# LAPORAN AKHIR

# PENELITIAN DOSEN PEMULA


# PENGEMBANGAN PROTOTYPE SINGLE SIGN ON UNIVERSITAS

Tahun ke 1 dari rencana 1 Tahun

**TIM PENELITI**

**YESI NOVARIA KUNANG, 0226117501**
**ILMAN ZUHRI YADI, 0229047501**

**UNIVERSITAS BINA DARMA**
**NOVEMBER 2013**

## HALAMAN PENGESAHAN

| | | |
|---|---|---|
| **Judul Kegiatan** | : | Pengembangan Prototype Single Sign On Universitas |
| **Peneliti / Pelaksana** | | |
| Nama Lengkap | : | YESI NOVARIA KUNANG ST.,M.KOM |
| NIDN | : | 0226117501 |
| Jabatan Fungsional | : | |
| Program Studi | : | Sistem Informasi |
| Nomor HP | : | |
| Surel (e-mail) | : | yesi_kunang@mail.binadarma.ac.id |
| **Anggota Peneliti (1)** | | |
| Nama Lengkap | : | ILMAN ZUHRI YADI |
| NIDN | : | 0229047501 |
| Perguruan Tinggi | : | UNIVERSITAS BINA DARMA |
| **Institusi Mitra (jika ada)** | | |
| Nama Institusi Mitra | : | |
| Alamat | : | |
| Penanggung Jawab | : | |
| **Tahun Pelaksanaan** | : | Tahun ke 1 dari rencana 1 tahun |
| **Biaya Tahun Berjalan** | : | Rp. 12.500.000,00 |
| **Biaya Keseluruhan** | : | Rp. 15.000.000,00 |

Mengetahui
Dekan Fakultas Ilmu Komputer

(M. Izman Herdiansyah, S.T., MM. Phd.)
NIP/NIK 090109088

Palembang, 23 - 11 - 2013,
Ketua Peneliti,

(YESI NOVARIA KUNANG ST.,M.KOM)
NIP/NIK030302208

Menyetujui,
Direktur Lembaga Penelitian

(Prihambodo H. Saksono,S.T.,M.Sc.,Ph.D)
NIP/NIK 110109348

# DAFTAR ISI

# DAFTAR TABEL

## DAFTAR GAMBAR

## DAFTAR LAMPIRAN

# RINGKASAN PENELITIAN

Banyaknya sistem dan aplikasi yang digunakan di suatu Universitas memberikan kendala tersendiri bagi pengguna, yaitu sulitnya pengguna sistem untuk mengingat *user* dan *password login* pada masing-masing sistem tersebut. Selain itu juga admin harus mengelola semua user yang ada pada masing-masing sistem yang terdistribusi tersebut. Kendala akan terjadi dengan banyaknya *user* dan *password* pada masing-masing sistem tersebut, mengakibatkan seringnya *user* lupa dengan *password* mereka. Ditambah lagi jika si *user* tidak lagi aktif di suatu aplikasi ataupun di seluruh sistem, maka si admin harus melakukan perubahan data user di masing-masing sistem. Untuk itu penelitian ini bertujuan untuk mengembangkan *prototype Single Sign On* yang aman dan bisa diimplementasikan di suatu Universitas, yang memungkinkan pengguna hanya perlu melakukan satu kali *login* pada keseluruhan sistem yang ada.

**Kata Kunci** : *Single Sign-on*, Autentikasi, Universitas, *LDAP, CAS*

# PRAKATA

Puji syukur disampaikan kehadirat Allah SWT yang telah memberikan rahmat dan karunia-Nya, sehingga peneliti dapat menyelesaikan penelitian yang berjudul "Pengembangan Prototype Single Sign On Universitas". Shalawat dan salam tak lupa dihaturkan kepada junjungan kita baginda Rasulullah Nabi Muhammad SAW.

Dalam penelitian ini peneliti telah banyak mendapatkan bantuan dan bimbingan dari berbagai pihak, baik secara moril maupun materil, sehingga peneliti dapat menyelesaikan penelitian ini. Untuk itu, peneliti mengucapkan terima kasih kepada semua pihak yang ikut berpartisipasi yang tidak bisa peneliti sebutkan satu persatu, semoga segala amal kebaikan ini mendapatkan balasan dari Allah SWT.

Akhirnya, dengan segala kerendahan hati peneliti menyadari bahwa laporan penelitian ini masih jauh dari kesempurnaan, maka pada kesempatan ini peneliti mengharapkan kritikan dan saran yang bersifat membangun demi kesempurnaan dari segenap pembaca. Akhir kata peneliti do'a-kan semoga semua amal yang diberikan mendapat imbalan dari Allah SWT, dan semoga penelitian ini bermanfaat bagi kita semua.

Palembang,

Peneliti

# BAB 1
# PENDAHULUAN

## 1.1. Latar Belakang

Teknologi informasi yang berkembang pesat telah umum digunakan pada semua bidang termasuk juga pada bidang pendidikan. Di suatu Universitas penggunaan teknologi informasi sudah menjadi keharusan untuk menunjang kelancaran proses belajar mengajar yang ada di Universitas tersebut. Universitas biasanya memiliki sistem akademis, dan beberapa aplikasi atau sistem lain seperti *elearning, email*, *digital library*, dan lain-lain yang digunakan mahasiswa, dosen, maupun karyawan di lingkungan Universitas tersebut.

Masing-masing sistem yang ada di Universitas biasanya dikembangkan oleh pengembang yang berbeda-beda dan memiliki platform yang berbeda-beda. Hal ini menjadi kendala bagi Universitas sendiri dikarenakan sistem-sistem tersebut memiliki database dan sistem autentikasi/login sendiri-sendiri. Dengan banyaknya sistem autentikasi tersebut mengakibatkan pengguna memiliki *user* dan *password* yang berbeda-beda pada masing-masing sistem tersebut. Hal ini tentu saja sangat berkendala yaitu *user* akan sulit mengingat *user* dan *password* yang mereka miliki. Selain itu juga kendala di sisi administrator yang harus mengelola user-user yang ada di seluruh sistem, sehingga sangat sulit bagi administrator untuk mensinkronisasi user yang ada. Sebagai contoh jika user yang merupakan mahasiswa yang sudah tamat maka sulit bagi administator untuk menghapus *account user* tadi di seluruh sistem yang ada.

Untuk mensinkronisasikan data pengguna sistem yang ada di Universitas, sejauh ini telah dilakukan antara lain dengan memanfaatkan *webservice* (Yesi, 2012 dan Nasir, 2012). Akan tetapi sinkronisasi tersebut baru dilakukan untuk di sistem autentikasi hotspot dan sistem otomasi perpustakaan dengan sistem akademis, belum menyeluruh ke seluruh layanan. Selain itu juga dengan sinkronisasi yang menggunakan konsep *SOA* tersebut memiliki kendala kurang *up to date* nya data pengguna sistem autentikasi, tergantung jadwal proses

sinkronisasi data tersebut. Untuk itulah dalam penelitian ini peneliti tertarik untuk mencoba mengembangkan *prototype* sistem *Single Sign On* untuk yang memudahkan pengguna layanan aplikasi yang tersedia di Universitas khususnya di Universitas Bina Darma tempat diadakannya penelitian ini. Sehingga cukup dengan melakukan proses 1 kali login maka pengguna bisa menggunakan aplikasi/layanan lain yang tersedia di Universitas.

Di perguruan tinggi lain yang sudah menggunakan sistem *Single Sign On* antara lain di ITB yang memiliki 2 sistem *SSO* untuk beberapa aplikasi, UGM (http://pasca.geologi.ugm.ac.id/download.phpfile=Buku_Panduan_Layanan_TIK_2011.pdf), UI yang sudah mengembangkan *SSO* berbasis teknologi *SAML (Security Assertion Markup Language)* dan *CAS (Central Authentication Service)* yang baru selesai 2011 dan baru akan diimplementasikan, Universitas Padjajaran yang memiliki *SSO* yang dinamakan PauS untuk 3 aplikasi dan di Universitas lain juga sudah mulai diterapkan. Jika dilihat dari keberhasilan Universitas/ Perguruan Tinggi tersebut mengembangkan sistem *SSO*, *SSO* ini bisa menjadi solusi untuk memudahkan prosis login di berbagai aplikasi, meskipun masih terdapat berbagai kendala penerapan yang diakibatkan *platform interoperabilitas* beberapa aplikasi yang mungkin tidak mendukung suatu teknologi *SSO*. Belajar dari pengalaman perguruan tinggi lain yang telah mengimplementasikan *SSO* dengan berbagai kendalanya tersebut, maka diharapkan dengan penelitian ini nantinya dihasilkan *prototype* sistem *SSO* yang bisa diimplementasikan di Universitas Bina Darma tempat penelitian ini diadakan.

### 1.2. Perumusan Masalah

Dengan banyaknya layanan aplikasi yang tersedia di Universitas memberikan kendala bagi pengguna diakibatkan banyaknya *user* dan *password* yang mesti diingat, selain itu bagi administrator sendiri menyulitkan mengelola akun pengguna di setiap aplikasi yang ada, untuk itu diperlukan sistem *SSO* yang memberikan kemudahan bagi pengguna dengan satu akun yang dimiliki, maka pengguna bisa masuk ke berbagai layanan yang tersedia di Universitas.

### 1.3. Tujuan Penelitian

Penelitian ini bertujuan untuk:

1. Merencanakan pengembangan integrasi seluruh sistem yang ada dengan sistem *Single Sign On* untuk autentifikasi pengguna.
2. Memetakan seluruh sistem yang ada di Universitas dan kendala integrasi sistem tersebut ke *SSO*.
3. Mendesain hirarki direktori *LDAP* pengguna yang bisa diterapkan di Universitas.
4. Mendesain teknologi *SSO* yang aman yang bisa diterapkan di Universitas.

### 1.4. Manfaat Penelitian

1. Penelitian ini menghasilkan *prototype SSO* Universitas yang bisa diimplementasikan.
2. Selain itu juga dengan penelitian ini bisa diketahui berbagai kendala aplikasi yang ada di Universitas untuk mengimplementasikan *SSO*.

### 1.5. Batasan Penelitian

Sistem *SSO* yang dikembangkan di Universitas ini berbasis teknologi *LDAP* dan *CAS,* yang juga akan ditinjau sisi keamanannya.

### 1.6. Sistematika Pembahasan

Sistematika pembahasan ini bab dalam penelitian ini adalah sebagai berikut :

BAB I PENDAHULUAN

Pada Bab ini menguraikan latar belakang masalah, tujuan dan manfaat penelitian, rumusan masalah, pembatasan masalah dan sistematika penulisan.

BAB II LANDASAN TEORI

Pada Bab ini menguraikan pengertian mengenai landasan pemikiran yang berisi teori-teori mengenai *Single Sign On (SSO)*, jenis-jenisnya dan *Lightweight Directory Access Protocol (LDAP)*, serta system flowchart.

BAB III METODE PENELITIAN

Pada Bab ini dimulai dengan metode penelitian, prosedur penelitian, dan kriteria pengukuran.

BAB IV PERANCANGAN DAN PENGEMBANGAN SISTEM SSO

Pada Bab ini dimulai dengan menguraikan rancangan yang akan dibuat untuk menyelesaikan permasalahan yang ada.

BAB V HASIL DAN PEMBAHASAN

Pada bab ini menguraikan tentang hasil pengujian dari implementasi pada bab sebelumnya seperti membahas kelebihan sistem yang digunakan serta kekurangannya.

BAB VI SIMPULAN DAN SARAN

Pada Bab ini menguraikan kesimpulan - kesimpulan dari pembahasan bab-bab di atas dan kemudian dilanjutkan dengan saran - saran.

# BAB 2
# 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 *useraccount* untuk beberapa *server* dengan hanya menggunakan satu *username* dan *password* saja. Sistem ini memiliki beberapa keuntungan, antara lain :

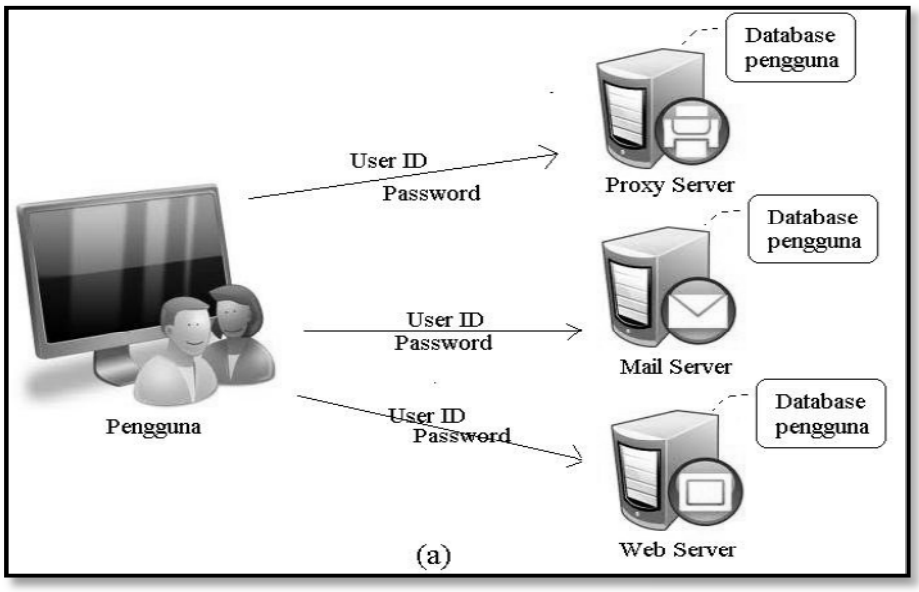
1. *User* tidak perlu mengingat banyak *username* dan *password*.
2. Kemudahan pemrosesan data.

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.

*SSO* merupakan teknologi yang memiliki kemampuan untuk memasukkan *id* dan *password* yang sama untuk *login* ke beberapa aplikasi dalam suatu perusahaan. Seperti *password* adalah mekanisme otentikasi paling aman, *SSO* kini

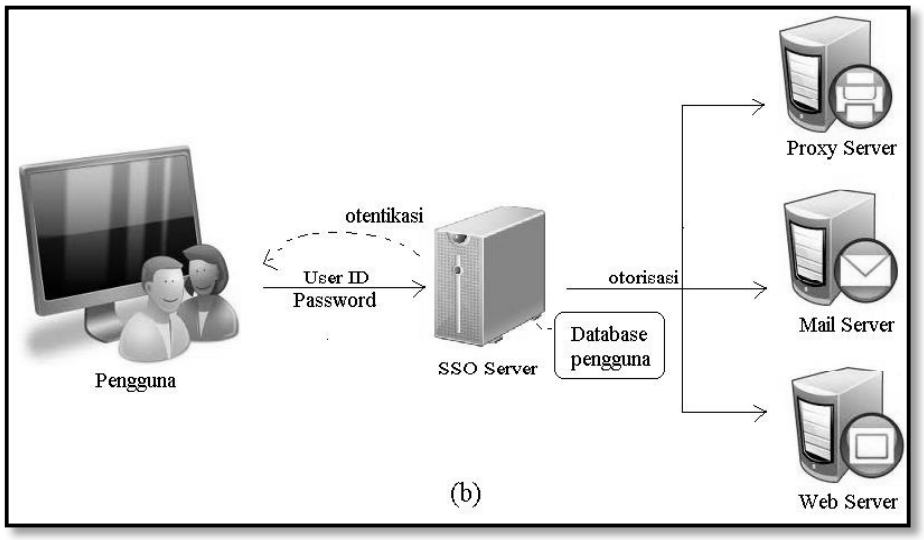
telah dikenal sebagai *reduced sign on* (*RSO*) sejak lebih dari satu jenis mekanisme otentikasi yang digunakan sesuai dengan model risiko perusahaan.

Untuk jaringan yang sangat besar dan bersifat heterogen, dimana pengguna diminta untuk mengisi informasi dirinya pada setiap aplikasi yang hendak di akses dibutuhkan *SSO*.Sistem *SSO* tidak memerlukan interaksi yang manual, untuk mengakses seluruh layanan aplikasi tanpa harus melakukan *login* dan mengetikan *password*-nya berulang kali. *SSO* meng-otentikasi pengguna pada semua aplikasi yang telah di-*authorized* untuk diakses. Ini menghilangkan permintaan *authentication* lagi ketika pengguna mengganti aplikasi selama *session* berlaku (Rudy, 2009). *SSO* juga memperkenankan informasi autentikasi dan mengidentifikasi subjek secara ketat guna menghindari *login* ganda pada sistem atau kelompok sistem yang terpercaya. Sistem *SSO* juga dapat memusatkan penglolaan dari parameter sistem yang relevan pada saat bersamaan dan meningkatkan penggunaan secara keseluruhan. Pengguna layanan bisa lebih menyukai sistem *SSO* dari pada sistem *sign-on* biasa.Gambaran perbedaan sistem *SSO* dengan sistem *sign-on* dapat dilihat pada gambar berikut.




(sumber : *repository.usu.ac.id/bitstream/.../3/Chapter%20II.pdf)*

**Gambar 2.1**(a) Sistem *Sign On*




(sumber : *repository.usu.ac.id/bitstream/.../3/Chapter%20II.pdf)*

**Gambar 2.2** (b) Gambaran Sistem *SSO*

Arsitektur Sistem *SSO* memiliki dua bagian utama yaitu *agent* yang berada di *web server* / layanan aplikasi dan sebuah *serverSSO* berdedikasi yang akan dijelaskan sebagai berikut:

1. *Agent* : permintaan setiap *HTTP* yang masuk ke *web server* akan diterjemahkan oleh *agent*. Di tiap-tiap *web server* ada satu *agent* sebagai *host* dari layanan aplikasi. *Agent* ini akan berinteraksi pada *server SSO* dari sisi lain aplikasi dan berinteraksi dengan *web browser* dari sisi pengguna.
2. *SSO server* : Dalam menyediakan fungsi manajemen sesi *cookies* temporer (sementara) menggunakan *server SSO. User-id, session creation time, session expiration time* dan lain sebagainya adalah informasi ada pada *cookies.*

Produk-produk sistem *SSO* yang berbasis *open source* yang umum digunakan pada saat ini adalah *CAS*, *OpenAM* (*Open Access Manager*), dan *JOSSO* (*Java Open Single Sign-On*).

### *2.2 Central Authentication Service (CAS)*

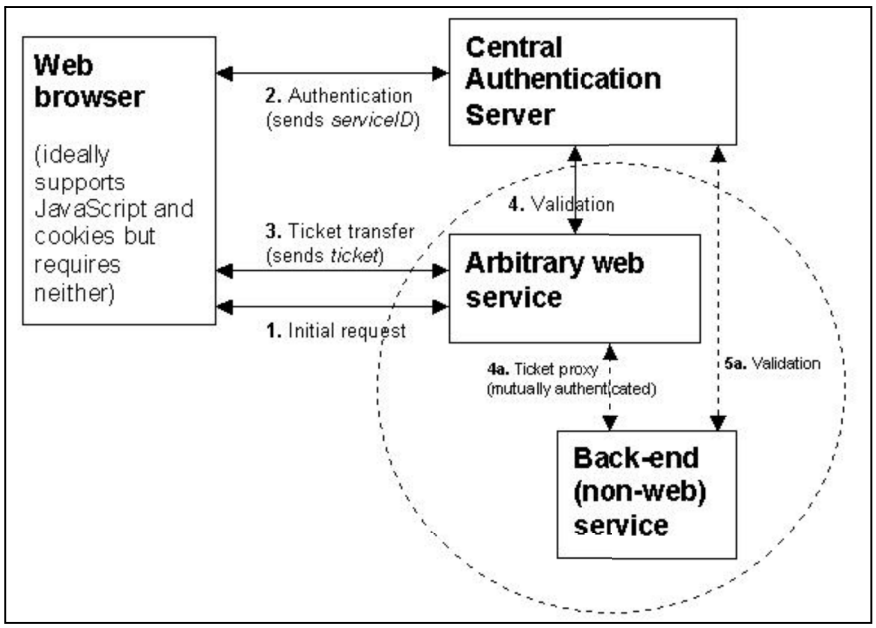
*CAS* adalah merupakan sebuah sistem autentikasi yang aslinya dibuat oleh Universitas Yale untuk menyediakan sebuah jalan yang aman untuk sebuah aplikasi untuk meng-autentikasi seorang user. *CAS* kemudian diimplementasi sebagai sebuah *open source* komponen server Java dan mendukung *library* dari

*client* untuk *Java, PHP, Perl, Apache, uPortal*, dan lainnya. *CAS* server sebagai sebuah dasar untuk beberapa *framework* untuk keamanan dan solusi *SSO* (Kristian Aaslund et al, 2007).

Dalam tahap pembuatannya, *CAS* memiliki beberapa fitur dasar layanan, diantaranya adalah :

1. Untuk memfasilitasi *Single Sign On* untuk berbagai aplikasi *web*. Sebagai sebuah servis inti, *CAS* tidak memerlukan *web-based* tetapi memiliki *front-end web*.
2. Memungkinkan layanan yang tidak memiliki akses ke suatu *organization* selain ITS (servis yang memiliki akses) untuk mengautentikasi user tanpa memiliki akses pada password-nya.
3. Untuk mempermudah prosedur pada aplikasi untuk melakukan autentikasi.
4. Untuk membatasi autentikasi menjadi hanya pada satu aplikasi web yang utama, yang mempermudah user untuk menjaga keamanan password-nya dan mengijinkan aplikasi yang dipercaya untuk mengubah logika autentikasinya jika diperlukan perlu, tanpa harus mengubah banyak aplikasi.

