BAB 2

LANDASAN TEORI

2.1 Teori Umum

Teori-teori pokok yang merupakan teori-teori landasan bagi teori-teori lainnya yang terdapat dalam skripsi ini, yaitu :

2.1.1. Analisis

Menurut Whitten (2004, p38), analisis sistem adalah pembelajaran dari *problem domain* bisnis untuk merekomendasikan perkembangan dan spesifikasi kebutuhan bisnis dan prioritas untuk solusi.

Analisis adalah kajian yang dilaksanakan terhadap sebuah bahasa guna meneliti struktur bahasa tersebut secara mendalam. Analisis merupakan evaluasi terhadap situasi dari sebuah permasalahan yang dibahas, termasuk di dalamnya peninjauan dari berbagai aspek dan sudut pandang.

2.1.2. Perancangan

Menurut O'Brien (2003, p511) perancangan berarti pengembangan secara spesifik dari hasil analisa kebutuhan untuk *hardware, software,* orang-orang, jaringan dan data serta produk informasi yang dapat memenuhi persyaratan fungsional dari suatu sistem.

Perancangan merupakan tahap penerjemahan dari keperluan atau data yang telah dianalisis ke dalam bentuk yang mudah dimengerti oleh pemakai (*user*).

2.1.3. *Engine*

Dalam pembuatan sebuah aplikasi diperlukan suatu alat (program) untuk menjalankan aplikasi tersebut. Program yang digunakan untuk menjalankan aplikasi tersebut adalah *Engine*. Contoh *engine* yang sering dipakai antara lain *search engine* dan *cheat engine.*

2.1.4. *Web Browser*

Web browser dikenal juga dengan istilah *browser* atau *internet browser*. *Browser* adalah *software* aplikasi yang memperbolehkan *user* dalam menampilkan dan berinteraksi dengan teks, gambar, video, musik, dan informasi lainnya yang terdapat di halaman *web*.

Teks dan gambar di halaman *web* bisa merupakan *hyperlink* yang mengarah ke halaman lain. *Web browser* menjadikan *user* lebih cepat dan mudah mengakses informasi. Dua program *web browser* yang cukup populer saat ini adalah *Microsoft Internet Explorer* dan *Mozilla Firefox.*

2.1.5. Bahasa Pemrograman *Internet*

Dalam proses pengembangan aplikasi yang berkaitan dengan *internet*, penggunaan *scripting language* merupakan salah satu hal pendukung utama yang menentukan seberapa dinamis atau statisnya aplikasi tersebut. Bahasa *scripting* adalah bahasa yang dapat menambah fitur-fitur tambahan secara tertulis pada HTML, mengingat terbatasnya kemampuan HTML. Bahasa *scripting* sendiri terbagi menjadi 2 jenis yaitu *client-side scripting* dan *server-side scripting*.

2.1.5.1. *Client-side Scripting*

Client-side scripting adalah *scripting* yang berjalan di sisi klien, dalam hal ini yaitu *web browser*. *Client-side scripting* ini biasa digunakan untuk fungsi-fungsi standar yang tidak berkaitan dengan basis data.

Client-side scripting digunakan sebagai bahasa pemrograman dalam *internet* apabila aplikasi tersebut membutuhkan suatu validasi ataupun hal-hal yang membuat aplikasi berbasis *internet* tersebut bersifat lebih interaktif**.** Masih terdapat banyak kekurangan dalam penggunaan *script* ini, antara lain adalah ketergantungannya yang tinggi terhadap *browser*. Aplikasi yang dibangun dengan *client-side scripting* bisa tidak berjalan apabila *browser* yang digunakan tidak mendukung penggunaan *script* tersebut. Selain itu jika ditinjau dari segi keamanan, penggunaan *script* ini dapat merugikan *programmer*.

Source code dapat dilihat oleh semua pihak yang mengaksesnya apabila tidak ada perlindungan terhadap *script* aplikasi tersebut.

Ada dua *client-side scripting* yang paling banyak digunakan saat ini adalah:

1. *Java Script* : *scripting* yang memberikan *interactivity* kepada halaman-halaman *HTML,* didukung oleh *Netscape.*
2. *VB Script* : *scripting* berbasiskan *Visual Basic* yang didukung oleh *Internet Explorer*.

2.1.5.2. *Server-side Scripting*

Server-side scripting adalah *scripting* yang dieksekusi melalui *web server* seperti *Apache*. *Server-side scripting* sangat berguna ketika dibutuhkan suatu fungsi ataupun validasi yang terhubung dengan basis data, keamanan *server,* ataupun proses melalui *server*. Penggunaan *server-side scripting* biasanya merupakan indikasi bahwa *web* aplikasi tersebut bersifat dinamis.

