Alma Mater Studiorum · Università di Bologna

facoltà di scienze matematiche, fisiche e naturali
Corso di Laurea Magistrale in Informatica

Un sistema di ticketing per i tecnici del Dipartimento di Informatica

Tesi di Laurea in Sistemi Mobili

Relatore:
Chiar.mo Prof.
Vittorio Ghini

Presentata da:
Matteo Romanelli

Correlatore:
Dott.
Paolo Marinelli

Sessione II
Anno Accademico 2011/2012

"So long, and thanks
for all the fish."
(Douglas Adams - Addio, e grazie per tutto il pesce)

Introduzione

Lo scopo di questa tesi è quello di implementare un sistema di ticketing per il dipartimento che sia in grado di gestire le richieste di supporto tecnico ed amministrativo in modo più efficiente di quanto non avvenga oggi. Nel sistema attuale, infatti, le richieste sono sottoposte al personale tecnico mediante il Newsgroup del dipartimento o tramite e-mail. Partendo dal lavoro dei colleghi Ivan Heibi e Ruggero Schirinzi si è cercato quindi di rendere disponibile uno strumento tale da fornire una gestione del lavoro che garantisse la possibilità di diminuire i tempi di risposta e di evitare che alcune richieste vengano perse o soddisfatte solo in parte.

È stato individuato un sistema di ticketing che fosse facilmente utilizzabile dagli utenti. Ovvero che permettesse, in modo semplice ed intuitivo, di sottomettere nuove richieste e che consentisse, al personale tecnico, una gestione delle richieste agevolata. Un'altra caratteristica richiesta è la possibilità di integrazione con l'SSO (Single Sign On), il nuovo sistema di autenticazione dell'Ateneo. L'SSO permette agli utenti di accedere a tutti i servizi dell'Ateneo per i quali si è abilititati senza ulteriori richieste di credenziali dopo il primo accesso.

Nel primo capitolo saranno introdotti i sistemi di ticketing, descrivendone finalità e architettura. Si passerà poi ad analizzare i requisiti che il nostro sistema dovrebbe avere per rispondere al meglio alle esigenze del dipartimento; inoltre verranno presentate le aree tematiche nelle quali saranno suddivise le richieste e i gruppi di utenti ad esse associati. A questo punto, nel terzo capitolo, si prenderanno in esame i diversi sistemi di ticketing pre-

senti nel mercato, confrontandoli al fine di determinare quello più idoneo da adottare, che verrà illustrato in dettaglio nel capitolo successsivo. Si passerà poi all'analisi del sistema di autenticazione SSO, alla sua implementazione mediante l'utilizzo di Shibboleth e alla sua integrazione con lo strumento scelto. Negli ultimi capitoli si mostrerà la procedura di installazione, configurazione e customizzazione dell'intero sistema. L'elaborato si concluderà con un capitolo riepilogativo in cui saranno illustrati possibili sviluppi futuri del sistema.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Sistemi di Ticketing

I sistemi di ticketing (anche conosciuti come trouble ticket system o issue tracking system) sono sistemi software che si occupano di gestire e mantenere liste di segnalazioni e rischieste sottoposte dagli utenti (clienti, personale tecnico, amministratori di sistema) secondo le necessità delle organizzazioni che forniscono il servizio.

Questi sistemi vengono solitamente utilizzati in organizzazioni (Enti, Pubbliche Amministrazioni, Privati, Professionisti) che necessitano di fornire supporto ai clienti dando la possibilità di creare, aggiornare e risolvere i problemi degli utenti o dei componenti dell'organizzazione.

1.1 Ticket

L'elemento principale di questi sistemi è il ticket. Questo contiene informazioni riguardanti un intervento di supporto da parte del personale tecnico ad una richiesta di un utente che ha riportato un qualche genere di problema. Le informazioni contenute in un ticket vengono aggiornate durante il suo ciclo di vita, aggiungendo, a quelle fornite durante la sua creazione, commenti o informazioni aggiuntive che il personale tecnico ha ritenuto importante al fine della risoluzione del problema. I Ticket sono identificati univocamente

da un codice (issue number) che permette di individuarlo in modo rapido sia agli utenti che allo staff.

I ticket vengono cosi chiamati per questioni storiche, infatti in precedenza le richieste venivano segnate dal personale su dei foglietti di carta che venivano poi adeguatamente catalogati.

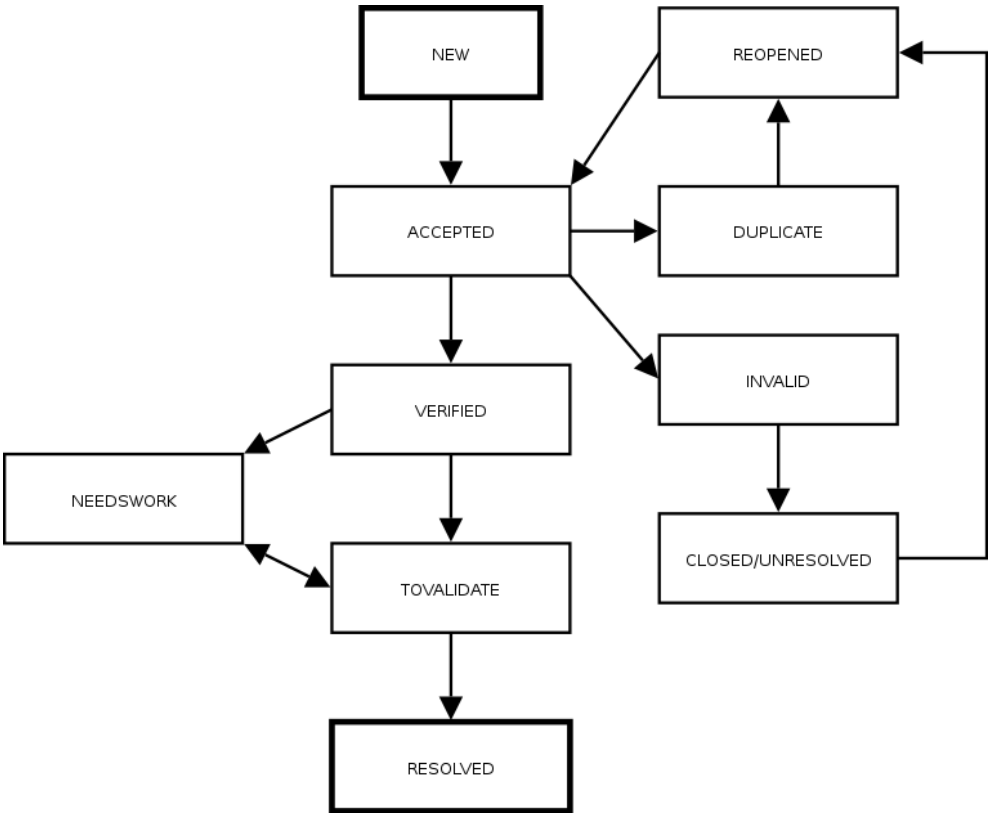

Figura 1.1: Ciclo di vita di un ticket

1.2 Architettura

L'architettura di un sistema di ticketing è relativamente semplice. È infatti composta da un database in cui vengono mantenute tutte le informazioni sugli utenti del sistema, sulle soluzioni ai problemi più comuni e altri dati simili. Queste informazioni vengono poi elaborate da un livello logico che si occupa di prendere i dati grezzi e renderli consultabili da un essere umano. Infine è presente un terzo componente che si occupa di presentare i dati precendentemente elaborati.

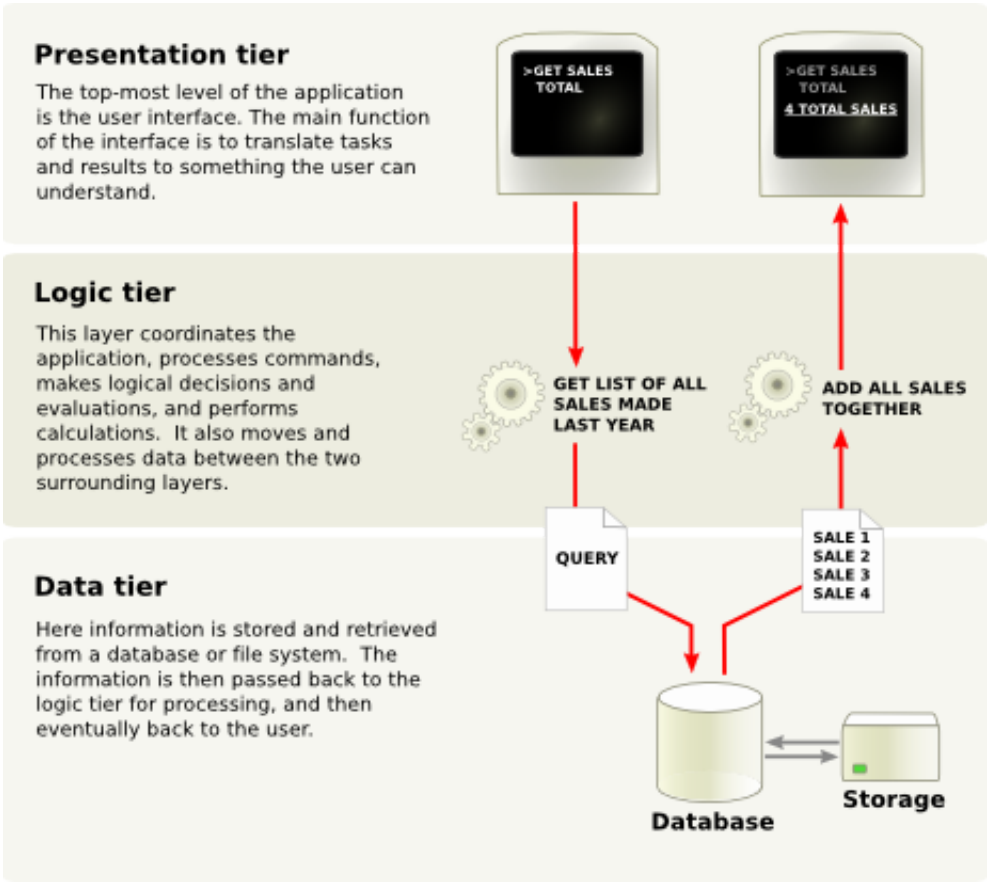

Figura 1.2: Architettura di un applicazione a tre livelli

Un utente che accede al sistema ha la possibilità di inserire un nuovo problema, modificarlo, consultare la lista di quelli già presenti o suggerirne una

soluzione. Ogni volta che una di queste azioni viene compiuta nel database vengono salvate le informazioni su chi l'ha compiuta e in che momento, mantenendo così una cronologia delle operazioni svolte.

Per questioni di sicurezza, l'accesso all'applicativo necessita di un autenticazione dell'utente per permettere a questo di agire.

Ad ogni richiesta viene associato un valore di urgenza basato sulla sua importanza; in questo modo, viene garantita la priorità a quei problemi che vengono ritenuti di maggiore rilevanza.
Solitamente livelli di urgenza differente corrispondono ad una gestione del problema in modo diverso, ad esempio richieste con priorità molto bassa possono essere gestite tramite un sistema di FAQ (Frequently Asked Questions) e solo in un secondo momento, nel caso in cui il problema non venga risolto, si può passare ad una gestione di livello superiore.

1.3 Casi d'uso

1. Un cliente riscontra un problema e consulta le FAQ per trovare una soluzione;
2. Se la soluzione non è presente:
   (a) invia una nuovo ticket mediante l'interfaccia utente del sistema;
   (b1) contatta tramite telefono o mail un tecnico del servizio e notifica il problema;
   (b2) il tecnico verifica la richiesta e provvede a creare il nuovo ticket;
3. I tecnici verificano che il problema sia vero e che siano state incluse tutte le informazioni necessarie;
4. I tecnici cercano di risolvere il problema, aggiungendo di volta in volta nuove informazioni al ticket;
5. Una volta che il problema è stato risolto il ticket viene contrassegnato come risolto.

Capitolo 2

Analisi dei requisiti

Prima di analizzare nel dettaglio le caratteristiche dei vari sistemi di ticketing disponibili sul mercato, è stato necessario delineare quali fossero le esigenze del dipartimento da tenere in considerazione. Per fare ciò è stato preso in esame il lavoro svolto dai colleghi De Donno e Pilato, i quali, attraverso uno studio degli strumenti attualmente utilizzati e mediante degli incontri con il personale tecnico-amministrativo, hanno descritto i possibili scenari verificabili nella nostra rete.

