Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea Triennale in Informatica

Sistema di ticketing

Tesi di Laurea in Architettura degli Elaboratori

Relatore:
Dott. Vittorio Ghini

Presentata da:
Ivan Heibi

Sessione I
Anno Accademico 2011/2012

Ai miei genitori,
che mi hanno sempre appoggiato in tutte le mie scelte
caricandomi e motivandomi per affrontare ogni ostacolo

Introduzione

Lo scopo di questa tesi è quello di progettare, implementare e valutare un valido sistema di ticketing, in grado di gestire le varie richieste di supporto tecnico e amministrativo sottoposte al personale tecnico del Dipartimento di Informatica. Questo sistema potrà essere visto come una alternativa all'attuale invio di richieste e segnalazioni tramite il Newsgroup fornito a tutti gli utenti afferenti al Dipartimento. tale servizio, per quanto elementare, non permette la risposta in tempo reale a tutte le richieste, perdendo così in efficienza.

Si è quindi avuta la necessità di trovare una soluzione che rendesse questo sistema più modulare, gestibile e di facile utilizzo, sia da parte degli utenti che sottomettono una richiesta, sia da parte del personale abilitato a gestire e risolvere la richiesta. Il software utilizzato, come vedremo in seguito, ci permetterà di monitorare le richieste in modo estremamente completo, dando una dimensione piuttosto professionale all'ambiente; inoltre, come richiesto dalle specifiche fornite dal CESIA, dovremo integrare il tutto con la nuova modalità di autenticazione.

La stesura della tesi si basa sul tirocinio svolto nei mesi precedenti assieme al collega Ruggero Schirinzi, prendendo spunto dal lavoro dei colleghi Nicola De Donno e Alfonso Davide Pilato. Per scelta, il nostro lavoro non ha avuto un preciso punto di separazione, sviluppando tutte le varie fasi insieme, suddividendoci il carico di lavoro solo in alcuni casi e dando ognuno il proprio contributo.

Dopo una breve panoramica e definizione delle caratteristiche principali

dei sistemi di ticketing, entreremo nel dettaglio dei requisiti e dei parametri utilizzati per la scelta del software più adatto alle nostre esigenze. Suddivideremo le varie possibili richieste e segnalazioni in delle aree, ad ognuna delle quali verrà assegnato un gruppo di corrispondenza. A tale gruppo, a sua volta, verranno associati diversi tipi di utente. Quindi parleremo delle scelte implementative prese durante i colloqui con i tecnici Andrea Barilli e Paolo Marinelli.

Inquadrata la situazione, ci soffermeremo ad analizzare e confrontare alcuni dei più diffusi software di ticketing presenti sul mercato Open Source; fatto ciò analizzeremo nel dettaglio OTRS, spiegandone la struttura interna, i vantaggi del suo utilizzo e le possibili configurazioni. Nel sesto capitolo cominceremo a descrivere in dettaglio la procedura di installazione e configurazione svolta di OTRS. Nel settimo capitolo ci occuperemo di descrivere ed analizzare il sistema di autenticazione Single Sign On (SSO), entrando nello specifico del software Shibboleth e modificando l'ambiente di utilizzo del sistema.

Concluderemo questo documento con un capitolo dedicato alla valutazione delle prestazioni del sistema e un ultimissimo capitolo che tratta i suoi possibili sviluppi futuri.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Sistemi di Ticketing

I sistemi di ticketing sono delle applicazioni web, generalmente utilizzate per l'organizzazione di un supporto tecnico rivolto alle problematiche degli utenti all'interno di una rete aziendale; essi forniscono gli strumenti necessari a tutti gli utenti che utilizzano tale sistema (semplici-personale tecnici-amministratori), la creazione di segnalazioni o richieste di natura hardware o software che chiameremo ticket. Questi sistemi, conosciuti anche come Issue Tracking System, si appoggiano su di un database all'interno del quale sono contenute tutte le informazioni necessarie all'utilizzo più consono da parte degli utenti.

1.1 Il Ticket

Come detto precedentemente un ticket sarà una segnalazione o richiesta che viene creata attraverso una interfaccia nel sistema di ticketing dedicata alla sua creazione (help desk). Una volta creato, il ticket conterrà alcune informazioni che verranno mantenute durante il suo ciclo di vita, le informazioni contenute saranno sia statiche (informazioni acquisite al momento della sua creazione), che dinamiche, ovvero informazioni e commenti inseriti dalle enti responsabili della sua gestione, durante il suo trattamento. Ogni ticket generato avrà un ID univoco per permettere una comunicazione estre-

mamente organizzata tra le parti. Questi ticket avranno un ciclo di vita predefinito:

1. Creazione del ticket da parte dell'utente.
2. Assegnazione del ticket al personale tecnico adeguato.
3. Gestione del ticket.
4. In caso di risoluzione positiva, chiusura del ticket e segnalazione all'utente proprietario del ticket.

1.2 Help desk

Un help desk è un servizio che fornisce informazioni e assistenza ad utenti che hanno problemi nella gestione di un prodotto o di un servizio. È possibile accedere e usufruire di tale servizio attraverso diverse modalità tra cui call center, posta elettronica, chat, e come detto in precedenza attraverso i sistemi di ticketing. Attraverso tali sistemi è possibile gestire un help desk tramite un software, che permette di monitorare le richieste degli utenti tramite un codice univoco.

Gli help desk sono spesso organizzati in vari livelli, a seconda delle priorità. Attraverso un sistema di ticketing è possibile gestire facilmente le priorità delle richieste, permettendo di gestire prima le richieste con maggior urgenza.

Capitolo 2

Analisi Sistemi di Ticketing

Per poter scegliere il miglior sistema di ticketing, ci siamo basati sul lavoro precedentemente svolto da Pilato e De Donno; seguendo le linee guida che avevano tracciato grazie ai colloqui con il personale tecnico, abbiamo potuto ampliare tale lavoro con una nostra ulteriore analisi, avendo così una conoscenza ancora più ampia riguardo i sistemi di ticketing attualmente in uso in ambito informatico. Questa ricerca è stata basata prendendo in considerazione alcune importanti caratteristiche che riteniamo importanti per una scelta più consona alle nostre esigenze .

2.1 Caratteristiche

Di seguito elenchiamo alcune importanti caratteristiche prese in considerazione nella analisi dei sistemi di ticketing :

Licenza: in particolare si è svolta una particolare attenzione sui sistemi con licenza di software libera (GPL,AGPL ...);

Linguaggio: il linguaggio con cui è scritto il sistema.

Data Base(Back End): il RDBMS usato dal sistema.

SystemLog: un file per il monitoraggio dello stato del sistema .

Interfaccia Web: l'interfaccia web utilizzata nell'interazione con il sistema.

Installazione: la procedura di installazione del sistema.

Single Sign On: la possibilità di integrare il sistema con il SSO .

Supporti esterni: applicazioni mobile .

Diffusione e Popolarità: questo fatto garantisce come conseguenza una comunità più ampia ed attiva in rete ,ed un sistema più soggetto ad aggiornamenti .

2.2 Tabella di valutazione

Di seguito nella tabella 2.1 vengono elencati alcuni sistemi di ticketing presi in analisi con particolare attenzione alle caratteristiche descritte nella sezione precedente. Come si puo notare si è voluto analizare anche un sistema di ticketing con licenza proprietaria come JIRA ,come particolare esempio di sistema di ticketing con tale licenza (questo sistema e stato successivamente abbandonato). La tabella fa anche riferimento al linguaggio adottato dal sistema, e si può notare che la maggior parte dei sistemi usa un linguaggio di tipo scripting ,questo probabilmente è dovuto al fatto che tali sistemi vanno ad interagire con delle pagine web. Un altro dato importante riguardava il Database(Back end) utilizzato dal sistema, MySQL risultò essere uno dei RDBMS più utilizzati in assoluto .

Di seguito viene descritta una breve presentazione dei gestori presi in considerazione, dettagliando nei prossimi capitoli il gestore che abbiamo deciso di utilizzare, e che quindi soddisfaceva di più le nostre esigenze.

Nome	Sviluppo	Licenza	Linguaggio	Database	Lancio
Bugzilla	Mozilla Foundation	*MPL*	*Perl*	MySQL	1998
JIRA	Atlassin	*Proprietaria*	*Java*	MySQL, Oracle	2003
MantisBT	Varie	*GPL*	*PHP*	MySQL, SQL Server	2000
OTRS	otrs.org	*AGPL*	*Perl*	MySQL, Oracle ,SQL Server	2002
Trac	Edgewall Software	*NewBSD*	*Python*	MySQL, SQLite	2003
OsTicket	OsTicket	*GPL*	*PHP*	MySQL	2009

Tabella 2.1: Sistemi di ticketing

2.3 Trac

Trac è un progetto open source. Il programma si ispira a CVS- Trac, ed è stato originariamente chiamato svntrac grazie alla sua capacità di interfacciarsi con Subversion. Trac è sviluppato e mantenuto da Edgewall Software, ed è scritto nel linguaggio di programmazione Python. Fino alla metà del 2005, era disponibile sotto la GNU General Public License, a partire dalla versione 0.9, è stato rilasciato sotto una licenza BSD modificata. Entrambi sono licenze di software libero. Trac attraverso un potente wiki permette la descrizione e il commit dei messaggi, la creazione di collegamenti e riferimenti per i bug, tasks, modifiche e file. Una cronologia mostra tutti gli eventi passati del progetto in ordine, rendendo molto facile il monitoraggio

del progresso di un progetto.

2.4 Bugzilla

Bugzilla è stato uno dei primi sistemi del suo genere ed è stato inizialmente sviluppato e usato dalla squadra che ha prodotto Mozilla, anche sè è stato rilasciato come un software open source da Netscape Communications nel 1998, Bugzilla è stato adottato anche come strumento per progetti con una licenza proprietaria.

2.4.1 Requisiti

Alcuni importanti requisiti necessari per poter usare Bugzilla :

- Un web server tipicamente Apache.
- Una adeguata versione di Perl 5.
- Un insieme di moduli Perl.
- Un server con un database compatibile (tipicamente MySQL).
- Un adeguato server SMTP (Protocollo di trasferimento postale).

2.5 Osticket

Grazie al lavoro svolto dai colleghi De Donno e Pilato, abbiamo potuto approfondire ed analizare meglio questo sistema sotto vari punti di vista. OsTicket è un pacchetto software per assistenza clienti, apparentemente semplice e affidabile, open source, scritto in php, con database MySQL, portabile, basato su web. OsTicket si basa principalmente sul concetto di ticket, attraverso i quali una segnalazione può essere inoltrata direttamente al reparto di competenza, scegliendo la categoria appropriata per il problema da esporre. L'apertura, la modifica e la chiusura del ticket vengono notificate tramite email.

2.5.1 Struttura

Dato che OsTicket basa il suo funzionamento sul concetto di ticket, a ciascun ticket aperto dall'utente corrisponde una richiesta di assistenza. Viene quindi assegnato un codice univoco (ID che identifica lo specifico ticket), attraverso il quale l'utente può seguire i progressi delle attività in corso. L'utente riceverà una mail con le istruzioni per accedere al sistema di gestione e avrà la possibilità di interagire con il Team di supporto per fornire ulteriori chiarimenti o seguire lo stato dei lavori e relative tempistiche fino al completamento delle attività e la relativa chiusura del ticket.

