BAB II
LANDASAN TEORI

II.1 Sistem Informasi

II.1.1 Konsep Dasar Sistem

Suatu sistem memiliki maksud tertentu, dimana ada dua pendapat yang menyebutkan maksud dari suatu sistem yaitu untuk mencapai suatu tujuan (*goal*) dan untuk mencapai suatu sasaran (*objective*). *Goal* biasanya dihubungkan dengan ruang lingkup yang lebih luas dan *objective* dalam ruang lingkup yang lebih sempit.

Menurut Jogiyanto (1999), ada dua kelompok pendekatan dalam mendefinisikan sistem, yaitu:

Pendekatan Sistem yang menekankan pada prosedur

"Suatu sistem adalah jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu."

Pendekatan sistem yang lebih menekankan pada komponennya

"Sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan tertentu."

II.1.2 Karakteristik Sistem

Sistem memiliki karakteristik atau sifat-sifat tertentu yang memiliki bagian atau komponen, interaksi antar bagian, tujuan batasan sistem dan lingkungan sistem yang saling berinteraksi membentuk satu kesatuan. Jika suatu sistem tidak memiliki tujuan, maka beroperasinya sistem tidak akan berguna. Tujuan dari

sistem sangat menentukan untuk masukan yang dibutuhkan sistem dan keluaran yang akan dihasilkan sistem.

Menurut Paulus Mudjiharto (1998), setiap sistem tidak selalu mempunyai kombinasi elemen yang sama, tetapi secara minimal sistem akan mempunyai :

1) *Input* (Masukan)

   Semua bahan atau sumber daya yang dimasukkan ke dalam sistem untuk keperluan proses sistem.

2) *Process* (Proses)

   Kegiatan yang mentransformasikan *input* ke *output*.

3) Model

   Model adalah bentukan yang dipakai proses dalam mentransformasikan *input* ke *output*.

4) *Output* (Keluaran)

   Semua bahan atau sumber daya yag dihasilkan dari dalam sistem untuk pengambilan keputusan.

5) Elemen *Feedback*

   Khusus untuk sistem *closed loop* terdapat elemen *feedback*, yaitu *output* yang ditangkap kembali oleh pengambil keputusan sebagai bahan pertimbangan mengambil keputusan.

6) Elemen *control* / Pengambil Keputusan

   Elemen *control* adalah elemen yang menerima *feedback* dan mengubah komponen *input* dan proses.

7) Elemen Lingkungan

   Semua elemen yang berada diluar sistem dan bukan bagian dari sistem tetapi tetap signifikan terhadap sistem.

8) Batasan Sistem

   Sistem dibatasi oleh lingkungan.

II.1.3 Pengertian Informasi

Menurut Jogiyanto (1999), informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya. Pada dasarnya informasi merupakan data yang telah diolah sedemikian rupa sehingga memberikan bentuk yang berarti bagi pemakai didalam membuat atau mengambil suatu keputusan.

II.1.4 Siklus Informasi

Data mentah merupakan bentuk yang paling sederhana yang belum bisa menjadi suatu informasi. Oleh karena itu, data ini harus diolah menjadi sebuah model kemudian baru didapatkan sebuah informasi. Untuk membentuk informasi ini diperlukan beberapa langkah yang digambarkan sebagai siklus. Didalam siklus ini data diolah dengan menggunakan model sehingga menjadi informasi. Dengan informasi tersebut kemudian diambil keputusan atau tindakan-tindakan yang kemudian diolah lagi melalui suatu model yang disebut juga dengan siklus proses data.

II.1.5 Kualitas Informasi

Kualitas dari suatu informasi tergantung dari tiga hal, yaitu :

1) Akurat (*Accurate*)

   Informasi tersebut harus bebas dari kesalahan-kesalahan dan tidak bias atau menyesatkan.

   Informasi harus secara jelas karena ketidak akuratan informasi akan mengakibatkan keputusan yang tidak tepat.

2) Tepat pada waktunya (*Timeliness*)

Informasi yang akan datang pada penerima tidak boleh terlambat. Karena informasi yang telah usang tidak akan mempunyai nilai, terlebih dalam pengambilan keputusan akan mengakibatkan fatal.

