**BAB II**

**TINJAUAN PUSTAKA**

**2.1 *Single Sign On (SSO)***

*Single Sign On (SSO)* adalah suatu mekanisme dimana masing-masing *user* hanya memiliki satu akun yang berfungsi sebagai identitas *user* satu-satunya. Satu akun ini dapat digunakan untuk meminta izin dari sistem supaya *user* dapat mengakses berbagai aplikasi dengan *username* dan *password* yang sama dalam session tertentu. Single Sign On mengurangi jumlah human error yang merupakan alasan kegagalan utama dari sebuah sistem.

(http://www.opengroup.org/security/sso/)

Teknologi *Single Sign On* sangat penting bagi portal, singkatnya portal membutuhkannya untuk mengkoordinasikan beberapa *website* dan penyimpanan data, *XML*, dan sistem transaksi lainnya. Kesemuanya ini memiliki paradigma yang berbeda dimana solusi *Single Sign On* akan diterapkan. Contoh dari vendor yang memanfaatkan teknologi ini adalah *Netegrity, Oblix, IBM, dan Entrust*.

*Single Sign On (SSO*) adalah sebuah sistem yang memfasilitasi penanganan *user* *account* untuk beberapa *server* dengan hanya menggunakan satu *username* dan *password* saja. Sistem ini memiliki beberapa keuntungan, antara lain:
1. *User* tidak perlu mengingat banyak *username* dan *password*.
2. Kemudahan pemrosesan data.

 Jika setiap *server* memiliki data *user* masing-masing, maka pemrosesan data *user* (penambahan, pengurangan, perubahan) harus dilakukan pada setiap *server* yang ada. Sedangkan dengan menggunakan *SSO*, cukup hanya melakukan 1 kali pemrosesan.

**2.2 *Lightweight Directory Access Protocol (LDAP)***

Menurut Imam Cartealy (2013:75) *Lighweight Directory Access Protocol (LDAP)* merupakan protokol yang mendefinisikan bagaimana data direktori dapat diakses melalui jaringan . *LDAP* biasa digunakan untuk menyimpan berbagai informasi terpusat yang dapat diakses oleh berbagai macam mesin atau aplikasi dari jaringan. Penggunaan *LDAP* di dalam sistem akan membuat pencarian informasi menjadi terintegrasi dan sangat mudah. *LDAP* seringkali digunakan untuk menyimpan nama pengguna dan sandi yang terdapat di dalam sistem secara terpusat.

Menurut Ratdhian Cipta Sukmana (2006:1) Dalam terminologi komputer, *directory* aplikasibisa dikatakan sebagai suatu *database* tempat penyimpanan data, yang dapat di gunakan untuk memberikan informasi-informasi yang berkaitan dengan objeknya. Bagian direktoridapat berisi kumpulan informasi tentang *user* seperti *sure name, first name, phone number, User ID, mail address* dan lain sebagainya.



Gambar 2.1. Konsep Direktori

Secara prinsip struktur *database* pada suatu *directory* aplikasiadalah hierarki seperti yang di tunjukkan pada gambar 2.1. Suatu *directory* aplikasi akan memiliki item yang di jadikan sebagai *root*. Untuk sebuah titik *root*, secara umum di tunjukkan dengan suatu *attribute* *dc (Domain Component)* atau *o (Organization)* mungkin juga *ou (Organization Unit).* Kemudian pada titik daun *(leaf)* biasanya akan berisi item dengan attribut *uid (User ID)* ataupun *cn (Common Name). Directory* aplikasibiasanya menyimpan informasi dalam bentuk struktur *tree* yang dinamakan *Directory Information Tree (DIT).* Setiap titik pada *DIT* diberi suatu alamat, baik secara relatif maupun secara absolut. Untuk suatu alamat relatif sering disebut sebagai *RDN (Relative Distinguish Name)* sedangkan alamat yang absolut di sebut sebagai *DN (Distinguish Name).* Untuk mendapatkan informasi tentang *user* *rsukmana* pada gambar diatas, dapat di tuliskan dengan pengalamatan misalnya:

*“dn=uid=rsukmana,ou=people,dc=smartbee,dc=com”.* Konsep seperti inilah yang di gunakan oleh direktori *LDAP*.

