bahasa pemodelan standar.

Konsorsium pun semakin berkembang, dan mengeluarkan versi 1.1 yang kemudian diadopsi OMG di tahun 1997. OMG pun terus mengembangkan UML dan menghasilkan revisi yang lebih banyak ditahun selanjutnya. UML kini telah menjadi standar secara *de-facto* di dunia industri perangkat lunak dan akan terus menerus berkembang. (Whitten et al, 2004, p430).

2.7.1 Pengertian UML

UML merupakan sekumpulan konvensi tentang pemodelan yang digunakan untuk menspesifikasi atau menggambarkan sistem perangkat lunak dalam hal-hal tentang objek (Whitten et al, 2004, p430).

UML terdiri dari sejumlah elemen grafikal yang digunakan untuk membentuk diagram-diagram. Tujuan diagram-diagram ini adalah untuk mempresentasikan berbagai pandangan terhadap sistem, dan sekumpulan pandangan ini yang disebut dengan model. Model dari UML berfungsi untuk

menggambarkan apa yang dilakukan oleh sistem, tetapi tidak menggambarkan bagaimana untuk mengimplementasikan sistem tersebut. (Schmuller, 1998, p8).

2.7.2 Diagram-diagram UML

Diagram-diagram yang terdapat dalam UML antara lain :

2.7.2.1 Diagram *Use Case*

Use case merupakan salah satu alat bantu dalam pengumpulan kebutuhan, perencanaan dan pengendalian proyek. Kebanyakan dari *use case* digunakan saat perencanaan proyek dan selebihnya ditemukan saat pengerjaan proyek.

Use case merupakan gambaran eksternal mengenai apa yang mampu dikerjakan oleh sistem atau aplikasi yang dibuat. Hal ini menguntungkan dalam mengkomunikasikan kepada pengguna terutama yang tidak mengerti hal teknis. (Martin Fowler, 2002, p64).

Unsur-unsur dari diagram *use case* antara lain :

1. Aktor

   Aktor merupakan segala sesuatu yang dibutuhkan dalam berinteraksi dengan sistem untuk pertukaran informasi. Dalam hal ini, aktor tidak selalu berupa manusia. Aktor dapat berupa suatu organisasi, sistem informasi lainnya, *external device* seperti sensor panas, atau bahkan konsep waktu.

2. *Use Case*

   Use case merupakan sekumpulan langkah-langkah (*scenario*) yang saling terhubung, baik secara otomatis maupun manual, yang bertujuan untuk meyelesaikan suatu proses bisnis.

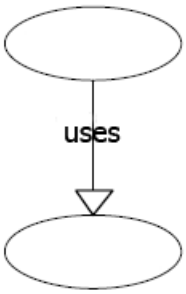
Gambar 2.7 - Aktor dan *Use Case* pada diagram *use case*

3. Hubungan antar-*use case*

   Ada beberapa hubungan antara *use case*, antara lain adalah:

   - *Uses*

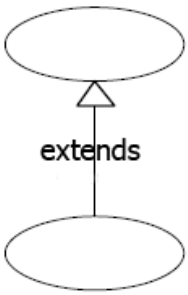
     Hubungan *uses* digambarkan dari suatu *use case* X ke suatu *use case* Y yang mengindikasikan bahwa proses untuk melakukan X selalu melibatkan Y, paling sedikit sekali. Singkatnya, menggunakan *uses* dalam bentuk X *uses* Y berarti X *has a* Y (proses X mengandung proses Y yang harus dikerjakan) sebagai bagian dari proses X.



Gambar 2.8 - Contoh *uses*

- *Extends*

  Hubungan *extends* digambarkan dengan *use case* X ke *use case* Y yang mengindikasikan bahwa proses X merupakan *case* dengan perlakuan spesial dari proses yang lebih umum dari *use case* Y. *Extends* dapat digunakan dalam situasi dimana sistem yang ada memiliki *use case* (proses) dimana ia memiliki beberapa subproses yang memiliki kesamaan, tetapi setiap subproses memiliki sesuatu yang berbeda yang tidak memungkinkan untuk mengelompokkan mereka didalam *use case* yang sama. (Whitten et al, 2004).



Gambar 2.9 - Contoh *extends*

2.7.2.2 Diagram *Class*

Diagram *class* menggambarkan data yang ada didalam sistem perangkat lunak. Diagram *class* merupakan dasar dari hampir seluruh metode *object oriented*. Oleh sebab itu, sudut pandang dari diagram *class* ini menjadi begitu luas dan terkadang ini menjadi masalah kita penggunaannya menjadi berlebihan. (Martin Fowler, 2002, p.64).