3) Relevan (*Relevance*)

Setiap orang mengambil suatu tindakan atau keputusan memerlukan informasi yang berbeda-beda sehingga informasi dikatakan relevan jika informasi tersebut diberikan kepada orang-orang yang betul-betul membutuhkan dan mempunyai manfaat bagi pemakainya.

II.2 Konsep Sistem Informasi Berbasis *Web*

Sistem informasi adalah serangkaian komponen yang terkait satu sama lain dan bekerja bersama-sama untuk mengumpulkan, mengolah dan menyimpan serta menyebarluaskan informasi guna mendukung pengambilan keputusan, koordinasi, kontrol, analisis dan visualisasi dalam suatu organisasi.

Sistem informasi berbasis *web* adalah sistem informasi yang sifatnya *on-line*, dimana informasi yang diberikan tidak hanya dapat dibaca atau statis melainkan juga dapat memberikan atau menerima tanggapan dari pengguna sistem. Sistem informasi berbasis *web* ini merupakan pengembangan dari sistem informasi yang bersifat konvensional.

II.2.1 *Web Server*

Web server adalah suatu perangkat lunak yang mengatur halaman *web* dan membuat halaman-halaman *web* tersebut dapat diakses di *client*, yaitu melalui

jaringan lokal atau melalui jaringan internet. Ada banyak *web server* yang tersedia diantaranya *Apache, IIS (Internet Information Service),* dan *IPlanet's Enterprise server.*

II.2.2 *Web Browser*

Web Browser digunakan untuk menjelajah situs *web* lewat layanan *HTTP.* Untuk mengakses layanan *www (World Wide Web)* dari sebuah komputer digunakan program *web client* yang disebut *web browser* atau *browser* saja. Jenis-jenis *browser* yang biasa digunakan adalah *Netscape, Internet Explorer, NCSA Mosaic, Arena,* dan banyak lainnya.

II.2.3 Web Statis

Web statis merupakan suatu halaman yang berisi skrip HTML editor dan disimpan sebagai *file .htm* atau *.html.* Disebut statis karena halaman tersebut dari waktu kewaktu isinya tidak berubah. Halaman web statis ini tidak memerlukan pemrosesan diserver sehingga pembuatannya dapat dilakukan menggunakan *editor HTML* dan hasilnya dapat dilihat pada *web browser.*

II.2.4 Web Dinamis

Pembuatan halaman *web* dinamis dapat dilakukan dengan dua cara yaitu secara *client side* atau secara *server side.* Penggunaan *client side* dan *server side* tidak saling bertentangan melainkan saling melengkapi. Seorang *web developer* harus dapat menentukan bagian mana yang diletakkan secara *client side* dan mana yang diletakkan secara *server side.*

II.3 Pemrograman *Web* (*Web Programming*)

Pemrograman *web* identik dengan pembuatan sebuah *homepage*. *Homepage* adalah halaman yang kita lihat pertama kali ketika kita membuka suatu situs tertentu sehingga situs dapat diartikan sebagai kumpulan halaman-halaman *web* di internet yang berisi informasi.

Tampilan situs-situs yang ada di internet sekarang lebih dinamis. Ciri-ciri situs yang dinamis adalah dapat berinteraksi dengan pengunjung situs, dapat menampilkan informasi-informasi dari *database* dan halaman-halaman *web* dapat berubah secara otomatis. Pemrograman *web* dapat dikategorikan menjadi dua, yaitu *server-side programming* dan *client-side programming*.

