Alma Mater Studiorum · Università di Bologna

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

UN PORTLET PER LA GESTIONE DI TICKET: implementazione e test case

Tesi di Laurea in Architettura degli Elaboratori

Relatore:
Chiar.mo Prof.
Vittorio Ghini

Presentata da:
Lorenzo Casanova

Sessione I
Anno Accademico 2011/2012

A mio nonno Giuseppe...

Indice

Elenco delle figure

Introduzione

Le aziende di qualsiasi settore sono alla continua ricerca della maggiore competitività ed efficienza. Per conseguire questi obiettivi si cerca di ottimizzare i processi interni e migliorare le relazioni con i principali stakeholder (clienti e fornitori) anche attraverso strumenti appositi e sfruttando il know-how maturato. In tale contesto, si inseriscono i sistemi informatici che raccolgono le segnalazioni inerenti la qualità dei prodotti e dei servizi erogati dalle aziende. L'attuale contesto economico è interessato da dinamiche evolutive che causano nelle aziende il bisogno di attuare un sempre più profondo e continuo adeguamento relativo non solo al modo di operare, ma anche – e soprattutto – alle modalità di gestione dei rapporti con l'esterno. Un aspetto importante è svolto dalle attività di ticketing e conseguente monitoraggio che permettono di raccogliere e seguire in modo strutturato l'andamento della segnalazione di un problema di prodotto o di un disservizio, tenendo sotto controllo i parametri ritenuti chiave al fine di arrivare alla sua risoluzione. E' evidente l'importanza, per un'azienda, di avere sistemi di segnalazione e di monitoraggio adeguati, in quanto, in una realtà ipercompetitiva come quella attuale, qualunque strumento permetta di avere un vantaggio competitivo viene adottato, adattato ed implementato. Grazie allo sviluppo delle tecnologie informatiche è possibile effettuare attività di controllo in tempo reale – anche a distanza – per fini decisionali, cosa che aumenta il potenziale competitivo. Attraverso i sistemi informatici è possibile raccogliere la segnalazione di un cliente, affidarla alle competenze professionali di uno staff interno e provvedere alla sua risoluzione. Questo permette di ridurre i tempi

di risposta dell'azienda che ha fornito un prodotto o un servizio, facendo sì che il cliente abbia un livello di soddisfazione (la cosiddetta customer satisfaction) più alto. Un sistema di ticketing permette una più facile gestione delle problematiche da parte delle aziende che devono dare un supporto ai prodotti e ai servizi che offrono. In pratica, attraverso un semplice form compilabile online, un utente può segnalare all'azienda un problema e ricevere una risposta per la risoluzione. Dall'altra parte, l'azienda dà vita ad un efficiente sistema di catalogazione di tutte le problematiche lamentate dagli utenti. Se l'azienda lo desidera, inoltre, è anche possibile dare al cliente la possibilità di esaminare lo storico delle segnalazioni ricevute, in modo da permettergli di fare ricerche prima di inviare la propria segnalazione. Se trova risposta al problema eviterà di contattare l'azienda, che potrà in tal modo ridurre il carico di lavoro interno. Il mercato offre una vasta gamma di questi sistemi, sia gratuiti sia a pagamento. Durante il tirocinio che abbiamo svolto in azienda presso la D'vel snc ci è stato proposto di lavorare sullo sviluppo di web application (nel nostro caso portlet) da integrare nel portale open source chiamato Liferay, che sta ottenendo anche in Italia il consenso di molte aziende e clienti. Dato che siamo rimasti piacevolmente colpiti da questo prodotto e considerando il fatto che l'11 marzo 2011 la Liferay ha aperto un marketplace per permettere a sviluppatori terzi di rendere disponibili le proprie portlet custom, abbiamo pensato di scrivere un software per la gestione di ticket, un tipo di portlet che attualmente il portale non implementa. Il portale Liferay è sviluppato in due versioni: la Community Edition, gratuita, e l'Enterprise Edition, a pagamento. La prima contiene tutte le ultime feature disponibili, però, non essendo testata in modo approfondito, può presentare alcuni bug. La seconda, essendo a pagamento, è sottoposta ad un maggiore controllo prima del rilascio, non contiene le ultime feature ed è molto più stabile della Community Edition. Per lo scopo della tesi, puramente accademico, abbiamo lavorato usando la versione Community Edition. Lo sviluppo della portlet è passato attraverso vari passaggi:

- analisi dell'offerta attuale del mercato

- analisi delle feature che la nostra applicazione deve implementare
- progettazione del database
- progettazione dell'applicazione
- elenco delle funzionalità implementate al fine di poterle testare prima di rilasciare il software.

Lo sviluppo dell'applicazione e la stesura di questa tesi sono state realizzate in collaborazione con il collega Alessandro Fogacci.

Capitolo 1

Issue tracking system

Un Issue tracking system è un software utilizzato per gestire in modo organizzato i problemi di prodotti o servizi segnalati dall'utente e giungere alla loro risoluzione. Il sistema è implementato con i dati e le informazioni fornite da chi segnala un problema, la presa in carico della segnalazione e il flusso di lavoro coerente con la risoluzione dello stesso. Il ticket è il cuore di tale sistema: un tempo cartaceo, oggi è un insieme di bit gestibili in modo molto più efficace rispetto ad un passato neppure tanto remoto.