2.5.2 Features

Di seguito vengono elencate alcune delle caratteristiche principali di OsTicket :

- Supporto Web e Email :
  I ticket possono essere creati tramite email, o attraverso un interfaccia web.

- Risposte Automatiche :
  Quando viene aperto un nuovo Ticket può essere inviata una risposta/segnalazione automatica. Le risposte automatiche sono personalizzabili dallo staff.

- Risposte Predefinite :
  È possibile impostare delle risposte predefinite per le domande più frequenti.

- Note interne :
  È possibile aggiungere ai ticket delle note e dei commenti interni, per organizzare al meglio il lavoro interno dello staff.

- Avvisi e notifiche :
  Lo staff e gli utenti sono aggiornati con avvisi e-mail quando necessario. Queste impostazioni sono configurabili e flessibili.

- Accesso basato su ruoli :
  È possibile controllare il livello di accesso del personale, in modo da controllare se un determinato utente è un amministratore oppure fa parte dello staff.

- Assegnamento e trasferimento Ticket :
  Un ticket può essere assegnato ad un determinato membro dello staff o in generale ad un dipartimento.

- Non è richiesta registrazione :
  Non è richiesta nessuna registrazione da parte dell'utente, viene utilizzato l'indirizzo email e un codice univoco per il login.

- Archivio Ticket :
  Tutti i ticket e le risposte vengono archiviati in un database.

2.5.3 Pannello controllo amministratore

Il pannello di controllo dell'amministratore di OsTicket è il cuore del sistema, grazie ad esso è possibile configurare il sistema in base alle esigenze dell'utilizzatore. Le operazioni possibili sono: Visualizzare i file di log del sistema, configurare le preferenze di visualizzazione, configurazione email tecnico/settore, aggiungere o modificare il template standard o le risposte automatiche di sistema, aggiungere o rimuovere membri dello staff, creare gruppi, aggiungere o rimuovere dipartimenti e infine si ha la possibilità di gestire l'email in entrata.

2.5.4 Vantaggi

OsTicket è un sistema multiutente: più operatori possono lavorare simultaneamente sulle segnalazioni inserite nel sistema leggendo, catalogando e

rispondendo ai messaggi in arrivo. OsTicket ha una buona scalabilità consente di gestire migliaia di segnalazioni al giorno e un numero praticamente illimitato di operatori attivi contemporaneamente.

2.6 MantisBT

Mantis è stato publicato nel 2000, e con il tempo questo sistema ottenne più popolarità, cosi da essere considerato attualmente uno dei sistemi di ticketing Open-Source più usati. Essendo Mantis scritto come un PHP script esso può essere installato su un qualsiasi sistema operativo supportato da PHP, e che supporti i RDBMS usati da Mantis.

2.6.1 Features

Di seguito elenchiamo alcune delle caratteristiche di Mantis:

- Notifiche:
  con MantisBT è possibile mandare delle notifiche via email, infatti gli utenti hanno la possibilità di specificare e filtrare le email da ricevere relativamente ai vari bug. In aggiunta Mantis rende disponibile la possibilità di tener traccia del proprio stato di ticket attraverso la sua interazione con un social network come Twitter, e dei link RSS.

- Plug-ins:
  nella versione 1.2.0 di mantis viene introdotto un sistema di plug-in che permette l'aggiornamento del sistema sia da enti ufficiali che da terze parti.

- Altro:
  MantisBT permette altre piccole Features come la possibilità di mantenere un record aggiornato dinamicamente per ogni ticket che tiene traccia dei cambiamenti effettuati su di esso, un meccanismo di controllo versione (Revision Control) dei campi di testo e la possibilità di effettuare una ricerca nel database del sistema.

2.7 Scelta del gestore di ticket

Abbiamo potuto notare fin da subito come il sistema scelto in precedenza (OsTicket) non rappresentasse la miglior soluzione vista sopratutto la mancanza di alcuni requisiti che sono via via venuti a mancare. In particolare, poter contare su una community attiva alle spalle del software, in grado quindi di rilasciare aggiornamenti con cadenza continua e di risolvere eventuali bug in modo tempestivo, era prerequisito indispensabile. Inoltre, OsTicket è risultato in più di una circostanza di difficile comprensione a livello di scelte implementative; questo, non tanto per la complessità delle sue funzioni, quanto per la poca modularità del suo codice. Con una distinzione poco evidente tra il codice HTML e quello PHP, si doveva mettere mano al codice in maniera piuttosto massiccia, correndo il rischio che ad un successivo aggiornamento software i problemi tornassero ancora.

2.7.1 Caratteristiche richieste

In particolare la nostra scelta è stata effettuata dopo aver preso nota di alcuni requisiti e caratteristiche ritenute importanti :

Licenza Open Source: il sistema deve avere una licenza libera (GPL ,AGPL etc)

Ricerca: La possibilità di cercare all'interno del sistema tutti i ticket ricevuti e quindi di poterli visualizzare in modo da poter tener traccia dei propri ticket e di apportare delle modifiche nel caso fosse necessario.

Interfaccia web: Deve essere possibile utilizzare il gestore di ticketing attraverso un interfaccia web semplice ed intuitiva.

SystemLog: La possibilità di avere un file aggiornato sullo stato del sistema è molto importante, grazie ad esso è possibile verificare che tutti gli accessi e tutte le operazioni sono avvenute correttamente.

Ticket by mail: la possibilità di ricevere una mail con un client di posta qualsiasi e poterla visualizzare in forma di ticket.

Single sign on: la possibilità di poter integrare il sistema con il modello Single sign on (SSO)

Il Codice: il codice del sistema deve essere quanto più possibile modulare in modo da permettere eventuali modifiche future.

Dopo aver analizzato i vari sistemi con particolare attenzione ai requisiti appena descritti e dopo una analisi delle esigenze del dipartimento, ci siamo concentrati sul sistema OTRS, il quale soddisfa al meglio le caratteristiche richieste .

Capitolo 3

Analisi dei requisiti

Prima di iniziare ad approfondire il sistema di ticketing scelto(OTRS), abbiamo analizzato il lavoro svolto in precedenza, il quale attraverso un accurato studio dei requisiti e delle esigenze del personale del dipartimento, aveva descritto i possibili scenari verificabili nella nostra rete. Ci siamo quindi limitati a studiare gli strumenti già esistenti, nei loro aspetti positivi e negativi e a verificare che fossero state considerate tutte le possibili situazioni.

Per capire quali tipi di richieste di supporto vengono sottomesse al personale da parte degli studenti e come poterle classificare si è scelto di analizzare subito il Newsgroup.[1]

3.1 NewsGroup

Il Newsgroup è uno strumento molto importante per gli studenti e i professori del dipartimento di Scienze dell'Informazione; viene utilizzato per lo scambio di qualsiasi informazione didattica, proveniente sia dai docenti sia dagli studenti, e soprattuto per le richieste di supporto tecnico e amministrativo. Tali richieste, oltre ad essere sempre più numerose con conseguente aumento del carico di lavoro che il personale è sottoposto a gestire, sono

[1]Si fa riferimento al Newsgroup per studenti del dipartimento di Scienze dell'Informazione disponibile all'indirizzo https://webnews.cs.unibo.it.

anche simili tra loro. Ecco perchè si è cercato di utilizzare un sistema di ausilio al Newsgroup che potesse snellire il sistema permettendo di gestire tutte le richieste nel miglior modo possibile, con la priorità adeguata ai vari temi trattati, evitando inefficienti ripetizioni.

3.2 Smistamento ticket

Per quando riguardo lo smistamento dei ticket al settore di competenza, inizialmente abbiamo definito delle macroaree, cioè una lista di argomenti visibili solo all'utente abilitato a chiedere un servizio. Oltre alle macroaree è possibile specificare anche delle microaree per poter informare in modo più dettagliato lo staff sul tipo di problema riscontrato.

Ogni macroarea viene associata ad un gruppo di appartenenza, all'interno del quale conferiranno sia gli utenti abilitati sia alcuni membri dello staff incaricati di prendere in consegna e gestire le richieste.
In questo modo è possibile isolare e definire in modo preciso ogni singolo problema, lasciando che uno specifico utente possa intervenire solo sulle aree problematiche collegate con il proprio ruolo all'interno della rete universitaria.

3.2.1 Macroaree e Microaree

La seguente struttura descrive tutte le possibili richieste che possono essere fatte da un utente che ha bisogno di un supporto tecnico o amministrativo. Alcune macroaree sono visibili solo agli utenti abilitati a chiedere un servizio e nascoste per sicurezza agli altri utenti. Lo scenario sarà più chiaro con le tabelle 3.1, con le quali si vedrà in modo chiaro quali sono le associazioni fatte tra i tipi di richieste creabili ed i tipi di utenti.

1. Attrezzature aule audio, video, inf.

2. Attrezzature informatiche dipartimento

3. Prenotazioni lab. per esercitazioni/esami
4. Allarme
5. Manutenzione infrastruttura
6. Forniture / Preventivi
7. Controllo accessi / Badge
8. Stampa
    - Problemi Hardware
    - Problemi Software
9. Attivazione Servizi
    - Pagine web dinamiche
    - Database
    - Gruppi e spazi di lavoro condivisi
    - Repository
    - Microaree aggiuntive solo per alcuni utenti
        - Newsgroup
        - Alias
        - Mailing List
10. Email
11. Macchine lab. e trusted
    - Bloccata
    - Problemi hardware
    - Processi
12. Installazione/Configurazione/Aggiornamenti

13. Connessione e rete
    - Wireless
    - Accesso da remoto
    - Microaree aggiuntive solo per alcuni utenti
        - Assegnamento IP/DNS
        - Rete Cablata
        - VPN
14. Account
    - Quota
    - Permessi
    - Microaree aggiuntive solo per alcuni utenti
        - Estensione validità
        - Creazione
15. Altro

Categoria	**Aree Visualizzate**
Docenti e Ricercatori	1 3 5 6 7 8 9 10 11 12 13 14 15
Studenti	5 7 8 9(in parte) 11 12 13(parte) 14(parte) 15
Tecnici	1 3 5 6 7 8 9 10 11 12 13 14 15
Amministrazione	5 6 7 8 10 14(parte) 15
Segreteria	1 2 3 4 5 6 7 8 10 14(parte) 15
Sorveglianti	1 2 4 5 7 8 11 12 13(parte) 14(parte) 15
Tutor	5 6 7 8 10 11 12 13(parte) 14(parte) 15
PhD, Post. dott, Assegnisti	1 3 5 6 7 8 9 10 11 12 13 14 15

(a) Associazione Categoria-Aree visualizzate

Gruppo	**Aree**
Gruppo1	1
Gruppo2	2 4
Gruppo3	3
Gruppo4	5 7 8 14 15
Gruppo5	6 10
Gruppo6	9
Gruppo7	11 12 13

(b) Associazione Gruppo-Aree

Categoria	**Gruppi**
Docenti e Ricercatori	1 3 4 5 6 7
Studenti	4 6 7
Tecnici	1 3 4 5 6 7
Amministrazione	4 5
Segreteria	1 2 3 4 5
Sorveglianti	1 2 4 7
Tutor	4 5 7
PhD, Post. dott, Assegnisti	1 3 4 5 6 7

(c) Associazione Categoria-Gruppi

Tabella 3.1: Relazioni tra ruoli

Capitolo 4

OTRS

OTRS Help Desk è un'applicazione installata su un server web che può essere usata attraverso un qualunque browser.