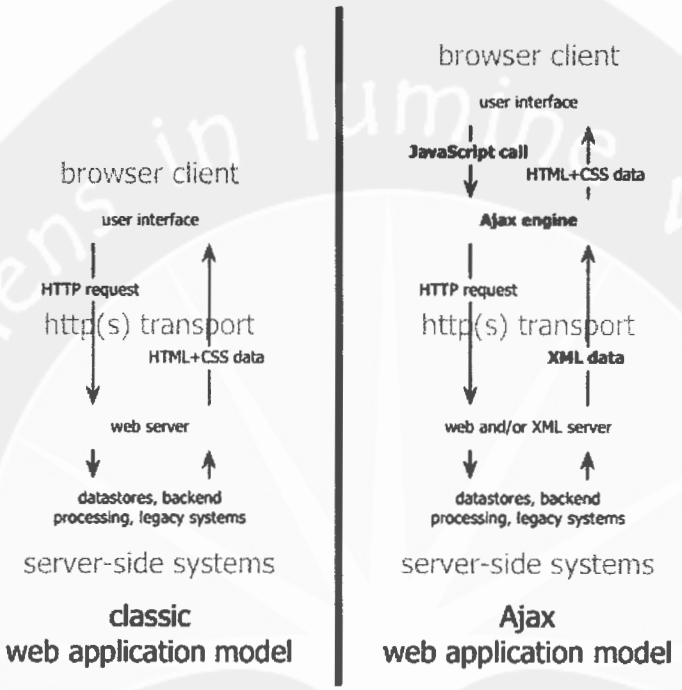
Pada *server-side programming* perintah-perintah program dijalankan di *web server*, sedangkan *client-side programming* menjalankan perintah-perintah program pada *web browser*. Aplikasi *web* berjalan pada protokol HTTP dan semua protokol yang ada di internet selalu melibatkan *server* dan *client*.

II.4 AJAX (*Asynchronous JavaScript And XML*)

II.4.1 Sejarah AJAX

AJAX bukanlah suatu hal yang baru. Sebenarnya AJAX sudah ada sejak *JavaScript* itu sendiri muncul. *Asynchronous JavaScript And XML* adalah suatu konsep pemrograman, dan bukan merupakan teknologi. Istilah AJAX pertama kali dikemukakan oleh Jesse James Garrett dalam sebuah artikel yang berjudul *AJAX: A New Approach to Web Applications (18 Februari 2005)*. Dalam artikel tersebut James Garrett mempertanyakan apa yang bisa

dilakukan oleh *user* pada saat *server* memproses permintaan dari *user*. Dalam artikel yang ditulisnya juga diberikan gambaran perbandingan antara *web* yang dikembangkan secara konvensional serta *web* yang dikembangkan dengan menggunakan AJAX.



Gambar 1. Perbandingan web konvensional dan AJAX

Dari skema di atas muncul suatu lapisan baru yang benama *AJAX Engine*. Lapisan inilah yang menjawab pertanyaan yang dikemukakan oleh James Garrett.

II.4.2 Pro dan Kontra

Perkembangan dunia perangkat lunak selalu menimbulkan pro dan kontra ketika perkembangan aplikasi untuk desktop selalu terbentur dengan masalah kesesuaian platform yang seringkali mempunyai aturan-aturan yang berbeda dalam penanganan kode (*handling code)*, sedangkan aplikasi internet jauh lebih sederhana untuk dapat diakses antar browser. Dikombinasikan

dengan fakta bahwa browser yang user base luas sangat sedikit maka hal ini berarti pemakaian aplikasi internet jauh lebih stabil pada penggunaan user yang berbeda.

Terdapat pula keuntungan dalam menciptakan dan me-*maintain* aplikasi online. Ketika menciptakan sebuah aplikasi desktop dan kemudian mengeluarkan *patch* untuk membenahi *bug* pada aplikasi tersebut, *user* harus menginstal kembali seluruh paket perangkat lunak atau mencari faslitas untuk mengekases *patch* tersebut dan menginstalnya. Lebih jauh lagi terdapat kesulitan untuk menentukan bagian mana yang terpengaruhi. Berbeda dengan aplikasi *web* yang letaknya dalam sebuah server, perubahan atau perbaikan dilakukan hanya pada server tersebut dan segara dapat menimbulkan efek pada sisi user.

Ketika mengirimkan aplikasi dari server sangatlah baik dari segi maintenance, tetapi masalah lain segera muncul ketika pengguna (*client*) membutuhkan piranti untuk mengaksesnya. Internet dapat menjawab masalah tersebut tetapi pertanyaan kembali muncul mengenai kecepatan aksesnya. Sebagai contoh ketika seseorang menggunakan Microsoft word, dengan mudahnya pengguna meng"klik" tombol dikomputer dan dengan segera menghasilkan respon. Berbeda dengan aplikasi web, dimana dibutuhkan koneksi untuk menjalankan aplikasi tersebut. Oleh karena itu, walaupun server dapat memproses aplikasi tersebut dengan cepat tetapi masih dibutuhkan waktu untuk mengirim respon kepada user.