*Central Authentication Servise (CAS)* merupakan aplikasi web yang berdiri sendiri. Untuk lebih jelasnya dapat dilihat pada gambar dibawah ini:




Gambar 2.3. Arsitektur *CAS* (sumber : Rudy : 2009)

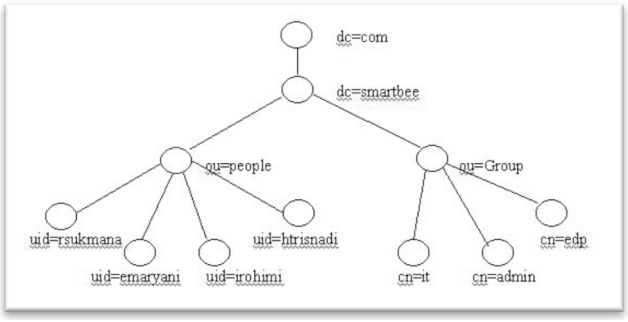
*URL login* menangani autentikasi yang utama.Karena itu *CAS* memaksa user dengan sebuah *NetID* dan password divalidasi ketika melakukan autentikasi.Pengembang *CAS* yang berbeda menggunakan *PasswordHandler* untuk memvalidasi *username* dan *password* melawan untuk menselarakan autentikasi.

Untuk mencegah kemungkinan dari pengulangan autentikasi, *CAS* juga mengirimkan user dan password dalam *memory cookie* (dihapus ketika browser ditutup).*Cookie* ini, bisa disebut "*ticket-granting cookie*", yang akan mengidentifikasi user setelah user melakukan satu kali *logged in*.

## 2.3 *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 2.4. Konsep Direktori

Secara prinsip struktur *database* pada suatu *directory* aplikasiadalah hierarki seperti yang di tunjukkan pada gambar 2. Suatu *directory* aplikasi akan memiliki item yang di jadikan sebagai *root*. Untuk sebuah titik *root*, secara umum di tunjukkan dengan suatu *attributedc (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 *userrsukmana* 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*.

2. Kelas Objek

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

3. 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 *clientserver*. Saat *client* terkoneksi ke *server* dan mengajukan *LDAPrequest*, *server* merespon dengan jawaban dan/atau dengan pointer ke arah mana *client* dapat mendapat tambahan informasi (khususnya ke *serverLDAP* yang lain). Tidak masalah *serverLDAP* yang mana yang merespon *request* dari *client*, karena *client* tersebut akan mendapat informasi yang sama. (http://www.scribd.com/doc/56286070/stuktur-*LDAP)*

## 2.4. Moodle

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

Menurut Amiroh (2012:1) *Moodle* merupakan program *open source* yang paling terkenal diantara program-program *e-learning* yang ada, misalnya ATutor, eLeaP$^{TM}$*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.

### 2.4.1 Kelebihan 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 : 2012).

Beberapa aktivitas pembelajaran yang didukung oleh *moodle* adalah sebagai berikut :

1. ***Assignment***

   Fasilitas ini ditugaskan untuk memberikan penugasan kepada peserta pembelajaran secara online.Peserta pembelajaran dapat mengakses materi tugas dan mengumpulkan tugas dengan cara mengirimkan file hasil pembelajarn mereka.

2. ***Chat***

Fasilitas ini digukan oleh pengajar dan peserta pembelajaran untuk saling berinteraksi secara online dengan cara berdialog teks (percakapan online).

3. ***Forum***

Merupakan forum diskusi secara online antara pengajar dan peserta pembelajaran yang membahas topic-topik yang berhubungan dengan materi pembelajaran.

4. ***Quiz***

Fasilitas iini digunakan oleh pengajar untuk melakukan ujian atau tes secara online (online test).

5. ***Survey***

Fasilitas ini digunakan untuk melakukan jajak pendapat. (Amiroh : 2012).

## 2.5. *Zimbra*

### 2.5.1 *Zimbra Collaboration Suite (ZCS)*

Menurut Tuxkeren (2012:3) *ZimbraCollaboration 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 *ZimbraNetwork Edition*, dan *ZimbraAppliance* yang bersifat virtualisasi serta sebuah lisensi lagi yang bersifat *open sources*.

*Zimbrawebclient* adalah 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 *internetbrowser*, juga ditambah lagi dengan*addons* yang dinamakan *Zimlet* sehingga membuat *Zimbra* menjadi kaya fitur (Tuxkeren : 2012).

Sementara untuk *Zimbraserver*, menggunakan teknologi dari beberapa aplikasi *open sources* yang sudah matang dan stabil, yang meliputi aplikasi *MTA, POP3*, *LDAP*, *MySQL* dan lain-lain. *Zimbraserver* 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 "*IZimbra*" yang dibawakan *group* band *Talking Heads*.

### 2.5.2 Versi-versi Zimbra

Zimbra secara keseluruhan mempunyai tiga versi untuk server, dan dua versi untuk client. Versi Zimbra untuk server adalah sebagai berikut:

1. Zimbra Network Edition
2. Zimbra Open Sources Edition
3. Zimbra Appliance

Sedangkan Zimbra untuk client ada dua, yaitu:

1. Zimbra Web Access
2. dan Zimbra Desktop.

**2.6. Wordpress**

Wordpress adalah sebuah aplikasi *open source*yang sangat popular digunakan sebagai mesin blog (*blog engine*).Wordprees dibangun dengan bahasa pemrograman PHP dan basis data (database) MYSQL.PHP dan MYSQL, keduanya merupakan perangkat lunak terbuka (*open source software*).Selain sebagai blog, wordpress juga mulai digunakan sebagai sebuah CMS (*Content Management System*) karena kemampuannya untuk dimodifikasi dan disesuaikan dengan kebutuhan penggunanya.Wordpress diusulkan oleh Christine Selleck, teman Matt Mullenweg. Wordpress saat ini menjadi *Platform content management system* (CMS) bagi beberapa situs web ternama seperti CNN, Reuters, The New York Times, TechCrunch, dan lainnya. (http://id.wikipedia.org/wiki/WordPress).

Rilis terbaru Wordpress adalah versi 3.5.2 (21 juni 2013).Wordprees didistribusikan dengan Lisensi Publik Umum GNU.

Sejarah wordpress dimulai saat Matt Mullenweg yang merupakan pengguna aktif dari b2 mengetahui bahwa proses pengembangan b2 dihentikan oleh pemrogramnya (*programmer*) yang bernama Michel Valdrighi, Matt Mullenweg merasa saying dan mulai melanjutkan pengembangan b2. (http://id.wikipedia.org/wiki/WordPress).

Wordpress muncul pertama kali di tahun 2003 hasil kerja keras Matt Mullenweg dengan Mike Little. Yang membuat wordpress makin terkenal, selain karena banyaknya fitur dan tampilan yang menarik adalah karena dukungan komunitas terhadap perangkat lunak terbuka (*open source*) untuk blog. (http://id.wikipedia.org/wiki/WordPress).

### 2.6.1 Wordpress.com dan Wordpress.org

Wordpress menyediakan dua alamat yang berbeda, yaitu Wordpress.com dan Wordpress.org.

1. **Wordpress.com**

   Merupakan situs layanan blog yang menggunakan mesin wordprees, didirikan oleh perusahaan Automattic. Dengan mendaftar pada situs Wordpress.com pengguna tidak perlu melakukan instalasi atau konfigurasi yang cukup sulit. Sayangnya, pengguna wordpress.com tidak dapat mengubah template standar yang sudah disediakan. Meski demikian, fitur yang disediakn oleh wordpress.com sudah cukup bagus.

2. **Wordpress.org**

Merupakan wilayah pengembang (*developer*).Dialamat ini, seseorang dapat mengunduh (*download*) aplikasi serta seluruh berkas CMS wordpress. Selanjutnya CMS ini dapat diubah ulang selama seseorang menguasai PHP, CSS dan skrip lain yang menyertainya. Wordpress dengan Bahasa Indonesia ada berkat kerja para contributor di Indonesia yang dipimpin oleh Huda Tori, seorang Mahasiswa Kedokteran dari Universitas Diponegoro (UNDIP) Semarang.

(http://id.wikipedia.org/wiki/WordPress).

## *2.7. Flowchart*

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).

### 2.7.1 Sistem *Flowchart*

Sistem *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 Pedoman-pedoman Dalam Membuat *Flowchart*

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

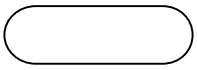
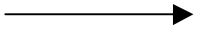
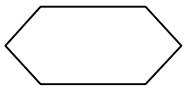
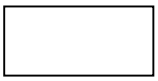
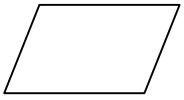
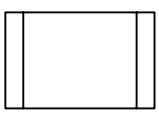
1. *Flowchart* digambarkan dari halaman atas ke bawah dan dari kiri ke kanan.
2. Aktivitas yang digambarkan harus didefinisikan secara hati-hati dan definisi ini harus dapat dimengerti oleh pembacanya.
3. Kapan aktivitas dimulai dan berakhir harus ditentukan secara jelas.
4. Setiap langkah dari aktivitas 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.

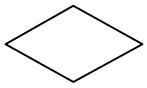
(Anharku, 2009:1)

### 2.7.3 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.

Tabel 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. |
| 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. |

| No | Simbol | Keterangan Fungsi |
|---|---|---|
| 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)

# BAB 3
# TUJUAN DAN MANFAAT PENELITIAN

## 3.1 Tujuan Penelitian

Penelitian ini bertujuan untuk :

1. Merencanakan pengembangan integrasi seluruh sistem yang ada dengan sistem *Single Sign On* untuk autentifikasi pengguna.
2. Memetakan seluruh sistem yang ada di Universitas dan kendala integrasi sistem tersebut ke *SSO*.
3. Mendesain hirarki direktori *LDAP* pengguna yang bisa diterapkan di Universitas.
4. Mendesain teknologi *SSO* yang aman yang bisa diterapkan di Universitas.

## 3.2 Manfaat Penelitian

Manfaat yang didapat dari penelitian ini yaitu :

1. Penelitian ini menghasilkan *prototype SSO* Universitas yang bisa diimplementasikan.
2. Selain itu juga dengan penelitian ini bisa diketahui berbagai kendala aplikasi yang ada di Universitas untuk mengimplementasikan *SSO*.

# BAB 4
# METODE PENELITIAN

Penelitian ini merupakan penelitian tindakan *(action research)* yang diimplementasikan dengan mengikuti situasi aktual yang ada di Universitas. Pemilihan metoda ini dikarenakan *action research* bersifat praktis dan bisa langsung diimplemetasikan. Kerangka kerja yang diikuti untuk mendapatkan pemecahan masalah adalah sebagai berikut :

1. Melakukan diagnosa (*Diagnosing*).

   Pada tahapan ini dilakukan identifikasi masalah-masalah pokok yang ada. Serta membuat hipotesa awal ; "bisakah dikembangkan *prototype SSO* yang aman di Universitas"

2. Membuat rencana tindakan (*Action Planning*)

   Pada tahapan ini kita memahami pokok masalah dengan melakukan studi pustaka yang berhubungan dengan teknologi *SSO* serta kelebihan dan kekurangannya yang ada, serta menyusun rencana tindakan yang tepat untuk menyelesaikan masalah yang ada.

3. Melakukan tindakan (*Action Taking*)

   Padatahapaninikitamengimplementasikanrencanatindakandenganharapandapatmenyelesaikanmasalah.Padatahapaniniakandibuat perancangan (desain) dari sistem *SSO* yang akan diterapkan di Universitas, yaitu berupa:

   a) Pengembangan struktur hirarki *LDAP* yang bisa diimplementasikan untuk sistem *SSO*.
   b) Perancangan topologi *SSO* yang akan diterapkan.
   c) Instalasi server LDAP
   d) Instalasi server SSO
   e) Instalasi Server yang akan diujicobakan
   f) Konfigurasi Server dengan LDAP
   g) Konfigurasi Server dengan SSO

4. Melakukan evaluasi (*Evaluating*)

   Pada tahapan ini kita evaluasi hasil dari implementasi. Setelah dirancang dan dibuat *prototype SSO* dilakukan evaluasi untuk melihat kestabilan sistem dan keamanan *SSO* yang dibuat.

5. Pembelajaran (*Learning*)

   Pada tahap ini kita melakukan review tahapan-tahapan yang telah berakhir dan mempelajari kriteria dalam prinsip pembelajaran. Dari hasil evaluasi bisa didapatkan kelebihan dan kekurangan *prototype* sistem *SSO* yang dikembangkan.

## 4.1. Rancangan Penelitian

### 4.1.1 Waktu dan Tempat

Penelitian akan dilakukan selama 8 (selapan) bulan dari bulan Maret 2013 s/d November 2013 bertempat di laboratorium Foresec Universitas Bina Darma.

### 4.1.2. Alat dan Bahan

Peralatan yang dibutuhkan untuk pengembangan sistem meliputi komponen *hardware*, beberapa perangkat komputer yang digunakan sebagai *serverSSO*, server DNS, server *LDAP* dan beberapa server aplikasi seperti *elearning, mail, blog, server hotspot, server digilib*, CMS, dan lain-lain, 1 buah *access point* dan 1 komputer *client* yang terkoneksi melalui jaringan. Beberapa *software* yaitu Sistem Operasi Linux Server, Ubuntu, Windows, dan lain-lain.. Tools Freeradius untuk radius server, database mysql, tools web server apache, *LDAP*, ZIMBRA, moodle, wordpress, CMS, dan lain-lain.

## 4.2. Prosedur Penelitian

Prosedur Penelitian menggunakan *Network Development Life Cycle* yang tahapannya sebagai berikut*:*

1. Analisis
2. Desain

3. Simulasi *Prototype*
4. Implementasi
5. Monitoring
6. Management.

Untuk penelitian ini hanya diterapkan sampai dengan tahapan ke-3 (Simulasi *Prototype*)

### 4.2.1 Analisa Kebutuhan Sistem

Pada tahapan ini melalukan pemetaan layanan-layanan yang ada di Universitas, platform pengembangannya serta kemungkinannya untuk diintegrasikan dengan *SSO* dengan mempelajari platform sistem itu sendiri dan literatur terkait.

### 4.2.2. Desain Perancangan Sistem

Berisi perancangan (desain) dari sistem *SSO* yang akan diterapkan di Universitas.

1. Pengembangan struktur hirarki *LDAP* yang bisa diimplementasikan untuk sistem *SSO.*
2. Perancangan topologi *SSO* yang akan diterapkan.
3. Perancangan alur proses*SSO.*

### 4.2.3. Simulasi *Prototype*

Pada tahapan ini *prototype SSO* yang dirancang/didesain akan disimulasikan mendekati sistem yang ada di Universitas.

## 4.3. Kriteria Evaluasi dan Teknik Pengukuran

Di dalam penelitian yang menjadi kriteria evaluasi adalah kestabilan dan keamanan teknologi SSO yang dibuat. Selain itu juga evaluasi dilakukan dengan melakukan survei ke calon pengguna jika teknologi SSO ini diterapkan di Universitas.

# BAB 5
# HASIL DAN PEMBAHASAN

## 5.1. Analisis Sistem

Tujuan dalam analisa sistem yang berjalan ini adalah untuk mendapatkan informasi tentang sistem atau aplikasi apa saja yang biasanya digunakan di suatu Universitas dan kemungkinan sistem atau aplikasi tersebut diintegrasikan menggunakan teknologi *SSO*.

Berdasarkan hasil wawancara dengan UPT SIM Universitas Bina Darma tempat diadakannya penelitian, ada beberapa sistem atau aplikasi yang digunakan di Universitas tersebut antara lain:

**Tabel 5.1 Sistem/Aplikasi yang Berjalan di Universitas**

| NO | Jenis Aplikasi | Platform Pengembangan | Platform server |
|---|---|---|---|
| 1 | Sistem Informasi Akademis | bahasa pemrograman PHP, Java, dan database MySQL | Linux |
| 2 | Sistem *elearning* | bahasa pemrograman PHP dan database MySQL | Linux |
| 3 | Sistem email | Zimbra | Linux |
| 4 | Sistem blog | PHP, MySQL dan Wordpress | Linux |
| 5 | Sistem HRIS | PHP, MySQL dan CMS | Linux |
| 6 | Sistem Digilib | PHP dan MySQL, GDLN versi 4 | Linux |
| 7 | Sistem Otomasi Perpustakaan | PHP dan MySQL | Linux |
| 8 | Website Fakultas | PHP dan MySQL | Linux |
| 9 | Website Program Studi | PHP dan MySQL | Linux |
| 10 | Sistem intranet akademis | PHP dan MySQL | Linux |
| 11 | Sistem intranet keuangan | PHP dan MySQL | Linux |
| 12 | Sistem Autentikasi Hotspot | freeradius, PHP, MySQL | Linux |
| 13 | Sistem eprint | PHP dan MySQL, | Linux |
| 14 | website unit kerja | PHP & MySQL | Linux |

### 5.1.1 Pemetaan Sistem yang Berjalan dengan Teknologi *LDAP*

Dari berbagai sistem aplikasi yang berjalan di Universitas dipelajari kemungkinan untuk diintegrasikan menggunakan *LDAP*. Dari berbagai literatur dipelajari kemungkinan aplikasi tersebut menggunakan *LDAP* untuk autentikasi login.

**Tabel 5.2 Pemetaan Aplikasi di Universitas dengan teknologi *LDAP***

| NO | JENIS APLIKASI | MEKANISME LOGIN | SUPPORT *LDAP* | Keterangan |
|---|---|---|---|---|
| 1 | Sistem Informasi Akademis | Halaman login berbasis PHP | Bisa | perlu modifikasi string login diarahkan ke *LDAP* dengan menambah library *LDAP* |
| 2 | Sistem *elearning* | Halaman login berbasis PHP | Bisa | |
| 3 | Sistem email | halaman login Zimbra | Bisa | |
| 4 | Sistem blog | wordpress | Bisa | |
| 5 | Sistem HRIS | Halaman login berbasis PHP | Bisa | |
| 6 | Sistem Digilib | Halaman login berbasis PHP | Bisa | |
| 7 | Sistem Otomasi Perpustakaan | Halaman login berbasis PHP | Bisa | |
| 8 | Website Fakultas | Halaman login berbasis PHP | Bisa | |
| 9 | Website Program Studi | Halaman login berbasis PHP | Bisa | |
| 10 | Sistem intranet akademis | Halaman login berbasis PHP | Bisa | |
| 11 | Sistem intranet keuangan | Halaman login berbasis PHP | Bisa | |
| 12 | Sistem Autentikasi Hotspot | Halaman login berbasis PHP | Bisa | |
| 13 | Sistem eprint | Halaman login berbasis PHP | Bisa | |
| 14 | website unit kerja | Halaman login berbasis PHP | Bisa | |

### 5.1.2 Pemetaan Sistem yang Berjalan dengan penerapan Teknologi *CAS SSO*

Dari aplikasi yang berjalan di Universitas berdasarkan platform dipelajari kemungkinan penerapan teknologi *SSO* berbasis *CAS*. Hasilnya sebagai berikut:

**Tabel 5.3 Pemetaan Aplikasi di Universitas dengan teknologi *CAS SSO***

| NO | JENIS APLIKASI | MEKANISME LOGIN | SUPPORT *CAS* | Keterangan |
|---|---|---|---|---|
| 1 | Sistem *elearning* | Halaman login berbasis PHP | Bisa | *moodle* suport library *CAS* |
| 2 | Sistem email | halaman login Zimbra | Bisa | Zimbra suport library *CAS* |
| 3 | Sistem blog | wordpress | Bisa | wordpress suport library *CAS* |
| 4 | Sistem Informasi Akademis | Halaman login berbasis PHP-MySQL | library php*CAS* | untuk aplikasi berbasis PHP memungkinkan untuk mendirect halaman login ke Server *CAS* akan tetapi perlu modifikasi script untuk mengaktifkan library php*CAS* yang |
| 5 | Sistem Digilib | | | |
| 6 | Sistem Otomasi Perpustakaan | | | |

| | | | | |
|---|---|---|---|---|
| 7 | Website Fakultas | | | perlu dilakukan pengujian aplikasi. Selain itu juga perlu diuji kompatibilitas library dengan versi PHP, versi webserver dan lain-lain. Untuk aplikasi web yang halaman loginnya dikembangkan berbasis CMS, beberapa CMS support library php*CAS* antara lain Joomla, drupal, dan memungkinkan juga dilakukan sistem autentikasi dengan merubah script login |
| 8 | Website Program Studi | | | |
| 9 | Sistem intranet akademis | | | |
| 10 | Sistem intranet keuangan | | | |
| 11 | Sistem Autentikasi Hotspot | | | |
| 12 | Sistem eprint | | | |
| 13 | website unit kerja | | | |
| 14 | sistem HRIS | | | |

Dari tabel 5.3 bisa dilihat penerapan *SSO CAS* di berbagai aplikasi sangat memungkinkan terutama untuk aplikasi yang pernah diujicobakan berdasarkan referensi di https://wiki.jasig.org/display/*CAS*/*CAS*ifying+Applications. Untuk aplikasi dengan platform zimbra, *moodle*, wordpress, *elearning* dan joomla sangat mungkin untuk menggunakan teknologi *SSO* berbasis *CAS*Ify. Demikian juga untuk aplikasi berbasius RADIUS pada referensi yang dipelajari juga sangat memungkinkan untuk menggunakan teknologi *SSO* berbasis *CAS* (https://wiki.jasig.org/display/*CAS*UM/RADIUS). Demikian juga untuk aplikasi lain, memungkinkan penggunaan teknologi *SSO* berbasis *CAS* dengan melakukan modifikasi script dan menambahkan library *CAS* seperti php*CAS* pada script berbasis PHP, *CAS* JSP untuk aplikasi berbasis Java, ASP.NET, CpldFusion, Perl Client, VBScript, Symfony dan lain-lain. Akan tetapi pada penelitian ini karea keterbatasan waktu hanya membahas pengujian implmentasi pada beberapa aplikasi saja yaitu *elearning* berbasis *moodle*, mailserver berbasis zimbra dan blog berbasis wordpress. Untuk aplikasi lain terutama yang codingnya dibuat oleh developer tertentu tidak diujicobakan karena coding dari aplikasi tersebut yang bersifat closesource.