Menurut Imam Cartealy (2013:76) Ada tiga definisi yang sangat penting di *LDAP*, yaitu skema, kelas, dan atribut. Ketiganya saling terkait dan menjadi tulang punggung *LDAP* .

* + 1. Skema

Skema bisa diibaratkan seperti sistem pemaketan untuk kelas objek dan atribut. Setiap kelas objek dan atribut harus didefinisikan di dalam skema, dan skema tersebut harus dideklarasi didalam berkas konfigurasi *deamon slapd*, *slapd.conf* .

* + 1. Kelas Objek

Kelas objek merupakan *container* yang berfungsi untuk mengelompokkan atribut. Kelas objek akan menentukan apakah suatu atribut harus ada, atau bersifat pilihan.

1. Atribut

Atribut merupakan struktur terkecil dari skema dan merupakan anggota kelas objek. Atribut memiliki nama dan juga memiliki nilai dan setiap atribut dapat memiliki lebih dari satu nilai.

Secara teknis, *LDAP* merupakan sebuah protokol untuk mengakses ke *Directory* Aplikasi *X-500*. Pertama-tama, *client* mengakses *gateway* ke X-500. *Gateway* tersebut akan menjalankan *LDAP* di antara *client* dan *gateway*, sekaligus menjalankan *Directory Access Protocol X-500* antara *X-500* dan *server*. Karena *LDAP* merupakan jenis simpel dari *DAP*, maka *LDAP* menyediakan sebagian besar dari fungsi *DAP* dengan biaya yang jauh lebih rendah. *LDAP* menggunakan model *client* *server*. Saat *client* terkoneksi ke *server* dan mengajukan *LDAP* *request*, *server* merespon dengan jawaban dan/atau dengan pointer ke arah mana *client* dapat mendapat tambahan informasi (khususnya ke *server* *LDAP* yang lain). Tidak masalah *server* *LDAP* yang mana yang merespon *request* dari *client*, karena *client* tersebut akan mendapat informasi yang sama. ([http://www.scribd.com/doc/56286070/stuktur-*LDAP*](http://www.scribd.com/doc/56286070/stuktur-ldap))

**2.3 *Remote Access Dial in User (RADIUS)***

*RADIUS* adalah protocol standar industry yang dijelaskan dalam RFC 2865, “Remote Authentication Dial-In *User* Service (*RADIUS*)”, dan RFC 2886, “*RADIUS* Accounting”. *RADIUS* digunakan untuk menyediakan layanan otentikasi, otorasasi, dan akuntasi. Client *RADIUS* mengirimkan data *user* dan informasi parameter koneksi dalam bentuk pesan *RADIUS* ke *server* *RADIUS*. *Server* *RADIUS* mengotentikasi dan mengotorisasi permintaan client *RADIUS*, dan mengirimkan kembali sebuah respon dari pesan *RADIUS*. Client *RADIUS* juga mengirimkan pesan akuntasi *RADIUS* ke *server* *RADIUS*. Sebagai tambahan, standar *RADIUS* mendukung penggunaan proxy *RADIUS*. Proxy *RADIUS* adalah computer yang meneruskan pesan *RADIUS* diantara komputer-komputer yang aktif-*RADIUS*.([http://technet.microsoft.com/en-us/library/cc781821(WS.10).aspx](http://technet.microsoft.com/en-us/library/cc781821%28WS.10%29.aspx))

 Menurut Imam Cartealy (2013:123) *RADIUS* sendiri tidak menyimpan informasi mengenai klien, tetapi *RADIUS* dapat menggunakan berbagai macam backend seperti *MySQL* dan *LDAP* untuk menyimpan informasi tersebut. Yang terjadi ketika *RADIUS* melakukan otentikasi terhadap klien *RADIUS* akan menanyakan informasi seperti sandi ke *back-end* tersebut.

 *RADIUS* biasanya bekerja dengan aplikasi klien *RADIUS*. Aplikasi inilah yang sebenarnya akan mengirimkan permintaan dari pengguna ke *RADIUS*. *RADIUS*-lah yang melakukan fungsi *AAA*. Klien *RADIUS* hanyallah mengirimkan ataupun meneruskan permintaan ke *server* *RADIUS*.