Dikaitkan dengan kebutuhan untuk me-*refresh* halaman setiap waktu yang membutuhkan respon dari

server akan menimbulkan masalah bagi pengguna aplikasi internet. Masalah tersebut adalah kebutuhan untuk menerima respon atau hasil eksekusi perintah secepat menggunakan aplikasi desktop. Flash dapat menjawab permasalahan tersebut, tetapi diperlukan kemampuan yang sangat mahir. DHTML juga mampu untuk mengatasi permasalahan tersebut dengan penggunaan JavaScript, tetapi kode untuk melakukan itu juga agak terbatas. Bahkan seringkali ditemukan browser yang menolak untuk berkooperasi dengan seperangkat standar. Untuk menjawab semua semua permasalahan ini muncullah Ajax. *Asynchronous JavaScript And XML* merupakan sarana untuk melakukan request ke server tanpa membutuhkan waktu yang lama dan mengurangi untuk me-load halaman secara keseluruhan.

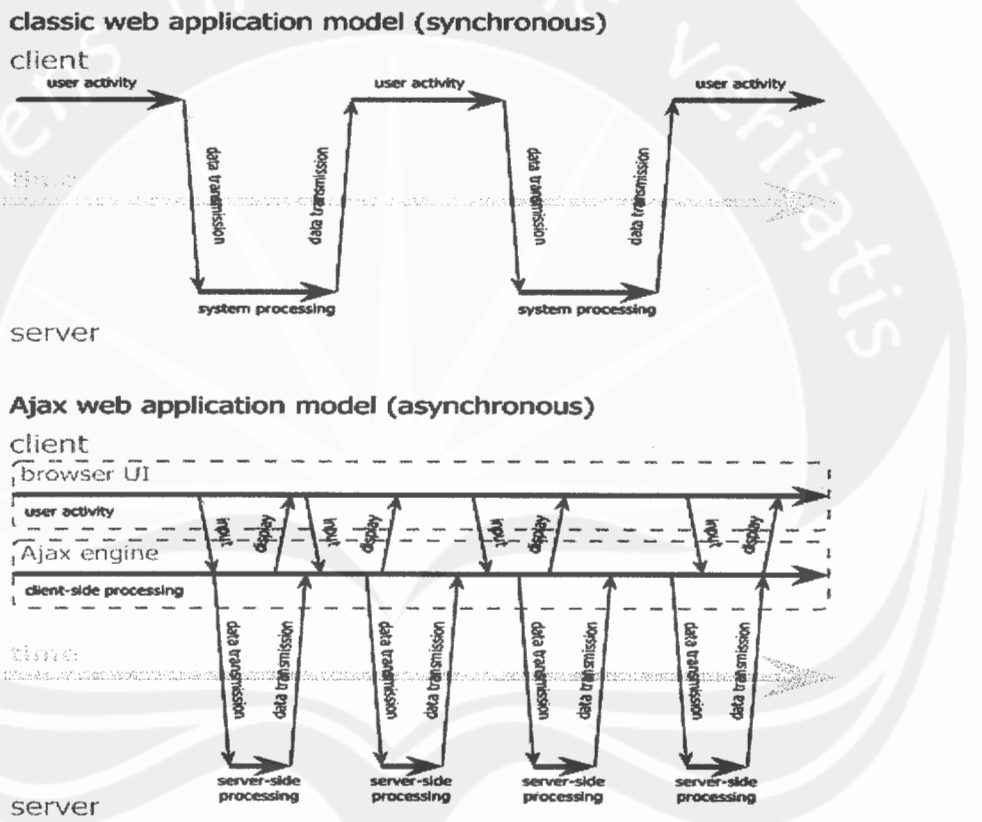
II.4.3 Konsep AJAX

James Garrett memaparkan bahwa *AJAX* memungkinkan *user* untuk melakukan komunikasi secara bebas (*asynchronous*) dengan *server*. Hal ini memberikan keuntungan bahwa *user* tidak harus melihat halaman kosong pada saat menanti respon dari *server*.