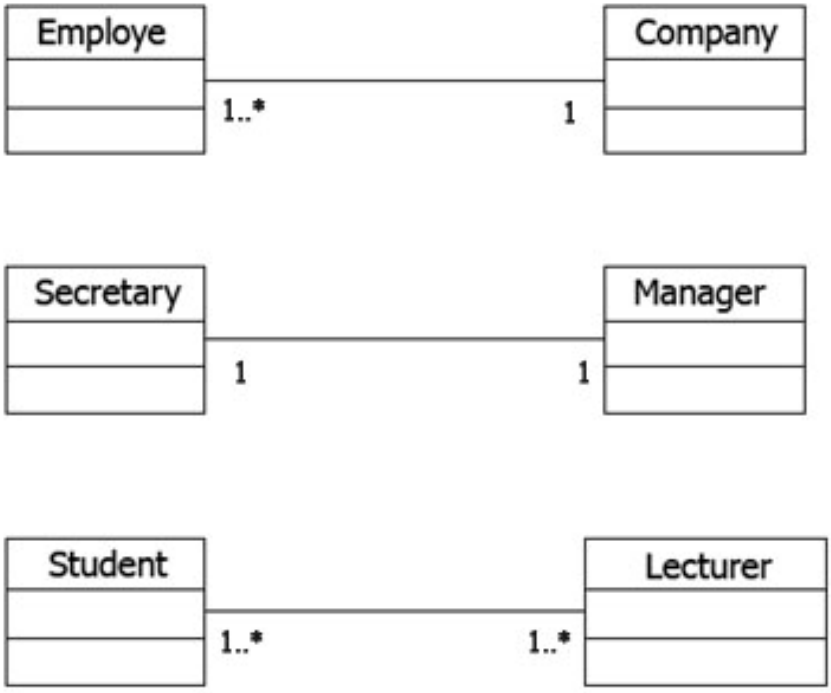
Berbagai simbol penting yang hadir didalam diagram *class* antara lain:

1. *Class*, yang berfungsi untuk mempresentasikan tipe dari data yang dimilikinya. Diagram *class* dapat ditampilkan dengan menunjukkan atribut dan operasi yang dimilikinya, atau hanya menunjukkan nama *class*-nya saja. Dapat juga kita tuliskan nama *class* dengan atributnya saja atau nama *class* dengan operasinya.

Gambar 2.10 – Beberapa bentuk penyajian diagram *class*

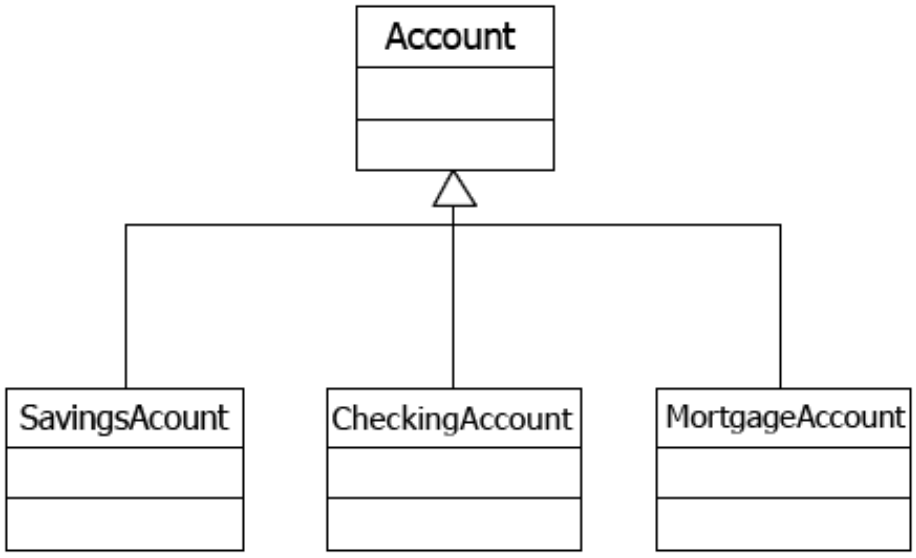
2. Atribut, merupakan data yang terdapat didalam *class* dan *instance*-nya dengan operator.
3. *Operation*, yang berfungsi untuk mempresentasikan fungsi-fungsi yang ditampilkan oleh *class* dan *instance*-nya dengan komputer.

4. Asosiasi, digunakan untuk menunjukkan bagaimana dua *class* berhubungan satu sama lainnya. Asosiasi ditunjukkan dengan sebuah garis yang terletak diantara dua *class*. Didalam sebuah asosiasi terdapat *multiplicity*, yaitu simbol yang mengindikasikan berapa banyak *instance* dari *class* pada ujung asosiasi yang satu dengan *instance class* diujung asosiasi yang lainnya.