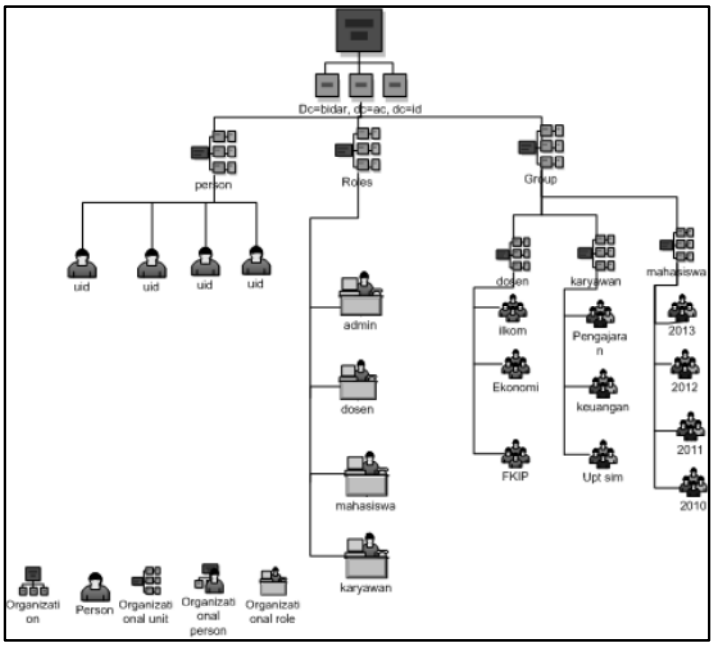
## 5.2. Desain Perancangan Sistem

Ada 3 hal yang dirancang dan didesain perancangan (desain) dari sistem *SSO* yang akan diterapkan di Universitas, yaitu struktur hirarki *LDAP* yang akan diterapkan, topologi *SSO* yang akan diuji cobakan dan perancangan alur proses *SSO*.

### 5.2.1 Pengembangan struktur hirarki *LDAP* yang bisa diimplementasikan untuk sistem *SSO*

*LDAP* (Ligweight Directory Access Protocol) merupakan sebuahprotokol yang mengatur mekanisme pengaksesanlayanan direktori (Directory Service). Protokol ini yang dimanfaatkan pada sistem *SSO* untuk pengelolaan pengguna semua layanan/aplikasi yang ada di Universitas.

Jika dilihat hasil pemetaan sistem aplikasi yang ada di Universitas, semua aplikasi tersebut memungkinkan untuk dimigrasikan menggunakan sistem autentikasi berbasis *LDAP*. Terutama untuk aplikasi/sistem yang menggunakan berbasis CMS. Pada penelitian ini penggunaan *LDAP* sudah diuji cobakan pada sistem autentikasi wireless berbasis radius, sistem *elearning* berbasis *moodle*, sistem berbasis CMS Wordpress, berbasis Joomla, sedangkan untuk aplikasi yang dikembangkan sendiri perlu melakukan perubahan script untuk halaman login agar bisa menggunakan *LDAP* (Timothy, 2003).



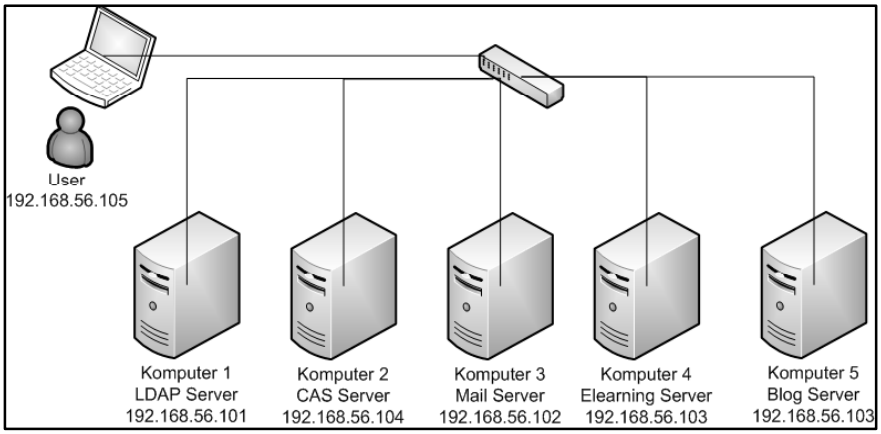

**Gambar 5.1. Struktur Hirarki *LDAP* yang dirancang**

Perancangan struktur hirarki *LDAP* menyesuaikan hirarki organisasi yang umumnya ada di Universitas. Di Universitas umumnya terdapat tiga kategori user

(pengguna) yaitu dosen, mahasiswa dan karyawan. Selain itu juga ada pembagian mahasiswa berdasarkan tahun anggkatan, pembagian mahasiswa berdasarkan program studi. Untuk dosen pengelompokan biasanya berdasarkan fakultas dan program studi, sedangkan untuk karyawan pengelompokannya berdasarkan unit kerja. Pengelompokan pengguna ini dimaksudkan untuk pengklasifikasian user untuk penggunaan aplikasi/ sistem. Karena biasanya beberapa aplikasi tidak ditujukan untuk semua pengguna, misalnya email yang hanya untuk dosen dan karyawan, blog yang hanya untuk untuk dosen dan lainnya.

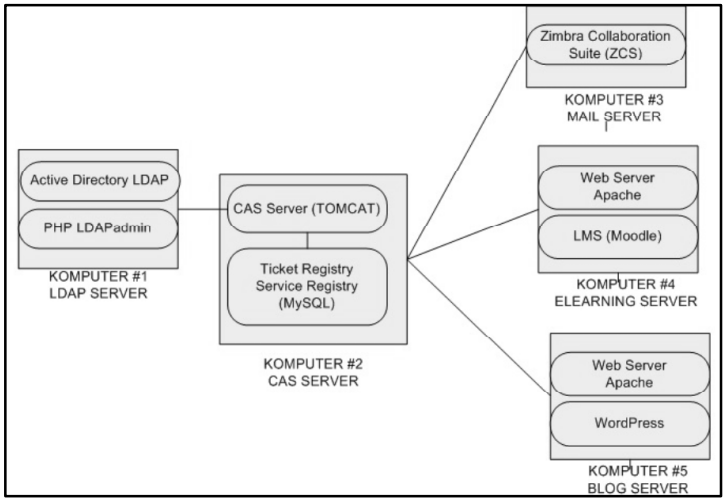
### 5.2.2. Perancangan topologi *SSO* yang akan diterapkan

Topologi *SSO CAS* yang diterapkan pada penelitian adalah sebagai berikut:




**Gambar 5.2. Topologi Jaringan Teknologi *SSO- LDAP* yang dirancang**

Pada penelitian ini menggunakan lima buah server, dan 1 pc notebook sebagai client. Server yang dibangun terdiri dari server *SSO CAS* untukLayanan *single sign on* sangat membantu pengguna untuk mengakses banyak layanan tanpa repot-repot memasukkan nama dan kata sandi lagi. Server yang akan diintegrasikan dengan server *CAS SSO* ini terdiri dari 3 server yaitu server *elearning* yang berbasis *moodle*, server mail server berbasis *ZCS* (*Zimbra Collaboration Suite*) dan server blog berbasis Wordpress.

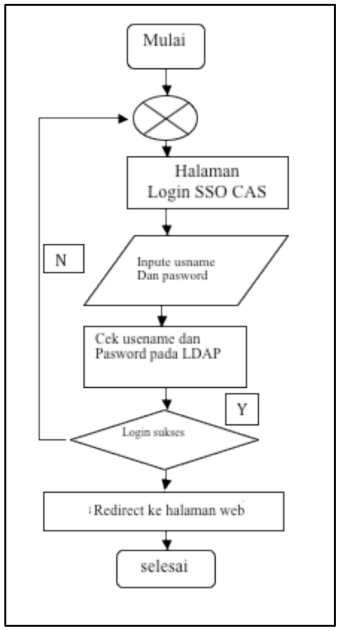



**Gambar 5.3. *Topologi Logic* Teknologi *SSO- LDAP* yang dirancang**

*Topogi logic* dari desain teknologi *LDAP* yang dirancang ini bisa dilihat dari gambar 5.3. Pada Komputer *SSO* didalamnya terdapat *CAS (Central Authentication Service)* berbasis server Tomcat yang menangani proses Single Sign Out. Selain itu di dalam server *SSO* terdapat juga Ticket Registry Service yang menangani proses ticket registry untuk mengatur sesi login user. Service registry dibutuhkan untuk menyimpan informasi aplikasi apa saja yang diizinkan untuk menggunakan infrastruktur *SSO*, demikian pula user attributes bisa diberikan untuk setiap aplikasi. Kedua *ticket registry* dan *service registry* disimpan di database MySQL yang disimpan di server *CAS* server. *LDAP* Server berfungsi untuk menyimpan dan memanajemen user yang menggunakan aplikasi server.

### 5.2.3. Perancangan Alur proses *SSO*

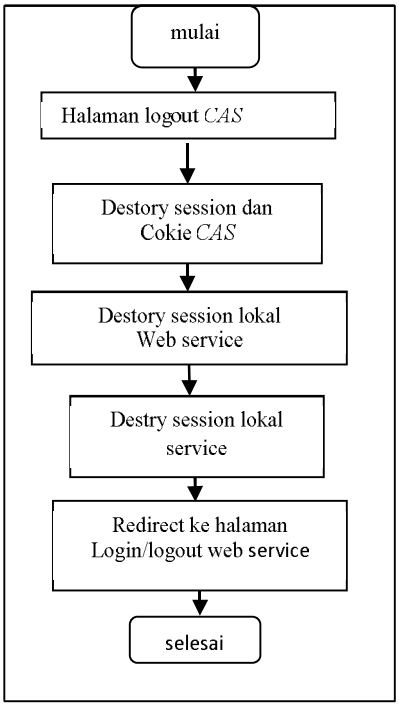
Perancangan alur proses *SSO* bisa dilihat pada gambar 5.4.




**Gambar 5.4. Perancangan alur proses login *SSO***

Gambar diatas adalah proses dari *login* pengguna agar dapat masuk kedalam portal *web*. Dimana pengguna masuk ke halaman *login CAS*, kemudian melakukan *login* dengan memasukan *username* dan *password* lalu dilakukan pengecekan *username* dan *password* pada *LDAP*. Jika *login* sukses maka proses selesai, jika tidak maka akan kembali kehalaman *login*. Selanjutnya adalah gambaran dari proses *logout*menggunakan *flowchart*, sebagai berikut.

Gambar 5.5 adalah proses *logout* pengguna dimana pengguna menuju kehalaman *logout* dimana pada proses ini dilakukan penghancuran *session* dan *cookie* dari *local web service* kemudian *local service* dan dikembalikan ke halaman *login CAS*.




**Gambar 5.5. Diagram alir proses *logout* aplikasi *CAS* server**

### 5.3. Pengembangan Prototype *SSO*

#### 5.3.1. Instalasi server *LDAP*

Pada penelitian ini menggunakan Server Ubuntu 12.10, cara instalasi dan konfigurasi Open*LDAP* server bisa di lihat di lampiran 1A. Cara instal paket open*ldap* dengan menjalankan perintah sebagai berikut:

```
sudo apt-get install ldap-utils ldapscripts slapd
ls /etc/ldap/schema/*.ldif | xargs -I {} sudo ldapadd -Y EXTERNAL -H ldapi:/// -f {}
```

Perintah baris kedua adalah untuk membuat membuat schema *LDAP*. Buat file konfigurasi dasar diberi nama **backend.example.com.ldif** seperti berikut:

```
# Load dynamic backend modules
dn: cn=module,cn=config
objectClass: olcModuleList
cn: module
#olcModulepath: /usr/lib/ldap
#olcModuleload: back_hdb

# Database settings
dn: olcDatabase=hdb,cn=config
objectClass: olcDatabaseConfig
objectClass: olcHdbConfig
olcDatabase: {1}hdb
olcSuffix: dc=bidar,dc=ac,dc=id
```

```
olcDbDirectory: /var/lib/ldap
olcRootDN: cn=admin,dc=bidar,dc=ac,dc=id
olcRootPW: xxxxxx
olcDbConfig: set_cachesize 0 2097152 0
olcDbConfig: set_lk_max_objects 1500
olcDbConfig: set_lk_max_locks 1500
olcDbConfig: set_lk_max_lockers 1500
olcDbIndex: objectClass eq
olcLastMod: TRUE
olcDbCheckpoint: 512 30
olcAccess: to attrs=userPassword by
dn="cn=admin,dc=bidar,dc=ac,dc=id" write by anonymous auth by self
write by * none
olcAccess: to attrs=shadowLastChange by self write by * read
olcAccess: to dn.base="" by * read
olcAccess: to * by dn="cn=admin,dc=bidar,dc=ac,dc=id" write by *
read
```

Langkah selanjutnya file **backend.example.com.ldif** ditambahkan ke server *LDAP* dengan perintah berikut :

```
sudo ldapadd -Y EXTERNAL -H ldapi:/// -f ~/backend.example.com.ldif
```

Selanjutnya perlu dibuat top-level object dalam domain, admin user, dan grup, secara detail bisa dilihat di lampiran 1A.

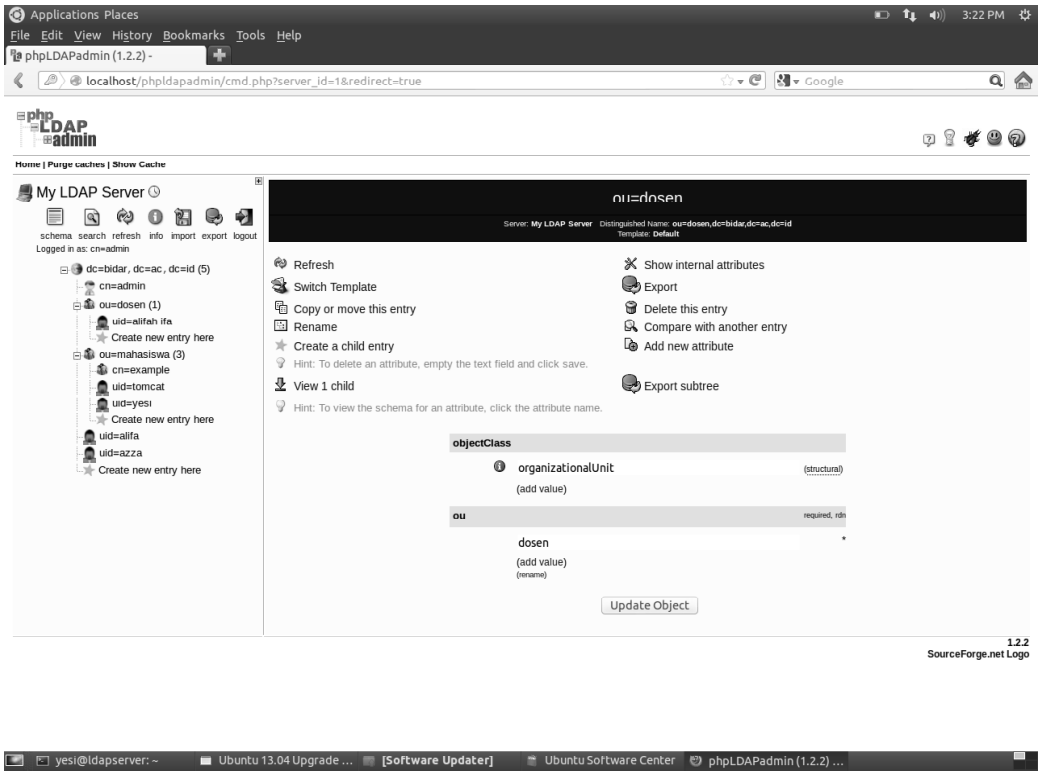
Untuk mempermudah penambahan user, group dan sebagainya maka diinstal php*ldap*admin versi 1.2.2, dengan menjalankan perintah berikut:

```
sudo apt-get install phpldapadmin

sudo ln -s /usr/share/phpldapadmin/ /var/www/phpldapadmin
```

Perintah pada baris ke dua untuk membuat symbolic link, dirktori php*ldap*admin. Kemudian yang perlu diedit adalah file /etc/php*ldap*admin/config.php sebagai berikut:

```
$servers = new Datastore();
$servers->newServer('ldap_pla');
$servers->setValue('server','name','My LDAP Server');
$servers->setValue('server','host','127.0.0.1');
$servers->setValue('server','base',array('dc=bidar,dc=ac,dc=id'));
$servers-
>setValue('login','bind_id','cn=admin,dc=bidar,dc=ac,dc=id');
```

Setelah direstart apache dan membuka port 80 dan port 389 di firewall, maka php*ldap*admin bisa dijalankan seperti berikut:




**Gambar 5.6. *phpldapadmin* dan *OpenLDAP* yang Sudah diinstal**

### 5.3.2. Instalasi server *SSO*

Proses instalasi dan konfigurasi server *SSO* ini merupakan proses yang paling penting pada tahapan penelitian ini. Pada penelitian ini server *SSO* diujicobakan pada Server Ubuntu 12.10 dan Server berbasis Windows XP. Masalah utama pada proses instalasi server *SSO* berbasis *CAS* ini adalah masalah sertifikat *SSL* yang dibuat sendiri yang tidak dikenali pada server *SSO* maupun server aplikasi yang menggunakan *SSO* sehingga proses komunikasi aplikasi server sebagai client dengan server *SSO* tidak berhasil. Untuk itu perlu dilakukan proses import *certificate* yang dibuat ke masing-masing server, dan karena proses pembuatan dan import *certificate* yang lebih mudah di sistem Operasi Windows maka pada penelitian ini server *SSO* yang akhirnya digunakan adalah yang berbasis Windows XP. Adapun langkah/proses instalasi server *SSO* ini membutuhkan beberapa aplikasi antara lain Java JDK, Maven, Tomcat, dan MySQL. Proses instalasi masing-masing aplikasi tersebut sebagai berikut:

a) Instalasi JDK pada penelitian ini menggunakan JDK version 1.6.0_43. Adapun langkah-langkah instalasi JDK secara detail bisa lihat di lampiran 1B. Konfigurasi variable JAVA_HOME system environment diarahkan pada C:\Program Files\Java\jdk tempat Java diinstal, dan direktori java executables PATH environment variable diseting ke %JAVA_HOME%\bin.

b) Instalasi Maven adalah sebuah build automation tool untuk java. Maven merupakan aplikasi project management yang menggunakan file xml untuk menetukan ketergantungan suatu project dan repository di dalamnya. Semua file-file yang dibutuhkan di download ke local system menjadi bagian proses yang dikembangkan. Sebuah project membutuhkan maven jika terdapat sebuah file xml, untuk maven 1 biasanya bernama project.xml sedangkan untuk maven 2 bernama pom.xml berisi informasi sebuah projek dan detil konfigurasi maven. File pom.xml ini merupakan Project Object Model adalah sebuah unit fundamental kerja di maven.

   Maven bisa di download di http://maven.apache.org/download.html. Pada penelitian ini maven yang digunakan versi 2.2.1.

   Yang perlu dilakukan konfigurasi M2_HOME environment mengarah ke folder yang di install, dan direktori maven executables pada PATH %M2_HOME%\bin secara detail bisa dilihat di lampiran 1B.

   Maven overlay method merupakan cara yang bisa digunakan untuk membangun *CAS* server dengan melakukan perubahan konfigurasi lokal. Sebagai contoh untuk melakukan seting registry dan interface *LDAP* memerlukan beberapa file konfigurasi, dan maven membantu membangun file war yang di dalamnya sudah termasuk modifikasi file konfig yang dibuat. Yang perlu dilakukan hanya menginstal file war ini ke komputer dimana server *CAS* akan dibangun.

c) Instal Apache Tomcat. Apache Tomcat merupakah salah satu servlet/web container yang paling populer di lingkungan pemrograman web Java. Apache Tomcat berada di bawah naungan Apache Software Foundation yang di sana terdapat project-project open source lainnya. Tomcat bisa didownload di http://tomcat.apache.org/download-60.cgi. Pada penelitian ini menggunakan

Tomcat versi 6.0.37. File ini diekstrak pada direktori C. Tomcat diinstal pada server *SSO* server dan server aplikasi yang menggunakan Java. *CAS* selain menggunakan tomcat bisa juga dijalankan pada server JBoss, akan tetapi pada penelitian ini menggunakan tomcat dengan alasan (http://zeroturnaround.com/rebellabs/the-great-java-application-server-debate-with-tomcat-jboss-glassfish-jetty-and-liberty-profile/) Tomcat lebih cepat dijalankan dan lebih mudah dikelola.