Cara kerja *server side scripting* dapat diibaratkan sebagai satu set instruksi yang diproses oleh *server,* dan menghasilkan HTML. HTML yang dihasilkan dikirim sebagai bagian dari tanggapan HTTP ke *browser*. *Browser* kemudian menampilkan HTML tersebut.

Server-side scripting sendiri memiliki beberapa kelebihan dibandingkan dengan *client-side scripting*. Selain bersifat lebih

fleksibel karena tidak adanya ketergantungan pada *browser*, penggunaan *server-side scripting* juga menguntungkan bagi para *programmer*. Hal ini dikarenakan *source code* yang telah dibuat tidak dapat dilihat begitu saja oleh para pengguna.

Beberapa contoh *server-side scripting* yang seringkali digunakan adalah *Active Server Pages* (ASP), *Java Server Pages* (*JSP*), dan *Hypertext Preprocessor* (PHP).

2.1.6. IMK (Interaksi Manusia dan Komputer)

2.1.6.1. Pengertian IMK

Menurut Ben Schneiderman (2005), Interaksi Manusia dan Komputer (IMK) adalah disiplin ilmu yang berhubungan dengan perancangan, evaluasi, dan implementasi sistem komputer interaktif untuk digunakan oleh manusia, serta studi fenomena-fenomena yang berhubungan dengannya. Fokus pada IMK adalah perancangan dan evaluasi antaramuka pemakai *(user interface)*. Antarmuka pemakai adalah bagian sistem komputer yang memungkinkan manusia berinteraksi dengan komputer.

2.1.6.2. Tujuan IMK

Tujuan IMK adalah :

1. Meningkatkan waktu belajar pengguna untuk mempelajari rancangan antar muka.

2. Mempercepat kinerja pengguna dalam melakukan tugasnya.
3. Mengurangi tingkat kesalahan yang dibuat oleh pengguna.
4. Memberikan ingatan jangka panjang pengguna terhadap penggunaan rancangan antar muka pengguna.
5. Meningkatkan kepuasan subjektif pengguna.

Demi tercapainya tujuan dari IMK, maka perancangan *interface* sebaiknya tidak lupa untuk mengikutsertakan evaluasi terhadap lima (5) faktor terukur dari manusia sebagai berikut (Schneiderman, 2005, p16) :

1. Waktu untuk belajar

   Ukuran berapa lama seorang *user* untuk mempelajari fungsi-fungsi di dalam sebuah aplikasi hingga pada akhirnya dapat menggunakannya dengan baik.

2. Kecepatan per*form*a

   Ukuran berapa lama suatu fungsi atau serangkaian tugas di dalam aplikasi tersebut dilakukan.

3. Tingkat *error* yang dilakukan pengguna

   Ukuran berapa banyak dan jenis *error* yang dilakukan oleh *user* di dalam melakukan serangkaian tugas.

4. Retensi waktu

   Ukuran berapa lama *user* mempertahankan ingatan dan pengetahuannya setelah beberapa jam, hari, atau bahkan beberapa minggu.

5. Kepuasan subjektif

   Ukuran seberapa puas *user* atas berbagai aspek dari suatu sistem.

Menurut Jacob Nielsen (2000), dalam IMK terdapat delapan aturan emas *(Eight Golden Rules)* yang digunakan dalam perancangan antarmuka pemakai yaitu :

1. Konsistensi

Konsistensi sangat diperlukan di dalam banyak hal, seperti : urutan aksi, istilah-istilah yang digunakan dalam *prompt*, menu, layar bantuan, warna, tata letak, huruf kapital, dan *font.*

2. Memungkinkan *frequent users* menggunakan *shortcuts*

Semakin sering pengguna memakai suatu fungsi, maka pengguna tersebut akan semakin membutuhkan *shortcut*. *Shortcut* dimaksudkan untuk meminimalisasi dan mempercepat interaksi yang dibutuhkan oleh pengguna sehingga pada akhirnya akan mempersingkat waktu yang dibutuhkan untuk menyelesaikan pekerjaannya.

3. Memberikan umpan balik yang informatif

Untuk setiap aksi yang dilakukan oleh pengguna, sebaiknya harus ada umpan balik dari sistem. Untuk aksi yang minor namun sering dipakai, umpan balik sebaiknya bersifat sederhana.

Sebaliknya, umpan balik yang lengkap diperlukan bagi aksi mayor namun jarang digunakan.

4. Desain kotak dialog

Urutan dari aksi-aksi harus diorganisasikan secara teratur apakah termasuk di dalam urutan awal, tengah, atau akhir. Kotak dialog akan mempermudah pengguna untuk mengingat urutan aksi yang telah dilakukannya. Hal ini akan membuat para pengguna dapat merencanakan aksi apa yang akan dilakukan berikutnya.