Gambar 2.11 - Contoh hubungan antar-*class* dengan berbagai jenis *multiplicity*

5. Generalisasi, yang berfungsi untuk mengelompokkan *class* ke dalam hirarki turunan.



Gambar 2.12 - Contoh generalisasi

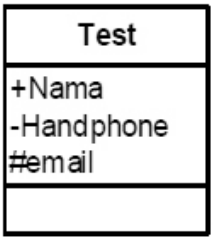
6. Agregasi, merupakan bentuk khusus dari asosiasi yang mempresentasikan hubungan *part-whole* dari hubungan ini sering disebut dengan *assembly* atau agregat. *Class* yang satu dapat dikatakan merupakan bagian dari *class* yang lain yang ikut membentuk *class* tersebut.

Gambar 2.13 - Hubungan agregasi di antara dua *class*

7. Komposisi, merupakan jenis agregasi yang lebih kuat diantara dua *class* yang memiliki asosiasi dimana jika semua ditiadakan, maka bagiannya juga ikut ditiadakan. Berbeda dengan agregasi, bagian akan tetap bisa berdiri sendiri meskipun bagian semuanya ditiadakan.

Gambar 2.14 - Hubungan komposisi di antara dua *class*

8. Penggunaan operator (+) pada diagram *class* diartikan *public*, operator (-) diartikan *private* dan operator (#) diartikan *protected*.



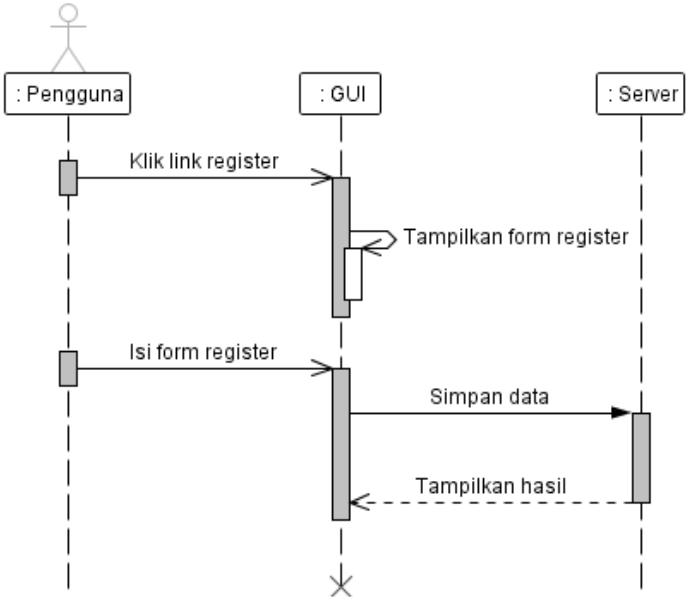
Gambar 2.15 - Penggunaan operator dalam *class*

2.7.2.3 Diagram *Sequence*

Diagram *sequence* menggambarkan serangkaian pesan yang mengalami pertukaran yang dilakukan oleh objek (dapat juga berupa aktor) yang melakukan tugas tertentu. Diagram *sequence* antara lain tersusun atas sekumpulan objek-objek yang akan dipergunakan untuk interaksi, *activation box* yang menyatakan daur hidup (*lifeline*) dari suatu tugas yang sedang dilakukan. Pesan yang dikirimkan dimulai dari suatu aktor atau objek, dan kita dapat memberikan label pada pesan yang dilakukan. (Lethbridge dan Laganiere, 2002, p270-271)

Diagram *sequence* menggambarkan hubungan antarobjek sesuai dengan urutan kejadiannya. Diagram *sequence* memudahkan pengembang untuk melihat kapan suatu objek mulai dan berakhir.

Dalam hal penggunaannya, diagram *sequence* digunakan untuk memperlihatkan dengan jelas urutan kejadian suatu proses karena didalamnya terlihat interaksi dari beberapa objek. (Martin Fowler, 2002, p75).



Gambar 2.16 Contoh diagram *sequence*

2.8 Bahasa Pemrograman *Web*

2.8.1 PHP

Menurut Andi Sunyoto (2007, p119), PHP adalah bahasa *server-side programming* yang menyatu dengan HTML untuk membuat halaman situs dinamis. Maksud dari *server-side programming* adalah sintaks dan perintah-perintah yang diberikan akan sepenuhnya dijalankan di server, tetapi disertakan pada dokumen HTML. Pengembangan PHP telah dipengaruhi sejumlah bahasa

pemrograman lain seperti PERL, C, Java, dan bahkan beberapa cakupan ASP (*Active Server Pages*).