2.1 Caratteristiche richieste

I principali requisiti richiesti al sistema sono:

- **Licenza Open Source:** Il sistema deve avere una licenza libera (GPL, AG-PL, etc);
- **Interfaccia web:** La possibilità di accedere al gestore di ticketing mediante un'interfaccia web semplice ed intuitiva;
- **SystemLog:** La possibilità di avere un file aggioranto sullo stato del sistema, grazie al quale è possibile visualizzare se tutti gli accessi e tutte le operazioni sono avvenute correttamente;

- **Ricerca:** La possibilità di cercare all'interno del sistema tutti i ticket ricevuti e di poterli visualizzare, rendendo così possibile tener traccia dei propri ticket e di apportarvi delle modifiche nel caso fosse necessario;

- **Ticket by mail:** La possibilità di ricevere e visualizzare i ticket mediante email su un client di posta qualsiasi;

- **Modularità del Codice:** Il codice del sistema deve essere quanto più possibile modulare, in modo da permettere eventuali modifiche future;

- **Autenticazione Customizzata:** La possibilità di integrare il sistema con il modello di autenticazione utilizzato dall'Ateneo (SSO);

- **Customizzazione delle aree di richiesta:** La possibilità di personalizzare le aree di richiesta del sistema e il settore di competenza delle varie categorie di utenti.

2.2 Autenticazione

L'autenticazione è il processo mediante il quale viene verificata la corretta identità di un utente. Nel nostro caso è stata fatta esplicita richiesta che l'accesso al sistema di ticketing venga effettuato mediante il sistema di autenticazione di Ateneo. Il sistema in uso dall'Ateneo al momento dell'analisi fatta dai miei collegi era il Directory Service d'Ateneo (DSA). Questo sistema sarà però dismesso alla fine del 2012, quindi già a partire dal lavoro di Ivan Heibi e Ruggero Schirinzi il parametro per la scelta dello strumento è stata la possibilità di integrarlo col nuovo sistema di autenticazione di Ateneo (SSO - Single sign on) fornito dal CeSIA, tramite il quale gli utenti potranno accedere a tutti i servizi per i quali sono abilitati autenticandosi un'unica volta. Vedremo nel dettaglio il Single sign on e la procedura di configurazione e integrazione di questo con il sistema scelto nei successivi capitoli.

2.3 Smistamento ticket

È stato necessario classificare i tipi delle richieste di supporto presentate al personale tecnico-amministrativo al fine di poter smistare i ticket al settore di competenza, permettendo solamente a determinate categorie di utenti di effettuare specifiche richieste. Questa classificazione è stata ottenuta attraverso un'analisi del principale strumento attualmente in uso per l'invio di richieste, il newsgroup. Da questa analisi sono state definite delle macroaree; alcune di queste sono state suddivise a loro volta in microaree così da 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.

2.3.1 Newsgroup

Il newsgroup è uno strumento molto importante per gli studenti ed 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 soprattutto per le richieste di supporto tecnico e amministrativo. Tali richieste, oltre ad essere sempre più numerose, con conseguente aumento del carico di lavoro sottoposto al personale, sono anche simili tra loro. Ecco perché si è cercato di utilizzare uno strumento di ausilio al Newsgroup che potesse snellire questo sistema, permettendo una migliore gestione delle richieste, attribuendo loro la priorità adeguata ed evitando inutili ripetizioni.

2.3.2 Macroaree e Microaree

Il seguente elenco descrive tutte le possibili aree a cui le richieste possono appartenere. Le tabelle che seguono saranno utili per illustrare le associazioni tra le diverse tipologie di utenti e i tipi di richieste per cui sono abilitati.

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

Tabella 2.1: Associazione Categoria - Aree Visualizzate

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

Tabella 2.2: Associazione Categoria - Gruppi

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

Tabella 2.3: Associazione Gruppo-Aree

Capitolo 3

Sistemi di Ticketing presi in esame

Al fine di scegliere quale fosse il sistema che più si adattasse alle esigenze del dipartimento, si è inizialmente stilata una lista dei più diffusi strumenti disponibili in commercio. In questa sono stati inseriti tutti i sistemi senza tenere conto dei requisiti precedentemente presentati, così da includere anche applicativi con licenza proprietaria, al fine di avere una panoramica il quanto più esaustiva possibile sulle possibili opzioni a disposizione.

In un secondo momento si è passati ad una fase di analisi dei sistemi rinvenuti. In questa fase sono stati tenuti in considerazione i seguenti criteri:

- Licenza Open Source
- Interfaccia web
- SystemLog
- Ticket by mail
- Modularità Codice
- Autenticazione Customizzata
- Customizzazione delle aree di richiesta

- Piattaforma di sviluppo
- Linguaggi e DBMS utilizzati

3.1 Tabella di valutazione

Di seguito è presentata una tabella riassuntiva dei sistemi che saranno poi valutati nelle sezioni successive.

Nome	**Sviluppo**	**Licenza**	**Linguaggio**	**Database**	**Lancio**
IssueTrak	IssueTrak Inc	Proprietary	ASP	SQL Server	2000
Assembla Tickets	Assembla	Proprietary	Ruby	MySQL	2008
Teamwork	OpenLab	Proprietary	Java	Hibernate	2003
JIRA	Atlassin	Proprietary	Java	MySQL, Oracle	2003
Bugzilla	Mozilla Foundation	MPL	Perl	MySQL	1998
MantisBT	Varie	GPL	PHP	MySQL, SQL Server	2000
Request Tracker	Best Practical Solutions	GPL	Perl	MySQL, PostgreSQL, Oracle, SQLite	1999
Trac	Edgewall Software	New BSD	Python	SQLite, PostgreSQL, MySQL	2006
OsTicket	OsTicket	GPL	PHP	MySQL	2009
OTRS	otrs.org	AGPL	PERL	MySQL, PostgreSQL, Oracle, SQL Server	2002

Tabella 3.1: Sistemi di ticketing valutati

3.2 Bugzilla

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

3.2.1 Requisiti

Esiste un insieme di requisiti necessari per poter usare Bugzilla:

- web server (Apache);
- Perl 5;
- un insieme di moduli Perl;
- un server con un database compatibile;
- server SMTP (Protocollo di trasferimento postale).

3.3 MantisBT

Mantis è stato pubblicato nel 2000 e con il tempo questo sistema ha ottenuto sempre più popolarità, così da essere considerato attualmente uno dei sistemi di ticketing Open-Source più usati. Essendo Mantis scritto come uno script PHP può essere installato su un qualsiasi sistema operativo con supporto PHP.

3.3.1 Caratteristiche

Saranno ora presentate alcune delle caratteristiche di Mantis:

- Notifice:
  con MantisBT è possibile mandare delle notifice via email. 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 tenere traccia dello stato del proprio ticket attraverso dei social network come Twitter oppure tramite link RSS.

- Plug-ins:
  nella versione 1.2.0 di Mantis è stato 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 personalizzazioni come la possibilità di mantenere un record aggiornato dinamicamente per ogni ticket, al fine di tenere 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 nei database del sistema.

3.4 Request Tracker

Request Tracker è un sistema di ticketing che consente ad un gruppo di persone, in maniera intelligente ed efficace, la gestione di task, problemi e richieste inviati da una comunità di utenti. Utilizzando la posta elettronica Request Tracker gestisce: richieste di assistenza, problemi tecnici, collaborazioni nella realizzazione di progetti e commesse e informazioni commerciali, costruendo un archivio di informazioni sulla relazione con il cliente. La piattaforma Request Tracker è in continuo sviluppo sin dal 1996 ed è usata da amministratori di sistema, da addetti al supporto cliente, dagli IT manager, da sviluppatori, dalle divisioni marketing su migliaia di installazioni a livello mondiale. Scritto in Perl onbect-oriented, Request Tracker è un sistema high-level, portabile e indipendente dalla piattaforma.

3.5 Trac

Trac è un progetto open source. Il programma si ispira a CVS-Trac, ed è stato originariamente chiamato svntrac per via della sua capacità di interfacciarsi con Subversion. Trac è sviluppato e mantenuto da Edgewall Software. Trac 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. Entrambe 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.

3.6 OsTicket

Lo studio di questo sistema è stato particolarmente approfondito da parte dei colleghi De Donno e Pilato, in quanto era stato considerato da loro come lo strumento più adatto da usare per rispondere alle esigenze del dipartimento.

OsTicket è un pacchetto software per assistenza clienti, apparentemente semplice e affidabile, open source, scritto in php, con database MySQL, portabile e basato sul 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.

3.6.1 Struttura

Come accennato OsTicket è basato sul concetto di ticket. L'apertura di un nuovo ticket da parte di un utente corrisponde ad una nuova richiesta; a questa viene assegnato un codice univoco che identifica il ticket e che può essere usato sia dall'utente che dal tecnico per seguire i progressi delle attività in corso. L'utente avrà inoltre la possibilità, attraverso il sistema di gestione, di interagire con il Team di supporto per fornire ulteriori chiarimenti o seguire lo stato dei lavori fino alla chiusura del ticket.

3.6.2 Features

Saranno ora presentate alcune delle caratteristiche di OsTicket:

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

- Risposte Predefinite:
  È possibile far sì che il sistema invii risposte automatiche, personalizzabili dallo staff, agli utenti. Ad esempio al momento della creazione di un nuovo ticket.

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

- Note interne:
  È possibile associare ai ticket delle note interne, in modo tale da organizzare al meglio il lavoro del personale tecnico.

- Notifice tramite email:
  È possibile configurare il sistema in modo tale che sia gli utenti che il personale tecnico ricevano notifiche e aggiornamenti riguardanti i ticket tramite email.

- Accesso basato su ruoli:
  È possibile controllare il livello di accesso del personale, distinguendo al momento dell'autenticazione se si tratti di un amministratore o di un membro dello staff.

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

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

3.7 OTRS

OTRS è l'acronimo di Open-source Ticket Request System (sistema open source per la richeista di ticket), è un pacchetto software open source che consente ad aziente, enti o istituzioni di assegnare dei ticket di segnalazione a ciascuna delle domande ricevute, rendendo molto più semplice la gestione delle richieste di assistenza (anche quelle via email o telefono) e degli altri scambi di informazioni con i propri utenti. Descriveremo più dettagliatamente questo sistema nel capitolo successivo.

3.8 Scelta del sistema

In un primo momento è stato scelto OsTicket, questo sistema però, nonostante la semplicità d'uso ed il fatto che possedesse molte delle caratteristiche ricercate, non è stato considerato adatto alle esigenze del dipartimento. Le scelte che hanno portato i mei colleghi a cambiare idea sono dovute principalmente a due fattori: il primo è la data di rilascio dell'ultima versione stable, che risale a Febbraio 2010, il secondo è da individuarsi nella poca modularità del codice. La distinzione poco evidente tra il codice HTML e quello PHP avrebbe comportato, se si fosse reso necessario customizzare parte del sistema, una revisione piuttosto importante. Si correva così il rischio di dover riscrivere parte del sistema ogniqualvolta si fosse presentato un nuovo aggiornamento software.

Sulla base di queste considerazioni si è quindi deciso di utilizzare OTRS. Questo sistema è forte di una serie di feedback di livello internazionale, offre una grande possibilità di personalizzazione grazie alla sua struttura modulare e possiede una community molto attiva, costituita da una Mailing List, un

Blog e un Forum. Inoltre le nuove versioni vengono rilasciate ad intervalli regolari.

Capitolo 4

OTRS

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

OTRS è divisa in diversi componenti. Il componente base è il framework che contiente tutte le principali funzioni utilizzate dall'applicazione e dal sistema di ticketing. Attraverso l'interfaccia web è possibile poi installare dei moduli aggiuntivi, come quello per le FAQ o per il monitoraggio della rete.

OTRS è implementato in linguaggio Perl e la sua interfaccia è resa più user-frendly grazie all'utilizzo di JavaScript.

4.1 Requisiti

OTRS può essere installato su diversi sistemi operativi (Linux, Solaris, AIX, FreeBSD, OpenBSD, Mac OS 10.x, Microsoft Windows).

I requisiti hardware minimi raccomandati sono:

- 2 GHz Xeon o CPU comparabili
- 2 GB RAM
- 160 GB memoria

Per quanto riguarda i requisiti software:

- Perl
  - Perl 5.8.8 o superiore
- Web server
  - Apache2 + mod_perl2 o superiore (raccomandato)
  - Webserver con supporto CGI (non raccomandato)
  - Microsoft Internet Information Server (IIS) 6 o superiore
- Database
  - MySQL 4.1 o superiore
  - PostgreSQL 7.0 o superiore
  - Oracle 10g o superiore
  - Microsoft SQL Server 2005 o superiore
- Web browser
  - Internet Explorer 8.0 o superiore
  - Mozilla Firefox 3.6 o superiore
  - Google Chrome
  - Opera 10 o superiore
  - Safari 4 o superiore

4.2 Features

Di seguito presenterò una lista delle principali caratteristiche del framework principale di OTRS.