OTRS è separato in diversi componenti; quello principale è un framework che contiene tutte le funzionalità utilizzate dall'applicazione e dal sistema di gestione dei ticket.
Attraverso l'uso dell'interfaccia web è possibile installare dei pacchetti addizionali come i moduli ITSM (*IT Service Management*), un modulo per le FAQ oppure per il monitoraggio della rete.

Fin dalla sua nascita OTRS è stato implementato in linguaggio Perl. L'interfaccia web è resa più user-friendly dall'utilizzo di JavaScript. Inoltre, essa utilizza un meccanismo di creazione dei modelli (template) chiamato DTL (*Dynamic Template Language*)per rendere più semplice la visualizzazione dei dati prodotti dal sistema.

4.1 Features

Di seguito possiamo menzionare in un elenco ,assolutamente non completo, delle principali caratteristiche di OTRS:

- Web interface:
  - Compatibilità con tutti i browser;

  - Compatibilità con smartphone(IPhoneHandle);
  - Supporto multi linguaggio;

- Mail interface:
  - supporto per allegati nelle mail (MIME);
  - filtraggio delle mail tramite gli header X-OTRS;
  - supporto PGP;
  - supporto gestione messaggi S/MIME;
  - risposte automatiche;
  - notifiche email per gli agenti;

- Tickets:
  - lock" dei ticket;
  - cronologia di ogni singolo ticket;
  - inserimento note personali;
  - controllo degli accessi alle liste dei ticket;
  - trasferimento ticket tra code;
  - ricerca tramite tag o tramite full text;

- Sistema:
  - supporto per qualunque S.O.;
  - supporto per i maggiori database (MySQL,PostgreSQL,Oracle,MSSQL);
  - integrazione con database (back-end) esterni;
  - autenticazione clienti tramite DB, LDAP, HTTPAuth;
  - sottocode;
  - differenziazione di risposte e firme per ogni coda;

4.2 Area ADMIN di OTRS

Gli amministratori di OTRS hanno a disposizione una pagina di amministrazione che permette la totale configurazione del sistema, aggiungendo agenti, clienti, gruppi, code, definendo i parametri di invio delle email, gestione dei ticket e molto altro.

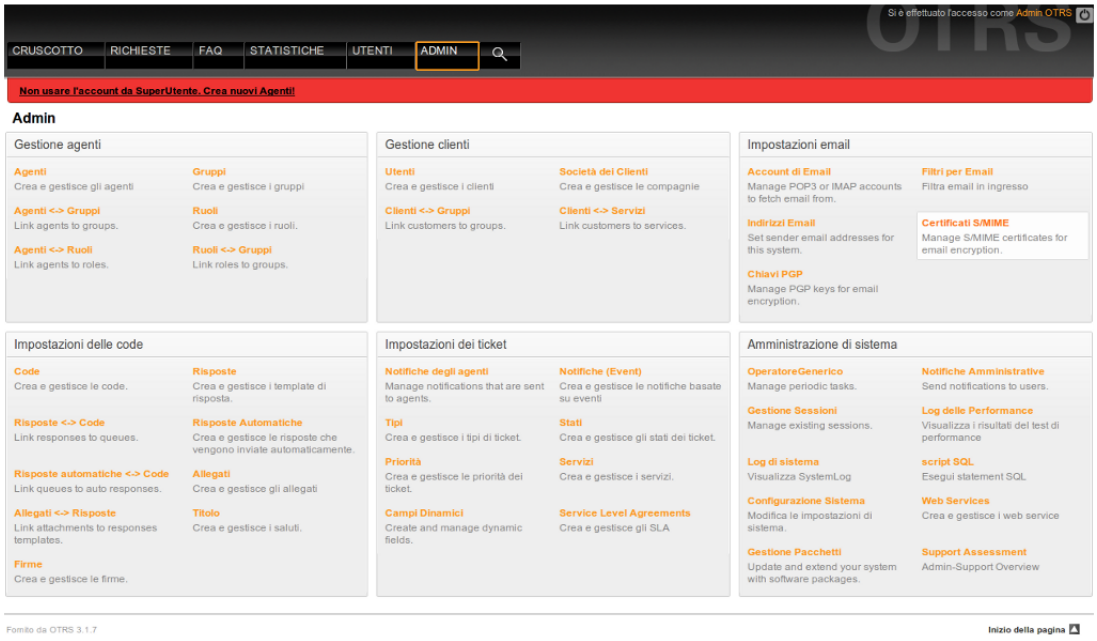

Figura 4.1: Area Amministrazione

4.2.1 Agenti, Gruppi, Ruoli

Agenti

Nella filosofia di OTRS, gli agenti non sono altro che dei "dipendenti" specializzati nella gestione e risoluzione di un determinato tipo di problema all'interno della rete. Nel nostro caso però, essi sono anche gli amministratori del servizio; questo avviene perchè sono proprio i tecnici del Dipartimento che si occuperanno di autogestire le proprie problematiche, distribuendosele secondo criteri da loro scelti.

Gruppi

Come già accennato nella sezione 3.2, in ogni gruppo confluiscono tutti gli utenti che potranno poi accedere ad una determinata coda di richieste ed ogni singolo utente può appartenere a più gruppi contemporaneamente. Una delle cose più interessanti di questo tipo di approccio è la possibilità di cambiare "in corsa" uno dei gruppi di appartenenza di un utente(per esempio in seguito ad un nuovo ruolo aziendale)senza intaccare le caratteristiche di appartenenza precedenti.

Ruoli

I ruoli sono uno strumento molto potente utilizzato per definire dei livelli di accesso per ogni singolo agente. In particolare, risultano essere molto utili in ambienti con un alto numero di agenti, gruppi e code.

L'associazione coda-ruolo è un vincolo più forte rispetto a quella coda-gruppo, perciò, se ci trovassimo di fronte alla necessità di aggiungere delle nuove problematiche da far gestire agli utenti, potremmo semplicemente creare un nuovo ruolo da assegnare ad un certo gruppo di agenti anzichè modificare manualmente ogni singolo permesso.
Una panoramica più esauriente dei permessi rilasciabili agli utenti è data dalla tabella 4.1.

Diritti	Descrizione
ro	accesso solo in lettura ai ticket e code di questo gruppo
move into	diritto a spostare ticket tra code di questo gruppo
create	diritto a creare ticket in code di questo gruppo
owner	diritto ad aggiornare il proprietario di ticket di code di questo gruppo
priority	diritto a cambiare la priorità di ticket di code di questo gruppo
rw	accesso totale al gruppo

Tabella 4.1: Tabella diritti gruppi

4.2.2 Clienti

I clienti sono l'utente più semplice presente nell'ambiente di un Issue Tracking System, nonchè l'utilizzatore finale di esso.

Anche i clienti possono essere associati a più gruppi contemporaneamente, dandogli così la possibilità di poter accedere a multiple code di richiesta.

4.2.3 Code

Le possibili richieste individuate nel paragrafo 3.2.1 prendono il nome di code di richieste. Un aspetto molto interessante riguarda la gestione di un ticket nel momento in cui viene bloccato, cioè preso in gestione, da un agente. È possibile infatti assegnare ad ogni coda un *unlock timeout* che, una volta scaduto, libera i ticket che non sono ancora stati segnalati come risolti.
Esistono tre differenti metodi di escalation:

Escalation - First Response Time: dopo la creazione di un ticket, se il timeout scade senza alcuna comunicazione verso il cliente allora il ticket viene rilasciato;

Escalation - Update Time: se ci sono dei follow-up, il timeout viene risettato; se a questo punto il timeout scade senza comunicazioni il ticket viene rilasciato;

Solution Time: se il ticket non viene chiuso prima della scadenza del timeout, il ticket viene rilasciato.

Come già accennato in precedenza, ogni coda può essere associata al più ad uno solo gruppo.

Capitolo 5

Scelte Tecniche

In questo capitolo, prima ancora di entrare nello specifico del lavoro svolto, analizzeremo le scelte prese in collaborazione con i tecnici per avere l'architettura di base più congeniale alle nostre esigenze; vedremo i motivi che ci hanno portato a sceglierne una piuttosto che un'altra senza tralasciare gli eventuali problemi.

5.1 Incontri con il personale tecnico

Durante i vari incontri con i tecnici del dipartimento sono stati trattati tre argomenti: la configurazione e lo smistamento corretto dei ticket, l'autenticazione degli utenti basata su un'architettura SSO e l'ambiente di lavoro per installare configurare e ottimizzare l'*Issue Tracking System* scelto.

Il primo è stato trattato nel paragrafo precedente,mentre il secondo argomento verrà sviluppato successivamente, quando parleremo dell'analisi dei requisiti e dell'autenticazione tramite Shibboleth; vediamo ora invece il terzo argomento.

5.2 Ambiente di sviluppo

Ovvero determinare un ambiente dove poter sviluppare, modificare e testare il nostro sistema. Nel nostro caso abbiamo adottato due approcci diversi.

5.2.1 Approccio Reverse proxy

Fin dal primo incontro con il personale tecnico, si è pensato di utilizzare un apposito spazio all'interno del server Loew del Dipartimento. Questa soluzione, in realtà, prevedeva la creazione di un apposito spazio di test ed uno per la produzione finale dell'applicazione.

L'architettura del server Loew prevede che la comunicazione client-server avvenga attraverso l'utilizzo di un reverse-proxy specifico.
Un reverse-proxy permette agli utenti di internet di accedere indirettamente ad alcuni server interni ad una rete; grazie ad esso, il server web è protetto dagli attacchi provenienti dall'esterno, cosa che rinforza la sicurezza della rete interna, d'altra parte, la funzione di cache del reverse-proxy può alleggerire il carico del server per cui è previsto ed è la ragione per cui un server del genere è detto anche acceleratore.

Va precisato che non tutte le applicazioni ospitate sul server Loew sono gestite tramite reverse proxy, bensì solo quelle che necessitano di una connessione https con certificato SSL rilasciato da un *certification authority* riconosciuta.