5. Memberikan pencegahan kesalahan dan penanganan *error* yang sederhana

Perancangan sistem yang baik sangat penting, sistem yang tidak memungkinkan penggunanya untuk dapat melakukan kesalahan yang serius. Jika sebuah *error* terjadi, maka sistem harus mendeteksinya, kemudian menawarkan mekanisme penanganan *error* yang sederhana namun terjamin dapat berhasil.

6. Memungkinkan pembalikan aksi yang mudah

Setiap aksi yang dilakukan pengguna sebaiknya dapat dibatalkan dengan mudah, misalnya dengan penggunaan *undo*.

Hal ini dimaksudkan agar pengguna tidak terlalu tegang ketika sedang mengeksplorasi suatu aplikasi.

7. Mendukung pusat kendali internal *(internal locus of control)*

Sistem harus memastikan agar pengguna benar-benar memegang kontrol akan sistem dan sistem tersebut merespon aksi yang dilakukan oleh pengguna.

8. Mengurangi beban ingatan jangka pendek

Tampilan dan fungsi sistem sebaiknya dibuat sesederhana mungkin agar pengguna dapat mengingatnya dalam jangka waktu yang cukup lama. Selain itu, akses *online* untuk *command-syntax forms*¸singkatan, kode, dan informasi lainnya juga harus disediakan oleh sistem.

Menurut Jacob Nielsen (2000), dalam perancangan antarmuka pemakai masih terdapat beberapa kesalahan yang sering dilakukan yaitu :

1. Penggunaan *frame*
2. Penggunaan teknologi baru dengan serampangan
3. Gerakan teks dan animasi yang berjalan terus
4. URL yang kompleks
5. Halaman tunggal

6. Halaman yang terlalu panjang gulungannya. Isi terpenting dan navigasi harus tampak di bagian atas
7. Kurangnya dukungan navigasi
8. Warna *link* yang tidak standar
9. Informasi yang kadaluarsa
10. Waktu *download* yang terlalu lama. Pemakai kehilangan minat pada *website* dalam 10-15 detik.

2.1.7. *Database*

Menurut Thomas Connolly dan Carolyn Begg (2005), *database* adalah suatu kumpulan *logical* data yang terhubung satu sama lain, deskripsi dari suatu data yang dirancang sebagai informasi yang dibutuhkan organisasi.

Menurut Wikipedia (http://id.wikipedia.org), *database* adalah kumpulan dari item data yang saling berhubungan satu dengan yang lainnya yang diorganisasikan berdasarkan sebuah skema atau struktur tertentu, tersimpan di *hardware* komputer dan dengan *software* untuk melakukan manipulasi untuk kegunaan tertentu. *Database* diperlukan karena merupakan salah satu komponen penting dalam sistem informasi, karena merupakan dasar dalam menyediakan informasi.

a) *Database* Lifecycle

Menurut Thomas Connolly dan Carolyn Begg (2005), tahapan dari *database lifecycle* adalah sebagai berikut :

1. Perencanaan *Database*
2. Pendefinisian Sistem
3. Pengumpulan *Requirement Database*
4. Rancangan *Database*
5. Rancangan Aplikasi
6. Pemilihan DBMS
7. *Prototyping (optional)*
8. Implementasi
9. *Loading* dan Konversi Data
10. *Testing*
11. Pemeliharaan Operasional

b) *Database Design*

Menurut Thomas Connolly dan Carolyn Begg (2005), *database design* adalah suatu proses menciptakan rancangan *database* yang nantinya digunakan untuk mendukung operasi perusahaan.

Faktor kesuksesan dalam merancang *database* adalah:

1. Kemungkinan bekerja secara *interactively* dengan *users*.
2. Kelengkapan mengikuti seluruh proses metodologi pembangunan model data.
3. Kelengkapan penggunaan pendekatan *data-driven.*
4. Pertimbangan struktur perusahaan dan kendala *integrity* ke dalam model data.
5. Kombinasi *conceptualization, normalization,* dan teknik validasi transaksi ke dalam metodologi pemodelan data.

c) *Database Management System (DBMS)*

Menurut Thomas Connolly dan Carolyn Begg (2005), DBMS adalah sebuah sistem *software* yang mengijinkan *user* untuk membuat, mengatur dan mengontrol akses ke *database*. DBMS terdiri dari koleksi data yang saling berhubungan dan koleksi program yang mengakses data tersebut. Tujuan utama dari DBMS adalah untuk menyediakan suatu lingkungan yang tepat dan efisien bagi *user* dalam memperoleh dan menyimpan informasi.

2.1.8. *Unified Modelling Language* (UML)

Menurut Thomas Connolly dan Carolyn Begg (2005), UML adalah alat untuk menggambarkan gambaran dari sistem yang akan dibuat melalui diagram dan simbol. Melalui seperangkat diagram, UML menyediakan standar yang memungkinkan sistem analis untuk merancang berbagai sudut pandang dari sistem, yang dinamakan model, yang dimengerti oleh *client, programmer* ,dan siapapun yang terlibat dalam proses pengembanganny a.