- Interfaccia Web:
  - Supporto dei maggiori browser, anche mobile;

  - Disponibile un interfaccia web per i clienti;
  - Disponibile un iterfaccia web per gli agenti;
  - Possibilità di customizzazione;
  - Supporto multi linguaggio.

- Interfaccia Mail:
  - Supporto per gli allegati (supporto MIME);
  - Conversione automatica dei messaggi HTML in messaggi di testo;
  - Possibilità di filtrare la mail tramite X-OTRS headers o indirizzo email;
  - Supporto PGP;
  - Supporto gestione messaggi S/MIME;
  - Risposte automatiche configurabili per i clienti;
  - Notifiche email per gli agenti;

- Tickets:
  - Visibilità alle nuovi richieste;
  - Possibiltà di bloccare i ticket;
  - Possibilità di configurare un template per le risposte automatiche;
  - Possibilità di consultare la storia dei ticket;
  - Possibilità di definire liste di accesso ai ticket;
  - Possibilità di spostare i ticket tra code;
  - Possibilità di cambiare la priorità ai ticket;
  - Possibilità di inserire note personali;
  - Possibilità di ricerca tramite tag o full text.

- Sistema:

- OTRS girà su molti sistemi operativi;
- Supporto ad ASP (active service providing);
- Integrazione con back-end esterni;
- Supporto dei maggiori database;
- Diversi metodi di autenticazione (LDAP, HTTPAuth, Radius);
- Supporto per gli account utente, i gruppi e i ruoli;
- Supporto per le sottocode;
- Informazioni sugli aggiornamenti tramite mail e interfaccia web.

4.3 Area ADMIN

Per gli amministratori di sistema è possibile gestire OTRS integralmente attraverso l'interfaccia web fornita e in particolare dalla pagina ADMIN. Da questa, l'amministratore potrà aggiungere un nuovo utente/agente o modificarne i dati, gestire i gruppi, le code, i ruoli, i ticket e molto altro.

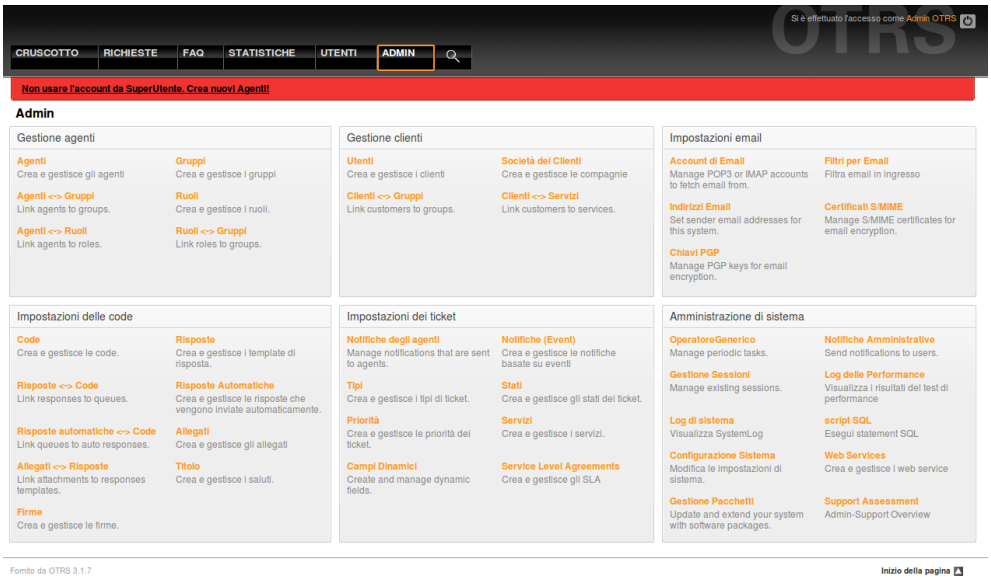

Figura 4.1: Area ADMIN di OTRS

4.4 Agenti

In OTRS, gli agenti sono i dipendenti dell'organizzazione che si occupano della gestione e della risoluzione dei problemi segnalati dai clienti. Nel nostro caso la figura dell'agente comprende anche quella dell'amministratore: i tecnici del dipartimento si occupano infatti sia della gestione del sistema che della risoluzione delle problematiche.

Un agente può essere creato attraverso l'inserimento dei dati richiesti nell'interfaccia che compare cliccando il bottone Add agent. Dopo che questo è stato creato può diventare un membro di uno o piu gruppi o ruoli.

L'account di un agente non può essere distrutto ma solamente disattivato. È possibile fare ciò impostando il valore del flag Valid a invalid o a invalid-temporarily.

4.5 Gruppi

Ad ogni gruppo verranno associati tutti quegli utenti che dovranno accedere ad una determinata coda di richieste; ogni utente può appartenere a più gruppi contemporaneamente.

Di default in OTRS sono presenti tre gruppi dei quali possono diventare membri gli agenti:

Gruppo	**Descrizione**
admin	Permette di svolgere compiti amministrativi
stats	Permette l'accesso al modulo stats per generare statistiche
users	Permette l'accesso a tutte le funzioni del sistema di ticketing

Tabella 4.1: Default groups in OTRS

Gli altri gruppi inseriti, durante la fase di customizzazione, sono stati presentati nella tabella 2.3.

Come per gli agenti un gruppo può soltanto essere disattivato ma non distrutto. Inoltre attraverso l'interfaccia ADMIN è possibile consultare la lista di tutti i gruppi presenti ed associare un gruppo ad un utente.

Ogni agente può possedere una qualsiasi combinazione dei seguenti diritti sul gruppo di cui è membro.

Diritto	**Descrizione**
ro	Accesso di sola lettura a ticket e code
move into	Diritto a spostare ticket tra code
create	Diritto a creare ticket
owner	Diritto ad aggiornare il proprietario dei ticket
priority	Diritto a cambiare la priorità dei ticket
rw	Accesso completo di lettura e scrittura

Tabella 4.2: Diritti associati ai gruppi in OTRS

4.6 Ruoli

I ruoli sono un potente feature per gestire i diritti di accesso di molti agenti in modo semplice e veloce. Sono usati principalmente in sistemi di dimensioni piuttosto elevate in cui sono presenti un gran numero di agenti, gruppi e code. In particolare è possibile usare un nuovo ruolo quando si devono cambiare i diritti d'accesso a molti agenti; anzichè farlo manualmente è possibile associare a questi un nuovo ruolo, in modo tale che i diritti degli agenti cambino automaticamente.

Come per i gruppi, i ruoli sono gestibili attraverso l'interfaccia ADMIN che dà la possibilità di consultare l'elenco dei ruoli del sistema, di modificare le impostazioni dei ruoli e di associarli agli utenti.

4.7 Clienti

I clienti sono gli utenti OTRS con il minor numero di permessi, questi possono infatti unicamente creare nuovi ticket e consultarli. Ogni cliente può essere associato ad uno o più gruppi, queste associazioni offrono la possibilità di gestire (creare/consultare) i ticket appartenenti a più code di richieste.

Come per gli elementi precedentemente descritti anche i clienti possono essere creati mediante l'interfaccia ADMIN e tramite questa possono essere amministrati e disattivati.

I clienti accedono al sistema mediante un'interfaccia che richiede l'autenticazione attraverso username e password. Ad ogni utente è associato un CustomerID che viene usato dal sistema per identificarlo e con esso i ticket a lui associati.

4.8 Code

In OTRS esistono di default quattro tipi di code (Raw, Junk, Misk e Postmaster): a queste sono state aggiunte quelle individuate nel paragrafo 2.3, inserite o come nuove code o come sottocode. Ognuna di queste è stata poi associata ad uno specifico gruppo.

Per ogni coda è possibile definire un unlock timeout, quindi se un agente dovesse bloccare un ticket senza chiuderlo prima che il timeout sia scaduto, questo verrebbe sbloccato, diventando disponibile agli altri agenti. Ci sono tre differenti metodi di escalation che possono essere associati alle code:

- First Response Time:
  Dopo la creazione di un ticket, se il tempo definito scade e non vi sono comunicazioni al cliente, allora questo viene rilasciato.

- Update Time:
  Se ci sono dei clienti che seguono il ticket allora il tempo viene resettato, altrimenti quando il tempo scade il ticket viene rilasciato.

- Solution time:
  Se il ticket non viene chiuso prima che il tempo termini allora viene rilasciato.

Capitolo 5

SSO - Single Sign On

Il Single sign-on (SSO), traducibile come autenticazione unica o identificazione unica, è un sistema di autenticazione centralizzata che permette ad un utente di inserire una sola volta le sue credenziali ed avere accesso a tutte le risorse informatiche alle quali è abilitato.

Grazie a questo sistema, le richieste di autenticazione non vengono più gestite dallo stesso sistema cui l'utente desidera accedere ma sono ridirette verso un altro sistema, che ha verificato in precedenza le credenziali d'accesso dell'utente.

Il compito del SSO è quindi quello di aumentare la sicurezza della rete, rendendone più facile la manutenzione e gestione . Grazie a questo sistema l'utente avrà la percezione di lavorare in un sistema sicuro, senza la sensazione che la sicurezza gravi su di esso. Il cliente non avrà infatti la necessità di reinserire username e password se già connesso, riducendo l'utilizzo di password banali o simili tra loro.

Vi sono tre approcci per la creazione di un sistema di SSO:

- Approccio centralizzato:
  Tutti gli utenti sono gestiti da un'autorità centrale, i loro dati sono contenuti in un database globale e centralizzato. In questo modo è possibile quindi centralizzare la politica di sicurezza.

- Approccio federativo:
  Con questo approccio esistono diversi gestori che che si occupano di amministrare i dati di uno stesso utente. L'accesso ad uno dei sistemi federati permette automaticamente l'accesso a tutti gli altri sistemi. In questo caso ognuno dei gestori mantiene il controllo della propria politica di sicurezza.

- Approccio cooperativo:
  Con questo approccio ogni utente dipende, per ciascun servizio, da uno solo dei gestori. Quando un utente cerca quindi di accedere ad un servizio, l'autenticazione verrà gestita da quello che ha in carico l'utente per quel servizio. Anche in questo caso ogni gestore avrà la propria politica di sicurezza, indipendente da quella degli altri gestori.

5.1 Obiettivi

Gli obiettivi raggiungibili attraverso l'uso del SSO sono:

- Semplificazione nella gestione delle password:
  l'utilizzo di un minor numero di password da parte degli utenti contribuisce a diminuire la possibilità che vengano impiegate password simili o banali, aumentando il livello di sicurezza del sistema.

- Semplificazione nella gestione degli accessi ai servizi:
  l'utilizzo del SSO garantisce agli utenti di doversi autenticare una sola volta, consentendo un accesso molto più rapido ai servizi.

- Semplificazione nella gestione delle politiche di sicurezza:
  la centralizzazione delle informazioni degli utenti garantisce di poter gestire la politica di sicurezza delle credenziali anche mediante l'uso di certificati X.509.

5.2 Architettura

L'architettura alla base dei sistemi di SSO è strutturata su tre livelli; questi possono essere descritti, assimilando l'architettura a quella client/server a tre livelli, come: livello utente, livello applicazione e livello dati. Ognuno dei livelli usufruisce dei servizi offerti dal livello inferiore e fornisce servizi al livello superiore.