PHP mempunyai beberapa kelebihan, antara lain:

- Mudah dibuat dan dijalankan.
- Mampu bekerja pada *web server* dengan sistem operasi yang berbeda-beda, contohnya, dapat bekerja dalam beberapa sistem operasi Linux.
- PHP bisa didapat secara gratis (*open source*).
- Dapat di-*embed*, yaitu skrip PHP dapat langsung diletakkan dalam tag HTML.

Perbedaan karakteristik *client-side scripting* dan *server-side programming*, yaitu:

- *Client-side scripting*
  a. Kode program diunduh bersama dengan halaman *web*.
  b. Bersifat *interpreter* dan diterjemahkan oleh *web browser*.
  c. Model eksekusinya mudah, dan skrip dapat dijadikan satu dengan HTML.
- *Server-side programming*
  a. Ada klien yang meminta *request*.
  b. Eksekusi program dilakukan di server.
  c. Mengirimkan hasil ke klien.

2.8.2 JavaScript

JavaScript merupakan modifikasi dari bahasa C++ dengan pola penulisan yang lebih sederhana. JavaScript terintegrasi dalam halaman HTML. JavaScript sendiri bukan merupakan bahasa pemrograman, melainkan suatu bahasa skrip (Bollinger, Gary, Natharajan, Bharathi, 2001, p32).

Perbedaan bahasa pemrograman dengan bahasa skrip adalah sebagai berikut:

Bahasa pemrograman:

- Lebih kompleks.
- Cocok untuk menyusun aplikasi yang berukuran besar.
- Memerlukan *compiler* untuk dijalankan.

Bahasa skrip:

- Tidak kompleks dan lebih sederhana.
- Cocok untuk menyusun aplikasi yang berukuran kecil.
- Tidak memerlukan *compiler* untuk dijalankan.

JavaScript adalah bahasa *interpreter*, yang tidak membutuhkan alat tambahan pendukung seperti *compiler* dan *debugger*. Untuk membuat skrip dan men-*debug* skrip tersebut yang diperlukan hanyalah sebuah *text editor* dan sebuah *web browser* yang mendukung JavaScript. Sebuah *file* HTML dipanggil ke dalam *web browser*, JavaScript tersebut diinterpretasikan dan dieksekusi.

2.8.3 *Asynchronous JavaScript and XML (AJAX)*

Asynchronous JavaScript and XML, atau disingkat AJAX, adalah suatu teknik permrograman berbasis *web* untuk menciptakan aplikasi *web* interaktif.

Tujuannya adalah untuk memindahkan sebagian besar interaksi pada komputer pengguna, melakukan pertukaran data dengan server di belakang layar, sehingga halaman situs tidak harus dibaca ulang secara keseluruhan setiap kali seorang pengguna melakukan perubahan. Hal ini akan meningkatkan interaktivitas, kecepatan, dan *usability*. AJAX merupakan kombinasi dari:

- DOM yang diakses dengan *client side scripting language*, seperti VBScript dan implementasi ECMAScript seperti JavaScript dan Jscript, untuk menampilkan secara dinamis dan berinteraksi dengan informasi yang ditampilkan.
- Objek XMLHTTP dari Microsoft atau XMLHttpRequest yang lebih umum diimplementasikan pada beberapa *web browser*. Objek ini berguna sebagai kendaraan pertukaran data asinkron dengan *web server*. Pada beberapa *framework* AJAX, element HTML *iframe* lebih dipilih daripada XMLHTTP atau XMLHttpRequest untuk melakukan pertukaran data dengan *web server*.
- XML umumnya digunakan sebagai dokumen transfer, walaupun format lain juga memungkinkan, seperti HTML, *plain text*. XML dianjurkan dalam pemakaian teknik AJAX karena kemudahan akses penanganannya dengan memakai DOM.
- JSON dapat menjadi pilihan alternatif sebagai dokumen transfer, mengingat JSON adalah JavaScript itu sendiri sehingga penanganannya lebih mudah.

2.9 Konsep Pengembangan Situs

Saat ini ada beberapa konsep pengembangan situs yang sering digunakan oleh para pengembang aplikasi berbasis *web*, tetapi saat ini yang paling banyak digunakan adalah konsep *web 2.0* karena konsep tersebut lebih mengedepankan interaksi antarpengguna dan aplikasi sehingga dapat tercipta suatu komunitas pada situs.

Pada penelitian ini, penulis menggabungkan konsep *web 2.0* dengan konsep *mashup*. Konsep *mashup* adalah penggabungan beberapa layanan *web service* atau biasa disebut sebagai *Application Programming Interface* (API) menjadi sebuah layanan baru yang inovatif.