Menurut Martin Fowler dan Kendall Scott (1999, p13), UML adalah bahasa pemodelan, bukan suatu metode. UML tidak memiliki notasi atas proses yang merupakan bagian penting dari metode.

Menurut Whitten, Bentley, dan Dittman (2004, p430), *Unified Modelling Language (UML)* adalah suatu pendekatan yang digunakan untuk mempelajari objek-objek yang ada untuk melihat apakah objek tersebut dapat digunakan kembali atau dimodifikasi untuk kegunaan baru, dan mendefinisikan objek baru atau yang telah dimodifikasi yang akan digabungkan dengan objek yang ada untuk membuat aplikasi bisnis.

UML dapat digunakan untuk memvisualisasikan, menspesifikasikan, membangun, dan mendokumentasikan alat dari sebuah sistem perangkat lunak. UML hanya sebuah bahasa, dengan demikian hanya merupakan suatu bagian dari suatu metode pengembangan perangkat lunak.

UML terdiri dari beberapa diagram, antara lain :

a) Use case diagram

Secara grafis menggambarkan interaksi antara sistem, sistem eksternal dan pengguna. Dengan kata lain, secara grafis mendeskripsikan siapa yang akan menggunakan sistem dan dalam cara apa pengguna mengharapkan interaksi dengan sistem itu.

b) Activity diagram

Secara grafis digunakan untuk menggambarkan rangkaian aliran aktivitas baik proses bisnis maupun *use case*. Diagram ini juga dapat digunakan untuk memodelkan aksi yang akan dilakukan saat sebuah operasi diekseskusi dan memodelkan hasil dari aksi tersebut.

c) Sequence diagram

Secara grafis menggambarkan bagaimana objek berinteraksi satu dengan yang lainnya melalui pesan pada eksekusi sebuah *use case* atau operasi. Diagram ini mengilustrasikan bagaimana pesan terkirim dan diterima diantara objek dan dalam urutan apa.

d) *Class diagram*

Class diagram berguna untuk menggambarkan struktur objek sistem. Diagram ini menunjukkan kelas objek yang menyusun sistem dan juga hubungan antar objek tersebut.

2.1.9. *Internet*

Menurut Barry Eaglestone dan Mick Ridley (2001), *internet* merupakan sebuah integrasi dari jaringan-jaringan dengan menggunakan protokol standar komunikasi dimana protokol ini mampu menghubungkan jaringan-jaringan yang ada. *Internet* adalah sistem komputer umum yang terhubung secara global dan menggunakan protokol pertukaran paket *(packet switching communication protocol)*. Jumlah pengguna *Internet* yang besar dan semakin berkembang, telah mewujudkan budaya *internet*. *Internet* juga mempunyai pengaruh yang besar atas ilmu, dan pandangan dunia. Dibandingkan dengan buku dan perpustakaan, *internet* melambangkan pengetahuan informasi dan data secara ekstrim.

a) Pengertian Protokol

Protokol adalah sebuah aturan atau standar yang mengatur atau mengizinkan terjadinya hubungan, komunikasi, dan perpindahan data antara dua atau lebih titik komputer. Protokol dapat diterapkan pada perangkat keras, perangkat lunak, ataupun kombinasi dari keduanya. Pada tingkatan yang terendah, protokol mendefinisikan koneksi perangkat keras.

b) *FTP (File Transfer Protocol)*

File Transfer Protocol adalah sebuah protokol yang berjalan di dalam lapisan aplikasi yang merupakan standar untuk pemindahan data komputer antar mesin-mesin dalam sebuah jaringan. Sebuah *server FTP* diakses dengan menggunakan *Universal Resource Identifier (URI)*, klien *FTP* dapat terhubung dengan *server FTP* dengan membuka URI tersebut. *FTP* menggunakan protokol *Transmission Control Protocol (TCP)* untuk komunikasi data antara klien dan *server*, sehingga di antara klien dan *server* akan tercipta sebuah sesi komunikasi sebelum transfer data dimulai.

b) *URL (Universal Resource Locators)*

Uniform Resource Locators (URL) adalah bagian dari *Universal Resource Identifier* (URI). URI sendiri merupakan *syntax* standar untuk *string* yang mengidentifikasi sebuah *resource*. URI juga dapat diartikan sebagai standar untuk nama dan alamat dari objek atau *resource* pada *World Wide Web*. *Resource* adalah hal fisik atau abstrak yang memiliki identitas. URL digunakan untuk mengidentifikasi *resource* dari cara pengaksesannya (Daconta, 2003, p.44).