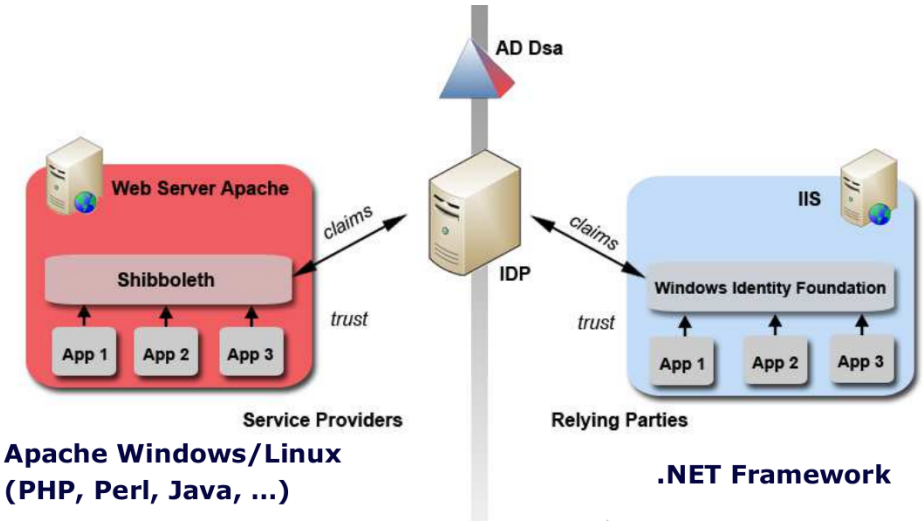

Figura 5.1: Architettura del SSO

Vediamo nello specifico chi copre il ruolo dei vari livelli nel nostro caso.

- Livello utente:
  corrisponde alla macchina sulla quale un cliente del sistema cerca di effettuare l'autenticazione mediante un qualsiasi browser.

- Livello applicazione:
  corrisponde al server sul quale gira l'applicazione e al quale verrà fatta la richiesta di autenticazione, che sarà poi girata all'identity provider.

- Livello dati:
  corrisponde al server su cui gira l'identity provider che si interfaccerà

con il registro delle identità dell'organizzazione al fine di autenticare l'utente.

Questa architettura permette ad ogni livello di accedere solo a quello immediatamente sottostante, rendendoli completamente indipendenti tra loro; ciò va a favore della gestione e manutenzione del sistema. Proprio per quest'ultima caratteristica l'architettura viene definita *chiusa*.

L'autenticazione e conseguente convalida di un client sono memorizzate attraverso un ticket emesso dal sistema di SSO, di cui solo l'utente entrerà in possesso. 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 verranno richiesti gli attributi utente precedentemente concordati, mentre in caso di esito negativo sarà generato un messaggio di errore di autenticazione.

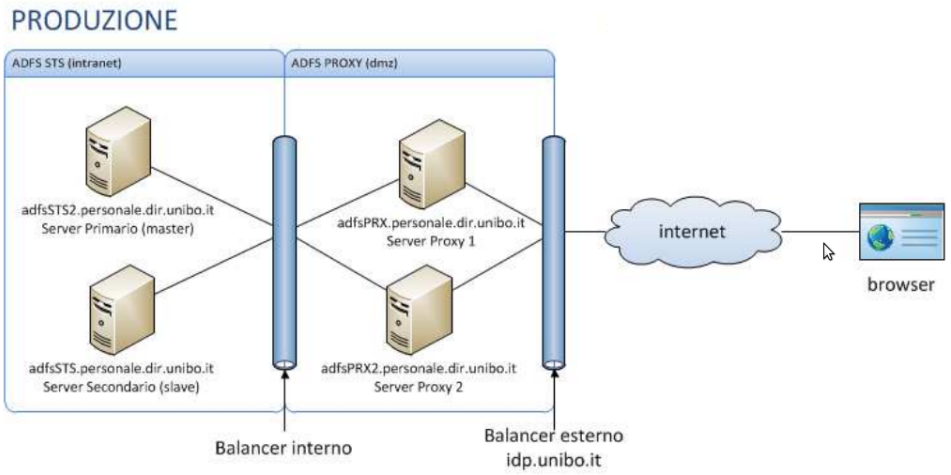

Figura 5.2: Architettura dell'ambiente di produzione

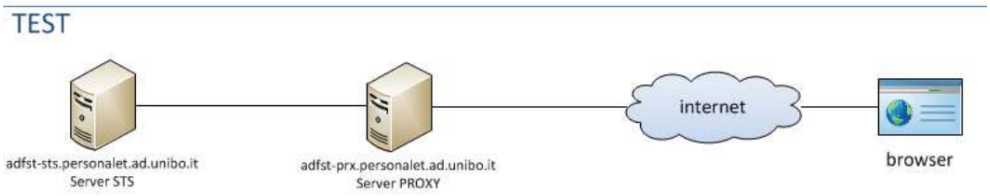

Figura 5.3: Architettura dell'ambiente di test

5.3 SAML

SAML (Security Assertion Markup Language) è uno standard aperto, basato sul formato XML, per lo scambio di dati di autenticazione ed autorizzazione tra parti, in particolare tra un identity provider (SAML autohority) e un service provider (Web service). In particolare SAML usa dei token contenenti delle asserzioni per inviare informazioni riguardanti un utente. SAML è mantenuto da OASIS Security Service Technical Committee.

Uno dei problemi più importanti affrontati da SAML è quello del single sign-on. Le specifiche SAML definisco tre ruoli: l'utente, l'identity provider e il service provider. In un tipico caso d'uso l'utente richiede un servizio al service provider, questo chiederà ed otterrà delle asserzioni dall'identity provider e sulla base di queste deciderà se permettere l'accesso al servizio. Per il rilascio delle asserzioni l'identity provider potrebbe richiedere un qualche tipo di autenticazione da parte dell'utente. Il tipo di autenticazione richiesto non è specificato e potrebbe essere per esempio la richiesta di username e password.

SAML è costruito su una serie di standard già esistenti:

- Extensible Markup Language (XML)
- XML Schema
- XML Signature
- XML Encryption

- Hypertext Transfer Protocol (HTTP)
- SOAP

5.3.1 Versioni

Dalla versione 1.0 SAML ha subito due revisioni, una minore e una maggiore:

- SAML 1.0 è stato adottato come standard OASIS nel novembre 2002.
- SAML 1.1 è stato sancito come standard OASIS nel settembre 2003.
- SAML 2.0 è divenuto uno standard OASIS nel marzo 2005.

5.3.2 Asserzioni e statement

Le asserzioni contengono pacchetti di informazioni di sicurezza costituiti dagli statement:

```
<saml:Assertion ...>
  ...
</saml:Assertion>
```

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

- Authentication Statement:
  Specifica che un utente è stato autenticato precedentemente.
- Attribution Statement:
  Specifica che ad un utente sono associati determinati attributi. Questi sono delle semplici coppie nome-valore.
- Authorization Statement:
  Specifica se un utente può avere accesso o meno a determinate risorse.

5.3.3 Protocols, Bindings e Profiles

Protocols

Un protocollo SAML descrive come certi elementi sono pacchettizzati all'interno di richieste e risposte SAML, dando delle regole che gli elementi dovranno seguire.

Il tipo più importante di protocollo è chiamato query. Esistono tre tipi di query, una per ognuno degli statement precedentemente presentati: authentication query, attribute query, authorization decision query.

Bindings

Un SAML binding è il mapping di un messaggio che usa un protocollo SAML in un formato standard o in un protocollo di comunicazione. Ad esempio il SAML SOAP binding specifica come un messaggio SAML è incapsulato in un messaggio SOAP, il quale è a sua volta incapsulato in un messaggio HTTP.

Profiles

I profili SAML specificano come assertions, protocols e bindings debbano essere usati insieme per supportare uno specifico caso d'uso.

5.3.4 Caso d'uso

Vedremo ora un caso d'uso riferito all'uso di SAML nel SSO.

1. Un utente attraverso un web browser effettua una richiesta ad una risorsa protetta da un service provider.

2-3. Il service provider per conoscere l'identità dell'utente effettua una richiesta di autenticazione all'identity provider attraverso l'utente.

4. L'idp risponde con un form XHTML

```
<form method="post" action="https://.../SAML2/SSO/POST" ...>
    <input type="hidden" name="SAMLResponse" value="response" />
        ...
     <input type="submit" value="Submit" />
</form>
```

5. Il valore del parametro SAMLResponse viene preso dal form ed inviato con una POST al service provider.

6-7-8. La risposta viene processata dal sp e l'utente viene ridiretto alla risorsa precedentemente richiesta.

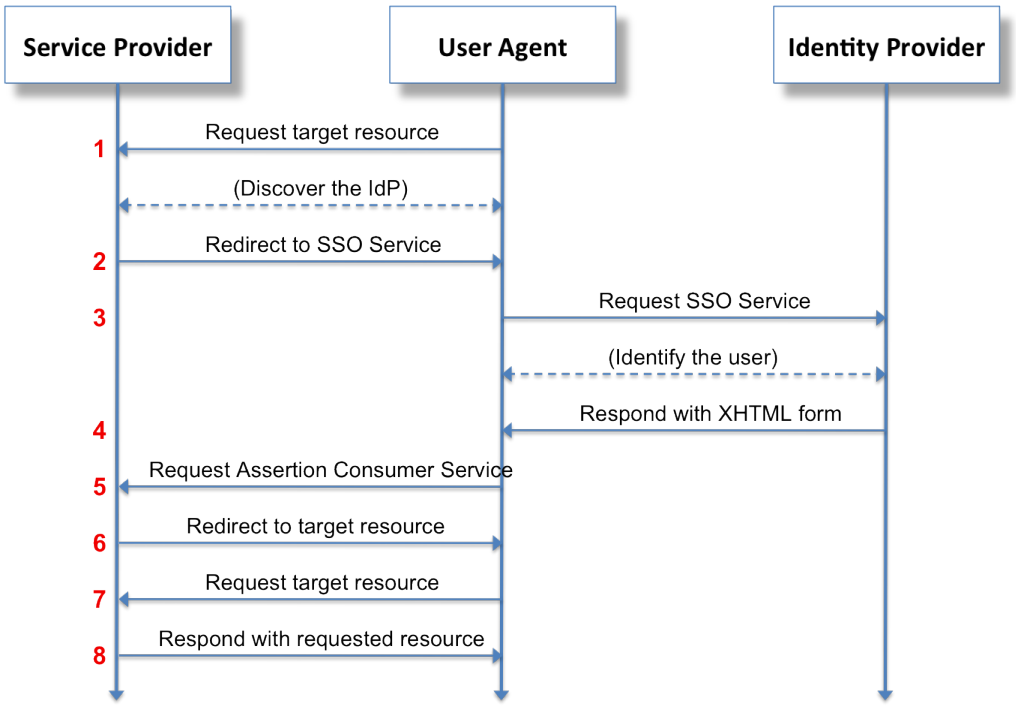

Figura 5.4: SAML: diagramma di sequenza

Capitolo 6

Shibboleth

Shibboleth è un software open source (rilasciato sotto licenza Apache 2.0), gestito da Internet2, tramite il quale è possibile implementare il single sign-on. È basato su diversi standard tra cui spicca SAML (Security Assertion Markup Language) per garantire uno scambio sicuro di attributi. Shibboleth attraverso i suoi componenti divide la gestione delle autenticazioni dalla gestione delle risorse autenticate. Gli utenti vengono infatti reindirizzati alla loro struttura di appartenenza, che avrà il compito di fornire garanzie sulla loro identità e sui loro diritti alla struttura che gestisce le risorse richieste mediante lo scambio di attributi. Shibboleth è pensato per essere usato all'interno di grandi gruppi federati come le aziende o le università.

Shibboleth consente all'utente di ottenere l'accesso a più servizi/risorse protetti, verso i quali si disponga dei necessari permessi, attraverso una singola autorizzazione. Questi possono essere forniti da diverse organizzazioni, l'importante è che queste appartenengano alla medesiama federazione. Tali organizzazioni hanno regole condivise per l'identificazione degli

utenti e l'accesso alle risorse, cos' da garantire uniformità nel trattamento delle informazioni.

6.1 Architettura

La struttura di shibboleth è quella classica che caratterizza tutti i sistemi di Single Sign-on. I principali componenti sono infatti:

- Web Browser: ovvero l'utente;
- Risorsa: che rappresenta la risorsa protetta cui l'utente vuole accedere;
- Identity Provider (IdP): che si occupa di autenticare l'utente;
- Service Provider (SP): effettua il processo di SSO per la risorsa.

Vediamo ora più nel dettaglio l'Identity Provider e il Service Provider.

Identity Provider

Questo componente si occupa di autenticare gli utenti richiedendo le credenziali di autenticazione, di gestirne le sessioni e di restituire gli attributi degli utenti autenticati in seguito ad una richiesta di uno dei Service Provider. L'idp è interfacciato con tutte le organizzazioni appartenenti alla federazioni in modo da avere accesso ai vari registri delle identità (LDAP, DBMS, ...).

L'Identity Provider è costituito dai seguenti elementi:

- Attribute Resolver Service:
  si occupa di trasformare gli attributi recuperati da un database in un insieme di dati relativi ad uno specifico utente, per poi inviarli al client dell'IdP.
- Attribute Filter Service:
  si occupa di filtrare gli attributi richiesti sulla base di regole precedentemente concordate.