Setelah tomcat diinstal, yang perlu dilakukan adalah mengedit C:\apache-tomcat-6.0.37\conf\tomcat-users.xml untuk menambahkan privileged user yang bisa mengakses tomcat console.

```
<tomcat-users>
        <role rolename="manager-gui"/>
        <role rolename="manager-script"/>
        <role rolename="manager-jmx"/>
        <user username="admin" password="tomcat" roles="manager-gui,manager-
script,manager-jmx"/>
</tomcat-users>
```

Konfigurasi ini penting agar bisa melakukan konfigurasi *CAS* pada server tomcat.

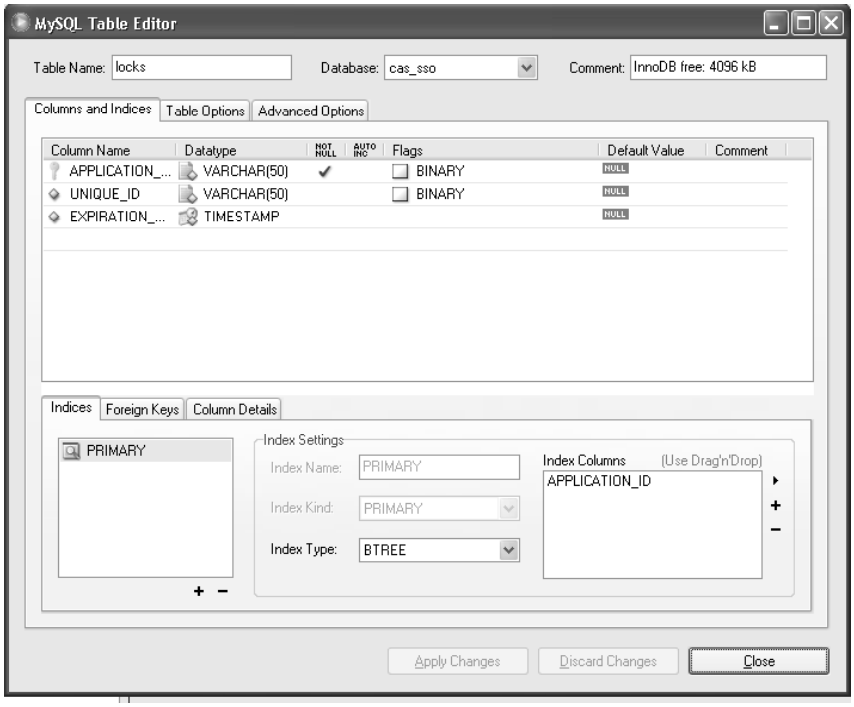
d) Proses instal MySQL yang pada penelitian digunakan sebagai registry ticket validator. MySQL bisa di download di http://www.mysql.com/downloads/. Pada penelitian ini menggunakan mysql versi 5.1.11 dan MySQL Administraor versi 1.2.17. MySQL ini diinstal di server *SSO* untuk mempermudah proses ticket registry dari server *CAS*.

Setelah diinstal pada server *SSO* server, langkah berikutnya membuat schema/database baru yang diberi nama *CAS_SSO*. Sebagian besar table yang ada pada database *CAS_SSO* akan dibuat secara otomatis oleh *CAS* saat system berjalan pertama kali kecuali untuk table lock, yang akan digunakan oleh registry cleaner.

Perintah berikut dijalankan pada MySQL Administrator untuk membuat table registry locks:

```
CREATE TABLE LOCKS (
  APPLICATION_ID VARCHAR(50) NOT NULL,
  UNIQUE_ID VARCHAR(50) NULL,
  EXPIRATION_DATE TIMESTAMP NULL );
ALTER TABLE LOCKS ADD CONSTRAINT LOCKS_PK PRIMARY KEY (APPLICATION_ID);
```

Seperti terlihat pada MySQL Administrator berikut:




**Gambar 5.7. Table locks yang dibuat pada database *SSO***

e) Instal Portecle X.509 adalah tools yang digunakan untuk membuat *certificate* dan melakukan manajemen *keystore* pada system operasi Windows. Menggunakan tools ini lebih mempermudah kita membuat *certificate* dibandingkan menggunakan Java keytool, terutama untuk proses ekspor dan import *certificate*. Portecle bisa didownload di http://sourceforge.net/projects/portecle dan kemudian diinstal pada komputer berbasis Windows (tidak harus pada server *SSO*). Pada penelitian menggunakan portecle versi 1.17.

Setelah proses install beberapa aplikasi tersebut tahapan selanjutnya adalah proses pembuatan X.509 *Certificate*s. *CAS* menggunakan SSL pada proses koneksi antara server *SSO* dan server yang diproteksi saat melakukan ticket validation, melakukan passing user attributes, dan lain-lain. SSL ini sebaiknya

digunakan pada semua proses koneksi antara client dan server yang proses autentikasi usernya berjalan di cookie.

*SSO* server dan semua server yang diproteksi sebaiknya masing-masing memiliki *certificate* server. Certicicate ini digunakan untuk melakukan proses koneksi yang dipercaya antara clients dan servers. Prinsip kerja trust relation antara server *SSO* dan server yang diproteksi sebagai berikut:

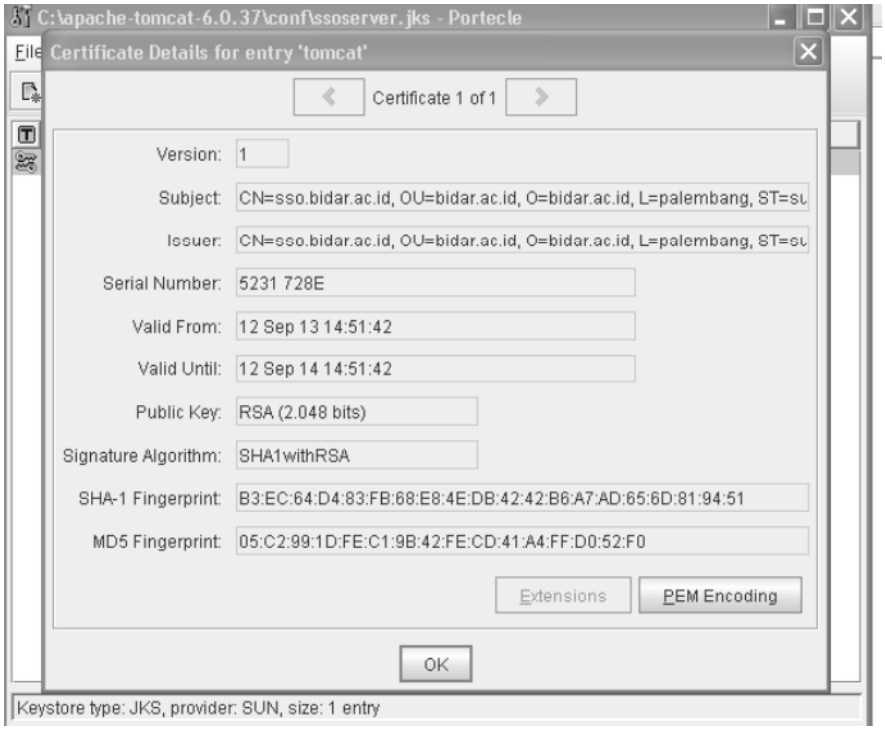
- Server *CAS* harus percaya pada server yang diproteksi, sehingga *certificate* server tersebut harus diinstal pada *SSO* server's trust store, misalnya pada JDK's cacerts *keystore*. Tanpa ini server tidak akan menerima permintaan validasi dari server lain.
- Server yang diproteksi juga harus percaya pada *CAS SSO* server, sehingga *certificate SSO* server juga harus diinstal pada masing-masing truststore server, misalnya pada JDK's cacerts *keystore*. Tanpa hal ini *SSO* client tidak akan menerima pesan *single sign-out*, dari server *CAS*.
- Semua server harus dikonfigurasi mengirim *certificate* masing-masing sebagai bagian dari SSL connection requests. Konfigurasi kersebut dilakukan pada file server.xml tomcat.

Pada penelitian ini menggunakan *self-signed certificate*. Pada proses implementasi yang sebenarnya sebaiknya menggunakan *certificate server* yang dikeluarkan oleh *local Certification Authority* atau menggunakan commercial CA. *Tools Portecle* digunakan untuk membuat *keystore* dan server *certificate* untuk *CAS SSO* server. *Keystore* ini diberi nama *keystore SSO*server.jks. Konfigurasi *certificate* yang dibuat bisa di lihat di gambar 5.8.

*Certificate* yang sudah dibuat ini disimpan sebagai file yang diberi nama *SSO*server.cer, dan akan diimpor ke server-server yang akan diproteksi oleh server *CAS SSO*. Proses pembuatan *certificate* ini diulangi untuk server yang akan diproteksi oleh *CAS*. Pada server *SSO*, *certificate* server yang akan diproteksi diimpor ke cacerts trust store, yang lokasinya ada di sini:

```
C:\Program Files\Java\jdk\jre\lib\security\cacerts
```

*SSO* server's *certificate* yang dibuat juga harus diimpor pada cacerts trust store masing-masing server.




**Gambar 5.8. *Certificate Self Signed* untuk *SSO* yang dibuat**

Langkah berikutnya adalah Konfigurasi Tomcat untuk Menerima Koneksi SSL.Pada server *SSO*, yang perlu dilakukan adalah mengedit file C:\apache-tomcat-6.0.37\conf\server.xml. Yang perlu dilakukan pada konfigurasi ini adalah membuang comment port 8443 connector dan menambahkan informasi *keystore certificate* pada local lomputer yang sudah dibuat sebelumnya, termasuk file path, password dan cert alias. Dengan konfigurasi ini memungkinkan server mengirim sertifikatnya ke client sebagai bagian dari koneksi SSL. Konfigurasi yang dilakukan pada penelitian ini adalah sebagai berikut:

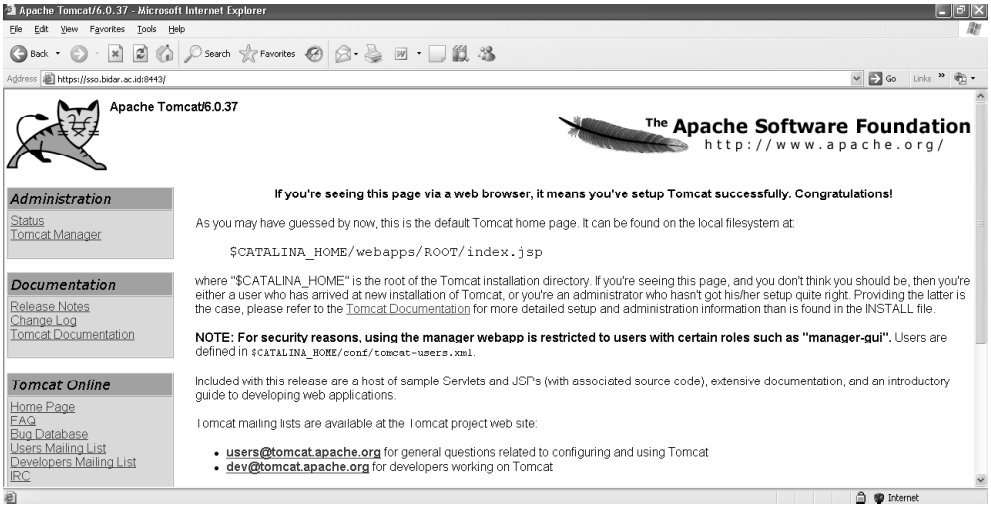
```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11Protocol"
     SSLEnabled="true"
     maxThreads="150" scheme="https" secure="true"
     keystoreFile="conf/SSOServer.jks" keyAlias="tomcat" keystorePass="changeit"
     clientAuth="false" sslProtocol="TLS" />
```

Dilakukan seting connector protocol menjadi org.apache.coyote.http11.Http11Protocol. *Keystore SSO*server.jks yang sudah dibuat

menggunakan portecle dikopikan ke C:\apache-tomcat-6.0.29\conf, dan *password* dan alias harus sama pada saat membuat *keystore* dan cert.

Untuk menguji konfigurasi tomcat di restart dan dipastikan tidak muncul errors yang terlihat di jendela console. Jika terjadi errors biasanya diakibatkan oleh file *keystore* yang tidak bisa ditemukan ataupun format file *keystore* yang tidak bisa dibuka oleh tomcat.

Tahap selanjutnya untuk memastikan konfigurasi dan sertifikat yang dibuat tidak bermasalah, dicoba membuka tomcat admin console menggunakan SSL URL, https://*SSO*.bidar.ac.id:8443/. Maka terlihat seperti berikut:




**Gambar 5.9. Server Tomcat yang sudah berjalan di SSL**

Setelah memastikan Tomcat sudah berjalan menggunakan SSL, berikutnya melakukan proses deploy *SSO* Server Menggunakan Maven Overlay Method. Proses deploy ini tahapannya sebagai berikut:

1. Membuat File Pom. File pom ini menentukan keterkaitan library yang akan digunakan saat mengkompile project menggunakan maven. Pada server *SSO* yang dikembangkan dibuat di direkori C:*CAS_SSO*\local-*CAS*. Di dalam direktori tersebut dibuat file pom.xml yang isinya lebih detail lihat di lampiran 1C.
2. Membuat Initial Server Build. Tahapan ini berfungsi untuk mendownload serangkaian komponen awal, termasuk file konfigurasi yang akan

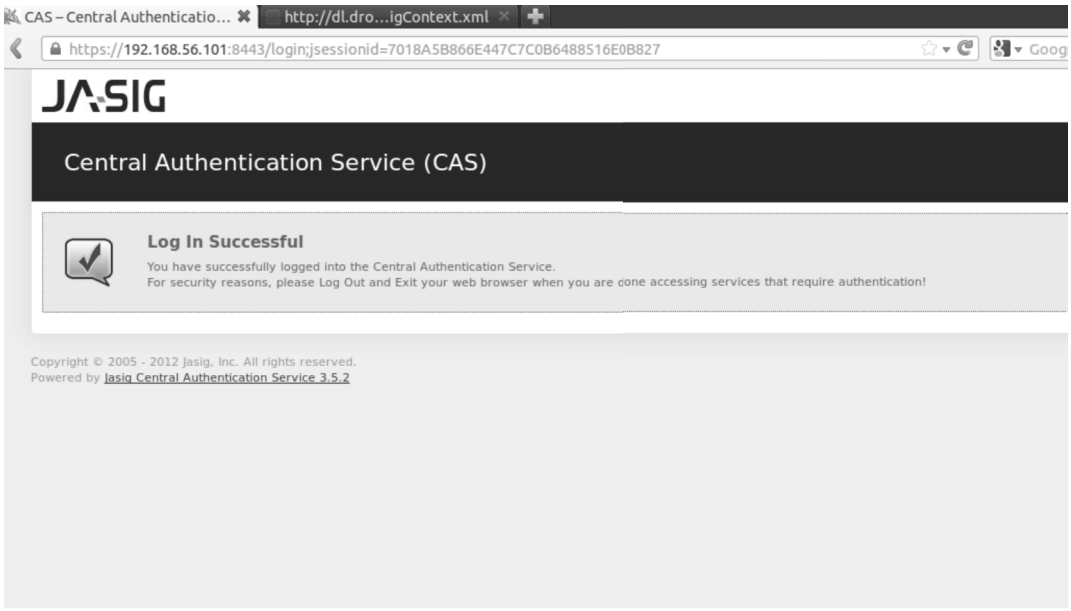
digunakan untuk mendeploy system server. Perintah maven berikut dijalankan di jendela console window yang diarahkan ke direktori C:*CAS_SSO*\local-*CAS* folder project akan dibuat untuk membangun project.

```
mvn clean package -DskipTests=true
```

Perintah tersebut akan mendownload ribuan file ke dalam local repository (yang secara default di simpan di C:\Documents and Settings\Your.Name\.m2\repository)

3. Setelah proses build selesai maka ada direktori target pada folder project. Di dalam direktori target tersebut berisikan hasil compile termasuk di dalamnya *CAS*.war archive.

   File *CAS*.war tersebut dikopikan ke C:\apache-tomcat-6.0.37\webapps. Kemudian selanjutnya menjalankan tomcat, arahkan ke Tomcat Manager admin console untuk memastikan *CAS* berjalan. Dengan memilih */CAS* application akan terlihat halaman login *CAS* login. Dengan mencoba memasukan username dan password ("test", "test") dengan mekanisme autentikasi default maka terlihat halaman *CAS* login successful page, seperti berikut:



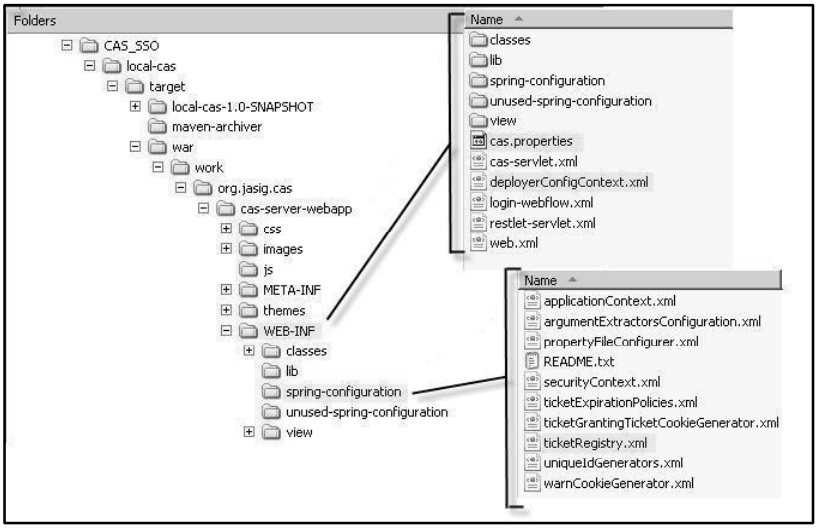

**Gambar 5.10. Tampilan *CAS SSO* login sucsess**

4. Melakukan konfigurasi Server *SSO*. Pada penelitian ini memerlukan modifikasi tiga file konfigurasi *CAS*. Hal ini bisa dilakukan dengan cara mengkopi direktori build target kemudian diberi nama direktori src directory yang akan dimodifikasi. Direktori src ini dibuat pada level yang sama dengan direktori target, dan sub direktorinya bisa dilihat pada gambar 5.11.Folder ini yang akan menyimpan file konfigurasi yang dibutuhkan.


**Gambar 5.11. Hirarki direktori src**

Yang perlu dilakukan adalah mengkopi file konfigurasi (seperti terlihat di gambar 5.11) yang ada di direktori target ke sub-direktori src seperti terlihat di bawah ini:

```
C:\CAS_SSO\local-CAS\src\main\webapp\WEB-INF
   CAS.properties
   deployerConfigContext.xml

C:\CAS_SSO\local-CAS\src\main\webapp\WEB-INF\spring-configuration
   ticketRegistry.xml
```




**Gambar 5.12. Hirarki direktori target yang perlu dikopi**

Selain itu juga pada direktori WEB-INF\classes terdapat file log4j.xml untuk mengatur log dari server *CAS*. Pada penelitian ini juga dilakukan modifikasi seting log4j sehingga dimasukkan ke file yang dikopi.

5. Tahap selanjutnya melakukan proses edit *CAS*.properties untuk melakukan Seting Services Management. Yang dilakukan menyesuaikan Services Management URL, dan jenis platform database yang mengatur services registry. Konfigurasi *CAS*.properties yang dibuat bisa dilihat seperti berikut:

```
# Point to the server that hosts Services Management
CAS.securityContext.serviceProperties.service=http://localhost:8080/CAS/services/j
_acegi_CAS_security_check
CAS.securityContext.CASProcessingFilterEntryPoint.loginUrl=http://localhost:8080/C
AS/login
CAS.securityContext.ticketValidator.CASServerUrlPrefix=http://localhost:8080/CAS

# Names of roles allowed to access the CAS service manager
CAS.securityContext.serviceProperties.adminRoles=ROLE_ADMIN

# The database platform is "SQL92" since MySQL meets that standard
ticket.cleaner.database.platform=SQL92
database.hibernate.dialect=org.hibernate.dialect.MySQLDialect

host.name=SSO.bidar.ac.id
CAS.themeResolver.defaultThemeName=default
CAS.viewResolver.basename=default_views
```

Host.name merupakan public domain yang dibuat dan disesuaikan dengan *certificate* https/ssl yang dibuat (ketidaksesuaian nama domain dengan

*certificate* yang dibuat akn mengikatkan gagalnya validasi *certificate* pada saat *CAS* dijalankan).