*Form*at umum dari sebuah URL adalah :

Protocol Transfer: //nama_domain//Path//nama_file

c) *HTTP (HyperText Transfer Protocol)*

HyperText Transfer Protocol adalah protokol yang dipergunakan untuk mentransfer dokumen dalam *World Wide Web*. Protokol ini dapat dipergunakan untuk berbagai macam tipe dokumen. HTTP adalah sebuah protokol yang menjembatani antara *client* dan *server*. Sebuah *client* HTTP seperti *web browser*, biasanya memulai permintaan dengan membuat hubungan TCP/IP melalui port tertentu ke *server*, kemudian *server* akan menjawab permintaan tersebut.

d) *TCP/IP (Transmission Control Protocol/Internet Protocol)*

Transmission Control Protocol/Internet Protocol adalah sekelompok protokol yang mengatur komunikasi data komputer di *internet* (Purbo, 1998, p1). TCP/IP merupakan sebuah standar jaringan terbuka yang bersifat independen sehingga dapat digunakan di mana saja. Protokol ini menggunakan skema pengalamatan yang sederhana yang disebut sebagai alamat IP *(IP address)* yang mampu membuat beberapa ratus juta komputer dapat saling berhubungan satu sama lainnya di *internet*. TCP/IP dikembangkan dengan tidak tergantung pada sistem operasi atau perangkat keras tertentu.

e) *Web*

Menurut Zimmerman (2003, p16), *Web* merupakan sebuah sistem penyedia informasi yang sangat besar bagi setiap orang di dalam jaringan. Informasi yang dapat disebarkan berupa teks, gambar, suara, video, dan tidak tertutup jenis informasi lainnya dalam pengembangannya. Untuk dapat mengakses halaman *web (web page)* diperlukan *browser* untuk merubah informasi dalam *web page*. Pada perkembangan awal, *HyperText Markup Language* dijadikan standarisasi bahasa dalam *web* dengan salah satu fasilitasnya yaitu *hyperlink*.

f) *WWW (World Wide Web)*

World Wide Web adalah suatu ruang informasi di mana sumber daya yang diidentifikasi oleh *Uniform Resource Locator (URL)*. WWW sering dianggap sama dengan *internet* secara keseluruhan, walaupun sebenarnya ia hanyalah bagian daripadanya. *HyperText* dilihat dengan sebuah program bernama *web browser* yang mengambil informasi dari *web server* dan menampilkannya, biasanya di sebuah monitor. *Web browser* memungkinkan kita mengikuti *hyperlink* di setiap halaman untuk pindah ke dokumen lain atau bahkan mengirim informasi kembali kepada *server* untuk berinteraksi dengannya. Aktivitas berpindah dokumen/halaman dengan mengikuti *hyperlink* disebut *"browsing"*.

2.2 Teori Khusus

Teori khusus yang berhubungan dengan topik yang dibahas dalam skripsi ini, terdiri dari :

2.2.1. *Game*

Menurut Salen dan Zimmerman (2003), sebuah *game* adalah sebuah sistem dimana pemain berinteraksi dengan konflik yang artifisial (tidak *real*), dibentuk oleh peraturan, yang hasilnya berupa hasil yang dapat dikuantifikasi (dapat dihitung).

Game adalah aktivitas atau konteks yang diatur oleh sekumpulan aturan tertentu. *Game* dapat memiliki jumlah pemain tertentu dan dapat dimainkan dalam bentuk kompetisi atau kooperatif. *Game* dapat diklasifikasikan dalam banyak cara, termasuk berdasarkan jumlah pemain yang memainkannya, atau dari tempat *game* tersebut dimainkan. Banyak *game* yang dapat masuk dalam lebih dari satu kategori di atas sehingga cara paling umum untuk mengklasifikasikan *game* adalah berdasarkan peralatan yang dibutuhkan untuk memainkannya.

Menurut Andrew Rollings dan Dave Morris (2003, p35-38), *game* bukanlah:

a) Kumpulan dari fitur yang bagus
b) Kumpulan grafik yang sangat fantasi
c) Kumpulan dari *puzzle*
d) *Setting* dan cerita yang bagus

2.2.2. *Java*

Java adalah sebuah bahasa pemrograman tingkat tinggi yang dikembangkan oleh *Sun Microsystems*. Awalnya, *Java* bernama *OAK*, dan didesain untuk peralatan mobile dan set-top boxes. *OAK* tidak sukses pada waktu itu sehingga pada tahun 1995, Sun mengganti namanya menjadi *Java* dan memodifikasi bahasa tersebut, untuk mengambil kesempatan dalam perkembangan *World Wide Web*.