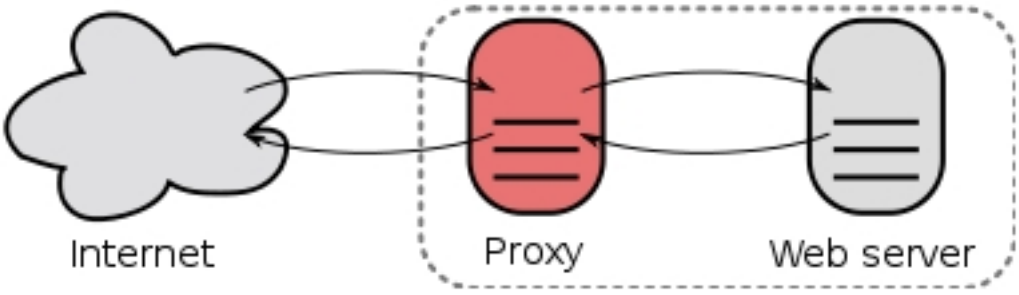

Figura 5.1: Reverse-proxy su server Loew

Lavorando in questo ambiente però, ci siamo imbattuti da subito in una serie di problemi che hanno rallentato notevolmente i nostri progressi.
In particolare:

- Mancanza di permessi sugli utenti *rschirin* e *heibi*
- Procedura d'installazione di OTRS non convenzionale
- Rottura link dinamici

Se lavorare con dei permessi limitati sui nostri account oppure controllare singolarmente i pacchetti, per evitare eventuali conflitti con altre applicazioni precedentemente installate sul server, erano risolvibili lavorando a stretto contatto con il personale tecnico, lo stesso non si può dire della rottura dei link generati dal software. Più in dettaglio, quando OTRS inviava al client un messaggio contenente un ulteriore link verso se stesso, il client non sapeva come interpretarlo visto il filtraggio eseguito dal proxy.

Una possibile soluzione sarebbe stata rappresentata da una massiccia riscrittura del codice dell'architettura del server, cosa che divergeva dall'obiettivo del nostro lavoro.

5.2.2 Approccio Virtual Machine

Abbiamo così deciso di installare una macchina virtuale sul server Storage del Dipartimento, proprio per risolvere i problemi riscontrati con la tecnica precedente.

Questa soluzione, a fronte di una leggera perdita prestazionale rispetto alla macchina fisica, ci permette di avere grande libertà di azione, seguendo passo per passo la procedura standardizzata di installazione di OTRS.
Un altro aspetto interessante riguarda le fasi di testing; esse non compromettono in nessun modo il sistema principale, permettendoci di non dover pensare ad avere uno spazio dedicato alla fase di testing ed uno alla fase di produzione, semplificando notevolmente il lavoro di configurazione. Infine,

la virtualizzazione ci ha permesso di avere una più semplice ed efficace procedura di backup e ripristino dell'intero sistema.
Sulla macchina virtuale, chiamata *ticket* e riferita all'IP *130.136.2.123*, abbiamo installato Ubuntu 12.04 LTS.

Capitolo 6

Configurazione OTRS

6.1 Requisiti Software

- Perl 5.8.8;
- Apache2 + modulo perl2;
- Webserver con supporto CGI;
- MySQL 4.1 o superiore;

6.2 Installazione

Una volta scaricato ed scompattato il pacchetto ufficiale, abbiamo installato il sistema nella cartella /opt/otrs.
A questo punto abbiamo controllato la presenza dei seguenti pacchetti tramite il comando *aptitude*:

- ldfibapache2-mod-perl2
- libdbd-mysql-perl
- libnet-dns-perl
- libnet-ldap-perl

- libio-socket-ssl-perl
- libpdf-api2-perl
- libsoap-lite-perl
- libgd-text-perl
- libgd-graph-perl
- libapache-dbi-perl
- mysql-server

A questo punto creiamo un utente specifico che possa eseguire i cron jobs di OTRS; questo verrà aggiunto al gruppo del webserver ed avrà come homedir /opt/otrs.

```
useradd -r -d /opt/otrs/ -c 'OTRS user' otrs
usermod -g www-data otrs
```

OTRS presenta dei file di configurazione di default, che vengono copiati nella cartella Kernel e che verranno modificati in seguito.

```
cp Config.pm.dist Config.pm
cp Config/GenericAgent.pm.dist Config/GenericAgent.pm
```

Utilizzando uno degli script di configurazione di OTRS, abbiamo impostato i permessi di utilizzo del webserver:

```
bin/otrs.SetPermissions.pl --otrs-user=otrs --otrs-group=otrs
--web-user=www-data --web-group=www-data /opt/otrs
```

Abbiamo aggiunto il file di configurazione di OTRS dentro apache2

```
cp /opt/otrs/scripts/apache2-httpd.include.conf \
/etc/apache2/conf.d/otrs.conf
service apache2 restart
```

A questo punto abbiamo seguito la procedura del web-installer che si occupa esclusivamente di associare il giusto DB (host, user db, user psw) al sistema installato. Infine abbiamo configurato i cron jobs che si occuperanno, tra i vari, di controllare la presenza di ticket pendenti, di ripulire la cache del sistema, di eliminare la presenza di precedenti ID di sessione.

```
for foo in *.dist; do cp $foo ‘basename $foo .dist’; done
bin/Cron.sh start otrs
```

6.3 Customizzazione

6.3.1 Configurazione modulo Perl

Una volta terminata l'installazione abbiamo modificato il file *apache2-perl-startup.pl*; in particolare abbiamo caricato i driver del database MySQL nella memoria del webserver che, seppur non obbligatorio, permette un aumento delle prestazioni.

```
use DBD::mysql ();
use Kernel::System::DB::mysql;
```

6.3.2 Configurazione Apache con SSL

Terminata la fase di installazione ci siamo occupati, insieme a Paolo Marinelli, di configurare il web server Apache in modo tale da permettere l'autenticazione tramite SSL. Abbiamo attivato il modulo apache che supporta l'autenticazione SSL:

```
a2enmod ssl
```

quello che permette la riscrittura delle regole di configurazione

```
a2enmod rewrite
```

e quello che gestisce il modulo perl

```
a2enmod perl
/etc/init.d/apache2 force-reload
```

Virtual Host e RewriteRule

A questo punto, abbiamo creato due Virtual Host per gestire le richieste di connessione in transito sulla porta 80 (http) e sulla porta 443 (https).

```
cp /etc/apache2/sites-available/default \
/etc/apache2/sites-available/ticket.cs.unibo.it


cp /etc/apache2/sites-available/default-ssl \
/etc/apache2/sites-available/ticket.cs.unibo.it-ssl
```

Per capire meglio la loro funzione vediamone i contenuti:

```
ticket.cs.unibo.it


<VirtualHost *:80>
ServerAdmin ticket@cs.unibo.it
ServerName ticket.cs.unibo.it
DocumentRoot /var/www
<Directory />
Options FollowSymLinks
AllowOverride None
</Directory>
<Directory /var/www/>
...
</Directory>
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}
....
</VirtualHost>
```

Il virtual host in questione resta in ascolto sulla porta 80, definendo come DocumentRoot la cartella di default /var/www, contenente un semplice file index.html che sarà in seguito modificato. Questo virtual host è di tipo *Ip-based* che permette di associare un unico indirizzo IP per ogni host virtuale. Nella parte finale abbiamo utilizzato il modulo rewrite per poter redirigere i visitatori verso il corrispondente URL in https.

`ticket.cs.unibo.it-ssl`

```
<IfModule mod_ssl.c>
<VirtualHost *:443>
ServerAdmin webmaster@localhost
ServerName ticket.cs.unibo.it
DocumentRoot /var/www
....
....
....
RewriteEngine on
RewriteRule ^/admin/?$ https://ticket.cs.unibo.it/otrs/\
index.pl [R,L]
RewriteRule ^/customer/?$ https://ticket.cs.unibo.it/otrs/\
customer.pl [R,L]
</VirtualHost>
</IfModule>
```

Questo secondo virtual host lavora sulla porta 443 e viene utilizzato per eseguire lo "split" grafico tra agenti e semplici utenti OTRS.
Mentre i primi vengono reindirizzati verso la pagina a loro dedicata index.pl, i secondi vengono reindirizzati verso la pagina di login customer.pl; il tutto sempre tramite URL https://.

Infine, abbiamo abilitato il plugin ssl:

```
a2ensite ssl
```

6.3.3 index.html

L'ultimo file ad essere modificato é index.html presente nella cartella /var/www che riguarda la homepage dello spazio web a nostra disposizione.

Seguendo le linee guida del Cesia, abbiamo creato una pagina di benvenuto per tutti gli utenti semplici che provano l'accesso dalla pagina `https://ticket.cs.unibo.it`.

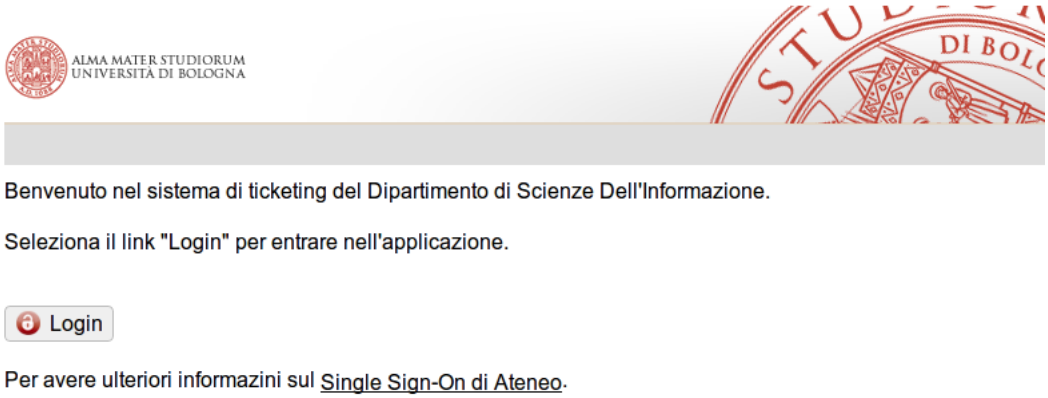

Figura 6.1: Homepage del sistema

L'utente accede all'applicazione web selezionando il link "Login", che lo reindirizzerà verso la pagina dell'IdP configurato precedentemente:

```
<div class="corniceLogin rounded">
<a href="https://ticket.cs.unibo.it/otrs/customer.pl"
id="linkLogin">Login</a>
</div>
```

6.4 Gestione DB

OTRS si occupa in fase di installazione di configurare nella maniera corretta il dabatabase creato, importando dei file come modelli necessari per la definizione del db.

otrs-schema.mysql.sql: è il primo file che viene importato e serve per creare tutte le tabelle necessarie per il giusto funzionamento del sistema.

otrs-initial-insert.mysql.sql: con esso vengono creati gruppi e code di default e viene inserito l'utente root. Questo file è stato modificato in fase di customizzazione, inserendo al suo interno tutte le macroaree ed i gruppi visti nel paragrafo 3.2.1.

otrs-schema-post.mysql.sql: con esso vengono create tutte le associazioni tra le varie tabelle grazie alle FOREIGN KEY.

6.4.1 PHPMyAdmin

L'interfaccia grafica di OTRS è estremamente professionale e completa, mettendo a disposizione degli amministratori tutti gli strumenti necessari per la gestione del DB presente alla base del sistema.
Non è però da escludere la necessità di voler modificare qualcosa che non può essere risolto passo dopo passo tramite interfaccia. Il programma mette così a disposizione una funzionalità chiamata *script SQL* accessibile dal pannello di amministrazione.