2.9.1 *Web 2.0*

Web 2.0, adalah sebuah istilah yang dicetuskan pertama kali oleh O'Reilly Media pada tahun 2003, dan dipopulerkan pada konferensi *web 2.0* pertama di tahun 2004, merujuk pada generasi yang dirasakan sebagai generasi kedua layanan berbasis web—seperti situs jaringan sosial, wiki, perangkat komunikasi, dan folksonomi—yang menekankan pada kolaborasi online dan berbagi antar pengguna. O'Reilly Media, dengan kolaborasinya bersama MediaLive International, menggunakan istilah ini sebagai judul untuk sejumlah seri konferensi, dan sejak 2004 beberapa pengembang dan pemasar telah mengadopsi ungkapan ini.

Walaupun kelihatannya istilah ini menunjukkan versi baru dari *web*, istilah ini tidak mengacu kepada pembaruan kepada spesifikasi teknis *world wide web*, tetapi lebih kepada bagaimana cara pengembang sistem di dalam menggunakan *web platform*. Mengacu pada Tim O'Reilly, definisi *web 2.0*

adalah sebuah revolusi bisnis di dalam industri komputer yang terjadi akibat pergerakan ke Internet sebagai *platform*, dan suatu usaha untuk mengerti aturan-aturan agar sukses di *platform* tersebut.

Prinsip-prinsip *web 2.0*, yaitu:

- *Web* sebagai *platform*.
- Data sebagai pengendali utama.
- Efek jaringan diciptakan oleh arsitektur partisipasi.
- Inovasi dalam perakitan sistem serta situs disusun dengan menyatukan fitur dari pengembang yang terdistribusi dan independen (semacam model pengembangan "*open source*").
- Model bisnis yang ringan, yang dikembangkan dengan gabungan isi dan layanan.
- Akhir dari siklus peluncuran (*release cycle*) perangkat.
- Mudah untuk digunakan dan diadopsi oleh user.

2.9.2 *Mashup*

Mashup adalah sebuah konsep aplikasi situs yang didalamnya mengintegrasikan *Application Programming Interface* (API) dan sumber data (*data source*) dari layanan lain untuk kemudian diproses menjadi sebuah layanan baru. *Mashup* mengkombinasikan teknologi *web* untuk menciptakan sebuah produk baru yang berorientasi pada konten dan layanan.

Mashup dibagi menjadi 3 *layer*:

- *Presentation*, merupakan *interface* yang digunakan oleh *mashup*, teknologi yang digunakan HTML/HTML5, CSS/CSS3, JavaScript, jQuery dan AJAX.
- *Web service*, merupakan proses utama dari *mashup* yang bertugas untuk mengakses layanan API yang akan digunakan, teknologi yang digunakan XMLHTTPRequest, XML-RPC, JSON-RPC, SOAP, REST.
- *Data*, bertugas menangani operasi transaksi data yang terjadi (*sending*, *storing*, *receiving*), teknologi yang digunakan XML, JSON, KML.

Kategori *mashup*:

- *Data mashup*, secara umum mengabungkan API dan *data source* yang memiliki media dan informasi yang sama. Kombinasi yang dihasilkan bertujuan menghasilkan layanan baru yang lebih komplit.
- *Consumer mashup*, secara umum menggabungkan API dan *data source* yang memiliki tipe data yang berbeda, biasanya yang digabungkan layanan visual dan layanan data dari berbagai sumber.
- *Business mashup*, secara umum merupakan penggabungan API dan *data source* umum dengan sumber yang dimiliki pribadi, fokusnya mendukung kepentingan bisnis yang memperoleh informasi dari luar dan dalam.

Terdapat beberapa API populer yang sering digunakan dalam mengembangkan situs yang memakai konsep *mashup*, antara lain Google Maps, YouTube, Twitter, Amazon eCommerce, Facebook, Google Search, Last.fm, Yahoo! Search, Yahoo! Weather, dan yang lainnya. Untuk sementara API dari

layanan Twitter, YouTube, Facebook, Google Maps, Flickr, dan LinkedIn merupakan API populer yang digunakan dalam mengembangkan *mashup*.

2.9.3 *Application Programming Interface (API)*

Application Programming Interface (API) adalah sekumpulan perintah, fungsi, *class*, dan protokol yang memungkinkan suatu perangkat lunak berinteraksi dengan perangkat lunak yang lain. Perangkat lunak yang mengimplementasikan API tersebut dapat mengunakan fitur-fitur yang disediakan oleh perangkat lunak penyedia API.