6. Langkah berikutnya mengedit file deployerConfigContent.xml. File inilah yang memegang peranan sebagian besar konfigurasi seting yang dibutuhkan pada penelitian ini. Detail konfigurasi file bisa di lihat di lampiran 1D. File deployerConfigContext.xml berisi sejumlah bean definitions beserta property yang digunakan. Termasuk di dalamnya authenticationManager bean (dengan property yang menentukan credentials resolvers dan penanganan autentikasi), contextSource bean, attributeRepository, serviceRegistryDao, dan lainnya. Yang perlu dilakukan pada file deployerConfigContent.xml ini antara lain Mendefinisikan dan mengkonfigurasi *LDAP* context source. Objek ini mengatur koneksi ke *LDAP*, seperti menambahkan URL dari server *LDAP*, menseting userDn dan password untuk akses service account yang dibuat untuk user lookups, seperti terlihat di bawah ini:

```
<bean id="contextSource"
class="org.springframework.ldap.core.support.LdapContextSource">
     <property name="anonymousReadOnly" value="false" />
     <property name="userDn" value="cn=admin,dc=bidar,dc=ac,dc=id" />
     <property name="password" value="ykunang" />
     <property name="pooled" value="true" />
     <property name="urls">
            <list>
                   <value>ldap://192.168.56.101:389/</value>
            </list>
     </property>
     <property name="baseEnvironmentProperties">
            <map>
                   <entry>

     <key><value>java.naming.security.authentication</value></key>
                          <value>simple</value>
                   </entry>
            </map>
     </property>
</bean>
```

Pada penelitian ini server *CAS* yang dibuat akan mengirim *LDAP* user attributes ke client. Untuk itu perlu dideklarasikan attribute repository *LDAP*, atribut ini diperlukan untuk menggunakan context source yang sudah didefinisikan sebelumnya. Properti baseDN disesuaikan dengan domain yang dibuat, seperti berikut:

```
<bean id="attributeRepository"
class="org.jasig.services.persondir.support.ldap.LdapPersonAttributeDao">
        <property name="contextSource" ref="contextSource" />
        <property name="baseDN" value="dc=bidar,dc=ac,dc=id" />
        <property name="requireAllQueryAttributes" value="true" />
        <property name="ldapTemplate" ref="ldapTemplate" />

        <!--
          Attribute mapping beetween principal (key) and LDAP (value) names
          used to perform the LDAP search.  By default, multiple search criteria
          are ANDed together.  Set the queryType property to change to OR.
          -->
        <property name="queryAttributeMapping">
                <map>
                <entry key="username" value="uid" />
                </map>
        </property>

        <property name="resultAttributeMapping">
                <map>
                        <!-- Mapping beetween LDAP entry attributes (key) and
Principal's (value) -->
                        <entry value="CN" key="cn"  />
                        <entry value="DN" key="distinguishedName"  />
                        <entry value="Groups" key="memberOf"  />
                </map>
        </property>
        </bean>
```

Properti *resultAttributeMapping* menentukan atribut user yang akan digunakan untuk client. Pada penelitian ini *CAS* Services Management dan client penting keduanya merefer ke attributes dengan nama CN, DN dan Groups.

Menambahkan *authenticationManager bean* yang berisi *credentialsToPrincipalResolvers.* Bean ini menambahan pencarian *LDAP* berdasarkan uid yang dimasukkan oleh user, yang akan didefinisikan pafa Principal available ke aplikasi client. Bean ini menggunakan context source dan atribut repository yang sudah didefinisikan sebelumnya. *SearchBase* disesuaikan dengan kebutuhan struktur direktoris ebagai berikut:

```
<beanclass="org.jasig.CAS.authentication.principal.CredentialsToLDAPAttributePrinc
ipalResolver">
        <!-- The Principal resolver form the credentials -->
        <property name="credentialsToPrincipalResolver">
             <bean
             class="org.jasig.CAS.authentication.principal.UsernamePasswordCredenti
             alsToPrincipalResolver" />
        </property>
        <!--The query made to find the Principal ID.  "%u" will be replaced by the
resolved Principal-->
        <property name="filter" value="(uid=%u)" />

        <!-- The attribute used to define the new Principal ID -->
        <property name="principalAttributeName" value="uid" />
```

```
        <property name="searchBase" value="dc=bidar,dc=ac,dc=id" />
        <property name="contextSource" ref="contextSource" />

        <property name="attributeRepository">
                <ref bean="attributeRepository" />
        </property>

</bean>
```

Menambahkan *LDAP authentication handler* ke a*uthenticationManager bean*. Bean ini memungkinkan ketika pencarian user selesai, selanjutnya akan mengautentikasi user dengan membuka context menggunakan pembeda berupa nama dan password yang dimasukkan. Bean ini juga menggunakan *bean context source* yang didefinisikan sebelumnya. *Property searchBase* disesuaikan dengan struktur direktori sebagai berikut:

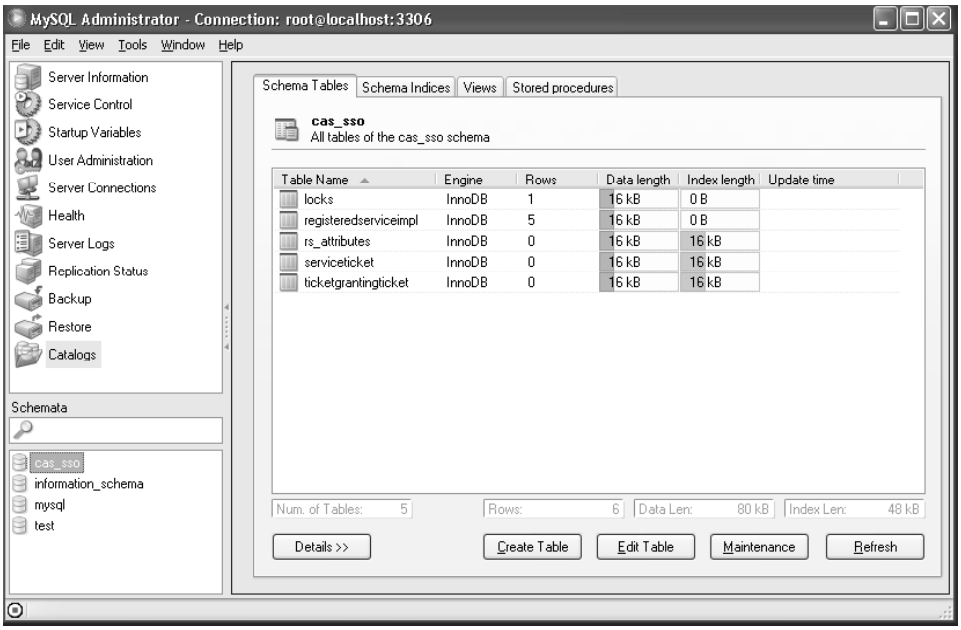
```
<bean class="org.jasig.CAS.adaptors.ldap.BindLdapAuthenticationHandler">
        <property name="filter" value="uid=%u" />
        <property name="searchBase" value="dc=bidar,dc=ac,dc=id"/>
        <property name="contextSource" ref="contextSource" />
        <property name="ignorePartialResultException" value="yes" /><!-- fix
because of how AD returns results -->
</bean>
```

Pada penelitian ini menggunakan service registry pada MySQL database yang diakses melalui *Java Persistence Architecture (JPA)* dan *Hibernate*. Beberapa beans perlu ditambahan (lihat lampiran 1D) untuk menentukan jenis database, enable automatic table creation dan memungkinkan koneksi ke database. Beberapa beans ini akan menshare ticket registry dan mengacu ke file ticketRegistry.xml file. Yang perlu dilakukan adalah mengedit property dataSource username dan password properties sesuai kebutuhan.

Selain itu juga dimasukkan *LDAP* uid dari user yang bisa menjalankan *CAS* Services Management. User ini terlebih dahulu dibuat pada *LDAP*. User yang dibuat ini bukan merupakan service account user di *LDAP*, akan tetapi hanya normal domain user account.

```
sec:user-service id="userDetailsService">

<sec:user name="admin2" password="notused" authorities="ROLE_ADMIN" />
<sec:user name="tomcat" password="notused" authorities="ROLE_ADMIN" />
</sec:user-service>
```

7. Mengedit File ticketRegistry.xml, bean ini memberikan akses database untuk ticket registry yang sudah didefinisikan di deployerConfigContext.xml, service registry dikelola pada database yang sama. Sehingga yang dilakukan adalah mendeklarasikan ticket registry dan menambahkan registry cleaner. File ticketRegistry.xml bisa dilihat di lampiran 1E.
8. Setelah melakukan modifikasi maven file pom file dan file *CAS* konfigurasi langkah selanjutnya membangun deployable *CAS* war. Proses ini dijalankan pada jendela console yang diarahkan ke C:*CAS_SSO*\local-*CAS* dan mengeksekusi perintah maven berikut untuk membangun project:
   ```
   mvn clean package -DskipTests=true
   ```
9. Tahap berikutnya menghentikan tomcat jika server tomcat sedang berjalan, dan mengkopikan file *CAS*.war ke C:\apache-tomcat-6.0.37\webapps. Menjalankan tomcat kembali dan memastikan tidak ada pesan error di jendela console. Pada penelitian ini, proses ini dilakukan beberapa kali dikarenakan muncul error yang diakibatkan kesalahan konfigurasi terutama pada file deployerConfigContext.xml.
10. Setelah dipastikan tidak terjadi error lagi, dilakukan pengujian dengan membuka Tomcat Manager admin console dan memverifikasi *CAS* bisa berjalan. Dilakukan pengujian membuka Tomcat Manager, dengan memilih /*CAS* application maka muncul halaman *CAS* login page. Kemudian dilakukan pengujian menggunakan login menggunakan user *LDAP*. *CAS* akan membuat sisa table di database yang dibutuhkan untuk registry, jika hal ini tidak berjalan berarti proses pengujian gagal. Pada saat diperiksa menggunakan MySQL Administrator, pada database *CAS_SSO* maka akan terlihat ada lima table service dan ticket registry, seperti pada gambar 5.13.




**Gambar 5.13. Tabel service dan registry setelah proses *CAS* berhasil**

### 5.3.3. Instalasi Server yang akan duijicobakan

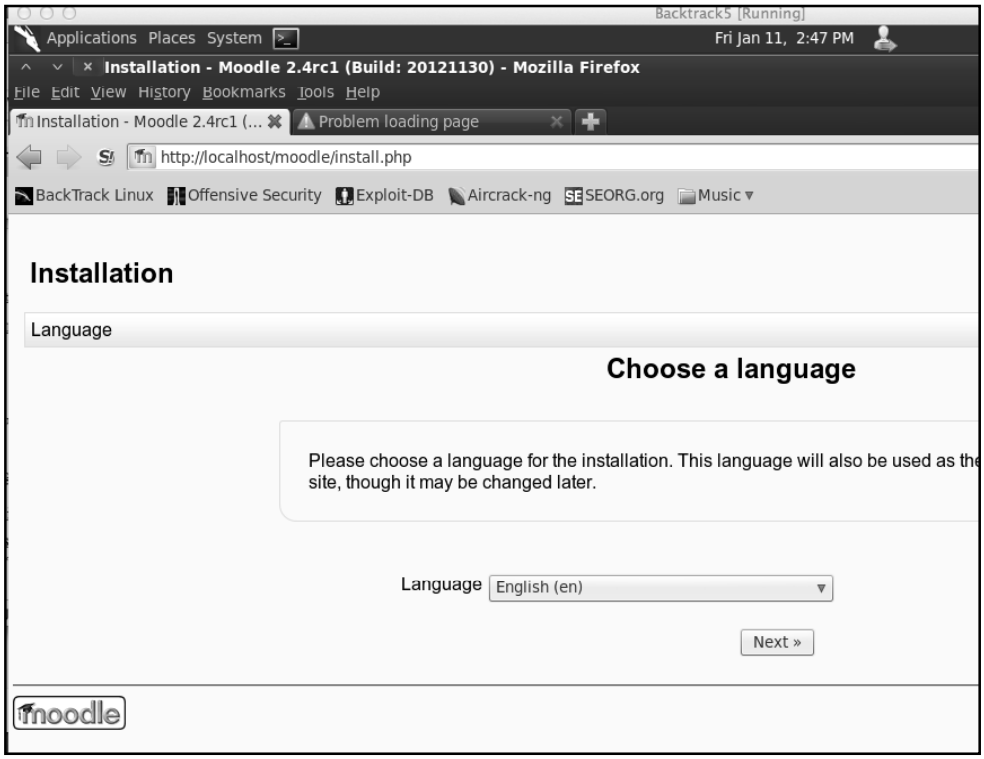
Pada penelitian ini ada tiga server yang diinstal yaitu server *elearning*, server mail dan server blog worpress.

#### 5.3.3.1. Instalasi Server *Elearning*

Server *elearning* diinstal pada Ubuntu 12.04, dengan versi *moodle* yang diinstal versi 2.2.6+ yang sudah support *LDAP* dan *CAS SSO*. *Moodle* bisa di download http://download.*moodle*.org/. Proses install *moodle* dilakukan setelah mengekstrak file yang sudah di download ke direktori /var/www. Yang perlu dilakukan adalah memastikan bahwa di server juga sudah diinstal mysql, library php5-curl, php5-gd, php5-intl, php5-xmlrpc, dan php5-*ldap*, dan phpmyadmin untuk mepermudah manajemen mysql dan *elearning*. Setelah proses install semua library dilakukan, maka yang perlu disiapkan adalah database *moodle* dengan menjalankan perintah berikut di jendela terminal mysql.

```
mysql> create database moodle charset=utf8;

mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,CREATE
TEMPORARY TABLES,DROP,INDEX,ALTER ON moodle.* TO
moodleuser@localhost IDENTIFIED BY 'yourpassword'
```

Setelah melakukan restart apache dan mysql, dengan membuka link browser ke halaman http://localhost/*moodle* (direktori tempat *moodle* diekstrak) akan tampil halaman instalasi seperti berikut:




**Gambar 5.14. Proses instalasi *moodle***

Dengan mengikuti petunjuk instalasi maka *elearning moodle* berhasil diinstal.

### 5.3.3.2. Instalasi Mail Server

Pada penelitian ini menggunakan Server Ubuntu Server 10.04.2 64 bit dan *ZCS* 7.1.1. Adapun tahapan instalasi dan konfigurasinya sebagai berikut:

1. Setelah Ubuntu server diinstal langkah pertama yang dilakukan adalah mengkonfigurasi Network dengan melakukan perubahan pada file /etc/network/interfaces. Kemudian direstart service networknya.
2. Ubah file /etc/hosts menjadi seperti berikut : dengan memasukan nama domain mail server yang dibuat

```
192.168.56.102  mail.bidar.ac.id mailserver
```

3. Ubah file /etc/resolv.conf agar memuat urutan DNS yang digunakan :

```
nameserver 192.168.56.102
nameserver 8.8.8.8
nameserver 208.67.222.222
```

IP pertama adalah IP Zimbra Server untuk DNS lokal. IP kedua adalah IP Google Public DNS, sedangkan yang ketiga adalah IP OpenDNS.

4. Lakukan update sistem. Jika diperlukan, ubah terlebih dahulu repo Ubuntu agar menggunakan repo mirror di Indonesia agar lebih cepat dalam melakukan proses instalasi paket yang diperlukan.
5. Setelah update, remove package apparmor (agar tidak menjadi bottle neck dari sisi security) dan install paket-paket yang diperlukan.

```
apt-get upgrade
dpkg --purge apparmor apparmor-utils
sudo apt-get install libidn11 libpcre3 libgmp3c2 libexpat1
libstdc++6 libltdl7 libperl5.10 sysstat fetchmail sqlite3
```

6. Download file binary Zimbra 7.2.2 untuk Ubuntu 10.04 LTS. Agar lebih cepat bisa menggunakan mirror lokal Komunitas Zimbra Indonesia : http://mirror.linux.or.id/zimbra/binary/
7. Seting DNS pada Zimbra Mail Server, konfigurasi sebagai berikut :

```
Nama domain : bidar.ac.id
Nama hostname : mail.bidar.ac.id
IP Address Server : 192.168.56.102
```

IP Address diatas akan digunakan untuk seluruh records yang digunakan. Untuk IP address yang berbeda untuk records tertentu silakan bisa ditambah nantinya.

a) Instal paket yang diperlukan

```
sudo apt-get install bind9
```

b) Buat zona baru untuk bidar.ac.id pada file named.

```
sudo -i
cd /etc/bind
nano named.conf
```

kemudian tambahkan baris konfigurasi berikut pada bagian paling bawah:

```
zone "bidar.ac.id" {
type master;
file "/etc/bind/db.bidar.ac.id";
};
```

c) Langkah selanjutnya adalah membuat konfigurasi zona forward untuk bidar.ac.id. Untuk memudahkan konfigurasi, copy file db.local menjadi db.bidar.ac.id

```
cp db.local db.bidar.ac.id
```

d) Lakukan pengubahan pada file db.bidar.ac.id

```
nano db.bidar.ac.id
```

Ubah konfigurasinya sehingga menjadi:

```
$TTL    604800
@       IN      SOA     ns1.bidar.ac.id. root.bidar.ac.id. (
                     2011062700         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      ns1.bidar.ac.id.
@       IN      A       192.168.56.102
@       IN      MX  0   mail.bidar.ac.id.
ns1     IN      A       192.168.56.102
mail    IN      A       192.168.56.102
```

e) Restart service dns dengan menggunakan perintah:

```
/etc/init.d/bind9 restart
```

f) Untuk melakukan testing DNS, kita bisa menggunakan perintah host nama domain, misalnya host bidar.ac.id atau menggunakan perintah nslookup sebagai berikut :

```
root@mailserver:/etc/bind# nslookup mail.bidar.ac.id
Server:           192.168.56.102
Address:    192.168.56.102#53

Name: mail.bidar.ac.id
Address: 192.168.56.102
```

g) Perhatikan jawaban dari hasil nslookup, pastikan bahwa IP yang muncul adalah IP server yang disetup DNS servernya.Selain dengan nslookup, untuk melakukan testing DNS bisa dengan menggunakan perintah dig.

h) Perhatikan pada bagian MX records menunjukkan bahwa MX ditujukan ke alamat mail.bidar.ac.id

8. Setelah selesai melakukan setup DNS, kita bisa melanjutkan proses instalasi Zimbra Mail Server. Proses instalasi Zimbra dengan proses sebagai berikut :
   a) Download file binary Zimbra dan menempatkannya pada folder /opt.
   b) Ekstrak file binary, masuk ke folder hasil ekstrak dan jalankan script instalasi:

```
cd /opt
tar -zxvf ZCS-7.1.1_GA_3196.UBUNTU10_64.20110527011124.tgz
cd ZCS-7.1.1_GA_3196.UBUNTU10_64.20110527011124
./install.sh
```

   c) Berikut adalah proses instalasi Zimbra, perhatikan bagian yang dicetak tebal (tanda # merupakan tanda bahwa semua perintah dijalan dengan hak akses root/sudo

```
# tar -zxvf ZCS-7.1.1_GA_3196.UBUNTU10_64.20110527011124.tgz
# cd ZCS-7.1.1_GA_3196.UBUNTU10_64.20110527011124
# ./install.sh
```

```
…
Select the packages to install
Install zimbra-ldap [Y] y
Install zimbra-logger [Y] y
Install zimbra-mta [Y] y
Install zimbra-snmp [Y] y
Install zimbra-store [Y] y
Install zimbra-apache [Y] y
Install zimbra-spell [Y] y
Install zimbra-memcached [N] n
Install zimbra-proxy [N] n
…
```

Jika mendapat pertanyaan soal “DNS ERROR resolving MX”. Jawab “Y” untuk mengubah nama domain dan kemudian ketik nama domain (bidar.ac.id) bukan mail.bidar.ac.id. Jangan by pass proses ini. Jika masih tetap bermasalah di tahap ini, periksa ulang konfigurasi DNS yang dilakukan pada tahap 2 karena kemungkinan besar ada masalah pada saat setting DNS Server.Setelah di enter, proses install akan berlanjut :

Zimbra akan memberikan informasi mengenai pilihan yang belum disetting, yaitu Zimbra Password :

```
…
Store configuration

   1) Status:                                  Enabled
```

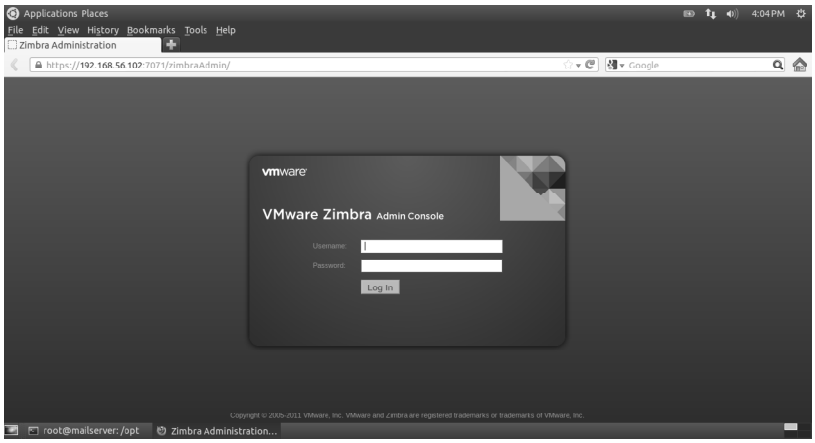
```
   2) Create Admin User:                          yes
   3) Admin user to create:                       admin@bidar.ac.id
** 4) Admin Password                              UNSET
   5) Anti-virus quarantine user:                 virus-
quarantine.gkbhtc4a@bidar.ac.id
...
```

Zimbra akan bertanya soal Zimbra Admin password pada konfirmasi akhir sebelum proses instalasi. Ketikkan saja password pada kotak yang disediakan.