Pensiamo se ci fosse il bisogno di azzerare tutti i campi "comments" delle code; per quanto possa sembrare un'operazione banale, dobbiamo ricordarci che un sistema di ticketing è pensato per ottimizzare al massimo il lavoro del personale tecnico di un ambiente lavorativo(azienda, facoltà etc). Con una semplice query è così possibile guadagnare tempo.

Se neanche questà funzionalità risultasse sufficiente, abbiamo deciso di installare sulla virtual machine PhpMyAdmin.

PhpMyAdmin è un'applicazione PHP libera che consente di amministrare in modo semplificato database di MySQL tramite un qualsiasi browser. PhpMyAdmin permette di creare un database da zero, creare le tabelle ed eseguire operazioni di ottimizzazione sulle stesse. Presenta un feedback sulla creazione delle tabelle per evitare eventuali errori. Sono previste delle fun-

zionalità per l'inserimento dei dati (popolazione del database), per le query, per il backup dei dati, ecc..

L'amministratore, invece ha a disposizione un'interfaccia grafica per la gestione degli utenti: l'interfaccia permette l'inserimento di un nuovo utente, la modifica della relativa password e la gestione dei permessi che l'utente ha sul database.
Anch'esso lavora tramite SSL, grazie alle RewriteRule viste nel paragrafo 6.3.2. Una volta installato si crea un link simbolico del suo file di configurazione del web server dentro Apache:

```
ln -s /phpmyadmin/apache.conf /apache2/conf.d/phpmyadmin.conf
```

per poi aggiungere dentro */phpmyadmin/config.inc.php* la voce:

```
$cfg['ForceSSL'] = true;
```

Di seguito mostriamo come PHPMyAdmin mostra uno schema di come vengono visualizzati le varie tabelle dei DB utilizzati in OTRS e le relazioni tra di loro già viste nei paragrafi precedenti.

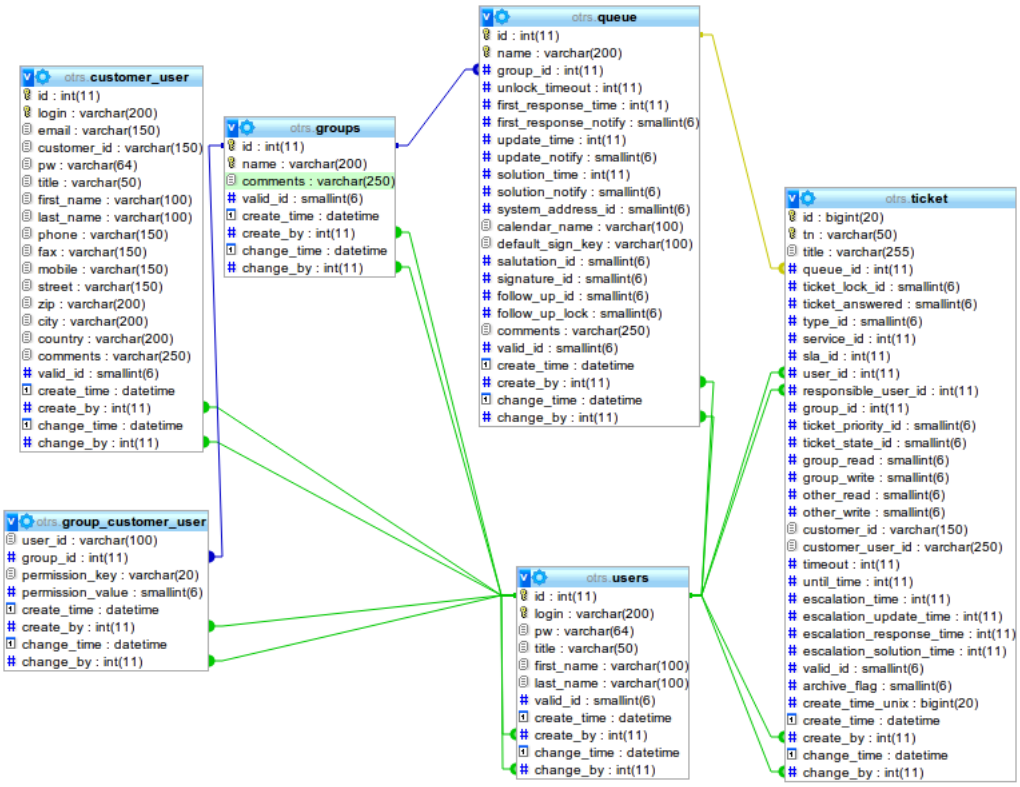

Figura 6.2: Relazione tabelle DB

6.5 OTRS Config.pm

Tutti i file di configurazione di OTRS sono contenuti nella directory *Kernel* e nelle sue sottodirectory. Questi vengono utilizzati come modelli dai quali prendere i parametri di configurazione per inserirli nell'unico file personalizzabile che è *Kernel/Config.pm*.
Inseriti all'interno di esso, i parametri possono essere modificati secondo le proprie necessità, sicuri che non verranno mai toccati durante il processo di aggiornamento.

Il file *Defaults.pm* contiene i parametri del framework centrale e definisce tutte le caratteristiche di base del sistema, come la configurazione mail, la connessione database, il charset ed il linguaggio di default.

Quando viene utilizzata l'interfaccia grafica di OTRS, tutti i file .xml dentro *Config/Files* vengono controllati in ordine alfabetico. Gli ultimi due, *ZZZAAuto.pm* e *ZZZAuto.pm* sono probabilmente i più importanti; il primo viene generato automaticamente dopo il primo accesso con l'utente root, mentre il secondo viene generato non appena effettuiamo qualche modifica configurativa.

```
ZZZAuto.pm file
$Self->{'CustomerGroupAlwaysGroups'} =  ['G4'];
$Self->{'CustomerGroupSupport'} =  '1';
delete $Self->{'PreferencesGroups'}->{'SpellDict'};
$Self->{'DefaultLanguage'} =  'it';
$Self->{'Organization'} =  '';
$Self->{'FQDN'} =  'ticket.cs.unibo.it';
$Self->{'SecureMode'} =  1;
}
# HttpType
# In case you use https instead of plain http specify it here
    $Self->{HttpType} = 'https';
```

Abbiamo abilitato la SecureMode, per impedire che la web-installation potesse essere ripetuta mettendo in serio pericolo la sicurezza del sistema, e abbiamo specificato il tipo di autenticazione. Infine, abbiamo eseguito il seguente script:

```
/bin/otrs.CryptPassword 'my-password'
```

per criptare con la funzione crypt() la password dell'amministratore del database di OTRS. Il risultato è stato poi inserito dentro Config.pm

```
# DatabasePw
# (The password of database user. run bin/
# otrs.CryptPassword.pl# for crypted passwords)
    $Self->{'DatabasePw'} = ********;
```

6.6 FAQ

Nel corso dei nostri incontri con il personale tecnico era emersa la necessità di potersi affidare ad un software in grado di fornire un micro wiki che potesse risolvere alcuni problemi prima ancora di aprire una richiesta di assistenza. Abbiamo così ritenuto opportuno installare il modulo estensivo FAQ che fornisce un metodo estremamente rapido di comunicazione tra agenti e/o clienti, indipendentemente dalla presenza o meno di un ticket da gestire o creare. Senza entrare nello specifico del suo funzionamento, questo modulo segue la filosofia utilizzata per la parte di Help Desk, dando la possibilità agli agenti di creare e gestire delle FAQ e concedere permessi ai clienti in modo che questi possano accedere al servizio autonomamente. Come è ovvio immaginare, gli utenti semplici non avranno i permessi per creare o modificare le FAQ.

6.6.1 Features

FAQ explorer: navigazione intuitiva attraverso le diverse tipologie di FAQ;

Articoli FAQ: accurato inserimento di una FAQ attraverso vari attributi come “Sintomi”, “Problemi”, “Soluzione”, “Commenti”, “Categoria”, “Stato”;

FAQ attachments: si possono inserire allegati che verranno poi visti dagli utenti;

Full-text and quick search: ricerca completa o rapida; si possono usare anche gli operatori AND e OR;

6.7 Modulo IPhoneHandle

Questo modulo fornisce gli strumenti necessari per la gestione e la presa in carico dei ticket da parte dei tecnici. Nativo per sistemi Apple, ora può essere utilizzato anche su sistemi Android.
La sua installazione segue la procedura standard, quindi viene scaricato dal menu di amministrazione “Gestione Pacchetti”. L’installazione prevede la creazione di uno script Perl che dovrà essere specificato durante l’accesso dal dispositivo mobile.

6.7.1 DS Helpdesk

L’applicazione utilizzata può essere scaricata facilmente dal market di Android; una volta avviata i dati da immettere sono i seguenti:

Nome Helpdesk: nome identificativo del sistema;

URL: http(s)://FQDN/ScriptAlias/json.pl;

username: username agente;

password: password agente;

6.7.2 OTRS app - IPhone

Applicazione ufficiale OTRS per IPhone, permette di creare, spostare o chiudere un ticket, aggiungere una nota, rispondere o semplicemente controllare la presenza di nuovi. Una volta avviata ci verranno richiesti gli stessi dati visti nella sezione precendente.

Capitolo 7

Single Sign On

Il Single sign on (SSO) può essere tradotto come autenticazione unica o identificazione unica, ovvero una proprietà che può essere integrata ad un sistema ,così da permettere ad un utente di autenticarsi al sistema una sola volta e di accedere a tutte le risorse informatiche alle quali è abilitato.

Quindi, un sistema basato su Single Sign On(SSO) fa si che le richieste di autenticazione da parte degli utenti non vengano direttamente gestite dal sistema stesso ma vengano ridirette verso un altro sistema che precedentemente ha già verificato le credenziali d'accesso dell'utente connesso. In questo modo si elimina la necessità di richiedere nuovamente una login e una password per un utente già connesso ma allo stesso tempo si permette la condivisione di risorse tra più organizzazioni.

Un sistema Single Sign On può essere di tre tipologie:

Approccio centralizzato: le informazioni di autenticazione di tutti gli utenti sono gestite da un'autorità centrale che comunica con i domini secondari attraverso protocolli come SAML.

Approccio federativo: l'autenticazione è realizzata da più SSO indipendenti (ma federati tra loro), ognuno dei quali possiede un'identità parziale dell'utente.

Approccio cooperativo: ogni singolo utente dipende da uno dei singoli gestori cooperanti. Per cui, se si tenta di accedere alla rete locale, l'autenticazione sarà effettuata unicamente dal gestore da cui dipende l'utente.

7.1 Obiettivi

Alcuni importanti obiettivi del suo utilizzo :

- gestione delle password:
  maggiore è il numero delle password da gestire, maggiore è la possibilità che vengano utilizzate password simili e facili da memorizzare, abbassando così il livello di sicurezza;

- semplificare la gestione degli accessi ai vari servizi:
  consente una gestione centralizzata degli utenti per tutti i portali e le applicazioni esistenti; in questo modo si evita la necessità di mantenere un database di utenti per ogni applicazione superando tutti i problemi di gestione che esso comporta;

- semplificare la definizione e la gestione delle politiche di sicurezza:
  in particolare aumenta la sicurezza delle credenziali mediante certificati X.509 in un unico ambiente centralizzato in modo tale da evitare di integrare in ogni applicazione la loro gestione ed interfacciarle singolarmente con il database utenti;

- gestire l'autenticazione e l'autorizzazione attraverso il sistema SSO e non attraverso le diverse applicazioni:
  in questo modo la manutenzione sarà effettuata modificando direttamente le impostazioni utente presenti nel database centrale.