Java merupakan bahasa berorientasi objek yang mirip dengan *C++*, tetapi disederhanakan untuk mengeliminasi fitur bahasa yang menyebabkan kesalahan *programming* yang umum. File-file *java source code* (file-file dengan *extension .java*) dikompilasi menjadi *form*at yang disebut *byte code* (file dengan *extension .class*), dimana ia dapat dieksekusi oleh interpreter *Java*. *Java code* yang telah dikompilasi dapat dijalankan pada sebagian besar komputer karena *Java interpreter* dan *runtime environment*, dikenal sebagai *Java Virtual Machines (VMs)*, dimiliki oleh sebagian besar sistem operasi, termasuk *UNIX, Macintosh,* dan *Windows*. *Bytecode* juga dapat langsung dikonversi menjadi instruksi bahasa mesin hanya dengan compiler *just-in-time (JIT)*.

2.2.3. *Java Server Pages (JSP)*

JavaServer Pages (*JSP*) merupakan cara yang sederhana dan cepat untuk membuat halaman *web (webpages)* yang menampilkan konten yang di-*generate* secara dinamis. Spesifikasi *JSP*, dikembangkan melalui suatu inisiatif dari industri yang didominasi oleh *Sun*

Microsystems, yang mendefinisikan interaksi antara *server* dan halaman *JSP*, dan mendeskripsikan *form*at dan syntax dari halaman tersebut.

Halaman *JSP* menggunakan *tag XML* dan *scriplets* yang ditulis dalam *Java programming* untuk meng-enkapsulasi logika yang meng-*generate* konten halaman. Halaman *JSP* tersebut melakukan *passing* pada *tag formatting* apapun (*HTML* atau *XML*) langsung kembali pada halaman respon *(response page)*. Cara tersebut memungkinkan halaman *JSP* memisahkan halaman logika dari desain dan tampilan.

Teknologi *JSP* adalah bagian dari keluarga teknologi *Java*. Halaman *JSP* dikompilasi ke dalam *Servlet* dan dapat memanggil komponen *JavaBeans* atau komponen *Enterprise JavaBeans* untuk melakukan *processing* di *server*. Teknologi *JSP* adalah komponen kunci dalam arsitektur skala besar untuk aplikasi berbasis *web*.

Halaman *JSP* tidak terbatas pada plat*form* atau *web server* tertentu. Spesifikasi *JSP* mewakili suatu spektrum yang luas dalam industrinya.

Halaman-halaman *JSP* akan dikompilasi ke dalam *servlet*, sehingga secara teoritis kita dapat menulis servlet untuk mendukung aplikasi berbasis *web* kita. Namun, teknologi *JSP* didesain untuk menyederhanakan proses pembuatan halaman dengan memisahkan penampilan *web* dari konten *web*. Di dalam banyak aplikasi, respon yang dikirim ke klien adalah sebuah kombinasi dari template data dan data yang di-*generate* secara dinamis. Dalam situasi seperti ini, lebih mudah

untuk bekerja dengan halaman-halaman *JSP* daripada melakukan semuanya dengan *Servlet.*

2.2.4. *Servlet*

Servlet adalah teknologi pada plat*form Java* yang dapat memperluas dan meningkatkan fungsi *web servers*. Servlet menyediakan suatu *component-based*, metode yang *platform-independent* untuk membangun aplikasi berbasis *web*, tanpa ada batasan per*form*a dari program-program CGI. Dan tidak seperti mekanisme *server extension* yang memiliki paten (seperti modul *Netscape Server API* atau *Apache)*, servlets tidak terbatas pada jenis *server* dan plat*form* (*server-independent* dan *platform-independent*). Ini membuat *user* bebas untuk memilih *platform* dan *tools* yang sesuai kebutuhan.

Servlet memiliki akses ke seluruh *Java APIs*, termasuk *JDBC API* untuk mengakses *database* enterprise. *Servlet* juga dapat mengakses library dari panggilan yang spesifik untuk *HTTP* dan memiliki seluruh keunggulan dari bahasa *Java* yang dewasa, termasuk *portability, performance, reusability* dan proteksi terhadap *crash*.

Servlet pada saat ini merupakan pilihan yang populer dalam membangun aplikasi *web* yang interaktif. Container servlet dari pihak ketiga tersedia untuk *Apache Web Server*. *Servlet container* biasa merupakan sebuah komponen dari *Web* dan *server* aplikasi, seperti *BEA*

WebLogic Application Server, IBM WebSphere, Sun Java System Web Server, Sun Java System Application Server, dan lainnya.

2.2.5. *AJAX (Asynchronous JavaScript and XML)*

AJAX adalah suatu istilah yang digunakan untuk mendeskripsikan suatu pendekatan dalam mendesain dan mengimplementasikan aplikasi *web*. *AJAX* adalah singkatan dari *Asynchronous JavaScript and XML*. Istilah ini pertama kali diperkenalkan dalam sebuah artikel oleh Jesse James Garrett dari Adaptive Path, sebuah firma desain *web* yang berbasis di San Fransisco. Dia menemukan istilah tersebut ketika dia menyadari kebutuhan akan cara yang mudah dan dapat dijual untuk memberikan gaya desain tertentu dan pembangunan untuk klien.