Ketika digunakan dalam konteks dari pengembangan situs, sebuah API bisa diartikan sebagai kumpulan dari pesan *request Hypertext Transfer Protocol* (HTTP), berhubungan dengan definisi dari struktur pesan respon, yang biasanya dalam format *Extensible Markup Language* (XML) atau *JavaScript Object Notation* (JSON). Semenjak "*web API*" menjadi persamaan kata dari *web service*, tren yang ada saat ini (disebut sebagai *web 2.0*) telah berubah cara berkomunikasinya, dari bentuk servis *Simple Object Access Protocol* (SOAP) menjadi bentuk *Representational State Transfer* (REST). *Web API* memungkinkan kombinasi dari beberapa servis menjadi sebuah aplikasi yang dikenal dengan sebutan *mashup*.

Dengan adanya API, memungkinkan komunitas situs untuk membuat sebuah arsitektur terbuka untuk berbagi konten dan data antara komunitas dan aplikasi. Dengan ini, konten yang dibuat di satu tempat bisa ditempatkan secara dinamis dan di-*update* di banyak lokasi pada situs. Contohnya:

1. Foto dapat dibagi dari situs seperti Flickr (*flickr.com*) dan Photobucket (*photobucket.com*) ke situs jejaring sosial seperti Facebook (*facebook.com*) dan Myspace (*myspace.com*).
2. Konten dapat disisipkan, contohnya sebuah presentasi dari Slideshare (*slideshare.com*) ke Linkedln (*linkedin.com*).
3. Konten dapat ditulis secara dinamis. Contohnya berbagi komentar secara langsung yang dibuat di Twitter (*twitter.com*) dengan akun Facebook (*facebook.com*) yang dimungkinkan dengan API mereka.
4. Konten video dapat disisipkan di situs yang dimiliki oleh orang lain.
5. Informasi pengguna dapat dibagi dari komunitas situs ke aplikasi luar, membawa fungionalitas baru ke komunitas situs untuk berbagi data mereka melalui sebuah API. Contoh terbaik dari itu adalah Facebook *Application Platform*.

2.9.3.1. *Facebook Application Platform* (API Facebook)

Pada tahun 2007, Facebook mengeluarkan *platform* mereka sendiri untuk pengembangan aplikasi. *Platform* tersebut terdiri dari:

1. *Facebook Markup Language* (FBML).

   FBML adalah sebuah *Application Programming Interface* (API) untuk memanggil *Representational State Transfer* (REST) terhadap Facebook yang memiliki format seperti bahasa *markup* berbasis HTML.

2. *Facebook Query Language* (FQL).

   FQL mirip dengan *Structured Query Language* (SQL) yang berinteraksi dengan Facebook.

3. *Facebook JavaScript.*

   Bahasa skrip untuk memperkaya pengalaman pengguna dan sebuah kumpulan dari *library* pemrograman klien.

Umumnya, alat bantu untuk membuat *platform* Facebook bisa disebut sebagai Facebook API. Dengan meluncurkan *platform* ini, Facebook membuat aplikasi yang memungkinkan pengembang aplikasi lain untuk membuat aplikasi eksternal yang memudahkan pengguna Facebook untuk berinteraksi dengan orang lain dengan cara yang baru dan berbeda (cara yang dikembangkan oleh pengembang aplikasi lain). Pengembang tidak terbatas hanya dapat membuat aplikasi situs, tetapi Facebook juga membuka *platform*-nya kepada aplikasi *desktop* yang terhubung dengan Internet dengan *library* Java klien.

2.9.3.2. API Yahoo! Weather

Yahoo! Weather adalah layanan dari salah satu situs pencarian terbesar di dunia yaitu *yahoo.com* yang memberikan informasi kepada pengguna Yahoo! tentang prakiraan cuaca yang ada di seluruh dunia. Pihak Yahoo! membuka akses API Yahoo! Weather kepada setiap pengembang aplikasi eksternal. API yang dibagi adalah dalam bentuk *Really Simple Syndication* (RSS 2.0) *feed* yang berformat *Extensible Markup Language* (XML). RSS *feed* tersebut memungkinkan pengunjung untuk mendapatkan

prakiraan cuaca yang *up-to-date* atas lokasi mereka. Ramalan cuaca yang dapat ditampilkan adalah hari ini dan dua hari kedepan karena pihak Yahoo! hanya memberikan akses maksimal dua hari setelah hari ini.

2.9.3.3. API Twitter

Twitter adalah layanan *micro-blogging* yang memungkinkan penggunanya untuk mempublikasikan hal-hal yang mereka alami melalui Internel hanya dalam 140 karakter. Twitter saat ini telah menjadi salah satu situs penyaing Facebook, karena telah memiliki pengguna yang sangat banyak. Hal ini dapat terjadi karena sejak awal Twitter diluncurkan, mereka sudah mengeluarkan API mereka untuk dipakai oleh setiap pengembang situs yang tertarik untuk mengaplikasikan Twitter pada situs yang mereka kembangkan.