7.2 Architettura

Il meccanismo che sta alla base dei sistemi di SSO è analogo a quello di una struttura Client/Server organizzati in 3 livelli: livello utente, livello applicazione, livello dati, dove ogni livello ha un ruolo ben preciso e si comporta contemporaneamente sia da Client che da Server. Da Client per il livello inferiore (usufruisce dei servizi offerti dal Server) e da Server per il livello superiore (offre determinati servizi).

Avremo il livello Utente che rappresenta il nostro livello utente, ossia le postazioni fisiche che accedono alle applicazioni esistenti e che richiedono servizi al livello inferiore, il livello Applicazione è l'insieme dei Server sui quali girano le applicazioni, che ricevono le richieste dai Client e che offrono il servizio richiesto ed il livello Dati indica l'insieme dei database dove risiedono le informazioni ed ha il compito di processare le richieste e di restituirle al livello superiore.

Questa struttura porta notevoli vantaggi, poichè anzitutto ogni livello può accedere solo al livello immediatamente sottostante e secondariamente perchè permette ai vari livelli di essere completamente indipendenti tra loro, cosa che tra l'altro favorisce la gestione del sistema stesso. Per tutti questi motivi, questo tipo di architettura è definita chiusa

Lo stato di autenticazione di un client è memorizzato attraverso un ticket emesso dal sistema di SSO (l'utente è in possesso di questo ticket). Quando un'applicazione richiede il riconoscimento di un utente, il ticket gli viene inviato in modo automatico. Sarà l'applicazione a contattare il sistema di SSO per verificare la validità del ticket: in caso di esito positivo saranno richieste le informazioni utente, mentre in caso di esito negativo sarà generato un messaggio di errore di autenticazione.

7.3 Shibboleth

Shibboleth rappresenta un software open source per l'accesso a risorse e servizi web condivisi tramite credenziali di autorizzazione. Esso modifica

il modo in cui vengono controllati gli accessi e sposta verso le strutture di appartenenza degli utenti il compito di dare garanzie sulla loro identità e sui loro diritti attraverso la realizzazione di una Infrastruttura di Autenticazione ed Autorizzazione. Lo scopo del progetto è l'implementazione di un sistema federato di autenticazione ed autorizzazione basato su Security Assertion Markup Language (SAML). Shibboleth consente all'utente di controllare gli attributi a cui un sito esterno potrà avere accesso. Shibboleth è pensato per le università o grandi aziende dove viene costituito un gruppo federato. Il codice è rilasciato sotto licenza Apache.

Il sistema Shibboleth è composto da diversi elementi che separano la gestione delle identità degli utenti dalla gestione delle risorse autenticate; è un sistema basato su standard affidabili e collaudati e offre adeguate garanzie di funzionamento e di aderenza agli standard. Permette al client di accedere ai servizi protetti che si trovano all'interno della propria organizzazione attraverso una singola autenticazione (web single sign-on). Tali servizi possono trovarsi anche in altre organizzazioni purchè appartenenti ad una determinata federazione in cui vigono regole condivise per l'identificazione degli utenti e l'autorizzazione per l'accesso a risorse e servizi.

Shibboleth in sostanza modifica il modo in cui vengono controllati gli accessi e sposta verso le strutture di appartenenza degli utenti il compito di dare garanzie sulla loro identità (autenticazione) e sui loro diritti (autorizzazione) attraverso la realizzazione di una Infrastruttura di Autenticazione ed Autorizzazione (AAI).

7.3.1 Struttura

Shibboleth è composto da tre componenti fondamentali: Identity Provider (IdP), Service Provider (SP) e il Discovery Service (DS)

Identity Provider (IdP): è il componente software che si interfaccia con il registro delle identità dell'Organizzazione (LDAP, database relazionale o altro), generalmente uno per ogni struttura che aderisce alla federazione. L'IdP si occupa della registrazione degli utenti e di mantenere le

loro informazioni, di gestire le sessioni di autenticazione (inserimento delle credenziali oppure verifica di una sessione già attiva per l'utente - Single sign on), del rilascio degli attributi degli utenti autenticati. Tali informazioni sono infatti richieste dai Service Provider (SP) per consentire all'utente di accedere al servizio.
L'Identity Provider Shibboleth comprende i seguenti elementi legati tra loro:

- Handler Manager: è un servizio che amministra i vari endpoint dai quali l'IdP può ricevere messaggi;
- Attribute Resolver: si occupa di recuperare gli attributi da un database e provvede alla loro combinazione e trasformazione in un insieme di dati relativi ad uno specifico utente; successivamente li invia, tutti o in parte, al client dell'IdP;
- Attribute Filter: è utilizzato per creare una collezione filtrata di attributi, sulla base di un insieme di regole. Queste regole rappresentano quali attributi e valori un client può ricevere;
- Attribute Authority (AA): questo servizio, che dipende dall'Attribute Resolver, riceve un insieme di attributi e li codifica in un'asserzione di attributi SAML. Inoltre, se i metadati SAML del richiedente contengono informazioni sul tipo di attributi che egli richiede, il servizio sceglierà e rilascerà solo gli attributi richiesti.

Service Provider (SP): questo componente si occupa di proteggere e condividere una risorsa o un servizio web da mettere in comune tra i vari utenti della federazione; inoltre provvede ad indirizzare l'utente al servizio DS, e successivamento al relativo IdP, raccoglie le informazioni inviategli dall'IdP, le utilizza per proteggere il servizio e permette l'autorizzazione all'utente che ha effettuato una richiesta.
Il SP Shibboleth è composto da tre elementi:

- Resource Manager (RM): si occupa della gestione delle richieste di accesso alle risorse protette del servizio, che vengono compiute dall'utente mediante un browser.
- Assertion Consumer Service (ACS): ha il compito di comunicare con l'handler dell'IdP, di elaborare asserzioni SAML e di estrarre da esse i contenuti.
- Attribute Requester (AR): ottiene attributi aggiuntivi sull'utente autenticato e delinea politiche di accettazione degli attributi (Attribute Acceptance Policy o AAP).

Discovery Service (DS): è un servizio che, interpellato dal Service Provider, permette all'utente di selezionare la propria federazione di appartenenza scelta tra un elenco di federazioni possibili.
Questo componente nel nostro caso non è utilizzato perchè abbiamo una sola rete di afferenza degli utenti (CESIA) .

Nelle prossime sezioni ci apprestiamo a trattare in dettaglio i componenti che più ci interessano.

7.3.2 Processo di autenticazione

Il processo di autenticazione tramite Shibboleth avviene in questo modo:

1. il client chiede la connessione ad una risorsa protetta, in modo che il SP intercetti la richiesta (la risorsa da proteggere è definita nei files di configurazione del web server che la ospita); Per compiere tale richiesta, occorre che l'utente acceda alla pagina iniziale, inserendo l'indirizzo della risorsa.

2. il SP determina a quale IdP far riferimento e quale protocollo utilizzare mediante il servizio DS. La richiesta di autenticazione passa così al DS che a sua volta la reindirizza all'IdP selezionato dall'utente. Da notare che nel nostro caso non avremo bisogno del DS per determinare l'IdP (è unico);

3. l'IdP decide se l'utente può o meno autenticarsi e quali attributi inviare al SP, una scelta fatta in base alle caratteristiche del SP stesso. In questa fase l'utente viene indirizzato alla pagina di login propria dell'IdP ;

4. l'IdP invia i dati come asserzione SAML al SP che li interpreta e li decodifica eseguendo una serie di controlli di sicurezza per decidere se colui che sta effettuando l'operazione di autenticazione abbia o meno diritto di accesso alla risorsa desiderata. Se l'esito del risultato è negativo, ovvero se le credenziali inserite non sono corrette, l'utente non avrà accesso alle risorse;

5. l'utente viene rediretto alla risorsa desiderata, se il processo di verifica è andato a buon fine.

Il processo appena descritto è illustrato nella seguente figura:

7.3.3 Configurazione SP Shibboleth - Cesia

Il Cesia ci forniva dei file di configurazione di Shibboleth predefiniti che dovevano essere modificati secondo i parametri della nostra macchina virtuale.

Per configurare correttamente Shibboleth è necessario sostituire il file shibboleth2.xml presente in /etc/shibboleth/ con il file di configurazione shibboleth2-test.xml (per configurarlo con l'IdP di test), rinominandolo opportunamente. A questo punto abbiamo modificato i seguenti campi:

- Host, dentro RequestMapper, sostituendo il valore dell'attributo name con quello del server su cui è ospitata l'applicazione (ticket.cs.unibo.it);

- Path, dentro RequestMapper, sostituendo il valore dell'attributo name con quello della directory in cui è contenuta l'applicazione sul server (otrs);

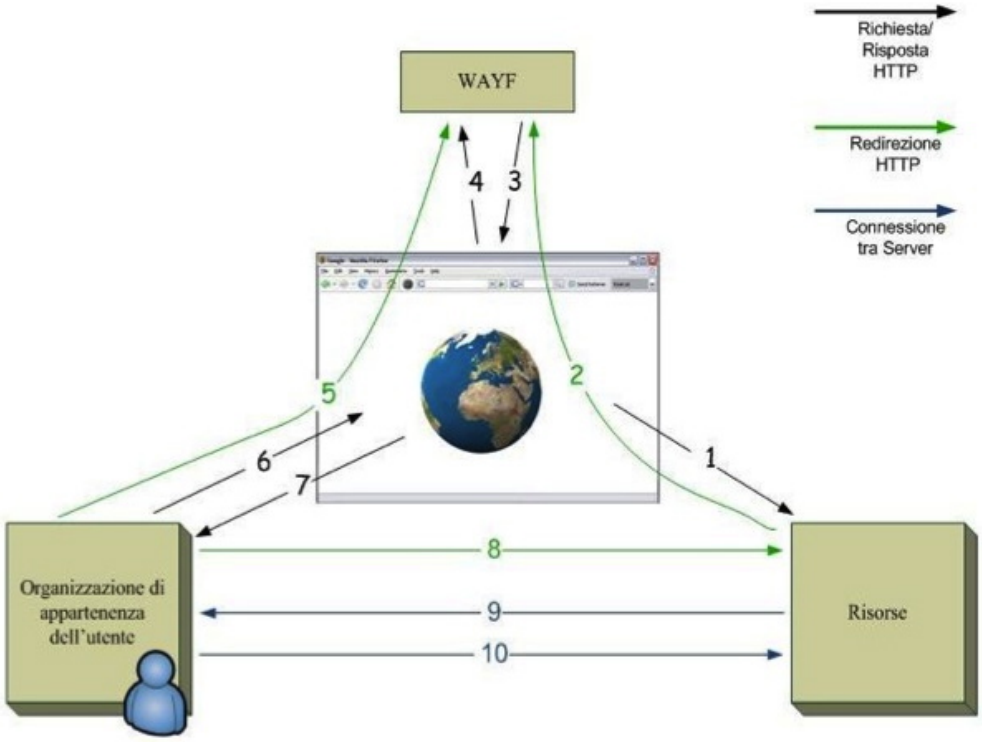

Figura 7.1: Processo autenticazione Shibboleth

- entityID dell'elemento ApplicationDefaults, questo attributo definisce l'identificativo univoco dell'applicazione all'interno del sistema federato (otrs);

- homeUrl dell'elemento ApplicationDefaults, questo attributo definisce l'indirizzo principale dell'applicazione (https://localhost/otrs/index.html);

- Modificare l'elemento Notify-Location dentro il file di configurazione di Shibboleth con il percorso della pagina di logout locale (logout.php è all'interno della cartella autenticata di otrs);

Poi, abbiamo abilitato il modulo di Shibboleth dentro apache2 con il comando:

```
a2enmod shib2
```

Per completare l'integrazione è necessario configurare il server web Apache in modo che l'accesso all'applicazione sia regolato da Shibboleth. Per farlo è necessario aggiungere alla propria configurazione una sezione come quella seguente:

```
/etc/apache2/apache.conf

<Location /otrs/customer.pl>
AuthType shibboleth
ShibRequireSession On
ShibUseHeaders On
require valid-user
UseCanonicalName On
</Location>
```

Come si può vedere, si è scelto di far passare dal modulo d'autenticazione solo gli utenti semplici che dovranno accedere al sistema dalla pagina customer.pl.