Tujuan utama dari *AJAX* adalah untuk membantu membuat fungsi aplikasi *web* lebih menyerupai sebuah aplikasi desktop. *HyperText Markup Language (HTML)*, bahasa yang menggerakkan *World Wide Web*, didesain berdasarkan ide *hypertext* – serangkaian teks yang dapat dihubungkan ke dokumen lain. Untuk *HTML*, sebagian besar aktifitas yang dilakukan oleh *end-user* di *browser*nya, mengirimkan sebuah *request* kembali ke *web server*. *Server* kemudian memproses request tersebut, mungkin mengirimkan request lagi, dan akhirnya merespon dengan apapun yang diminta oleh *user*.

Seiring dengan baiknya per*form*a pendekatan ini pada masa awal *Internet*, untuk aplikasi *web* modern, waktu menunggu yang konstan antara klik mulai membuat frustasi para *user* dan mengakibatkan

pengalaman yang kurang menyenangkan. Biasanya dalam semua aplikasi *web*, *user* memasukkan data ke dalam *form* dan kemudian melakukan klik pada tombol *submit* untuk mengirim request ke *server*. *Server* memproses request tersebut dan kemudian mengarahkan tampilan ke halaman baru (dengan me-*reload* seluruh halaman tersebut). Proses ini tidak efisien, menghabiskan banyak waktu, dan cukup membuat frustasi para *user* jika hanya sedikit data yang dibutuhkan. Sebagai contoh pada *form* registrasi *user*, ini dapat menjadi hal yang menyebalkan untuk *user*, karena keseluruhan halaman di-*reload* hanya untuk memeriksa ketersediaan *user name*. *User* menjadi terbiasa dengan respon yang sangat cepat dari aplikasi *desktop* dan tidak puas ketika sebuah *web*site tidak dapat memberikan respon yang sama cepatnya. Aplikasi *AJAX* membuat proses antara antarmuka *user* dan respon aplikasi menjadi lebih cepat dengan menambahkan suatu layer diantara antarmuka *user* dan komunikasi dengan *server*,. Seiring dengan makin populernya *AJAX* di aplikasi *web*, *user* menjadi terbiasa terhadap respon yang cepat ini, membantu untuk membuat lebih banyak bisnis untuk mengadopsi metodologi *AJAX*.

Sebuah aplikasi *AJAX* terdiri dari sejumlah aplikasi yang digunakan konjungsi untuk membuat proses yang lebih mulus. Ini termasuk *Extensible HTML (XHTML)* dan *Cascading Style Sheets (CSS)* untuk membangun struktur pokok halaman dan tampak visualnya. Interaksi semacam ini menggunakan *Document Object Model*, manipulasi data menggunakan *Extensible Markup Language (XML)*, pengambilan data menggunakan *XMLHttpRequest*, dan *JavaScript* untuk membantu

elemen-elemen yang berbeda ini berinteraksi satu sama lain. *AJAX* menyebar dengan sangat cepat melalui *web*, dengan contoh-contoh yang dapat dilihat di *web*site-*web*site besar. Sebagai contoh, *Google Maps*, dalam banyak hal melambangkan inti dari model *AJAX*, dengan fungsionalitas yang kompleks dan sangat mulus.

Cara kerja *AJAX* adalah sebagai berikut:

Ketika *user* pertama kali mengunjungi suatu *page*, engine *AJAX* akan diinisiasi dan di-*load*. Dari saat itu, *user* berinteraksi dengan *engine AJAX* untuk berinteraksi dengan *server web*. *Engine AJAX* beroperasi secara *asynchronous* ketika mengirimkan request ke *server* dan menerima respon dari *server*.

Langkah-langkah kerja *AJAX* dapat dibagi dalam beberapa tahap:

- *User* mengunjungi suatu *page*: *user* mengunjungi suatu *URL* dengan mengetik *URL* di *browser* atau melakukan klik pada *page* lain.
- Inisiasi dari engine *AJAX*: *engine AJAX* akan diinisiasi bersamaan dengan proses *loading page*. *Engine AJAX* juga dapat diset untuk me-*refresh* konten *page* secara terus-menerus tanpa me-refresh *page* secara keseluruhan.

- Putaran Proses Event:
    - *Browser event* dapat meminta *engine AJAX* untuk mengirimkan *request* ke *server* dan menerima data respon.
    - Respon *server – engine AJAX* menerima respon dari *server*. Kemudian dia memanggil fungsi *call back* dari *JavaScript.*
    - *Browser (View) update* – fungsi *request call back JavaScript* digunakan untuk meng-update *browser*. *DHTML* dan *CSS* berfungsi untuk mengupdate display *browser*.