Asynchronous JavaScript And XML digunakan supaya sebuah *website* menjadi lebih lebih interaktif. Tujuan utamanya adalah membuat sebuah *website* lebih responsif terhadap aktifitas *user* dan melakukan perubahan presentasi data hanya pada data yang perlu diubah (di-*update* atau di-*refresh*). Jadi, sebuah *website* tidak perlu di-*refresh* secara keseluruhan untuk menampilkan presentasi data terbaru.

update atau di-*refresh*). Jadi, sebuah *website* tidak perlu di-*refresh* secara keseluruhan untuk menampilkan presentasi data terbaru.

Dengan menggunakan *JavaScript* tidak semua *request* dari *user* diteruskan ke *server*. *Request* yang dapat ditangani di sisi *user* diproses oleh *AJAX Engine*. Bila AJAX memerlukan data dari *server*, AJAX mengirimkan *request* ke *server* dengan menggunakan XML.



Gambar 2. Transfer data AJAX dan web konvensional

Berikut ini adalah salah satu kerangka dasar AJAX.

```
function createRequestObject() {
   var ro;
   var browser = navigator.appName;
   if(browser =="Microsoft Internet Explorer"){
   ro = new ActiveXObject("Microsoft.XMLHTTP");
    }else{
        ro = new XMLHttpRequest();
    }
    return ro;
```

```
    http.onreadystatechange = handleResponse;
    http.send(null);
}

function handleResponse() {
    if(http.readyState == 4){
        var response = http.responseText;
        var update = new Array();
        update = response.split('|');
        if((response.indexOf('|' != -1)) && (update[0] ==
        "OK")) {
            document.getElementById("hasil").Style['color'] =
            '#00f';
            document.getElementById("hasil").innerHTML =
            update[1];
        }
        else
        {
         document.getElementById("hasil").style['color'] =
         '#f00';
         document.getElementById("hasil").innerHTML =
         "Respond dari server tidak sesuai";
        }
    }
}
```

Untuk melaksanakan proses *request*, dibutuhkan obyek *XMLHttpRequest*. Pada kode di atas hal tersebut dilakukan pada fungsi *createRequestObject()* yang hasilnya disimpan dalam obyek *http*.

Dalam fungsi tersebut, jika *browser* yang digunakan adalah *Internet Explorer*, maka yang diciptakan adalah obyek *Microsoft.XMLHTTP* dan *XMLHttpRequest* untuk *browser* lainnya.

Fungsi berikutnya, yaitu *sndReq()* berfungsi sebagai pengirim *request* kepada *server*. Pada contoh di atas, *request* yang dilakukan adalah dengan *method* '*get*', dan *action*/datanya diletakkan pada *URL*-nya.

Pada fungsi *sndReq()* ditentukan bahwa yang akan menangani hasil *request* adalah fungsi *handleResponse()* dan setelah melakukan pengiriman (*send*), tugas dari fungsi *sndReq()* telah berakhir.

Fungsi yang akan sangat sibuk dalam AJAX adalah fungsi *handleRequest()*, karena di sinilah segala respon dari *server* ditangani.

Dalam contoh ini, yang ditangani adalah ketika *request* telah seluruhnya diterima (*readyState == 4*). Data yang diambil adalah *responseText*, yaitu seluruh teks hasil *feedback* dari *server*. Jika ingin menangani *feedback* data yang berupa XML, maka data yang diambil adalah *responseXML*.

Untuk kasus di atas, diharapkan respon dari *server* berupa:*"status|Pesan"* sehingga *script* akan langsung mengetahui status *request*, dan merubah isi *div* yang telah disediakan dengan teks respon dari *server*.

II.4.4 Kebutuhan untuk Mengaplikasikan AJAX

Dasar dari Ajax adalah JavaScript maka untuk menjalankan aplikasi-aplikasi yang dibuat dengan menggunakan AJAX, *browser* yang digunakan harus mendukung JavaScript.