Pada awalnya Twitter mengeluarkan API dalam bentuk oAuth yang akan digunakan untuk melakukan validasi apakah parameter *username* dan *password* yang dikirimkan melalui situs yang berbeda adalah benar atau salah. Jika benar, maka API Twitter akan memberikan otentifikasi bahwa pengguna tersebut dapat menggunakan layanan Twitter walaupun dari situs yang berbeda. Kekurangan dari oAuth ini adalah tidak semua pengembang dapat melalukan hal tersebut karena hal ini terbilang cukup rumit.

Sekarang Twitter menggunakan API dalam bentuk JavaScript yang sangat memudahkan para pengembang situs untuk memakai API Twitter pada situs yang mereka kembangkan. Twitter juga telah mengeluarkan beberapa *plugin*, seperti "*Tweet button*" yang dapat digunakan untuk

berbagi berita dari situs lain ke Twitter melalui akun Twitter sang pengguna.

2.9.3.4. API Google Maps

Saat ini, solusi pemetaan adalah suatu hal yang menjadi unsur alami dalam situs. Banyak orang menggunakan peta untuk melihat lokasi suatu tempat, mencari posisi alamat, mendapatkan petunjuk arah saat mengemudi, dan untuk melakukan hal-hal lainnya. Sebagian besar informasi memiliki lokasi, dan jika informasi tersebut memiliki lokasi, maka dapat ditampilkan pada peta.

Ada beberapa solusi pemetaan termasuk Yahoo! Maps dan Bing Maps, namun yang paling populer adalah Google Maps. Bahkan, menurut *programmableweb.com*, Google Maps merupakan API yang paling populer di Internet. Menurut statistik, situs pada Mei 2010, 43% dari semua *mashup* menggunakan API Google Maps (*www.programmableweb.com/apis*). Sebagai perbandingan, API kedua yang paling populer adalah Flickr dengan persentase 11%, dan API pemetaan kedua yang paling populer adalah VirtualEarth (Bing Maps) dengan persentase sebesar 3%.

Solusi pemetaan adalah salah satu unsur penting dalam banyak *mashup*. API Google Maps memungkinkan *programmer* memanfaatkan kekuatan Google Maps untuk digunakan dalam aplikasinya, untuk menampilkan datanya (atau data orang lain) dengan cara yang efisien dan bermanfaat.

2.10 Desain *Web*

2.10.1 HTML5

HTML5 merupakan standar baru untuk HTML, XHTML, dan DOM HTML. Sejak munculnya HTML versi 4.01, perkembangan situs dunia semakin berkembang. Saat ini HTML5 masih dalam pengembangan, namun hanya beberapa browser sudah mendukung HTML5. (Ianpanrita, 2010)

HTML5 merupakan hasil proyek dari W3C (*World Wide Web Consortium*) dan WHATWG (*Web Hypertext Application Technology Working Group*). WHATWG bekerja dengan bentuk situs dan aplikasi, sedangkan W3C merupakan pengembang dari XHTML 2.0 pada tahun 2006, kemudian mereka memutuskan untuk bekerja sama dan membentuk versi baru dari HTML.

Tujuan dibuatnya HTML5 antara lain:

a. Fitur baru harus didasarkan pada HTML, CSS, DOM, dan JavaScript.
b. Mengurangi kebutuhan untuk *plugin* eksternal (seperti Flash).
c. Penanganan kesalahan yang lebih baik.
d. Lebih banyak *markup* untuk menggantikan *scripting.*
e. HTML5 merupakan perangkat mandiri.

Fitur baru dalam HTML5:

a. Unsur kanvas untuk menggambar.
b. Video dan elemen audio untuk media pemutaran.
c. Dukungan yang lebih baik untuk penyimpanan secara *offline*.

d. Elemen konten yang lebih spesifik, seperti artikel, *footer, header, navigation, section.*

e. Bentuk kontrol form seperti kalender, tanggal, waktu, *e-mail,* URL, *search*.

Beberapa *browser* sudah mendukung HTML5 seperti Safari, Chrome, Firefox, dan Opera. Kabarnya IE9 (Internet Explorer) akan mendukung beberapa fitur dari HTML5.

Beberapa kelebihan yang dijanjikan pada HTML5:

a. Dapat ditulis dalam sintaks HTML (dengan tipe *media text*/HTML) dan XML.

b. Integrasi yang lebih baik dengan aplikasi situs dan pemrosesannya.

c. Integrasi ('*inline*') dengan *doctype* yang lebih sederhana.

d. Penulisan kode yang lebih efisien.

e. Konten yang ada di situs lebih mudah terindeks oleh *search engine*.