7.4 Lo standard SAML

Security Assertion Markup Language (SAML) è uno standard informatico per lo scambio di dati di autenticazione e autorizzazione (dette asserzioni) tra domini di sicurezza distinti, tipicamente un identity provider e un service provider. Il formato delle asserzioni SAML è basato su XML. SAML è mantenuto da OASIS Security Services Technical Committee.

SAML prevede opportuni meccanismi di estendibilità e definisce il formato dei metadati, utilizzati da Shibboleth per descrivere informazioni di vario tipo che caratterizzano entità SAML, come ruoli o certificati dei membri di una federazione. Fornisce funzionalità di Single Sign On federato e un framework per lo scambio di attributi; fornisce funzionalità estese relative alla privacy e riesce a comunicare informazioni di autenticazione e autorizzazione tra domini federati attraverso le asserzioni.

7.4.1 Asserzioni SAML

Le asserzioni e gli statement costituiscono la base dello standard SAML. SAML definisce tre tipi diversi di statement che possono essere contenuti in un'assertion:

- Authentication Statement (statement di autenticazione), specifica che un client è stato autenticato precedentemente attraverso uno degli svariati metodi (come password, chiave pubblica X.509)
- Authorization Statement (statement di autorizzazione), specifica se un soggetto può avere o meno accesso a determinate risorse
- Attribution Statement (statement di attributo), specifica che quell'utente è associato a determinate caratteristiche individuali (gli attributi) espresse generalmente da una coppia nome-valore.

7.4.2 La specifica SAML

La specifica SAML si basa su alcuni componenti fondamentali:

- asserzioni: sono dichiarazioni relative all'autenticazione e all'autorizzazione di un utente, sia esso una persona o un sistema hardware/software. Le asserzioni sono di tre tipi diversi. Le prime, quelle di autenticazione, stabiliscono che un client ha verificato la propria identità ad un determinato asserting party (emesso da una SAML authority). Le seconde, quelle di attributo, indicano alcuni dati di un utente come ad esempio il ruolo che quest'ultimo ha all'interno di un'organizzazione. Le ultime asserzioni, i permessi (AuthorizationDecision), stabiliscono le operazioni precise che un utente può effettuare;

- protocolli: le asserzioni devono avere la possibilità di scambiarsi opportunamente tra le entità che interagiscono contattando l'asserting party. Per questo motivo sono stati definiti opportuni protocolli che consentono lo scambio di asserzioni attraverso un meccanismo propriamente detto di domanda e risposta: ovvero dei messaggi SAML request e SAML response.

- binding: sono utilizzati per indicare come avviene lo scambio di informazioni di sicurezza in SAML attraverso appropriati protocolli di trasporto (HTTP o SOAP). Questo perchè i protocolli non stabiliscono le modalità di trasporto ma solo la struttura delle informazioni che possono essere scambiate;

- profili: rappresentano un insieme di asserzioni, protocolli e binding SAML, organizzati in diversi casi d'uso, che specificano come questi elementi vengono combinati per raggiungere lo scopo del caso d'uso in questione.

Un esempio può essere la realizzazione del Single Sign On tra svariati servizi. Il livello di fiducia che si trova all'interno di un'assertion non è specificato da SAML; questo compito è affidato ai sistemi locali i quali decidono se le politiche di sicurezza di una determinata applicazione siano sufficienti a proteggere un'azienda nel caso in cui sorgano dei problemi circa una decisione di autorizzazione basata su un'assertion errata. Questa caratteristica di

SAML fa si che le attività basate sul web, prima di accettare un'asserzione, debbano aderire ad un livello base di verifica. SAML può fondersi con diversi protocolli di comunicazione e trasporto (esempio : SOAP (Simple Object Access Protocol) , HTTP (HyperText Transfer Protocol)).

7.4.3 Funzionamento SAML

Il funzionamento di SAML è il seguente: l'utente viene registrato presso un IdP che provvede ad autenticarlo; il SP deve ovviamente identificare l'utente che chiede l'autenticazione quindi riceve dall'IdP un'asserzione SAML sulla base della quale il SP decide se concedere o vietare l'accesso dell'utente ai propri servizi.

Ad esempio, supponiamo che un utente di un'organizzazione X viene autenticato dal sistema di identificazione di X e può accedere ai servizi di Y, questo perchè l'organizzazione di Y verifica che sia valido il token di autenticazione rilasciato da X e non chiede un'ulteriore processo di verifica da parte del token rilasciato da Y. I token del mittente e del ricevente sono infatti equivalenti. Affinchè il token SAML sia valido bisogna redirigere la richiesta del client al server di autenticazione, questo nel caso in cui venisse meno il token relativo all'identità dell'utente e al ruolo che gli è stato associato. In questo caso, l'applicazione dovrà contattare il server di autenticazione dove ci saranno le credenziali dell'utente ed interrogare il servizio di autenticazione del dominio in cui l'utente è registrato. Se l'esito sarà positivo, sarà restituita un'asserzione con i dati dell'utente.

Quindi in sintesi un utente richiede una risorsa web protetta da un SAML service provider(SP). Il service provider cerca di identificare l'identita del richiedente ,quindi l'SP richiede una richiesta di autenticazione al SAML IdP passata attraverso il web browser. Questo processo è descritto in questa immagine .

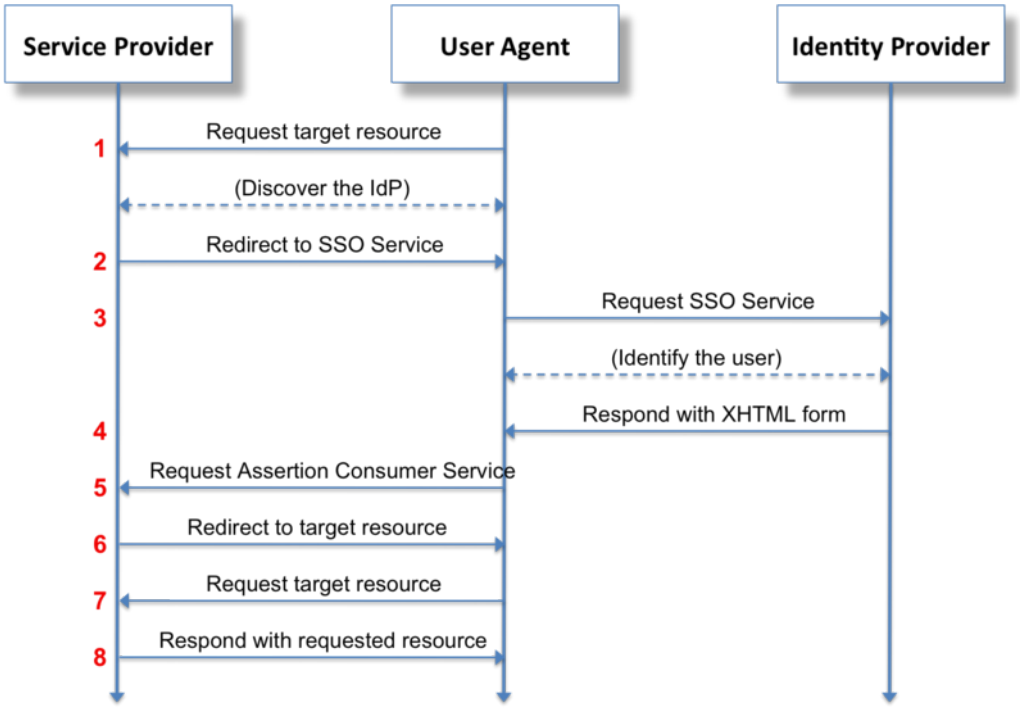

Figura 7.2: SAML Web Browser SSO

Capitolo 8

Testing

Al termine del nostro lavoro abbiamo eseguito qualche test prestazionale sulla nostra macchina virtuale.
Va ricordato che nell'analizzare i dati ottenuti dai benchmark non é importante tenere conto di quanto tempo il web server impiega per restituire una pagina: un utente che richiede una pagina tramite internet solo in linea teorica potrà ottenerla con un ritardo tra lo 0.05 (ms) e lo 0.1 (ms).
Ciò che risulta essere davvero importante é il tempo medio impiegato quanto il sistema é realmente sotto stress, cioè si ritrova a dover soddisfare un alto numero di richieste concorrenti.

Per i nostri test ciò ci siamo serviti del comando fornito insieme al web server Apache, *ab*. Abbiamo creato un file in PHP in grado di accedere, tramite MySQL, al database di OTRS ed eseguire una semplice query.

```
<html>
<head><title>Php+MySQL</title></head>
<body>
<?php
   $link = mysql_connect("localhost", "USERNAME", "PASSWORD");
   mysql_select_db("DATABASE");
   $query = "SELECT * FROM TABLENAME";
   $result = mysql_query($query);
```

```
   while ($line = mysql_fetch_array($result))
   {foreach ($line as $value)
       {print "$value\n";
      }}
    mysql_close($link);
?></body>
</html>
```

Lanciando il comando

```
ab -n 1000 -c 20 https://ticket.cs.unibo.it/psql.php
```

abbiamo generato 1000 richieste totali divise in 20 richieste contemporanee; il sistema, nonostante avesse poco più di 250 megabyte di RAM libera, non ha subito particolari rallentamenti.

```
output ab
Concurrency Level:      20
Time taken for tests:   20.009 seconds
Complete requests:      1000
Failed requests:        0
Write errors:           0
Non-2xx responses:      1000
Total transferred:      494000 bytes
HTML transferred:       291000 bytes
Requests per second:    49.98 [#/sec] (mean)
Time per request:       400.183 [ms] (mean)
Time per conc-request:  20.009 [ms]
Transfer rate:          24.11 [Kbytes/sec] received

Connection Times (ms)
```

```
              min   avg   max
Connect:       31   272  720
Processing:   106   121  552
Total:        137   393 1272
```

É interessante notare come il tempo di gestione di ogni singola richiesta concorrente é di soli 20 ms ed il tempo di gestione di 20 richieste é di circa 400 ms, un valore sufficientemente basso per poter giudicare in modo positivo il nostro sistema. Inoltre, tutte le richieste sono state soddisfatte senza errori.