- Attribute Authority Service:
  si occupa di codificare gli attributi che riceve in asserzioni SAML.

- Handler Manager Service:
  si occupa di amministrare i vari endpoint dai quali l'IdP può ricevere messaggi.

Service Provider

Questo componente si occupa di verificare l'autenticazione ed i permessi dell'utente quando cerca di accedere ad una risorsa. In un primo momento indirizza l'utente all'IdP, poi rielabora le informazioni che questo gli invia e le utilizza per autorizzare o meno l'accesso alla risorsa.

Il SP è costituito dai seguenti elementi:

- Assertion Consumer Service:
  si occupa della comunicazione con l'IdP e dell'elaborazione delle asserzioni SAML.

- Attribute Requester Service:
  si occupa si ottenere attributi aggiuntivi sull'utente autenticato e delinea politiche di accettazione degli attributi.

6.2 Single Sign-on Steps

Vediamo ora in soli sei passi come funziona il processo di Single Sign-on con Shibboleth:

1. L'utente richiede la risorsa:
   L'utente cerca di accedere ad una risorsa, se non ha una sessione già aperta inizia il processo di SSO.

2. Il SP effettua una richiesta di autenticazione:
   Il SP prepara una richiesta di autenticazione che viene inviata all'IdP attraverso l'utente.

3. L'utente di identifica all'IdP:
   Se l'utente ha già una sessione aperta l'IdP passa alla fase successiva, altrimenti richiede le credenziali per l'autenticazione.

4. L'IdP risponde alla richiesta di autenticazione:
   Dopo aver identificato l'utente, l'IdP prepara una risposta e la invia al SP attraverso l'utente.

5. Il SP controlla la risposta:
   Il SP controlla la validità della risposta, dopodiché crea una nuova sessione utente e rende disponibili alla risorsa le informazioni estratte dalla risposta dell'IdP. Infine reindirizza l'utente alla risorsa.

6. L'utente accede alla risorsa:
   Ora l'utente ha una sessione aperta e la risorsa conosce le sue informazioni, viene quindi permesso l'accesso.

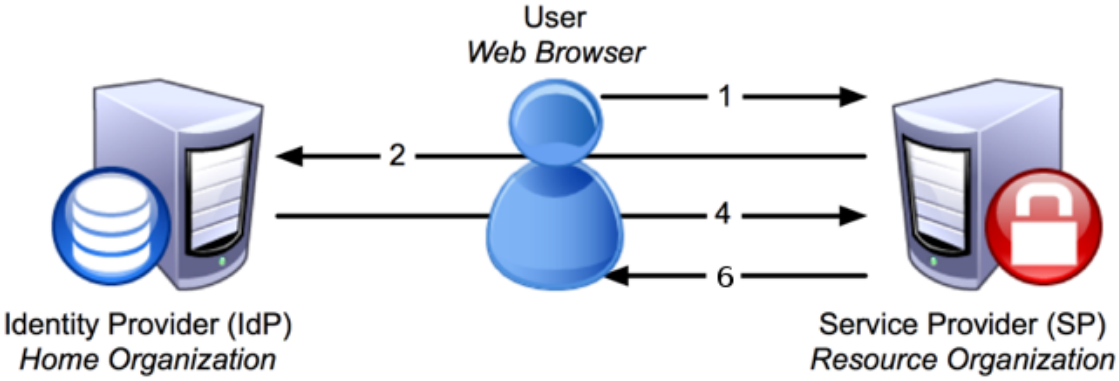

Figura 6.1: Single Sign-on Steps

6.3 Matrice di compatibilità

La seguente tabella rappresenta la matrice di compatibilità per le coppie Sistema Operativo/Browser per le quali il CeSIA garantisce il funzionamento dellautenticazione con Single Sign-On di Ateneo.

	IE 7	**IE 8**	**IE 9**	**Firefox**	**Safari 4**	**Safari 5**
Windows XP SP3	X	X		X		
Windows Vista	X	X	X	X		
Windows 7		X	X	X		
Mac OSX 10.6				X	X	X
Mac OSX 10.7						X
Ubuntu 8.4 LTS				X		
Utuntu 10.4 LTS				X		

Tabella 6.1: Matrice di compatibilità

Capitolo 7

Ambiente di lavoro

In questo capitolo, prima di analizzare il processo di istallazione, configurazione e personalizzazione del sistema, vedremo quali sono state le architetture che i tecnici del laboratorio hanno preso in considerazione al fine di ottenere quella più congeniale alle nostre esigenze.

Nello specifico le soluzioni che sono state prese in esame e che andremo ad analizzare sono:

- L'uso di un reverse proxy per permettere la comunicazione client-server;
- L'uso di macchine virtuali.

7.1 Reverse Proxy

L'idea iniziale era quella di utilizzare un apposito spazio all'interno del server Loew del dipartimento. 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 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. D'altra parte, la funzione di cache del reverse proxy può alleggerire il

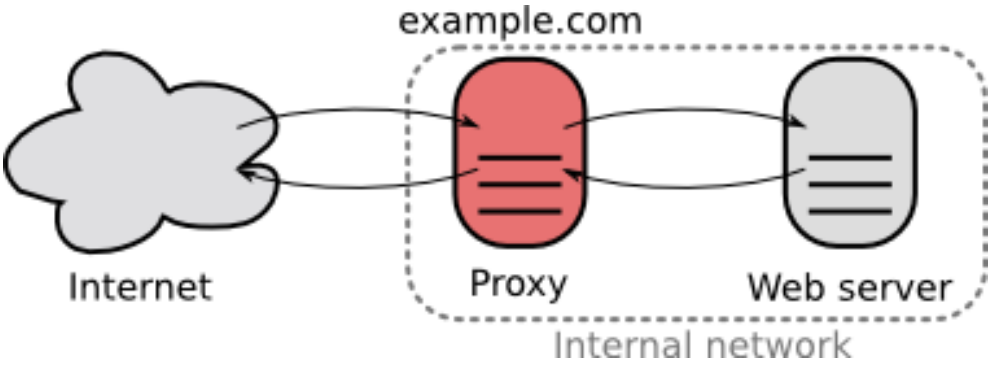

Figura 7.1: Reverse Proxy

carico del server per cui è previsto ed è la ragione per cui un approccio del genere è detto anche acceleratore.

Non tutte le applicazioni presenti sul server Loew sono però gestite tramite reverse proxy, ma solo quelle che necessitano di una connessione https, con certificato SSL rilasciato da una certification authority riconosciuta.Ovvero il nostro caso.

Lavorando in questo ambiente si sono però imbattuti da subito in una serie di problemi di difficile soluzione.

In particolare:

- mancanza di permessi sugli utenti dei tesisti
- procedura d'installazione di OTRS non convenzionale
- rottura dei link dinamici

Per evitare di dover effettuare una massiccia riscrittura dell'architettura del server, al fine di risolvere i problemi riscontrati, si è quindi deciso d'intraprendere una strada alternativa.

7.2 Virtual Machine

Per risolvere i problemi riscontrati con la precedente tecnica sono stae te installate delle macchine virtuali sul server Storage del dipartimento.

Questa soluzione, a fronte di una leggera perdita prestazionale rispetto alla macchina fisica, ha permesso di seguire la procedura standardizzata d'installazione di OTRS e una maggiore libertà di azione. Nello specifico le macchine virtuali non compromettono in nessun modo il sistema principale, permettendo una più semplice ed efficace procedura di backup e ripristino dell'intero sistema.

Sono state quindi installate due macchine virtuali con Ubuntu 12.04 LTS: una per l'ambiente di testing, chiamata ticketdev e una per l'ambiente di produzione chiamata ticket.

Capitolo 8

Linee guida del CeSIA

Il CeSIA ha rilasciato una serie di linee guida che gli sviluppatori di applicazioni per l'Ateneo sono tenuti a seguire scrupolosamente per l'integrazione di applicazioni web con il sistema di Single Sign-On.

Gli scopi sono principalmente due:

- Facilitare la fruizione dei servizi applicativi da parte degli utenti, i quali devono essere consapevoli di lavorare in un ambiente di autenticazione integrata.
- Uniformare l'aspetto delle applicazioni di Ateneo.

Queste lienee guida costituiscono inoltre un pre-requisito per il passaggio in produzione di una qualsiasi applicazione che integra il sistema di SSO.

8.1 Flusso di navigazione

Accesso all'applicazione

L'impostazione descritta per il contesto non autenticato prevede, a centro pagina:

- Logo Single Sign-On (lucchetto rosso)

- Link Login per ridirigere al server per l'autenticazione.

L'utente accede all'applicazione web selezionando il link Login, dopodichè viene ridiretto alla pagina di Login del server Single Sign-On. Se l'utente fornisce delle credenziali valide, il controllo viene fatto tornare all'applicazione web che avrà a disposizione le informazioni dell'utente.

Figura 8.1: Pagina iniziale dell'applicazione

Gestione degli errori

Sarà la pagina di Login dell'IdP a gestire il caso in cui le credenziali inserite non siano valide. Sarà invece compito dello sviluppatore dell'applicazione reindirizzare l'utente ad una pagina apposita, contenente informazioni su chi fornisce il supporto all'applicazione, nel caso in cui l'utente non è autorizzato ad accedere all'applicazione nonostante abbia inserito delle credenziali valide.

Contesto autenticato

Per il contesto autenticato l'impostazione descritta prevede, in alto a destra:

- Username dell'utente che ha effettuato il login (Es: *nome.cognome@unibo.it*).

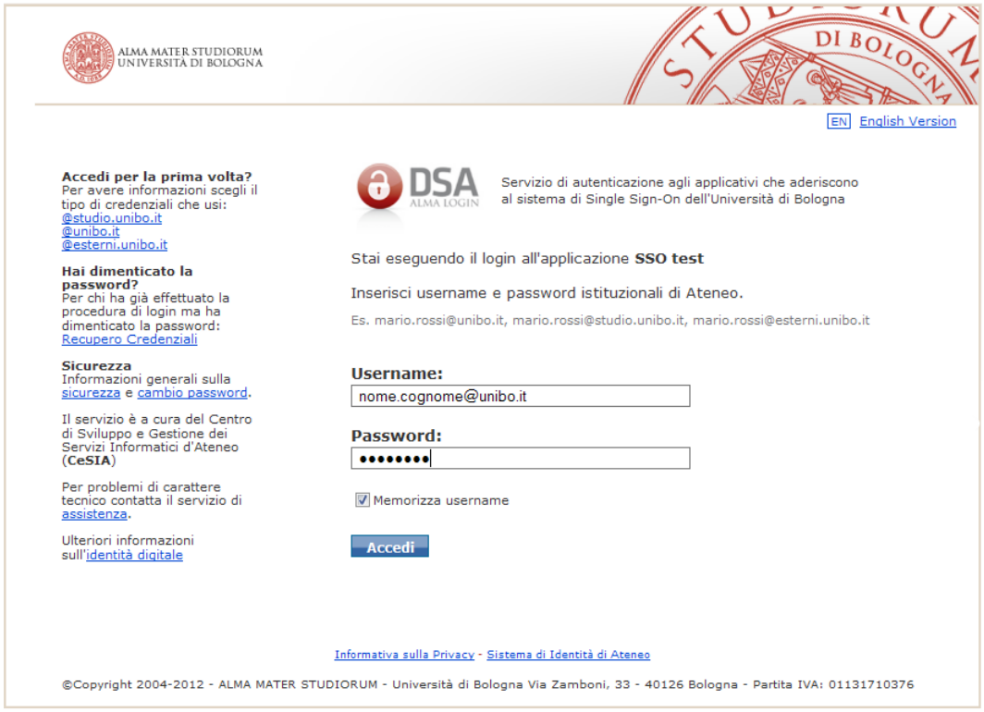

Figura 8.2: Pagina di Login dell'IdP

- Logo Single Sign-On (lucchetto rosso), con il seguente testo per i tag *alt* e *title*:

  - Italiano: *Informazioni sul Single Sign-On di Ateneo.*
  - Inglese: *About University of Bologna Single Sign-On system.*

  Il logo deve essere implementato come link che punta ad una pagina informativa che elenca le applicazioni a cui si è correntemente connessi.

- Link Logout per ridirigere l'utente al server IdP per la gestione del processo di Single Logout.