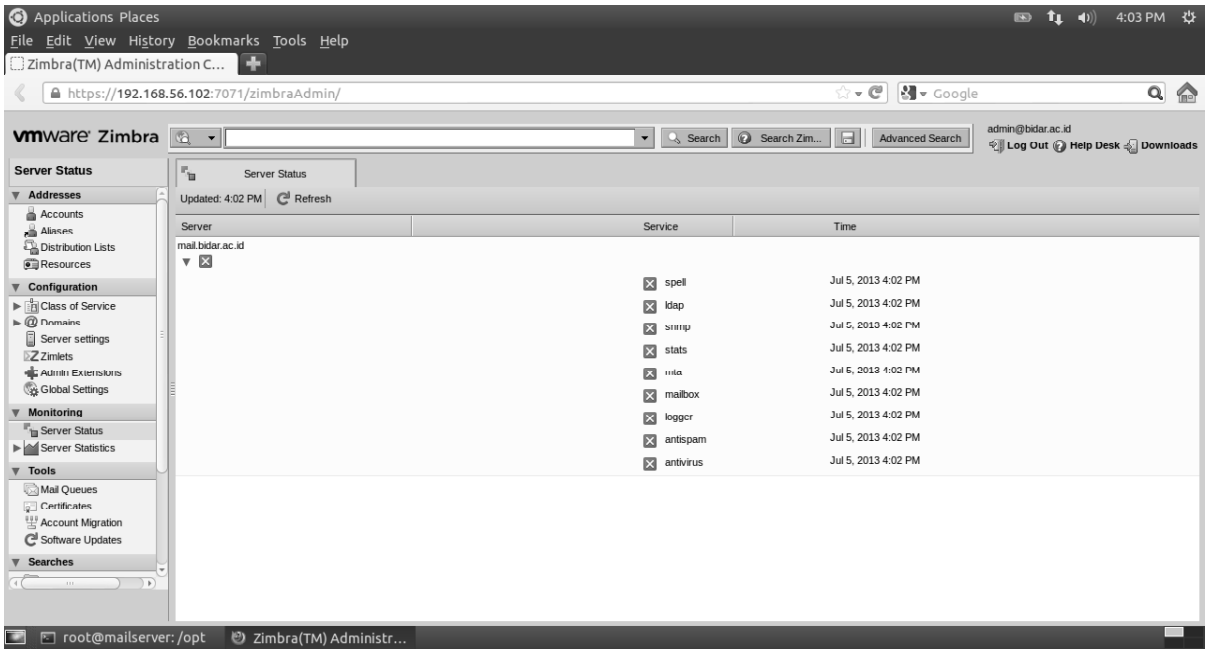
Setelah proses instalasi selesai, kita bisa melakukan proses pengecekan status menggunaka perintah zmcontrol status :

```
su - zimbra
zimbra@mail:> zmcontrol status
Host mail.bidar.ac.id
antispam Running
antivirus Running
ldap Running
logger Running
mailbox Running
mta Running
snmp Running
spell Running
stats Running
zimbra@mail:~> zmcontrol -v
Release 7.1.1_GA_3196.UBUNTU10_64 UBUNTU10_64 FOSS edition.
```

Zimbra web mail dapat diakses menggunakan host name atau IP Address (http://mail.bidar.ac.id atau http://192.168.56.102) sedangkan Zimbra Admin dapat diakses menggunakan protokol https pada port 7071 (https://mail.bidar.ac.id:7071 atau https://192.168.56.102:7071)




**Gambar 5.15. Login Admin Zimbra**




**Gambar 5.16. Jendela Admin Zimbra**

### 5.3.3.3. Instalasi Server Blog

Pada penelitian yang dibuat ini Server Blog yang dibuat berupa blog berbasis CMS wordpress yang diinstal pada Server Ubuntu 12.04. Adapun langkah yang dilakukan untuk menginstal wordpress adalah sebagai berikut:

1. Download wordpress terbaru dengan perintah

   ```
   wget http://wordpress.org/latest.tar.gz
   ```

2. Setelah proses download selesai extract dengan perintah:

   ```
   tar -zxvf latest.tar.gz
   ```

3. Setelah selesai, silahkan rename file wp-config-sample.php menjadi wp-config.php dengan perintah

   ```
   cp/wordpress/wp-config-sample.php wp-config.php
   ```

4. Buat database didalam MYSQL.

   ```
   mysql -u root -p. create database wordpress;
   ```

5. Edit file wp-config.php dengan perintah berikut

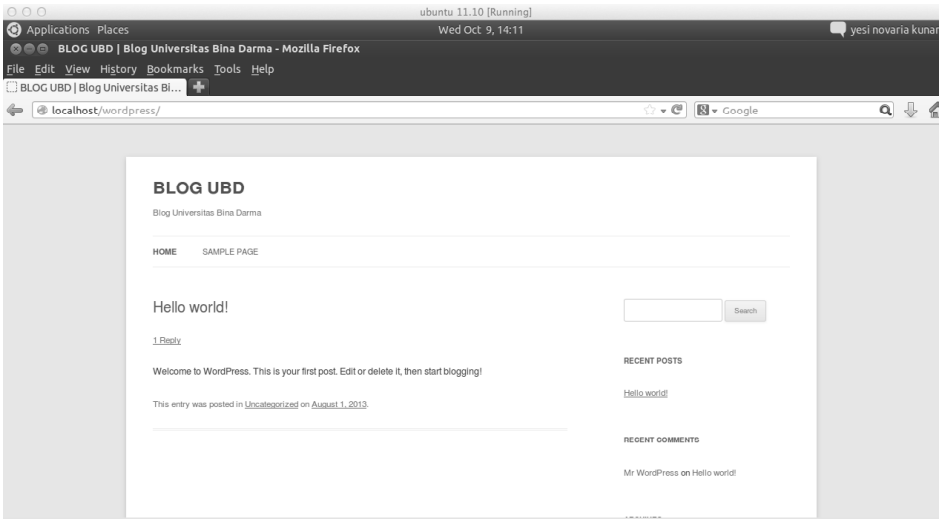
   ```
   nano /wordpress/wp-config.php
   ```

   silahkan edit bagian-bagian berikut :

   - DB_NAME diganti dengan nama database yang sudah dibuat tadi (wordpress)

- DB_USER diganti dengan nama user untuk database yang sudah dibuat (root)
- DB_PASSWORD diisi dengan password MySql

6. Ketikkan alamat http://localhost/wordpress di address bar browser, akan muncul tampilan Welcome. Silahkan isi form yang ada sesuai keinginan anda, kemudian klik Install WordPress. Jika sukses akan muncul pesan Success.
7. Klik login untuk masuk ke dashboard wordpress.




**Gambar 5.17. Halaman Utama Server Blog**

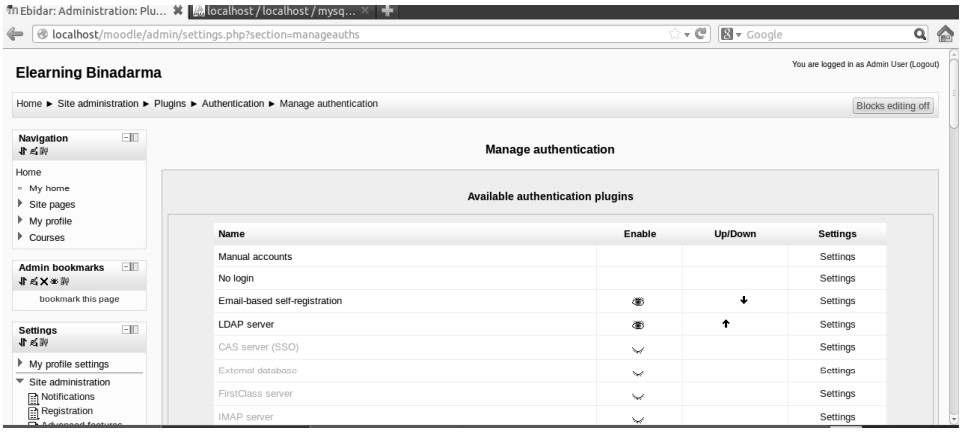
### 5.3.4. Konfigurasi Server dengan *LDAP*

Sebelum melakukan pengujian dengan Server *SSO CAS* yang sudah dibangun terlebih dahulu dilakukan pengujian autentikasi sistem server yang sudah dikembangkan dengan user autentikasi *LDAP* yang sudah dibuat pada Server Open *LDAP*. Akan tetapi yang perlu diperhatikan meskipun konfigurasi server *elearning*, zimbra dan wordpress yang dibuat dialihkan mekanisme loginnya menggunakan *LDAP* maupun *SSO* berbasis *CAS*, pada masing-masing server terlebih dahulu tetap dibuatkan user account di masing-masing server tersebut untuk pembuatan mailbox pada Zimbra, pembuatan dashboard di wordpress dan di LMS *moodle*, hanya saja kita tidak perlu membuat password karena authentikasi akan diproses oleh *LDAP*.

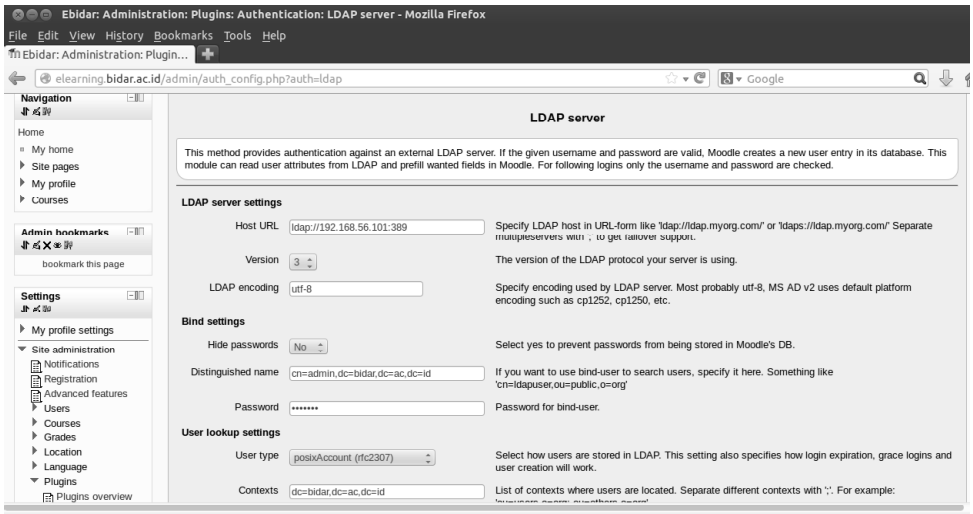
### 5.3.4.1Konfigurasi Server *Elearning* dengan *LDAP*

Setelah server *elearning* diinstal, maka sebelum mengintegrasikan server *elearning* dengan teknologi *SSO* yang dibuat maka sebelumnya dicoba terlebih dahulu mengintegrasikan *moodle* dengan *LDAP*. Karena *SSO CAS* server yang dikembangkan berbasis *LDAP* untuk manajemen usernya, sehingga perlu dipastikan terlebih dahulu apakah server yang kita kembangkan ini bisa mengakses user sesuai dengan domain group yang dibuat di server *LDAP*.

Adapun langkah konfigurasi *moodle* ke *LDAP* adalah dengan login ke sistem *moodle* sebagai admin dan melakukan modifikasi plugin authentication. Jika pluginnya tidak jalan cek php.ini untuk mengaktifkan *LDAP*. Kemudian aktifkan sehingga tampilannya seperti pada gambar 5.18 di bawah ini:




**Gambar 5.18. Mengaktifkan plugin *LDAP* di Server *Elearning***



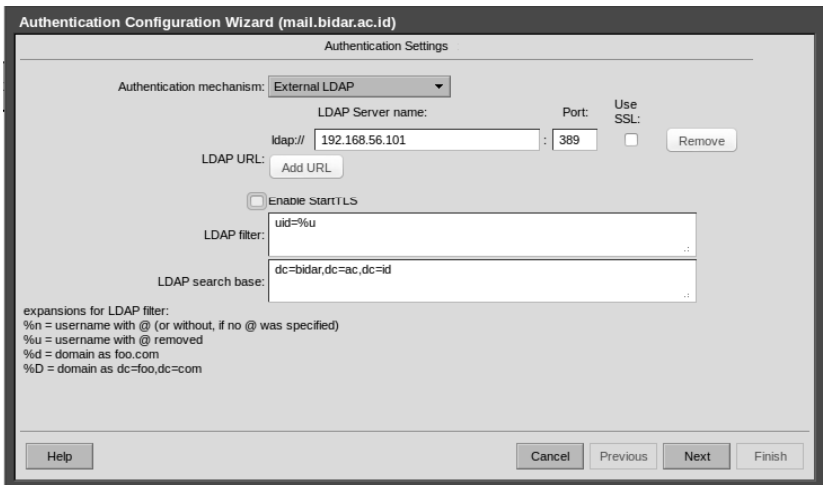

**Gambar 5.19. Parameter *LDAP* di Server *Elearning***

Setting parameter *LDAP* sesuai dengan informasi *LDAP* yang dimiliki oleh server Open *LDAP* yang sudah dibuat sebelumnya.Simpan parameter tesebut, dan lakukan tes login ke aplikasi *moodle* dengan menggunakan akun user dan password yang sudah dibuat di *LDAP*.

### 5.3.4.2 Konfigurasi Server Mail dengan *LDAP*

Salah satu prinsip dasar *Single sign on* adalah kesamaan user name dan password pada berbagai aplikasi, misalnya antara Active Directory/File Server, Zimbra Mail Server, Squid Proxy Server dan aplikasi web server. *Single sign on* dapat ditempuh dengan menggunakan *LDAP* sebagai database penyimpanan user name dan password.Zimbra mail server menggunakan *LDAP* sebagai database penyimpanan daftar accountnya, termasuk menggunakan eksternal *LDAP*. Untuk Zimbra konfigurasi eksternal *LDAP* nya dilakukan pada console Zimbra Admin.Langkahnya sebagai berikut:

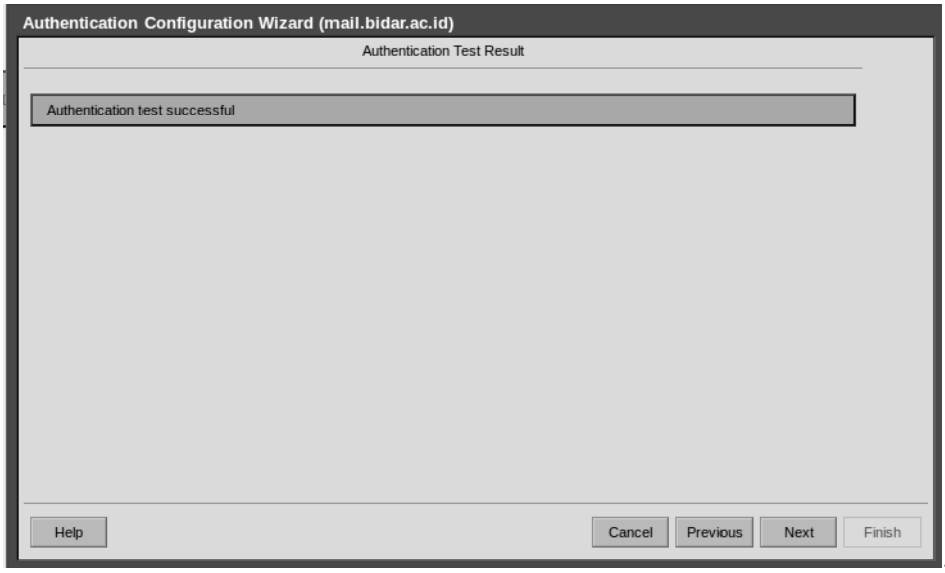
1. Pada jendela admin pilih nama domain yang ingin dikonfigurasi. Jika menggunakan skema multi domain, maka harus dilakukan konfigurasi untuk masing-masing domain, meski semuanya merujuk ke *LDAP* yang sama.
2. Pilih menu Configure Authentication. Menu ini ada pada tab bagian atas, deretan Save, Close & New.
3. Pada Authentication Mode pilih External *LDAP*, pada gambar 5.20 konfigurasi *LDAP* yang dilakukan.




**Gambar 5.20. Parameter *LDAP* di Server Zimbra**

4. Tampilan berikutnya adalah konfigurasi bind DN *LDAP*. Bind DN ini maksudnya konfigurasi user admin/manager yang digunakan untuk mengakses *LDAP*. Untuk memasukkan konfigurasi bind DN, beri tanda centang pada pilihan Use DN/Password to bind to external serverkemudian isikan bind DN-nya, cn=admin,dc=bidar,dc=ac,dc=id**.**

Pada wizard berikutnya, masukkan nama user dan password *LDAP* sebagai media testing, kemudian klik Test. Jika konfigurasi benar, Zimbra akan memberikan konfirmasi,**Authentication Test Result : Authentication test successful.** Jika masih ada pesan kesalahan, periksa ulang kesesuaian konfigurasi *LDAP* dengan konfigurasi yang dimasukkan.

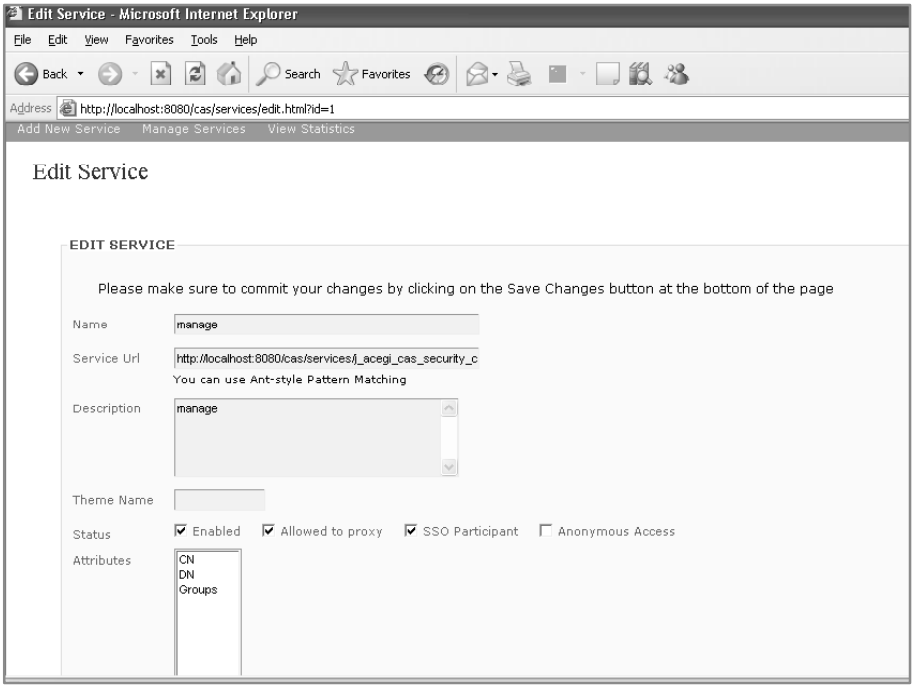



**Gambar 5.21. Hasil Test *LDAP* di Server Zimbra**

### 5.3.5. Mengenable Client Services pada Services Management *SSO-CAS*

Pada server *CAS SSO*, aplikasi server yang dikelola oleh server *CAS* harus didaftarkan pada Service Management. Untuk mengelola Service Management maka dilakukan dengan membuka aplikasi Services Management pada halaman http://localhost:8080/*CAS*/services/manage.html. Untuk loginnya digunakan user yang sudah ditentukan pada file deployerConfigContext.xml yang sebelumnya sudah dibuat pada langkah 5.3.2. Pada saat login maka tampil halaman pesan yang mengingatkan untuk menambahkan Management service

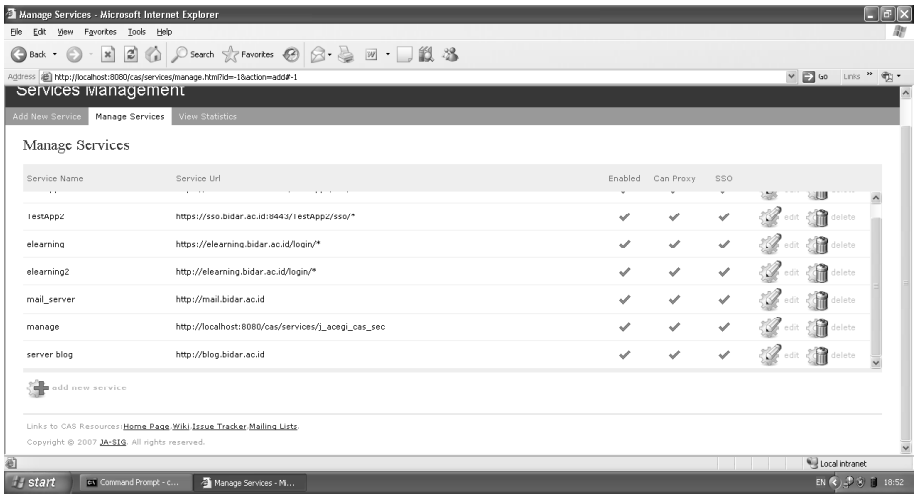
sebagai layanan pertama yang harus dimasukkan. Langkah berikutnya mengambah URL berikut sebagai layanan:http://localhost:8080/*CAS*/services/j_acegi_*CAS*_security_check




**Gambar 5.22. Penambahan layanan pada *Service Management SSO***

Pada penelitian ini sempat terjadi kesalahan melupakan proses penambahan URL di atas sehingga berakibat Services Management *CAS* tidak bisa diakses kembali. Kesalahan ini bisa diatasi dengan membuang semua table *CAS* di database kecuali table LOCKS, *CAS* server akan membuat kembali table-tabel tersebut setelah dijalankna kembali

Setelah semua proses berjalan layanan aplikasi server lain bisa ditambahkan pada halaman *ServiceManagement*.