**2.4 *FreeRADIUS***

*FreeRADIUS* adalah sebuah produk *RADIUS open-source* yang dilengkapi degan modul, performa tinggi, dan banyak fitur termasuk didalamnya adalah *server, client, development libraries*, dan berbagai kegunaan tambahan yang berhubungan dengan *RADIUS*.

 Sebagai *open-source* *RADIUS* yang pertama, *freeRADIUS* ini dimasukkan sebagai paket standar dengan banyak sistem operasi, memiliki paket *binary*, dan sumber yang bisa digunakan untuk membangun hampir di semuanya. Penyebaran produksi ini mencakup instalasi skala besar yang berisikan banyak *server* *AAA* dengan pengguna lebih dari sepuluh juga dan jutaan permintaan setiap harinya. *FreeRADIUS* juga mendukung permintaan *proxy* dengan *fail*-*over* (kemampuan untuk berpindah ke *server* yang redudansi atau siaga ketika terjadi kegagalan) dan *load balancing* (teknik mendistribusikan pekerjaan pada *port-port* jarungan untuk meningkatkan penggunaan jaringan), sebaik dengan kemampuan untuk mengakses berbagai macam tipe basis data *back-end*. (<http://freeradius.org/>)

**2.4 *Zimbra Collaboration Suites (ZCS)***

Menurut Tuxkeren (2012:3) *Zimbra* *Collaboration Suites (ZCS*) atau yang lebih dikenal dengan *Zimbra* adalah sebuah produk *Groupware* yang dibuat oleh *Zimbra*, *Inc* yang berlokasi di Palo Alto, California, Amerika Serikat. Pada masa awal-awalnya perusahaan ini dibeli oleh Yahoo! Tepatnya pada bulan September 2007.

 Pada awal tahun 2010, *Zimbra* dibeli oleh VMWare. Aplikasi *Zimbra* sendiri dibuat menjadi dua bagian komponenn yang bersifat *Client* dan *Server*. *Zimbra* juga memiliki dua model lisensi, yaitu komersial dengan nama produk *Zimbra* *Network Edition*, dan *Zimbra* *Appliance* yang bersifat virtualisasi serta sebuah lisensi lagi yang bersifat *open sources*.

 *Zimbra* *web* *client* ada sebuah perangkat kolaborasi yang penuh dengan menggunakan teknologi *AJAX*, sehingga fitur-fitur seperti *tool tips, drag and drop* item, dan klik kanan dapat dilakukan di *internet* *browser*, juga ditambah lagi dengan *addons* yang dinamakan *Zimlet* sehingga membuat *Zimbra* menjadi kaya fitur.

 Sementara untuk *Zimbra* *server*, menggunakan teknologi dari beberapa aplikasi *open sources* yang sudah matang dan stabil, yang meliputi aplikasi *MTA, POP3*, *LDAP*, *MySQL* dan lain-lain. *Zimbra* *server* sendiri dapat di install di beberapa distro *linux* yang terkenal.

 Pembuat *Zimbra* pertama kalinya adalah Mr. Scott Dietzen, nama *Zimbra* sendiri terinspirasi dari lagu pop tahun 70an yang berjudul “*I* *Zimbra*” yang dibawakan *group* band *Talking Heads*.

**2.4 *Moodle***

**2.4.1 Pengertian**

Menurut Amiroh, S.Kom (2012:1) *Moodle* merupakan program *open source* yang paling terkenal diantara program-program *e-learning* yang ada, misalnya ATutor, eLeaPTM *LEARNING MANAGEMENT SYSTEM (LMS)*, dan seterusnya. Aplikasi *moodle* ini dikembangkan pertama kali oleh Martin Dougiamas pada Agustus 2002 dengan *moodle* versi 1.0. Versi terbaru dari *moodle* saat buku ini ditulis adalah *Moodle* 2.2.2+ dengan kapasitas file 91.4 MB.

 Karena bersifat *open source*, maka *moodle* dapat diunduh secara gratis dari situs resminya [http://www.*moodle*.org](http://www.moodle.org) dan dapat dimodifikasi oleh siapa saja dengan lisensi *GNU* (*General Public License*).