Figura 8.3: Esempi di pagine in contesto autenticato

Logout

Quando l'utente seleziona il link Logout, presente nel contesto autenticato, viene ridiretto su una pagina dell'IdP in cui sono elencate tutte le applicazioni per le quali ha effettuato l'autenticazione. In questa pagina l'utente potrà decidere se effettuare il Logout da queste. In tal caso l'IdP invierà un messaggio a tutte le applicazioni elencate in modo tale che la sessione di autenticazione venga invalidata. L'avvenuta invalidazione è confermata dalla comparsa delle spunte verdi accanto al nome dell'applicazione.

Una volta effettuato il logout è possibile far comparire sulla pagina il link Torna all'applicazione; per fare ciò basta aggiungere all'URL della pagina di logout il parametro *wreply* con il link di ritorno.

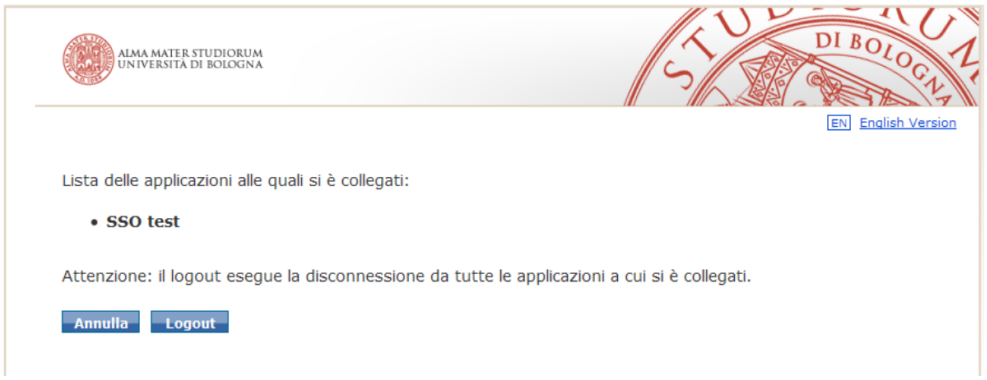

Figura 8.4: Pagina di Logout dell'IdP

Va fatto notare che la pagina di logout è comune a tutte le applicazioni e

visualizza l'elenco di tutte quelle per cui sia stata effettuata l'autenticazione. Scegliendo quindi di effettuare l'operazione di logout non si avrà quindi la possibilità di selezionare le applicazioni da invalidare, infatti verrà negata l'autenticazione a tutte.

Capitolo 9

Istallazione, configurazione e personalizzazione

Nel momento in cui è stato preso in mano il progetto si aveva a disposizione una sola macchina virtuale (*ticket*). I ragazzi che avevano lavorato al sistema in precedenza avevano infatti ritenuto superfluo avere una macchina per l'ambiente di test e una per quello di produzione. Questa decisione è stata però rivalutata.

Il lavoro ha avuto quindi inizio con due macchine virtuali, una (*ticket*) con l'ambiente di lavoro già installato ma non configurato per il suo effettivo scopo (quello di ambiente di produzione) e una (*ticketdev*) con un sistema operativo appena installato.

Verrà ora l'intero processo di installazione e configurazione dei vari componenti del sistema.

9.1 Installazione

Dopo aver configurato le macchine per permettere l'accesso tramite autenticazione RSA sono passato all'installazione dei componenti.

Per iniziare si è installato il web server Apache tramite il comando:

```
apt-get install apache2
```

Al termine dell'installazione che richiede all'incirca venti minuti si è passati ad installare OTRS. Dopo aver scaricato il pacchetto *.tra.gz* contenente il sorgente, questo è stato estratto nella directory */opt/otrs*. Si è poi passati ad installare i moduli Perl necessari al suo funzionamento e il database MySQL, a tal fine tramite il comando aptitude sono stati istallati i seguenti pacchetti:

- libapache2-mod-perl
- 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

Il processo si è concluso con l'installazione del Service Provider Shibboleth:

```
apt-get install libapache2-mod-shib2
```

9.2 Configurazione: OTRS

È stato necessario creare un nuovo utente con home directory */opt/otrs*, che potesse eseguire i cron jobs di OTRS, ed è stato aggiunto al gruppo *webserver*

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

A questo punto sono stati preparati i file di configurazione OTRS. Per fare ciò sono stati copiati i file di configurazione di default presenti nella cartella */opt/otrs/Kernel* e rinominati adeguatamente.

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

Sono stati poi settati i permessi in modo tale che la directory fosse accessibile sia al web server che agli utenti OTRS. Questo è stato fatto mediante uno degli script presenti in */opt/otrs/bin*.

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

A questo punto è stato necessario aggiungere all'interno di apache il file di configurazione di OTRS. Tra i vari compiti di questo file c'è anche quello di aggiungere l'alias *otrs/* che potrà quindi essere usato al posto di *opt/otrs/* al fine di ottenere URL più leggibili.

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

Per concludere è stato necessario creare il database locale di OTRS e preparare i cron jobs. Per quanto riguarda la creazione e associazione del database, OTRS mette a disposizione un web installer raggiungibile tramite browser all'indirizzo *http://serverAddress/otrs/installer.pl*. Al termine della

procedura è stato necessario modificare il file */opt/otrs/scripts/apache2-perl-startup.pl* per configurare *Apache::DBI* al fine di aumentare le prestazioni. Sono quindi state decommentate le righe per l'uso del database MysQL:

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

Vediamo ora quali sono i comandi per la preparazione dei cron jobs e per aggiungerli alla crontab:

```
cd /opt/otrs/var/cron
for foo in *.dist; do cp $foo `basename $foo .dist`; done

cd /opt/otrs
bin/Cron.sh start otrs
```

9.3 Configurazione: Shibboleth

Per la configurazione del Service Provider Shibboleth e l'integrazione di OTRS nel sistema di SSO il CeSIA ha fornito una documentazione più che esaustiva e una serie di file da modificare pronti per essere usati. Di seguito verrà mostrata la procedura di configurazione.

Per iniziare, una volta installato Shibboleth, è necessario creare una coppia certificato-chiave. Questo certificato non sarà esposto agli utenti ma sarà utilizzato solo per la comunicazione tra Shibboleth e l'Identity Provider. La creazione viene fatta col comando *shib-keygen*:

```
shib-keygen -y 5 -h app.esempio.cs.unibo.it
```

Passiamo ora all'utilizzo dei file forniti dal CeSIA. Iniziamo col file *shibboleth2test.xml/shibboleth2prod.xml*, questo file (l'uno o l'altro a seconda se si sta lavorando sull'ambiente di test o di produzione) dovrà essere rinominato per andare a sostiuire il file */etc/shibboleth/shibboleth2.xml*. Prima però deve essere opportunatamente modificato:

- Sostituzione del valore dell'attributo *Host*, dentro *RequestMapper*, con quello del server (Es: *https://ticket.cs.unibo.it*).

- Sostituzione del valore dell'elemento *Path*, dentro *RequestMapper*, con quello della directory in cui è contenuta l'applicazione sul server.

- Sostituzione del valore dell'elemento *entityID*, dentro *ApplicationDefaults*, con il valore che identifica univocamente l'applicazione all'interno del sistema federato.

- Sostituzione del valore dell'elemento *homeUrl*, dentro *ApplicationDefaults*, con l'indirizzo principale dell'applicazione.

A questo punto dovranno essere sostituiti i file *attribute-policy.xml* e *attribute-map.xml*, sempre in */etc/shibboleth*, con quelli forniti e copiare il file *logout.php* all'interno della cartella dell'applicazione autenticata e il file *greencheck.gif* all'interno di una directory non protetta. Il file *logout.php* deve essere adeguatamente modificato in modo che contenga la corretta ubicazione di *greencheck.gif*.

Per concludere è necessario configurare il server web Apache in modo che l'accesso all'applicazione sia regolato da Shibboleth. Per fare ciò è necessario aggiungere alla propria configurazione una sezione del tipo:

```
<Location /directoryapp>
  AuthType shibboleth
  ShibRequireSession On
  ShibUseHeaders On
  require valid-user
  UseCanonicalName On
</Location>
```

Vedremo questo passaggio nella prossima sezione.

9.4 Configurazione: Apache

Per quanto riguarda la configurazione di Apache, sono stati prima di tutto abilitati i moduli che ci servivano per la gestione di perl, l'uso di regole di riscrittura, l'autenticazione SSL e l'uso di Shibboleth:

```
a2enmod ssl
a2enmod rewrite
a2enmod perl
a2enmod shib
```

A questo punto sono state generate le coppie chiave-certificato. Per quanto riguarda l'ambiente di test il certificato è stato autofirmato, mentre per l'ambiente di produzione, per cui è necessario un certificato rilasciato da una certification authority riconosciuta, è stata fatta richiesta al CeSIA. Una volta creata la coppia chiave-certificato si è passati alla creazione e abilitazione dei Virtual Host.

Sono stati creati due Virtual Host, uno per le connessioni sulla porta 80 (http) e uno per le connessioni sulla porta 443 (https).

I file creati (*ticket.cs.unibo.it* e *ticket.cs.unibo.it-ssl*) sono stati posizionati in */etc/apache2/sites-available* e sono stati poi abilitati con i comandi:

```
a2ensite ticketdev.cs.unibo.it
a2ensite ticketdev.cs.unibo.it-ssl
```

Questi comandi non fanno altro che creare un link simbolico a ticket.cs.unibo.it e *ticket.cs.unibo.it-ssl* nella directory */etc/apache2/sites-enabled*. Vediamo ora un po' più nel dettaglio quale è il loro scopo.

Il Virtual Host *ticket.cs.unibo.it* resta in ascolto sulla porta 80 e il suo compito principale è quello, mediante l'utilizzo del modulo *rewrite* di ridirigere l'utente verso il corrispondente URL in https.

```
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}
```

Il secondo Virtual Host (*ticket.cs.unibo.it-ssl*) è invece in ascolto sulla porta 443. Nel file corrispondente vengono specificate le locazioni in cui sono stati messi la chiave privata e il certificato precedentemente creati, inoltre è qui che è stata aggiunta la sezione:

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

che fa in modo che l'accesso all'applicazione sia regolato da Shibboleth. In questo file sono state inoltre aggiunte alcune regole di riscrittura per la semplificazione dell'url dell'applicazione.

```
RewriteRule ^/admin/?$
https://ticketdev.cs.unibo.it/otrs/admin.html [R,L]

RewriteRule ^/customer/?$
https://ticketdev.cs.unibo.it/otrs/customer.html [R,L]
```

9.5 Personalizzazione

9.5.1 Inizializzazione del DB

È stato visto che in fase di installazione OTRS si occupa di configurare il maniera corretta il database, questo viene fatto usando tre file *.sql* presenti in OTRS. Al fine di inserire direttamente nel database le macroaree e i gruppi descritti nel paragrafo 2.3 uno di questi file è stato modificato. Vediamo brevemente quale è il compito di questi modelli:

- otrs-schema.mysql.sql:
  ha il compito di creare tutte le tabelle.
- otrs-initial-insert.mysql.sql:
  inserisce nel DB gruppi e code di default, inoltre si occupa della creazione dell'utente root. Questo è il file che è stato modificato per aggiungere i nostri campi.
- otrs-schema-post.mysql.sql:
  ha il compito di creare le varie asserzioni tra tabelle mediante l'utilizzo di chiavi esterne.

9.5.2 Implementazione delle linee guida

La home page dello spazio web a nostra disposizione è stata modificata in modo tale che rispettasse in pieno le linee guida forniteci dal CeSIA. L'accesso all'applicazione web viene fatto mediante il link Login che una volta

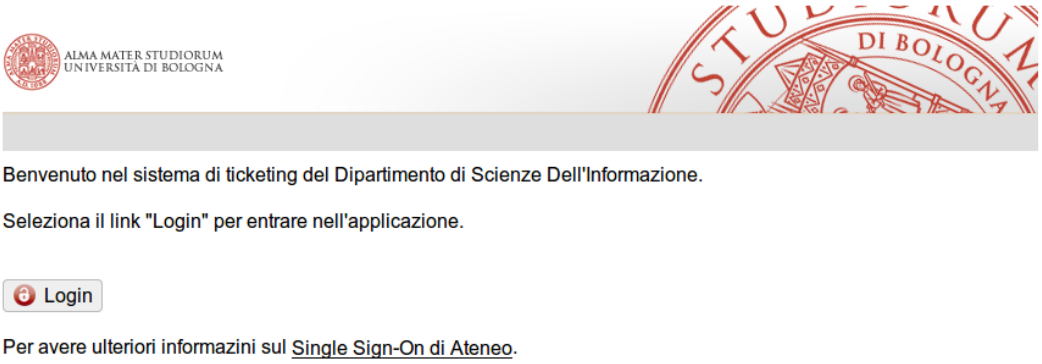

Figura 9.1: HomePage: index.html

selezionato reindirizzerà l'utente verso la pagina di autenticazione dell'IdP.