Browser-browser yang mendukung Ajax antara lain Firefox (semua versi yang ada), Internet Explorer (mulai versi 4.0 dan versi berikutnya), Apple Safari (mulai versi1.2 dan versi berikutnya), Conqueror, Netscape (mulai versi 7.1 dan versi berikutnya) dan Opera (versi 7.6 dan berikutnya). Oleh kerena itu sebagian besar browser yang ada saat ini dapat digunakan untuk mengaplikasikan Ajax.

Untuk mengaplikasikan dan mengembangkan Ajax secara efisien dibutuhkan suatu kreatifitas dari pengembang (*developer*) sehingga tercipta aplikasi yang fungsional dan revolusioner.

II.4.5 Kelemahan dan Kelebihan AJAX

Masalah utama AJAX yaitu tidak bersahabat dengan *Search Engine* karena AJAX memprioritaskan pada JavaScript dan XML. Hal ini berarti semua fitur AJAX tidak dapat diakses oleh *search engine* atau pengunjung yang men-*disable* JavaScript-nya.

Dengan digunakannya AJAX untuk membangun suatu *website* menyebabkan isi dari *website* tersebut tidak dapat terindeks dan terindeks hanya halaman terdepannya saja sehingga jika dilakukan pencarian dengan menggunakan *search engine*, yang dapat dikenali hanya halaman yang terindeks saja atau halaman terdepan saja. Hal ini menurut pandangan dunia bisnis sangat merugikan karena dapat mengurangi jumlah calon pengunjung yang mencari *website* tersebut dengan menggunakan *search engine*.

Asynchronous JavaScript And XML merupakan alternatif untuk kenyamanan bagi *user*, di mana *browser* tidak akan mengambil seluruh halaman, tapi hanya *loading* bagian yang perlu diganti saja dari halaman tersebut. Sehingga *user* tidak memerlukan waktu lama untuk menunggu informasi yang dibutuhkannya tampil di layar.

II.5 Basis Data

Basis data terdiri dari dua kata yaitu basis dan data. Basis data dapat diartikan sebagai markas atau gudang, sedangkan data mewakili suatu obyek seperti manusia, barang, hewan, dan sebagainya. Dengan demikian basis data dapat didefinisikan sebagai himpunan

kelompok data yang saling berhubungan dan diorganisasikan sedemikian rupa tanpa adanya pengulangan (redudansi) agar dapat dimanfaatkan kembali dengan mudah dan cepat. Prinsip utama dari basis data adalah kemudahan dan kecepatan dalam mengambil kembali data/arsip.

Pengelolaan basis data secara fisik tidak dilakukan oleh pemakai secara langsung, tetapi ditangani oleh sebuah perangkat lunak yang disebut dengan *Database Management System* (*DBMS*). Perangkat lunak inilah yang akan menentukan bagaimana data diorganisasi, disimpan, diubah, dan diambil kembali.

Basis data *(database)* adalah kumpulan informasi yang berhubungan dengan subyek atau tujuan khusus seperti sistem stok barang dan penggajian pegawai. Pada pengelolaan basis data, informasi akan dimasukkan dalam tabel. Tabel adalah kumpulan informasi dengan topik yang khusus seperti karyawan, produk dan pemasok. Penggunaan tabel yang terpisah untuk tiap topik berarti memasukkan informasi sekali saja.

Untuk menghindari adanya informasi ganda, maka informasi yang akan kita masukkan ke dalam basis data dipecah menjadi beberapa tabel yang terpisah dengan topik khusus. Tabel akan mengorganisasikan data ke dalam kolom (*field*) dan baris (*record*). Informasi dikelola dengan cara berhubungan (*relational*) di mana tabel satu dengan lainnya saling berhubungan melalui *field* yang sama yang disebut dengan *field* kunci.

Ada dua macam field kunci:

1. *Primary Key* (kunci utama) yaitu satu atau lebih *field* unik yang mengidentifikasikan tiap-tiap *record* dalam suatu tabel.
2. *Foreign Key* (kunci tamu) yaitu satu atau lebih *field* yang menunjuk kepada *field primary key* atau *field-field* pada tabel lainnya.