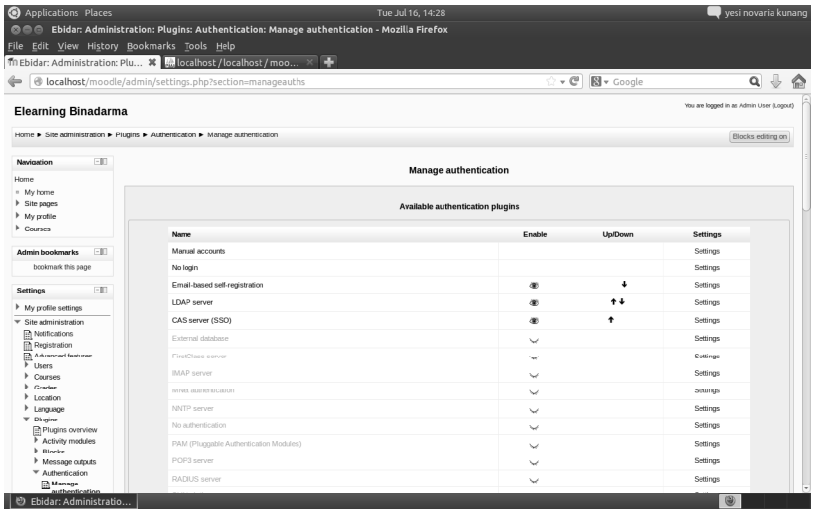

**Gambar 5.23. Layanan pada *Service Management SSO CAS***

### 5.3.6. Konfigurasi Server dengan *SSO*

Setelah layanan aplikasi server semuanya didaftarkan pada Service Management, maka langkah berikutnya melakukan konfigurasi server *elearning*, *moodle* dan Zimbra untuk diintegrasikan menggunakan layanan server *SSO* berbasis *CAS*.

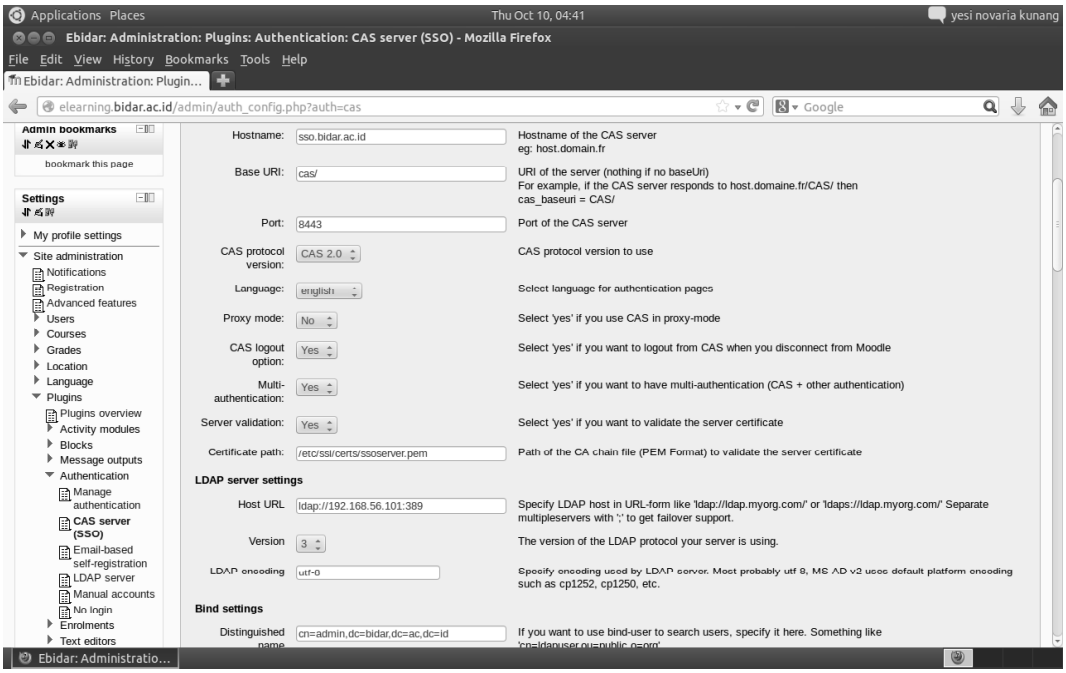
#### 5.3.6.1. Konfigurasi Server *Elearning* dengan *SSO CAS*

Untuk mengaktifkan Autentikasi menggunakan Server *CAS SSO*, maka perlu mengaktifkan plugin *CAS* Server *SSO* di bagian Manage Authentication setelah login sebagai administrator *elearning* seperti pada gambar 5.24.




**Gambar 5.24. Mengaktifkan plugin *CAS* server (*SSO*)**

Pada bagian Setings lakukan konfigurasi menyesuaikan konfigurasi di server *SSO* maupun di *LDAP*, seperti berikut ini:




**Gambar 5.25. Seting *CAS* server *SSO* di *elearning***

Yang perlu diperhatikan di sini kunci komunikasi antara server *SSO* dan server *elearning* sebagai client menggunakan komunikasi SSL yang sangat tergantung dengan mekanisme trust relationship. Untuk itu sertifikat *SSO*server.cer yang sudah dibuat menggunakan portecle itu harus diimpor terlebih dahulu ke masing-masing server aplikasi yang menjadi client server *SSO*. Adapun perintah untuk mengimpor file CA di server Ubuntu adalah dengan perintah buat:

1. buat direktori extra di /usr/share/ca-*certificate*s/
2. Kopikan file sertifikat

   `cp SSOserver /usr/share/ca-certificates/extra/`
3. edit file

   nano /etc/ca-*certificate*s.conf

   tambahkan file di direktori extra yang dibuat
4. `dpkg-reconfigure ca-certificates`
5. `update-ca-certificates`

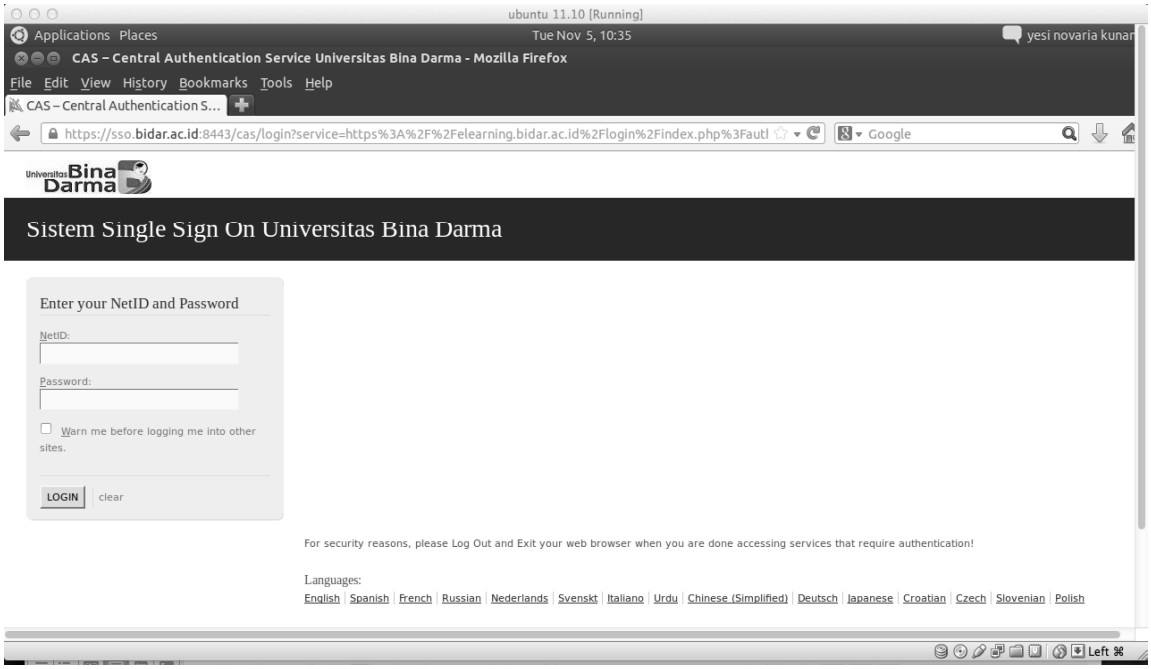
Proses ini akan menghasilkan file *SSO*server.pem yang disimpan di direktori /etc/ssl/certs/*SSO*server.pem. Pada setingan di gambar 5.25 tinggal diarahkan

*Certificate* Path yang sudah dibuat. Jika konfigurasi *certificate* path ini salah misalnya dikarenakan format *certificate* yang tidak dikenali maka proses login melalui server *SSO* tidak akan berhasil. Untuk itu pastikan pada proses ekspor melalui Portecle *certificate* diekspor menggunakan format pem.

Selain itu juga jika terjadi kesalahan konfigurasi baik untuk *CAS* dan *LDAP* bisa mengakibatkan *moodle* tidak bisa dibuka karena user termasuk admin tidak bisa login. Untuk itu melalui phpmyadmin buka tabel mdl_config kemudian edit field authdengan menghilangkan value auth (*LDAP* atau *CAS* yang bermasalah). Buang sistem autentikasi yang bermasalah, kemudian coba buka kembali login sebagai admin.

Pada penelitian ini juga untuk membantu memeriksa kesalahan login menggunakan server *SSO CAS* ini, peneliti mengenable debug *CAS* di php dengan menambahkan php*CAS*::setDebug('/var/log/php*CAS*.log') di file var/www/*moodle*/auth/*CAS*/auth.php.

Setelah melakukan konfigurasi tersebut maka proses login *moodle* berhasil diarahkan ke server *SSO CAS* yang hasil pengujiannya akan di bahas pada bab selanjutnya.




**Gambar 5.26. Login *elearning* yang di redirect ke *SSO***

### 5.3.6.2. Konfigurasi Server Zimbra Mail Server dengan *SSO CAS*

Zimbra merupakan software open source server untuk email dan collaboration - email, group calendar, contacts, instant messaging, file storage dan web document management. Untuk mengkonfigurasi Zimbra dengan *CAS SSO* langkahnya sebagai berikut:

1. Konfigurasi Zimbra CACerts *keystore*Import *CAS* Server *certificate*s yang sudah dibuat ke dalam Zimbra CACerts *Keystore* dengan menjalankan perintah berikut sebagai user root :

   ```
   /opt/zimbra/java/bin/keytool -import-file SSOserver.cer -
   alias CAScert -trustcacerts -keystore
   /opt/zimbra/java/jre/lib/security/cacerts -storepass
   changeit
   ```

2. Import Java *CAS* Client library

   Library ini digunakan untuk mengimplementasikan fungsi *CAS* dan menyederhanakan aplikasi web *CAS*ifying menggunakan aplikasi filter.

   a) Download dari http://www.ja-sig.org/downloads/*CAS*-clients/. Client version 3.1.x berjalan baik pada Zimbra 7.0.x dan *CAS* Server 3.3.x.

   b) Kopikan *CAS*-client-core-3.1.x.jar ke /opt/zimbra/jetty/common/lib.

3. Modifikasi file web.xml Pada Zimbra Webapp, Tambahkan baris berikut ke /opt/zimbra/jetty/etc/zimbra.web.xml.in sebelum bagian <servlet> (~baris 230) dan ganti *CAS*.url.com:port dan zimbra.url.com:port.

   Default ports yang digunakan untuk *CAS* Server menggunakan port 8443 dan port 443 untuk Zimbra Web Client (atau 80 jika menggunakan port HTTP selain of HTTPS) . Jalankan perintah berikut:

   ```
   nano /opt/zimbra/jetty-6.1.22.z6/etc/zimbra.web.xml.in
   ```

   Kemudian edit `https://CAS.url.com:port` sesuaikan dengan url server *SSO CAS* dan portnya, serta `https://zimbra.url.com:port`, disesuaikan dengan URL zimbra mail server serta portnya. Contoh file bisa dilihat di lampiran 1F.

4. Membuat PreAuth key dengan menjalankan perintah berikut sebagai Zimbra user :

   **zmprov gdpak bidar.ac.id**

   Perintah tersebut akan membuat PreAuth key

5. Membuat file preauth.jsp Pada Zimbra Webapp. Kopikan file preauth.jsp-zimbra file (download dari http://wiki.zimbra.com/wiki/Preauth#Sample_preauth.jsp) ke /opt/zimbra/jetty/webapps/zimbra/public/preauth.jsp.

   Yang perlu dilakukan adalah mengganti DOMAIN_KEY dengan key yang sudah digenerate sebelumnya dengan **zmprop**,Ganti yourdomaine.com dengan domain bidar.ac.id pada baris 90. Konfigurasi file preauth.jsp bisa di lihat di lampiran 1G.

6. Jalankan perintah berikut sebagai user root :

   ```
   chown zimbra:zimbra
   /opt/zimbra/jetty/webapps/zimbra/public/preauth.jsp
   ```

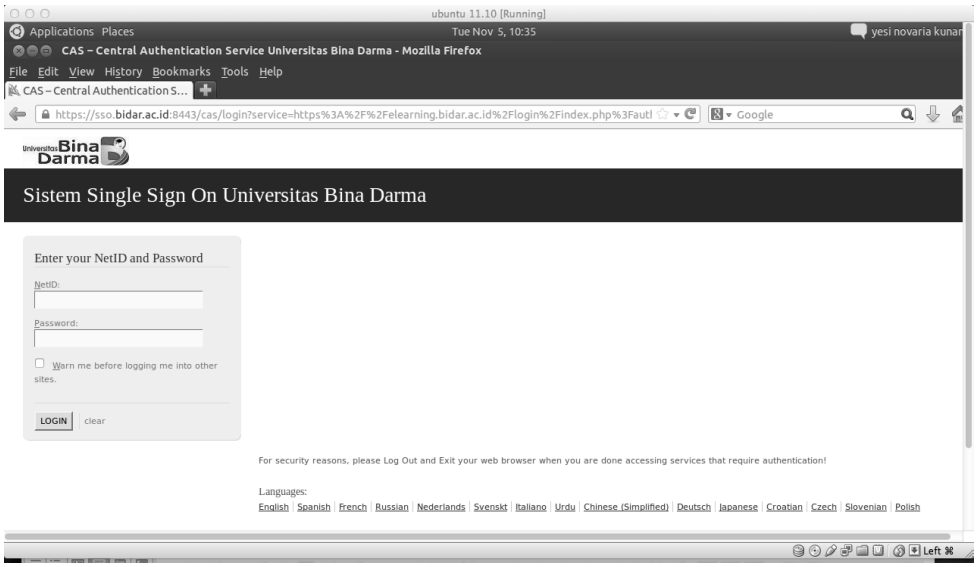
7. Rubah URL login dan logout Jalankan perintah berikut sebagai Zimbra user :

   ```
   zmprov md bidar.ac.id zimbraWebClientLoginURL
   https://mail.bidar.ac.id:443/zimbra/public/preauth.jsp

   zmprov md bidar.ac.id zimbraWebClientLogoutURL
   https://SSO.bidar.ac.id:8443/CAS/logout
   ```

8. Restart ZimbraJalankan perintah berikut sebagai Zimbra user :

   ```
   zmcontrol restart
   ```

Maka setelah melakukan konfigurasi dan merestart zimbra, selanjutnya untuk login zimbra akan diredirect ke server *SSO CAS* login.




**Gambar 5.27. Halaman *Login* Server Mail Zimbra setelah redirect ke Server *SSO CAS***

### 5.3.6.3. Konfigurasi Server Blog Wordpress dengan *SSO CAS*

Untuk mengintegrasikan *CAS* dengan klien php di server blog wordpress, maka kita perlu mengunduh terlebih dahulu pustaka untuk menghubungkan *CAS* server dengan klien berbasis php. Pustaka php*CAS* dapat diunduh pada url http://www.ja-sig.org/downloads/*CAS*-clients/php/1.3.2/*CAS*-1.3.2.tgz pada server linux yang digunakan pada penelitian dengan menggunakan perintah :

```
#wget
http://www.jasig.org/downloads/CASclients/php/1.3.2/CAS-
1.3.2.tgz
```

Setelah selesai mengunduh pustaka klien, ekstrak dengan perintah :

```
#tar -xzvf CAS-1.3.2.tgz
```

Setelah diekstrak akan ada 1 direktori dan satu file. Direktori tadi dikopi ke direktori /var/www

```
#cp CAS-1.3.2 /var/www/CAS
```

Selain menginstal library *CAS*-client, yang perlu dilakukan juga adalah mengimport sertifikat digital *SSO*server.pem yang langkahnya sama seperti pada server *elearning* seperti pada langkah 5.3.6.1.

Setelah sebelumnya menginstal Wordpress, langkah selanjutnya adalah

instalasi plugin otentikasi *CAS*.Plugin ini dapat diunduh padahttp://downloads.wordpress.org/plugin/wp*CAS*.zip, setelah terunduhekstrak plugin dengan perintah :

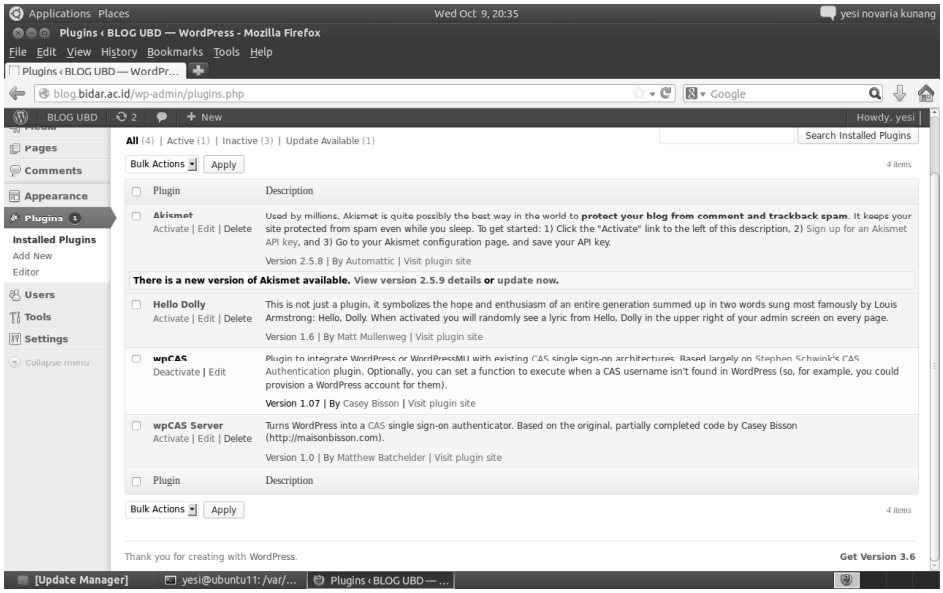
```
# unzip wpCAS.zip
```

Hasil ekstraksi berupa direktori wp-*CAS*. Salin direktori tersebut ke dalam direktori wordpress/wp-content/plugins. Ubah nama wp*CAS*conf-sample.php menjadi wp*CAS*-conf.php, sesuaikan parameter pada file konfigurasi tersebut sesuai parameter yang ada pada server *CAS*.