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

Come indicato dalle linee guida è stata anche creata una pagina cui l'utente deve essere reindirizzato nel caso che, nonostante l'autenticazione sia andata a buon fine, esso non sia in possesso dei diritti per accedere all'applicazione. In questa pagina, oltre alla descrizione del problema riscontrato, devono essere presenti le informazioni necessarie per contattare chi gestisce il sistema, nel nostro caso i tecnici del dipartimento.

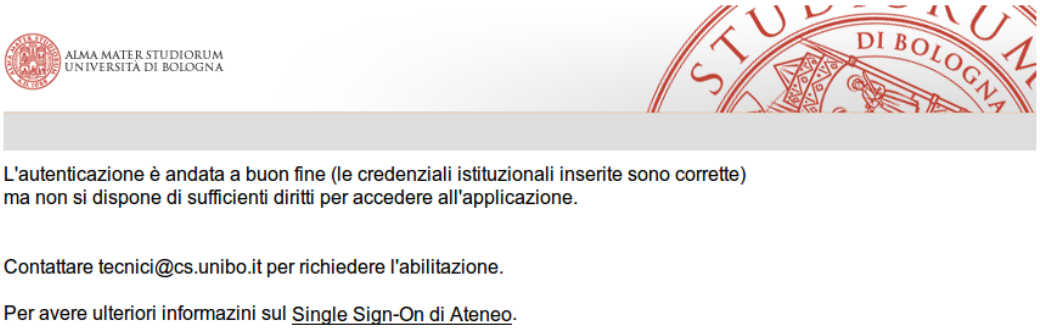

Figura 9.2: Pagina di errore per mancanza di diritti

Nelle linee guida viene specificato che il logout deve avvenire tramite una specifica pagina dell'IdP nel momento in cui l'utente seleziona il link Logout presente nel contesto autenticato. È stato molto semplice far sì che ciò accadesse in quanto il tipo di autenticazione scelto, tra quelli forniti da OTRS, prevedeva già la possibilità di inserire un proprio link verso il quale l'utente deveva essere reindirizzato qualora avesse scelto di disconnettersi dall'applicazione. Il processo di configurazione verrà descritto nella prossima sottosezione.

Per il completamento dei pre-requisiti fondamentali per la messa in produzione dell'applicazione mancava all'appello la gestione del contesto autenticato. Il CeSIA richiedeva che in alto a destra fossero presenti l'username dell'utente, il simbolo del SSO implementato come link e il link Logout che reindirizzasse alla pagina di logout dell'IdP. Per risolvere questo problema è stato necessario creare un nuovo Tema OTRS, partendo da quello di base

(*Standard*) creando così *SSOTheme*. Il file più rilevante al nostro scopo è *CustomerNavigationBar.dtl* che è stato opportunamente modificato al fine di rispondere ai requisiti richiesti. A questo punto, una volta creato, il tema è stato importato in OTRS ed assegnato come tema di default per i nuovi clienti.

9.5.3 OTRS: Login e Logout

OTRS offre la possibilità di gestire l'autenticazione di utenti e agenti in vari modi, potendo anche scegliere di usare una tipologia di autenticazione per gli utenti e una diversa per gli agenti.

Le possibili tipologie sono:

- DB (Default):
  l'autenticazione avviene attraverso il database interno di OTRS, vengono richiesti username e password.

- LDAP:
  l'autenticazione avviene attraverso i dati contenuti in una directory LDAP. Il modulo può accedere ai dati solo in lettura, non è quindi possibile modificare i dati dei clienti e degli agenti attraverso l'interfaccia web di OTRS.

- HTTPBasicAuth:
  l'autenticazione avviene attraverso il confronto dei valori presenti nel DB con quelli presenti in delle specifice variabili d'ambiente. Questo modulo è la soluzione ottimale se si vuole integrare il SSO.

- Radius:
  l'autenticazione avviene attraverso un server Radius.

La scelta più ovvia è stata quella di utilizzare il modulo HTTPBasicAuth; è stato però deciso insieme ai tecnici di utilizzare questo tipo di accesso solo per gli utenti del sistema e di lasciare il sistema di autenticazione di default, quello tramite il database interno, per gli agenti.

Per modificare il sistema di autenticazione per i clienti è stato necessario andare a modificare il file di configurazione di OTRS aggiungendo le seguenti istruzioni:

```
$Self->{'Customer::AuthModule'} =
'Kernel::System::CustomerAuth::HTTPBasicAuth';

$Self->{CustomerPanelLoginURL} =
'https://ticketdev.cs.unibo.it/noAuthorization.html';

$Self->{CustomerPanelLogoutURL} =
'https://idptest.unibo.it/adfs/ls/Prelogout.aspx?
wreply=http://ticketdev.cs.unibo.it';
```

Con la prima andiamo a specificare che il sistema di autenticazione che vogliamo usare per i clienti è l'HTTPBasicAuth, con la seconda andiamo a specificare l'URL verso il quale reindirizzare l'utente nel caso in cui non abbia i diritti necessari ad accedere ad OTRS, mentre con l'ultima specifichiamo l'URL verso il quale gli utenti devono essere reindirizzati quando effettuano una richiesta di logout. Impostando quest'ultimo URL con quello della pagina dell'IdP che gestisce i logout viene soddisfatto uno dei pre-requisiti del CeSIA per il passaggio in produzione dell'applicazione.

9.5.4 PHP: Scripts e Utility

Ci sono stati diversi motivi che hanno portato all'installazione di PHP sulla macchina: il primo è stato sicuramente che il CeSIA ha fornito uno script PHP da usare per testare il sistema di SSO e sarebbe stato superfluo riscrivere un nuovo script per questo scopo. Si è poi deciso di sfruttare ulteriormente questo linguaggio prima installando sulla macchina phpMyAdmin al fine di semplificare la gestione del database e successivamente per la creazione di uno script che si occupasse del popolamento della base di dati di OTRS.

Vedremo ora questi punti con maggiore dettaglio.

viewer.php

Come detto precedentemente, il CeSIA ha fornito insieme alla documentazione per l'integrazione del nuovo sistema di autenticazione di Ateneo uno script di test (*viewer.php*) con l'obiettivo di verificare il corretto funzionamento dell'autenticazione e del rilascio degli attributi richiesti.

Per utilizzare lo script questo deve essere protetto da autenticazione al fine di poter accedere agli attributi rilasciati. Sono state quindi apportate delle leggere modifiche alla configurazione durante la prima fase di test del sistema al fine di utilizzare lo script fornitoci. Una volta finiti i test preliminari la configurazione è stata poi riportata al suo stato originale.

Lo script si occupa, basandosi sul file *attribute-map.xml* presente nella directory di Shibboleth, di estrarre tutte le informazioni ricevute dall'IdP per poi visualizzarle in formato HTML. La pagina inoltre avvisa l'utente nel caso in cui l'IdP non abbia restituito gli attributi riferiti all'utente o nel caso in cui lo script non sia protetto da autenticazione restituendo un errore : *No valid Shibboleth session!*.

phpMyAdmin

Nonostante la possibilità di eseguire comandi SQL da OTRS e quella di utilizzare MySQL da remoto si è pensato che sarebbe potuto risultare utile per gli amministratori avere la possibilità di avere uno strumento che permettesse un accesso semplificato al database senza alcuna sorta di limitazione. Si è quindi deciso di installare sulla macchina PhpMyAdmin.

PhpMyAdmin è un applicazione libera scritta in PHP che permette di amministrare in modo semplice ed intuitivo database MySQL tramite un qualsiasi browser. Il tool è stato prima installato tramite il comando aptitude e poi configurato attraverso l'interfaccia grafica proposta. Successivamente

SWITCH Attribute Viewer

Attributes	Values
Shib-Application-ID	default
Shib-Session-ID	_ad4e3ac9d249aad004e72cc7c73e02e1
Shib-Identity-Provider	http://idptest.unibo.it/adfs/services/trust
Shib-Authentication-Instant	2012-10-26T18:10:35.681Z
Shib-Authentication-Method	urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
Shib-AuthnContext-Class	urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
eppn	paola.rossi@unibo.it
givenName	Paola
idAnagraficaUnica	573036
sn	Rossi
HTTP_SHIB_SESSION_ID	_ad4e3ac9d249aad004e72cc7c73e02e1
HTTP_SHIB_SESSION_INDEX	
HTTP_SHIB_IDENTITY_PROVIDER	http://idptest.unibo.it/adfs/services/trust
HTTP_SHIB_AUTHENTICATION_METHOD	urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
HTTP_SHIB_AUTHENTICATION_INSTANT	2012-10-26T18:10:35.681Z
HTTP_SHIB_AUTHNCONTEXT_CLASS	urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
HTTP_SHIB_AUTHNCONTEXT_DECL	
HTTP_SHIB_ASSERTION_COUNT	
HTTP_SHIB_SCHAC_HOMEORGANIZATION	
HTTP_SHIB_APPLICATION_ID	default

Figura 9.3: Risultato di viewer.php

è stato necessario includere in apache il file di configurazione di phpMyAdmin e, visto che anche phpMyAdmin andrà a lavorare tramite SSL è stata aggiunta in */phpmyadmin/config.inc.php* la voce:

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

Oltre che per gli scopi amministrativi per cui è stato inizialmente installato, phpMyAdmin è risultato particolarmente utile durante lo studio della struttura del database di OTRS. Uno studio approfondito del database è stato infatti necessario al fine di creare lo script per il suo popolamento.

Popolamento del database locale

Nonostante l'autenticazione venga gestita tramite SSO, OTRS richiede comunque un database locale con le informazioni di tutti gli utenti. Questa necessità ha richiesto di dover trovare una soluzione al problema del

popolamento della base di dati. A tal fine sono state pensate due possibili soluzioni:

1. L'inserimento di un utente all'interno del DB viene fatto al momento del suo primo login.
2. La base di dati viene popolata coi dati di tutti i possibili utenti ed aggiornata periodicamente.

Analizziamo le due opzioni un po' più nel dettaglio. La prima soluzione prevede la creazione di uno script che deve essere eseguito non appena l'utente si è autenticato mediante SSO. Questo script avrà il compito di rielaborare gli attributi ricevuti dall'IdP per controllare se l'utente possiede i diritti di accesso ad OTRS e se l'utente è già presente nel database. A seconda del risultato di questi due controlli verranno compiute le seguenti azioni:

- L'utente ha i diritti di accesso:
  - L'utente è presente nel DB:
    viene reindirizzato ad OTRS.
  - L'utente non è presente nel DB:
    viene inserito nel database e a seconda della categoria di appartenza (studente, docente, ...) gli vengono associati i gruppi ai quali può accedere. Infine viene reindirizzato ad OTRS.
- L'utente non ha i diritti di accesso:
  - L'utente è presente nel DB:
    l'utente viene disabilitato, non può essere rimosso per garantire coerenza nel sistema. Una volta disabilitato viene reindirizzato ad una pagina che lo informa di non essere autorizzato all'accesso.
  - L'utente non è presente nel DB:
    viene reindirizzato ad una pagina che lo informa di non essere autorizzato all'accesso.

La seconda soluzione prevede la creazione di uno script che periodicamente, basandosi su una lista degli utenti validi fornita esternamente, vada ad aggiungere i nuovi utenti al database e disabiliti quelli che sono già presenti nella base di dati ma non lo sono più nella lista.

Entrambe le soluzioni presentano dei pro e dei contro, la seconda però aveva uno svantaggio che avrebbe potuto portare diversi problemi. Questo svantaggio è costituito dal fatto che questo approccio è basato su una lista di utenti, lista che deve essere costantemente tenuta aggiornata e deve essere richiesta ad una fonte esterna. Per evitare tutti i possibili problemi che l'ottenimento della lista e il suo mantenimento avrebbero potuto portare si è quindi deciso di adottare il primo approccio.

A questo scopo è stato realizzato uno script PHP che viene eseguito una volta che l'utente si autentica con l'IdP e si comporta nel modo precedentemente descritto.

9.5.5 FAQ

Durante la descrizione dell'architettura dei sistemi di ticketing si è parlato di diversi livelli di gestione delle richieste e della caratteristica di molti sistemi di fornire un qualche strumento mediante il quale l'utente può cercare una soluzione al suo problema prima di creare un ticket.