2.10.2 CSS 3 (*Cascading Style Sheet* 3)

CSS merupakan bahasa yang paling mudah dimengerti, dipelajari, dan tidak mempunyai tanda atau karakter khusus untuk mendeklarasikan di halaman situs.

Beberapa standar baru untuk CSS3 menggantikan CSS2 dan mungkin akan membuat bereksplorasi lebih dalam lagi untuk membuat tampilan situs lebih menarik dan mulai meninggalkan situs yang membuat menunggu dengan *loading* yang lama atau *browser* yang tidak cocok untuk situs yang dikunjungi.

Beberapa kelebihan yang ada pada CSS3:

a. CSS3 bisa lebih detail untuk mendeklarasikan objek yang akan diberikan *style*. Contohnya didalam objek '*blockquote*' bisa membuat (*quote*) di awal dan akhirnya menggunakan *font* lebih besar dari konten *blockquote* itu sendiri.

b. CSS3 kaya akan fitur untuk animasi dan efek untuk text atau objek, yang sebelumnya tidak bisa dilakukan oleh CSS2/CSS2.1, dan bisa menggantikan peran gambar.

c. Standar *web 2.0* atau situs interaktif dan efisien berdasar dari penggunaan CSS. Dengan CSS3 situs akan bisa lebih berkembang dan bisa lebih interaktif lagi dengan pengunjung.

d. Bisa mengurangi ukuran *file* yang di-*load* dan lebih ringan, secara otomatis mengurangi *bandwidth inbound/outbound* situs.

CSS3 memiliki fasilitas untuk *shadow* dari suatu *div layout,* fitur transparansi, gradien warna pada *border*, warna pada teks yang diseleksi, fitur skala memperkecil atau memperbesar *layout*, kolom pada teks, dan fitur gradien pada *background*.

2.10.3 JQuery

JQuery adalah *library* JavaScript yang ringkas, cepat dan menyederhanakan dokumen HTML, animasi, penanganan *event* dan interaksi AJAX. JQuery juga disebut *framework* sekaligus kumpulan skrip yang berguna untuk membuat situs menjadi lebih interaktif, terlihat *powerful,* dan memiliki

animasi yang bagus. JQuery ditulis dengan menggunakan JavaScript sebagai sebuah file tunggal. Dibuat oleh John Resig pada awal tahun 2006,

Saat ini jQuery merupakan salah satu *library* JavaScript yang banyak dipakai, bahkan oleh Microsoft didukung penggunaannya. Kemampuan jQuery untuk membuat instruksi JavaScript yang singkat dan jelas merupakan salah satu keunggulannya. JQuery ini memiliki keunggulan *pluginable*, artinya jQuery bisa ditambahi dengan berbagai *plugin*.

2.11 Teknik Penyerangan *Web*

2.11.1 *SQL Injection*

SQL injection adalah teknik penyisipan kode yang menyerang kelemahan celah keamanan yang terjadi dalam lapisan basis data dari sebuah aplikasi. Celah ini terjadi ketika parameter yang dimasukkan oleh pengguna tidak disaring secara benar sebelum dieksekusi menjadi sebuah perintah SQL, sehingga muncul pesan kesalahan pada aplikasi. Celah ini adalah contoh dari sebuah kategori celah keamanan yang dapat terjadi setiap kali sebuah bahasa pemrograman atau skrip digabungkan di dalam bahasa yang lain. Serangan *SQL injection* juga dikenal dengan sebutan serangan *SQL insertion*.

2.11.2 *Cross Site Scripting (*XSS*)*

Bagi beberapa orang, untuk mengerti mengenai XSS agak sulit. Pengetahuan mengenai HTML dan bahasa pemrograman *web* diperlukan agar seseorang dapat mengerti XSS ini karena memang XSS memanfaatkan HTML serta kecerobohan *programmer* dalam pengolahan HTML melalui bahasa

pemrograman *web* yang digunakan. Oleh sebab itu, XSS seringkali juga disebut *HTML Injection*.

Beberapa *programmer web* pemula umumnya tidak terlalu memperhatikan bahaya XSS sehingga saat ini banyak celah XSS yang bisa ditemukan pada situs yang dirancang. XSS pun cenderung tampak "kalah pamor" dibanding dengan *SQL injection* sehingga tidak banyak orang yang memperhatikan hal ini.

XSS merupakan salah satu jenis serangan injeksi kode (*code injection attack*). XSS dilakukan oleh penyerang dengan cara memasukkan kode HTML atau *client script code* lainnya ke suatu situs. Serangan ini akan seolah-olah datang dari situs tersebut. Akibat serangan ini antara lain penyerang dapat mem-*bypass* keamanan di sisi klien, mendapatkan informasi sensitif, atau menyimpan aplikasi berbahaya.