**2.4.1 Kelebihan *Learning Management System (LMS)* *Moodle***

Banyak hal yang membuat *moodle* berbeda dengan yang lain, diantaranya:

* + 1. Sederhana, efisien, dan ringan serta kompatibel dengan banyak browser.
		2. Instalasi yang sangat mudah.
		3. Dukungan berbagai bahasa termasuk Bahasa Indonesia.
		4. Tersedia manajemen situs untuk melakukan pengaturan situs secara keseluruhan, perubahan modul dan lain sebagainya.
		5. Tersedia manajemen pengguna (*user* management).
		6. Tersedianya manajemen course yang baik.
		7. Tersedia modul, modul polling , modul forum, modul untuk jurnal, modul untuk kuis, modul untuk workshop dan survey, serta masih banyak lagi.

(Amiroh, S.Kom, 2012:2)

**2.7 *System Flowcart***

**2.7.1 *Flowchart***

Adalah penyajian yang sistematis tentang proses dan logika dari kegiatan penanganan informasi atau penggambaran secara grafik dari langkah-langkah dan urut-urutan prosedur dari suatu program. *Flowchart* menolong analis dan programmer untuk memecahkan masalah kedalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian. (Anharku, 2009:1)

**2.7.2 *System flowchart***

Adalah urutan proses dalam system dengan menunjukkan alat media input, output serta jenis media penyimpanan dalam proses pengolahan data. (Aharku, 2009:1)

**2.7.2.1 Pedoman-Pedoman Dalam Membuat *Flowchart***

Jika seorang analis dan programmer akan membuat *flowchart*, ada beberapa petunjuk yang harus diperhatikan, seperti :

1. *Flowchart* digambarkan dari halaman atas ke bawah dan dari kiri ke kanan.

2. Aktivitas yang digambarkan harus didefinisikan secara hati-hati dan definisi ini harus dapat dimengerti oleh pembacanya.

3. Kapan aktivitas dimulai dan berakhir harus ditentukan secara jelas.

4. Setiap langkah dari aktivitas harus diuraikan dengan menggunakan deskripsi kata kerja, misalkan Melakukan penggandaan diri.

5. Setiap langkah dari aktivitas harus berada pada urutan yang benar.

6. Lingkup dan *range* dari aktifitas yang sedang digambarkan harus ditelusuri dengan hati-hati. Percabangan-percabangan yang memotong aktivitas yang sedang digambarkan tidak perlu digambarkan pada *flowchart* yang sama. Simbol konektor harus digunakan dan percabangannya diletakan pada halaman yang terpisah atau hilangkan seluruhnya bila percabangannya tidak berkaitan dengan sistem.

7. Menggunakan simbol-simbol *flowchart* yang standar.

(Anharku, 2009:1)

**2.7.2.2 Simbol – simbol *Flowchart***

 Dalam membuat *system flowchart* terdiri dari beberapa simbol-simbol yang mana simbol-simbol ini saling berhubungan satu sama lainnya. Adapun simbol-simbol tersebut diantara lain seperti yang terdapat pada tabel 2.1 dibawah ini.

Table 2.1. Simbol-simbol *Flowchart* Sistem

|  |  |  |
| --- | --- | --- |
| **No** | **Simbol** | **Keterangan Fungsi** |
| 1 | Terminator  | Pemulaan atau akhir dari program. |
| 2 | Garis Alir *(Flow Line)* | Arah aliran program |
| 3 | *Preparation* | Proses inisiasi atau pemberian harga awal. |

Tabel 2.1 Lanjutan

|  |  |  |
| --- | --- | --- |
| **No** | **Simbol** | **Keterangan Fungsi** |
| 4 |  Proses | Proses perhitungan atau proses pengolahan data. |
| 5 | *Input/output data* | Proses *input* atau *output data,* parameter, informasi |
| 6 | *Predefined Process (Sub Program)* | Permulaan sub program atau proses menjalankan sub program. |
| 7 | *Decision*  | Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya. |
| 8 | *On page connector* | Penghubung bagian-bagian *flowchart* yang berada pada satu halaman. |
| 9 | *Off page connector* | Penghubung bagian-bagian *flowchart* yang berada pada halaman yang berbeda. |
|  |  |  |

(sumber: anharku, 2009:2)