Keuntungan dari *AJAX* adalah sebagai berikut:

- *AJAX* dapat digunakan untuk menciptakan aplikasi berbasis *web* yang penuh fitur, dimana tampilan dan per*form*anya seperti aplikasi *desktop.*
- *AJAX* memiliki tingkat kesulitan yang rendah untuk dipelajari. *AJAX* berbasis pada *JavaScript* dan teknologi yang sudah ada seperti *XML, CSS, DHTML*, yang membuatnya *familiar* bagi *programmer*.
- *AJAX* dapat digunakan untuk mengembangkan aplikasi *web* yang dapat mengupdate *page* data secara terus-menerus tanpa me-refresh keseluruhan *page.*

2.2.6. *JQuery*

JQuery adalah library yang sangat baik untuk mengembangkan aplikasi berbasis *AJAX*. *JQuery* sangat baik untuk programmer *JavaScript*, yang menyederhanakan pengembangan dari aplikasi *web 2.0*. *JQuery* membantu *programmer* untuk membuat code tetap sederhana dan singkat. *JQuery library* didesain untuk membuat hal-hal menjadi sederhana dan dapat digunakan kembali.

Library JQuery menyederhanakan proses transversal dari pohon *HTML DOM*. Kita dapat menggunakan *JQuery* untuk menangani event, menjalankan animasi, dan menambahkan dukungan *AJAX* ke dalam aplikasi *web* dengan mudah. Kemudahan ini ditambahkan lagi dengan sifatnya yang *cross-browser* atau dapat digunakan di beberapa jenis *browser* yang berbeda.

JQuery membuat kita mudah untuk membuat aplikasi *JavaScript* yang baik dan menciptakan efek-efek animasi yang menarik perhatian, bersaing dengan *Flash*. *JQuery* sangat baik untuk:

- Menambah efek animasi pada elemen-elemen

  JQuery membuat kita mudah untuk memberikan efek-efek seperti *fading in/out, sliding in/out*, dan expanding/contracting.

- Membuat *request XML (Ajax)*

  Dengan menggunakan *JavaScript*, kita dapat merequest data tambahan dari *Web Server* tanpa harus me-reload data.

- Memanipulasi *DOM*

  Kita dapat dengan mudah menambah, menghilangkan, dan menyusun ulang konten di dalam halaman *Web* hanya dengan beberapa baris code.

- Membuat slideshow image

  Dengan *JQuery*, kita dapat membuat slideshow yang menarik dengan tambahan efek-efek animasi.

- Membuat *drop-down menu*

  JQuery memberikan kemudahan untuk membuat menu *drop-down* dengan animasi.

- Membuat *drag-and-drop interfaces*

  Programmer bisa mengubah posisi dan menyusun urutan elemen-elemen dengan hanya *drag-and-drop* dalam membangun suatu halaman dengan *JQuery*.

- Memberikan banyak keunggulan pada *form*

  Programmer dapat dengan mudah menambah *complex client-side form validation*, membuat text field *auto-complete Ajax* yang dapat menarik data dari *server-side database*.

2.2.7. *MySQL*

MySQL saat ini merupakan sistem manajemen *database* yang paling populer, yang dikembangkan oleh *Oracle Corporation*. *SQL* pada *MySQL* merupakan singkatan dari *Structured Query Language*. *SQL* merupakan bahasa standar yang paling umum digunakan untuk mengakses *database* dan didefinisikan oleh standar *SQL ANSI/ISO*. Standar *SQL* telah berevolusi sejak 1986 dan beberapa versi sudah muncul.

Software *MySQL* merupakan software yang bersifat *OpenSource*, yang berarti semua orang dapat menggunakan dan memodifikasi software tersebut tanpa membayar sepeser pun. Kita dapat mempelajari *source code* dan menggantinya untuk menyesuaikan dengan kebutuhan. *MySQL* menggunakan *GPL (GNU General Public License)*, untuk mendefinisikan apa yang kita boleh dan tidak boleh lakukan pada *MySQL* dalam beberapa situasi yang berbeda.

Database Server MySQL bekerja sangat cepat, reliable dan mudah untuk digunakan. *MySQL Server* dikembangkan pada awalnya untuk menangani *database* yang besar dan telah berhasil untuk digunakan dalam lingkungan produksi yang tinggi permintaannya. Software *Database MySQL* merupakan sistem *client/server* yang terdiri dari *multi-threaded SQL server* yang mendukung *backends* yang berbeda-beda, beberapa program dan *library* client yang berbeda, *administrative tools*, dan *application programming interfaces (APIs)* yang cakupannya luas.