Skrip di bawah ini adalah contoh dari konfigurasi plugin wp-*CAS*

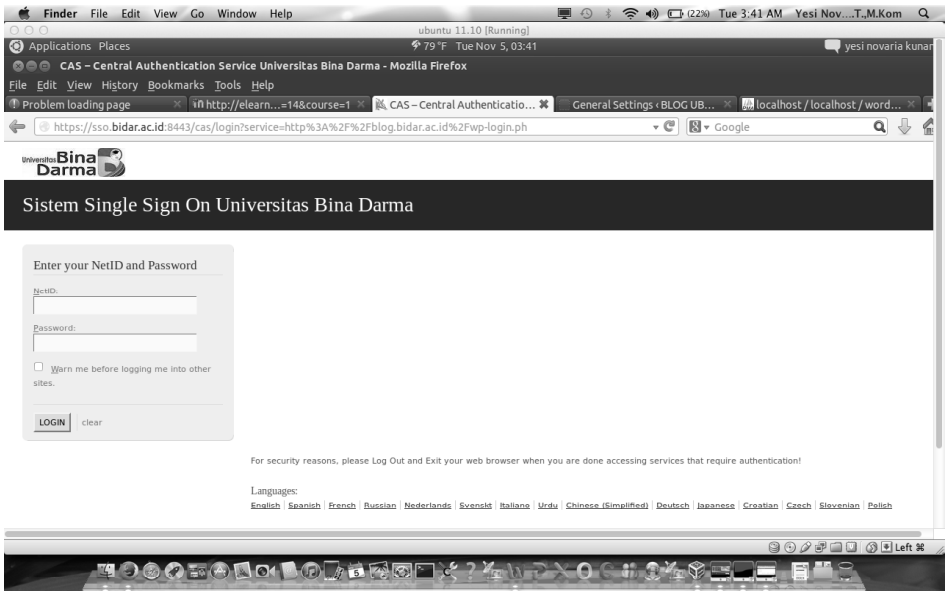
```
// the configuration array
$wpCAS_options = array(
      'CAS_version' => '2.0',
      'include_path' => '/var/www/CAS/CAS.php',
      'server_hostname' => 'SSO.bidar.ac.id',
      'server_port' => '443',
      'server_path' => '/CAS/'
      );

// this function gets executed
```

Aktifkan plugin melalui halaman administrasi pada bagian menu Plugins pilih wp*CAS* kemudian klik Activate .




**Gambar 5.28. Plugin wpCAS di wordpress**

Setelah mengaktifkan plugin tersebut, maka setelah logut dilakukan ujicoba login ke wordpress akan di redirect ke server *SSO CAS* seperti pada gambar 5.29.




**Gambar 5.29. Proses login wordpress yang diredirect ke *CAS* server**

## 5.4. Pengujian Sistem *SSO CAS*

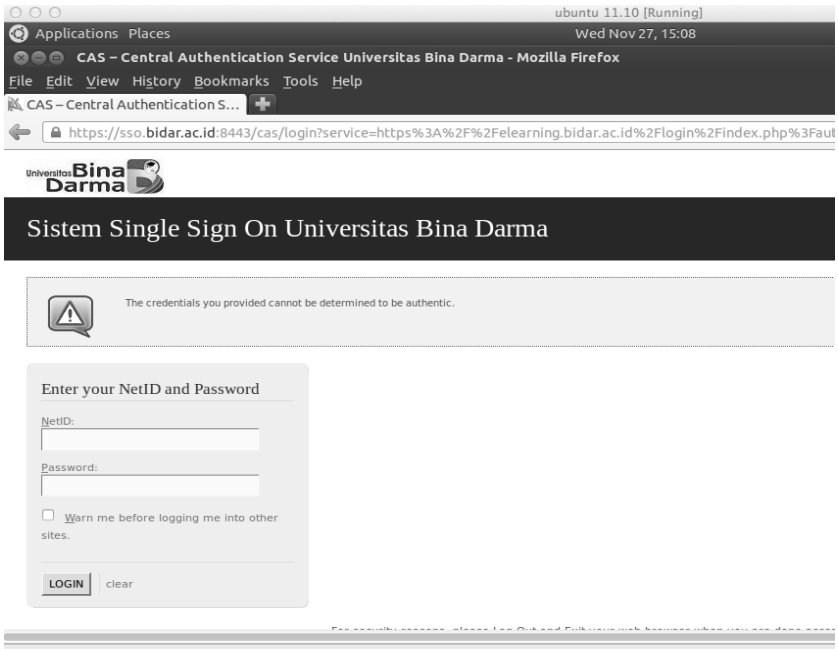
### 5.4.1. Pengujian Sistem *SSO CAS*

Pada penilitian ini dilakukan Pengujian *SSO* berdasarkan hirarki user. Pengujian ini akan lebih fokus ke hirarki Group *LDAP* dengan asumsi tidak semua group user di Universitas memiliki hak ases ke suatu sistem aplikasi. Misal aplikasi yang ditujukan hanya untuk dosen seperti *blog* yang dtujukan hanya untuk dosen. Di sini akan dipelajari apakah hal tersebut memungkinkan filterisasi user di *LDAP*.

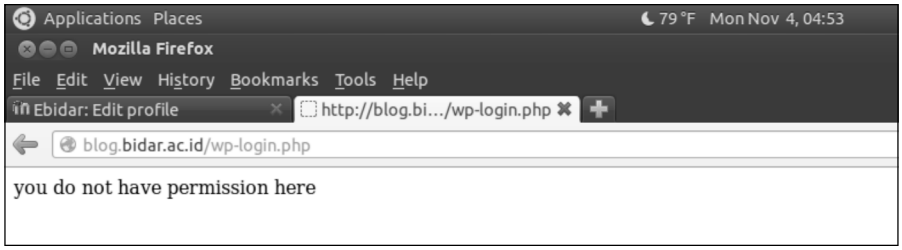
**Tabel. 5.4. Skenario Pengujian dan Hasil Pengujian berdasarkan Hiraki Group User di *LDAP***

| User | Otoritas | Hak Akses | | | Login *SSO* | Akses ke server | | |
|---|---|---|---|---|---|---|---|---|
| | | *elearning* | blog | mail | | *elearning* | blog | mail |
| User1 | Tidak ada | tidak | tidak | tidak | ditolak | gagal | gagal | gagal |
| User2 | mahasiswa | ya | tidak | ya | berhasil | berhasil | ditolak | berhasil |
| User3 | dosen | ya | ya | ya | berhasil | berhasil | berhasil | berhasil |

Hasil pengujian menunjukan teknologi *SSO CAS* ini sangat memungkinkan memfilterisasi user berdasarkan user group yang ditentukan. Diujicobakan dengan membuat group user dosen dan mahasiswa. Pada pengujian dicoba server *blog* yang hanya ditujukan untuk user dosen. Pada saat *login* diujicobakan *login SSO* menggunakan *account* user mahasiswa. Dari file log *LDAP* menunjukan bahwa proses *login* menemukan user ditemukan akan tetapi respon *LDAP* memberitahukan user tidak berhak seperti pada gambar 5.30.




**Gambar 5.30. User 1 ditolak karena tidak ada hak akses *SSO CAS***

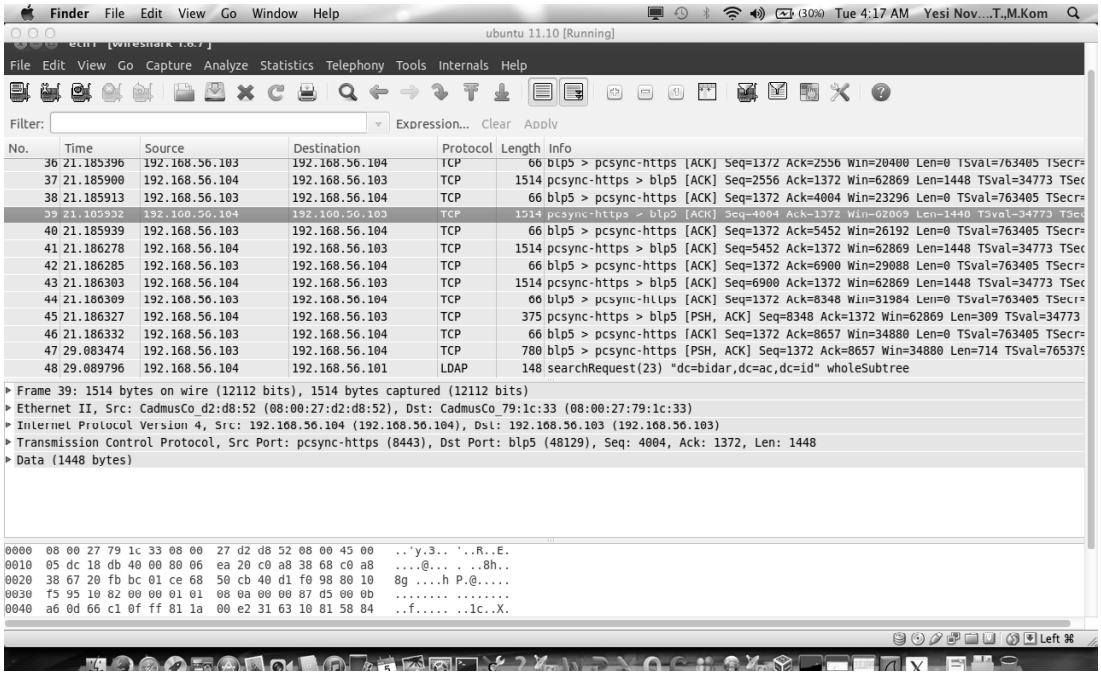



**Gambar 5.31. User 2 yang tidak memiliki akses ke server blog setelah login *SSO* akan ditolak untuk mengakses server blog**

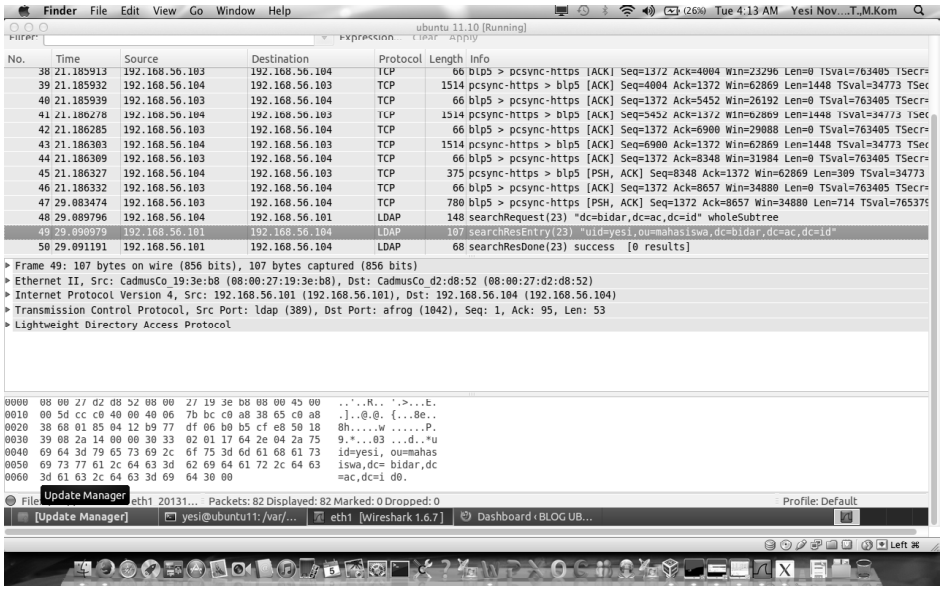
Stabilitas sistem *SSO* juga diujicobakan dengan melakukan simultan *login*, untuk beberapa aplikasi. Hasilnya user yang membuka aplikasi *elearning login* dengan *SSO CAS*, bisa langsung masuk ke halaman *blog* dan *email* tanpa melakukan *login* lagi jika user memiliki otoritas untuk mengakses aplikasi tersebut.

### 5.4.2. Pengujian Keamanan Sistem *SSO CAS*

Pada pengujian juga dilakukan pengujian untuk meninjau keamanan sistem. Pengujian dilakukan dengan mengcapture paket menggunakan tools wireshark. Pada saat *login* menggunakan server *SSO CAS* terlihat proses *login* dienkripsi menggunakan protokol https. Akan tetapi setelah proses *login* maka server *LDAP* akan mengirim respon query yang di dalamnya akan terlihat user dan domain dari query *LDAP* yang tidak dienkripsi menggunakan protokol *LDAP* 389. Tetapi hasilya hanya berupa success atau bindresponse *LDAP* bukan *password.*




**Gambar 5.32. Paket *login SSO* terenkripsi menggunakan https**




**Gambar 5.33. Respon Query *LDAP* yang tidak terenkripsi**

Jadi mekanisme *login* menggunakan *Single sign on* berbasis *CAS LDAP* ini aman karena komunikasi dilakukan menggunakan protocol https yang dienkripsi. Akan tetapi untuk respon query dengan protokol *LDAP* terlihat tidak terenkripsi, sehingga bisa terlihat query *LDAP* berupa respon *LDAP* yang bisa terlihat nama user dan group user tapi tidak untuk password.

Untuk pengembangan perlu dipikirkan untuk mengusahakan jalur terenkripsi untuk protokol *LDAP*. Untuk itu ada 2 solusi agar saat pengiriman data *ldap*server terencypt, yaitu menggunakan:

a) *Ldap* over TLS ( Transport Layer security) = penggunaan saat configure client masih *ldap*://…. dan masih menggunkan port default *ldap* server yaitu 389.

b) *Ldap* SSL (Secure Socket Layer ) = jika menggunakan ini saat saat configure client harus *ldaps*://…….. dan akan mengubah port dari *ldap* server menjadi 636.

Kemudian juga untuk website yang menggunakan Sistem *SSO CAS* ini sebaiknya dikonfigurasi untuk menggunakan protokol https yang terenkripsi.

### 5.4.3. Survei ke Calon Pengguna

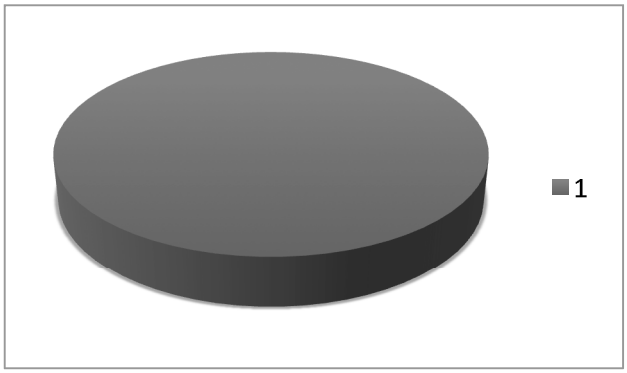
Untuk melihat kelayakan sistem maka dilakukan survei kelayakan ke beberapa calon pengguna yang terdiri dari dosen, mahasiswa dan karyawan. Survei dilakukan ke 30 orang dengan sebaran sebagai berikut:

**Tabel 5.5. Sebaran Kuisoner ke Calon pengguna**

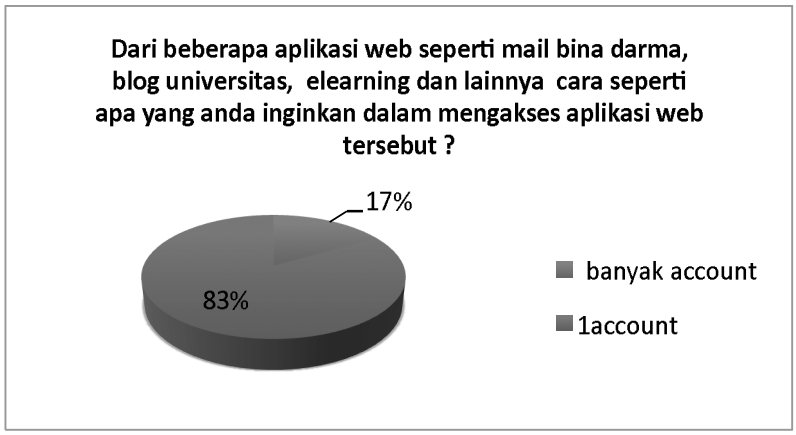
| User | Unit | Jumlah | Total |
|---|---|---|---|
| dosen | ilkom | 9 | 12 |
| | non ilkom | 3 | |
| Karyawan | karyawan | 0 | 10 |
| | upt | 1 | |
| | ppm | 3 | |
| | adm | 5 | |
| | pengajaran | 0 | |
| | lainnya | 1 | |
| Mahasiswa | Ilkom | 5 | 8 |
| | Non Ilkom | 3 | |
| Total | | | 30 |

Hasil dari kuisioner yang dilakukan bisa dilihat sebagai berikut:

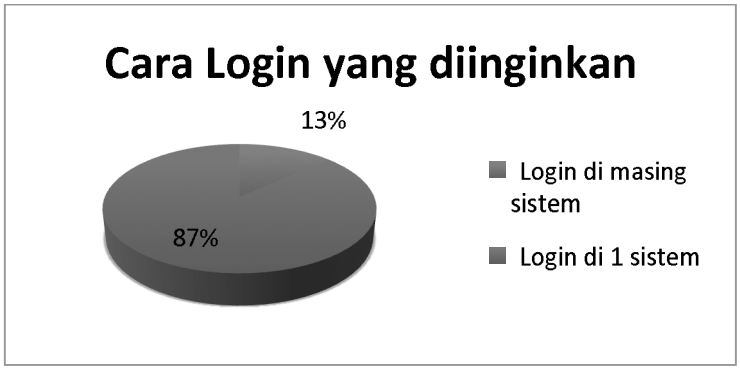
Kuisioner 1. Apakah anda sering menggunakan beberapa system/aplikasi/website Universitas Bina Darma.



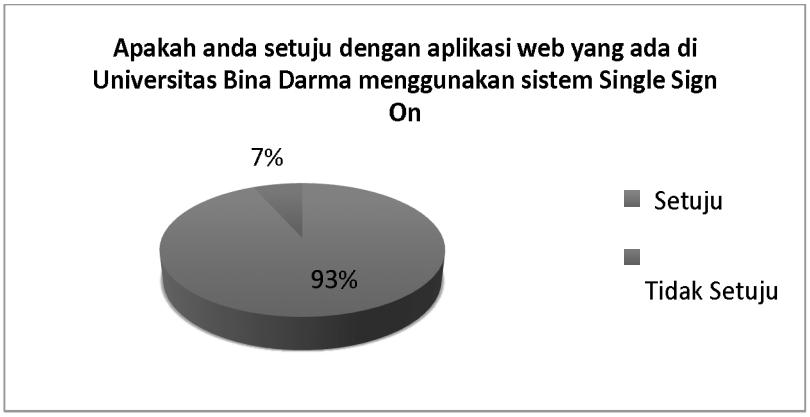

Kuisioner 2. Dari beberapa aplikasi web seperti mail bina darma, blog universitas, *elearning* dan lainnya cara seperti apa yang anda inginkan dalam mengakses aplikasi web tersebut ?



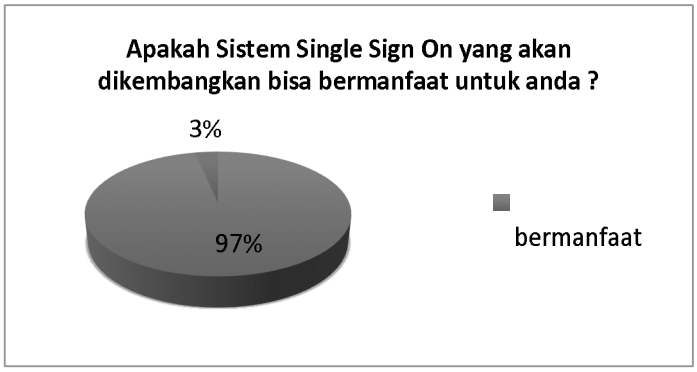

Kuisioner 3. Bagaimana cara login yang anda inginkan untuk menggunakan aplikasi *web* di atas ?




Kuisioner 4. Apakah anda setuju dengan aplikasi *web* yang ada di Universitas Bina Darma menggunakan sistem *Single sign on* (satu account, satu password dan satu kali login) ?



Kuisioner 5. Apakah sistem *Single sign on* ini dapat bermanfaat untuk anda ? Jelaskan alasannya !




Kesimpulan dari hasil kuisioner yang telah dilakukan yaitu:

- Keseluruhan responden merupakan pengguna aktif layanan aplikasi/sistem yang ada di Universitas Bina Darma.
- Keinginan untuk menggunakan satu *account* (satu *username* dan *password*) pada beberapa aplikasi *web* yang diajukan seperti blog binadarma, mail binadarma serta *elearning* binadarma mendapat lebih dari 83% responden.
- Kemudian cara satu kali *login* untuk dapat mengakses semua aplikasi *web* juga mendapat respon lebih dari 87% responden.
- 93% setuju untuk menggunakan sistem *SSO* (satu account, satu password dan satu kali login) pada aplikasi web yang ada di Universitas Bina Darma.
- Serta 97% responden menyatakan bahwa sistem *SSO* dapat bermanfaat

Jika dilihat dari hasil kuisioner tersebut maka prototype sistem *Single sign on* Universitas yang dikembangkan ini sangat mungkin untuk diimplemetasikan khususnya di Universitas Bina Darma.

# BAB 6
# KESIMPULAN DAN SARAN

Kesimpulan pada penelitian ini adalah Sistem Single Sign On ini sangat memungkinkan diimplementasikan pada Universitas berdasarkan hasil sementara yang sudah diujikan pada server SSO berbasis CAS yang digunakan untuk mengintegrasikan layanan elearning, email dan blog.

Sistem SSO yang dikembangkan ini sangat memungkinkan dilakukan filterisasi berdasarkan group user di LDAP. User bisa diatur untuk akses ke beberapa aplikasi saja berdasarkan group user tersebut, meskipun login menggunakan sistem SSO.

Dari sisi keamanan sistem SSO yang dikembangkan cukup aman terutama dengan penggunaan https dan fitur ldap over TLS sehingga komunikasinya terenkripsi. Dengan demikian tidak bisa dilakukan proses sniffing password oleh hacker.

Saran pada penelitian ini agar pengujian sistem SSO bisa dujicobakan pada sistem akademis dan sistem lainnya yang didevelop oleh developer tertentu. Selain juga sistem SSO ini bisa dikembangkan penelitiannya untuk sistem berbasius Radius, Shiboleth dan lainnya.