Successivamente abbiamo ripetuto il test aumentando però il livello di concorrenza:

```
ab -n 1000 -c 60 -S https://ticket.cs.unibo.it/psql.php
```

mantenendo uguale il numero di connessioni totali, il numero di richieste concorrenti é salito fino a 60. Il tempo necessario per soddisfare gruppi di 60 connessioni concorrenti possiamo vedere essersi più che triplicato (un valore al limite del tollerabile), mentre quello per ogni singola connessione é rimasto invariato (circa 23 ms). In questo caso però, abbiamo notato come alcune connessioni sono state rifiutate per un errore SSL:

```
SSL read failed - closing connection
SSL read failed - closing connection
Completed 600 requests
SSL read failed - closing connection
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
```

la cosa viene dettagliatamente spiegata dall'output del comando ab:

```
Concurrency Level:      60
Time taken for tests:   23.225 seconds
Complete requests:      1000
Failed requests:        5
   (Connect: 0, Receive: 0, Length: 3, Exceptions: 2)
Write errors:           0
Non-2xx responses:      997
Total transferred:      492518 bytes
HTML transferred:       290127 bytes
Requests per second:    43.06 [#/sec] (mean)
Time per request:       1393.528 [ms] (mean)
Time per request:       23.225 [ms]
Transfer rate:          20.71 [Kbytes/sec] received
Connection Times (ms)
              min   avg   max
Connect:      237  1041 3140
Processing:    88   315 4147
Total:        325  1356 7287
```

Il campo Failed Request ci mostra come ben 5 richieste non siano andate a buon fine con un innalzamento medio del tempo di connessione di circa 5 volte (da 272 ms a 1042 ms).
L'output del comando vmstat ci giustifica il deperimento prestazionale anche con la mancanza quasi totale di memoria RAM libera.

```
estratto output vmstat -S M 1
procs -----------memory----------
 r  b   swpd   free   buff  cache
 1  0      0    239      2     46
11  0      0    237      2     46
11  0      0    231      2     46
```

```
13  0      0    220      2     46
17  0      0    202      2     46
25  0      0    171      2     46
41  0      0    142      2     46
36  0      0     40      2     46
44  0      0     33      2     46
49  0      0     26      2     46
55  1      0      4      2     45
55  2      0      5      1     27
54  0      0      6      1     18
42 14      0      5      0      5
49 10      0      4      0      5
48 21      0      4      0      2
9   9      0      4      0      2
----- terminato il test ab ------
procs -----------memory----------
 r  b   swpd   free   buff  cache
21  1      0     10      0      4
50  0      0     12      0      3
0   0      0     24      0     21
0   0      0     30      0     21
```

Capitolo 9

Sviluppi futuri

Giunti al termine del nostro tirocinio, ci siamo soffermati ad analizzare gli aspetti ancora incompleti o totalmente mancanti del sistema sviluppato da implementare per raggiungere l'obiettivo prefissato.

- I tecnici dovranno occuparsi di ottenere un certificato valido per l'indirizzo ticket.cs.unibo.it e comunicare al CESIA i dati mancanti, affinchè loro possano configurare correttamente l'applicazione sul server IdP. Solo a questo punto l'applicazione potrà essere realmente testata e, risolti gli ultimi problemi, passare in produzione.

- Il database di OTRS, fatta eccezione per alcune voci di prova, é ancora vuoto, quindi bisognerà popolare le varie tabelle con l'elenco dei tecnici e degli altri utenti semplici che potranno usufruire del software.

- Dovrà essere valutata una eventuale autenticazione DSA anche per gli amministratori; in questo perciò la loro autenticazione non passerà più dal db locale.

- Un altro aspetto tanto importante quanto delicato sarà il configurare OTRS in modo tale da autenticare i propri agenti ed i propri clienti tramite SSO. Si potrà usare il modulo *HTTPBasicAuth* presente nella cartella Kernel/System/Auth/ ed aggiungere nel file Config.pm, visto nella sezione **??**, il seguente frammento:

```
$Self->{'AuthModule'} =
'Kernel::System::Auth::HTTPBasicAuth';
#redirect in caso di deny access
$Self->{LoginURL} =
'http://host.example.com/not-authorised-for-otrs.html';
#redirect verso IdP in caso di logout
$Self->{LogoutURL} =
'http://host.example.com/thanks-for-using-otrs.html';
```

con le ultime due direttive che serviranno qualora l'utente non sia autorizzato (deny access) oppure sia stato effettuato il Logout (redirect verso IdP).
Lo stesso procedimento andrà utilizzato anche per gli utenti che accederanno dalla pagina customer.pl:

```
$Self->{'Customer::AuthModule'}=
'Kernel::System::CustomerAuth::HTTPBasicAuth';
#redirect in caso di deny access
$Self->{CustomerPanelLoginURL} =
'http://host.example.com/not-authorised-for-otrs.html';
#redirect verso IdP in caso di logout
$Self->{CustomerPanelLogoutURL} =
'http://host.example.com/thanks-for-using-otrs.html';
```

Va ricordato che di default OTRS utilizza l'autenticazione tramite database locale.

- Collegamento tra i dati passati via SSO (attributi dell'utente) e i dati presenti nel database locale di OTRS.

Una probabile soluzione a questo problema potrebbe essere cercare l'utente autenticato nel database locale di OTRS, se l'utente è presente, allora si caricano i dati dal database locale e OTRS procede come se andasse avanti senza SSO, altrimenti viene inserito l'utente nel database locale e gli vengono associati altri dati utili a OTRS, come per esempio il gruppo di appartenenza, in base agli attributi passati dal SSO.

Conclusioni

In questo lavoro di tesi è stata analizzata la possibilità di progettare e sviluppare un valido sistema di ticketing per la gestione delle richieste di supporto tecnico-amministrativo sottoposte al dipartimento di scienze dell'informazione. Il lavoro è stato condotto con il collega Ruggero Schirinzi e partendo da una base di lavoro e studio svolta in precedenza dai colleghi De Donno e Pilato .

Per poter effettuare la scelta più corretta tale da poter soddisfare al meglio le nostre esigenze ,sono state effettuate diverse riunioni con il personale tecnico del dipartimento, in modo da avere una chiara visione sui requisiti principali del sistema di ticketing da implementare. Una delle principali richieste era l'eventuale integrazione del sistema ad un metodo di autenticazione unica (SSO), quindi la scelta del sistema è stata effettuata prendendo in considerazione e analizzando queste informazioni. Dopo una accurata ricerca si è deciso di procedere con il sistema OTRS.

OTRS è risultato essere un sistema molto professionale e completo sotto molti punti di vista, con una interfaccia semplice ed intuitiva. La sua customizzazione ed il suo adattamento alle nostre esigenze è stato ampiamente realizzato grazie alle vaste opzioni che il sistema offre , in più il sistema offre la possibilità di integrare e installare dei servizi (patch) esterni ad esso (esempio: modulo FAQ).

Il passo successivo è stato l'integrazione del sistema con il nuovo sistema di autenticazione richiesto dal CESIA (Centro Servizi Informatici dell'Università di Bologna) ,ovvero il Single Sign On. Per realizzare questo scopo

abbiamo utilizzato il software Shibboleth .

Il completo procedimento di installazione del sistema ha avuto alcuni rallentamenti con l'utilizzo del reverse-proxy come ambiente per il suo sviluppo, come conseguenza di questi problemi si è deciso di procedere alla sua implementazione con una VM (virtual machine) all'indirizzo **ticket.cs.unibo.it**.

In conclusione ,la base del sistema è stata consolidata per eventuali sviluppi futuri per perfezionare al meglio soprattutto l'aspetto riguardate l'autenticazione con il SSO ed una analisi e gestione dei dati sugli utenti passati dal CESIA. Una sua completa finalizzazione ed utilizzo in futuro potrebbe alleggerire notevolmente il carico di lavoro del reparto tecnico del dipartimento.

Bibliografia

[1] Wikipedia - The Free Encyclopedia. "SAML". http://en.wikipedia.org/wiki/SecurityAssertionMarkupLanguage

[2] Wikipedia - The Free Encyclopedia. "Issue Tracking System". http://en.wikipedia.org/wiki/Issue tracking system

[3] Wikipedia - The Free Encyclopedia. "Comparison of issue-tracking systems". http://en.wikipedia.org/wiki/Comparison of issue tracking systems

[4] The Trac Project. http://trac.edgewall.org.

[5] Bugzilla Software. http://www.bugzilla.org.

[6] Mantis Bug Tracker. http://www.mantisbt.org/.

[7] OsTicket - Open Source Support Ticket System. http://osticket.com.

[8] OTRS - Open-source Ticket Request System. http://www.otrs.com/en.

[9] Shibboleth. http://shibboleth.net.

[10] Jesus Blanco, Daniel Lopez. *Apache.* Addison Wesley Pearson, 2007.

[11] Marc Delisle. *Mastering phpMyAdmin 3.4 for Effective MySQL Management.* Packt Publishing, 2012.

[12] Ivana Ruggiero. *Porting su RedHat e integrazione di nuovi servizi nell'infrastruttura di Autenticazione ed Autorizzazione Federata IDEM.* Università di Napoli Federico II, 2010.

Ringraziamenti

Inizio ringraziando i miei genitori per il loro continuo sostegno, la loro costante vicinanza e il loro instancabile incoraggiamento e sostegno sin dal primo giorno di inizio del mio percorso di studi. Ringrazio i miei fratelli e famigliari per il loro appoggio e aiuto di ogni giorno. Desidero ringraziare il Professor Vittorio Ghini per i preziosi insegnamenti, la sua completa disponibilità ed il suo continuo aiuto al mio lavoro di tesi. Intendo poi fortemente ringraziare il personale tecnico del Dipartimento di Informatica, sottolineando in particolare Paolo Marinelli e la sua grandissima disponibilità, serietà e le numerose ore dedicate durante tutto il mio periodo di tirocinio. Un grandissimo ringraziamento al collega ed amico Ruggero Schirinzi per i numerosi consigli, aiuti e sopratutto i tantissimi momenti di svago passati insieme in questi ultimi due anni. Intendo poi ringraziare in particolare Alberto, Manuela e il Dr.Antonello con il quale ho condiviso molti momenti di studio e di divertimento in questo ultimo anno. Vorrei esprimere la mia sincera gratitudine ad altrettanti colleghi e amici che mi hanno aiutato durante questo periodo,in particolare Enrico Raspadori per il suo appoggio e aiuto in svariati corsi, Matteo Fagiolino per i suoi appunti di diversi corsi e tantissimi altri colleghi senza specificare i nomi per paura di dimenticare qualcuno.