Questa funzionalità è fornita in OTRS mediante l'utilizzo del modulo FAQ. Questo, una volta installato attraverso il Gestore dei Pacchetti presente nel menù di amministrazione, permette il trasferimento di conoscenze tra agenti e utenti senza che ce ne sia un esplicita richiesta, portando ad un notevole risparmio di tempo.

Il modulo fornisce agli agenti la possibilità di aggiungere nuove FAQ, di suddividerle in categorie e di fornire agli utenti semplici i diritti per poterle consultare.

Vediamo ora qualche altra funzionalità del modulo:

- FAQ Explorer:
  navigazione intuitiva grazie alla suddivisione delle FAQ in temi e all'organizzazione gerarchica.

- WYSIWYG Editor:
  editor per la formattazione degli articoli che permette anche l'inserimento di immagini.

- FAQ articles:
  possibilità di associare diversi attributi agli articoli.

- FAQ attachments:
  possibilità di associare degli allegati agli articoli in modo che siano disponibili agli utenti.

- Full-text and quick search:
  possibilità di effettuare ricerche sull'intero database, anche attraverso l'uso degli operatori relazionali.

- Top 10 articles:
  vista dei 10 articoli con il maggior numero di accessi.

- Ranking/Voting FAQ articles:
  possibilità di votare le FAQ e dare informazioni sulla qualità degli articoli.

9.5.6 Mobile

OTRS dispone di un modulo, chiamato IPhoneHandle, che aggiunge la possibilità di far comunicare OTRS con l'applicazione OTRS iPhone App, che come suggerito dal nome è rilasciata per dispositivi iOS. Su Google Play

è presente un'applicazione (DS Helpdesk) rilasciata dalla dacher-systems che permette di accedere alle stesse funzionalità dell'applicazione per dispositivi Apple anche su device Android.

Questo modulo è facilmente installabile seguendo la procedura standard di OTRS per l'installazione di pacchetti aggiuntivi e cioè tramite l'accesso alla voce Gestione Pacchetti del menù di amministrazione.

L'accesso alle funzionalità dell'applicazione è possibile aggiungendo un nuovo account OTRS attraverso l'inserimento dello specifico URL nell' OTRS App Account Manager. Dopodichè è necessario fornire gli stessi User e Password dell'applicazione web.

Figura 9.4: Confronto tra OTRS App e DS Helpdesk

Capitolo 10

Test

Una volta completata la fase di installazione e quella preliminare di configurazione, si è passati a testare i vari componenti del sistema in modo da riuscirne a completarne la configurazione e l'integrazione tra le diverse parti.

10.1 SSO

Al fine di testare il sistema di autenticazione è stato necessario richiedere la creazione di alcuni utenti di prova al CeSIA, poichè, in questa fase, non era possibile utilizzare le credenziali DSA. È stata quindi richiesta la creazione di diversi utenti, uno per ogni categoria di appartenenza prevista, e a questi sono stati associati dei profili così che, ad ognuno, corrispondesse un utente dei claim (attributi) SAML.

Una volta ricevuta conferma della creazione delle credenziali di test è stato quindi possibile testare il SSO. Come prima cosa la configurazione è stata leggermente modificata al fine di reindirizzare l'utente verso lo script viewer.php e non verso OTRS. Lo scopo di questa fase era infatti quello di controllare che l'autenticazione andasse a buon fine e che l'IdP restituisse tutti gli attributi richiesti.

In un primo momento ci si è trovati difronte a due problematiche:

- La pagina viewer.php restituiva un messaggio di errore.

- Andando sulla pagine del logout, l'utente non risultava autenticato su nessuna applicazione.

Fortunatamente lo script viewer.php restituendo il messaggio ne ha suggerito la motivazione. La modifica apportata alla configurazione fino a quel momento prevedeva infatti solo un reindirizzamento allo script, senza tenere conto del fatto che, per un buon funzionamento del sistema, le pagine che hanno necessità di utilizzare i claim forniti dovevano essere protetette da autenticazione. In seguito a questa piccola modifica si è potuto constatare che l'IdP restituiva tutti gli attributi richiesti e che quindi il processo di autenticazione non aveva problemi.

Anche le difficoltà relative al logout sono state facilmente risolte, infatti erano dovute ad un errore durante la modifica della configurazione che serviva a portare il flusso dell'applicazione verso viewer.php. Shibboleth infatti richiede che, per un corretto funzionamento del Single Logout, l'URL del punto da cui parte l'autenticazione dell'applicazione coincida con l'entityID che identifica univocamente quell'applicazione. Risolto anche questo problema si è raggiunto lo scopo di avere un sistema di SSO funzionante che avesse come applicazione di riferimento, per il controllo degli attributi restituiti dall'IdP, lo script viewer.php .

10.2 OTRS

Subito dopo l'installazione in OTRS è presente solo l'utente root. La prima cosa fatta è stata quindi quella di accedere con le credenziali dell'utente root attraverso l'interfaccia degli agenti. Una volta verificata la possibilità di autenticarsi come amministratore è stato quindi creato un primo agente di prova. Una volta autenticati come agente con le nuove credenziali si è poi passati alla creazione di un primo cliente di prova per verificare che anche l'accesso attraverso l'interfaccia per i clienti (autenticazione tramite database locale) funzionasse a dovere.

A questo punto, una volta verificato il funzionamento dell'autenticazione mediante DB sia per gli agenti che per i clienti si è provveduto a fare dei semplici test di creazione/gestione dei ticket e delle altre funzionalità ritenute utili. Si è quindi passati all'integrazione col sistema di SSO.

Come prima cosa è stato modificato il file Config.pm nella cartella Kernel di OTRS in modo da impostare come metodo per l'autenticazione dei clienti HTTPBasicAuth. In questa maniera invece di visualizzare il form per la richiesta dell'username e della password, OTRS va a cercare i dati per l'autenticazione all'interno delle variabili d'ambiente in cui Shibboleth va a posizionare gli attributi restituiti dall'IdP, se trova un riscontro all'interno del database locale permette l'accesso all'utente. Allo scopo di testare questo processo è stato creato un nuovo cliente all'interno di OTRS con gli stessi attributi di uno degli utenti di test creati per testare il SSO. In seguito ad una piccola modifica del file che si occupa di gestire questo tipo di autenticazione, in cui sono state cambiate le variabili da cui OTRS cercava di recuperare i dati necessari, il login dell'utente è avvenuto con successo senza nessun problema.

Per poter avere un sistema completo mancava ancora lo script per il popolamento automatico del database, che ha lo scopo di evitare l'inserimento manuale dei nuovi utenti come nel caso dell'utente di test appena visto. Si è quindi passati alla progettazione ed integrazione nel sistema dello script descritto nella sezione 9.5.4.

Conclusioni

Il lavoro svolto può essere suddiviso in due fasi: una prima in cui sono stati analizzati tutti gli strumenti necessari a soddisfare, nel miglior modo possibile, le esigenze dei tecnici del dipartimento. Una seconda in cui si è passati alla realizzazione del sistema utilizzando lo strumento scelto.

La prima fase si è incentrata sullo studio dei sistemi di ticketing, sulla loro struttura e funzione, così da valutare i diversi strumenti presenti sul mercato. Questi sono stati poi filtrati tenendo conto delle caratteristiche emerse in seguito all'analisi dei requisiti suggeriti dal dipartimento. Tra tutti i sistemi è stato scelto OTRS, si è quindi passati ad un suo studio più approfondito. Questa fase si è conclusa con l'analisi del nuovo sistema di autenticazione dell'Ateneo (Single Sing On) e dello strumento indicato dal CeSIA per la sua implementazione (Shibboleth).

Nella seconda fase il lavoro si è inizialmente concentrato sulla definizione dell'architettura dell'ambiente di sviluppo. Una volta definita la necessità di utilizzare un doppio ambiente di lavoro (produzione e sviluppo) si è passati all'installazione e alla configurazione dei vari componenti necessari per la realizzazione del sistema. Questi componenti sono stati inizialmente gestiti singolarmente e solo dopo un attenta fase di test si è deciso di passare alla loro integrazione e customizzazione.

Il sistema allo stato attuale segue tutte le linee guida indicate dal CeSIA come pre-requisiti necessari alla messa in produzione; la successiva fase del lavoro ne prevede quindi la richiesta formale al CeSIA.

Altre operazioni che devono essere compiute nell'ambiente di produzione sono:

- l'inserimento dei dati relativi ai tecnici del dipartimento all'interno della tabella degli agenti;
- l'aggiunta all'interno del sistema delle FAQ in modo che queste, grazie al modulo installato, possano essere disponibili ai clienti del sistema.

Si potrebbe inoltre pensare di apportare delle ulteriori modifiche al tema creato per OTRS in modo da avere una maggiore corrispondenza tra l'interfaccia dell'applicazione e quella dei siti dei corsi di laurea del dipartimento.

Bibliografia

[1] Apache Software Foundation. http://www.apache.org/

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

[3] DS Helpdesk. https://play.google.com/store/apps/details?id=org.dacherSystems.dsHelpdesk

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

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

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

[7] OTRS App. https://itunes.apple.com/it/app/otrs/id383841790

[8] phpMyAdmin. http://www.phpmyadmin.net/home_page/index.php

[9] Request Tracker. http://www.bestpractical.com/rt/

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

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

[12] Wikipedia - Help Desk. http://it.wikipedia.org/wiki/Help_desk

[13] Wikipedia - Issue Tracking System. http://en.wikipedia.org/wiki/Issue_tracking_system

[14] Wikipedia - Comparison of Issue Tracking System. http://en.wikipedia.org/wiki/Comparison_of_issue-tracking_systems

[15] Wikipedia - SAML.
http://en.wikipedia.org/wiki/Security_Assertion_Markup_Language

[16] Wikipedia - SAML 2.0. http://en.wikipedia.org/wiki/SAML_2.0

[17] Wikipedia - Single Sign-On.
http://en.wikipedia.org/wiki/Single_sign-on

[18] CeSIA. Single Sign-On: concetti generali e stato dell'arte in Ateneo. 2011.

[19] CeSIA. Single Sing-On: nuove implementazioni e dismissione Web Service DSAAuthentication. 2011.

[20] CeSIA. Integrazione del Single Sign-On di Ateneo in applicazioni web. 2012.

[21] CeSIA. Specifiche tecniche per la configurazione del Single Sign-On di Ateneo in applicazioni web gestite con Shibboleth. 2012.

[22] 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

Desidero innanzitutto ringraziare il Professor Vittorio Ghini per i preziosi insegnamenti durante questi cinque anni di università e per le ore dedicate alla mia tesi. Inoltre, ringrazio il Dr Paolo Marinelli che è stato sempre disponibile a dirimere i miei dubbi durante il lavoro.

Vorrei poi esprimere la mia sincera gratitudine verso tutte quelle persone che mi sono state vicine, in un modo o nell'altro, durante il percorso che mi ha condotto al raggiungimento di questo obiettivo che è solo il primo passo del lungo viaggio che ho deciso di intraprendere cinque anni fa.

I primi che devo ringraziare sono sicuramente i miei genitori che mi hanno dato la possibilità e i mezzi per scegliere la strada che ritenevo più giusta; fornendomi sempre il massimo appoggio in ogni situazione. Non penso che sarei stato in grado di ottenere questo risultato senza la grande fiducia che hanno avuto nei miei confronti e che almeno in parte spero di aver ripagato.

Devo poi ringraziare mio fratello Roberto e la mia ragazza Francesca. Roberto è sempre stato per me come una guida, chiunque stia al suo fianco è invaso dalla voglia di imparare cose nuove e combattere la propria ignoranza. Devo anche rendergli merito di aver temprato la mia, comunque poca, pazienza; dover combattere la follia che si alterna al suo genio è stata per me una palestra per affrontare il pazzo mondo in cui ci troviamo. Francesca è la mia salvezza, mi aiuta e mi sostiene nell'affrontare i problemi di tutti i giorni ed è sempre li ad aspettarmi quando sono lontano. Se in questi ultimi anni non mi fosse stata accanto non avrei potuto godere di alcuni dei momenti più belli della mia vita.

Ringrazio anche tutti gli amici che mi hanno accompagnato in questo percorso, in particolar modo i miei coinquilini Andrea, Daniele, Manuel, Maria, Paolo e i miei compagni di studio e di svago, Andrea Pola, Domiziana Suprani, Enrico Succi, Eugenia Panella, Lorenzo Maiani, Marco Perrone, Michele Villani che mi hanno accolto e fatto sentire a casa.

E ora, che il viaggio continui!
Gotta catch 'em all!