1.1 La storia

I sistemi di gestione, assistenza e monitoraggio delle segnalazioni relative ad eventuali problemi di vari prodotti esistevano già nell'era preinformatica. Il termine ticket nasce dal fatto che la raccolta delle comunicazioni di quanti segnalavano un problema ad un determinato prodotto o ad un servizio era affidata originariamente a piccoli biglietti che venivano attaccati ad una lavagna a muro. La lavagna con il proprio corredo di ticket rappresentava il sistema di gestione di tutte le segnalazioni ricevute. Quando riceveva una segnalazione da un utente, il responsabile tecnico compilava a mano un biglietto cartaceo con i dettagli dell'utente e un breve riassunto della richiesta, per poi appenderlo sulla lavagna in una determinata posizione – di norma

l'ultima, ma modificabile in base alla priorità data – nella colonna "lavori da effettuare" di un determinato operatore dello staff di supporto. In questo modo il responsabile decideva chi avrebbe dovuto lavorare alla segnalazione e con quale priorità. Oggi i sistemi sono completamente informatizzati e prendono il nome di Issue tracking system.

1.2 Che cosa è un ticket

Un ticket è una richiesta di supporto che, all'interno del sistema, è identificata da un insieme di informazioni utili all'azienda per giungere alla risoluzione del problema, in quanto contiene tutte le indicazioni fornite dall'utente in merito al problema riscontrato nel prodotto o nel servizio acquistati. E' composto generalmente da:

- numero univoco di referenza, utilizzato per permettere allo staff o all'utente stesso di localizzare, modificare o aggiungere dati alla segnalazione effettuata

- titolo e descrizione dettagliata del problema riscontrato

- data di apertura della segnalazione

- priorità, basata sulla gravità descritta nella segnalazione. I ticket con priorità "critica" sono i più gravi e dovrebbero essere risolti nel minor tempo possibile e prima di tutti gli altri, in quanto identificano problemi che di norma impediscono, in parte o in toto, il corretto funzionamento del sistema su cui è stata effettuata la segnalazione. I ticket con priorità "bassa" o "nulla" sono di importanza minore. Di solito si tratta di piccoli difetti che non impediscono il funzionamento del prodotto e che vanno risolti quando nel sistema non sono presenti segnalazioni di importanza e criticità più elevate.

1.3 Funzionamento

L'architettura di un sistema di ticketing è molto semplice: abbiamo un database che è lo storage principale per tutti i dati necessari al sistema (ticket, staff, utenti...). L'utente finale può creare segnalazioni completamente nuove, leggere quelle già presenti e aggiungere dettagli per agevolare la risoluzione del problema. Quando un utente effettua una modifica, il sistema registra l'azione (cosa è stato fatto, chi l'ha fatto e in che data) per mantenere uno storico degli avvenimenti. Ogni operatore registrato nel sistema ha segnalazioni a lui assegnate, diventando quindi responsabile della risoluzione di un determinato problema.

1.4 Workflow

Lo scenario-tipo di come funziona normalmente un sistema di ticketing è il seguente:

1. un utente apre una segnalazione attraverso una chiamata telefonica, una mail o tramite un modulo compilabile online per registrare il problema nel sistema

2. quando la segnalazione è registrata in modo corretto, lo staff tecnico può visualizzarla, prenderla in carico o modificare alcuni parametri (magari l'utente segnala come "critico" un problema che l'operatore può riassegnare come "bassa criticità")

3. ogni volta che l'operatore incaricato della risoluzione del problema lavora e avanza verso la chiusura del ticket, i dati sulla segnalazione sono aggiornati in modo da rendere osservabile da terzi (utenti ed altri operatori) il progresso effettuato

4. quando l'operatore risolve il problema, il ticket relativo è marcato come "risolto" nel sistema.

Capitolo 2

Analisi di alcuni software esistenti

Analizziamo i principali software per la gestione dei ticket messi in commercio dalle softwarehouse al fine di confrontarli tra loro e valutarne gli aspetti positivi e negativi. Vogliamo capire, soprattutto, quali delle loro funzionalità possano essere adottate per le esigenze degli obiettivi che ci siamo proposti. Questa analisi è di valido aiuto per capire bene come funziona un sistema per la gestione dei ticket e quali sono le feature base che un sistema del genere deve implementare. Di seguito presentiamo l'elenco dei sistemi analizzati, che ci sono stati suggeriti dall'azienda D'vel durante il periodo del tirocinio.

2.1 OpenTT

Il sistema si propone come soluzione rivolta alla piccola e media impresa per la risoluzione del problema della gestione delle richieste attraverso l'integrazione di:

- un potente motore di gestione delle richieste
- un semplice ed intuitivo contenitore di informazioni strutturate nella forma domanda/risposta o titolo/contenuto (FAQ - Frequently Asked Question)

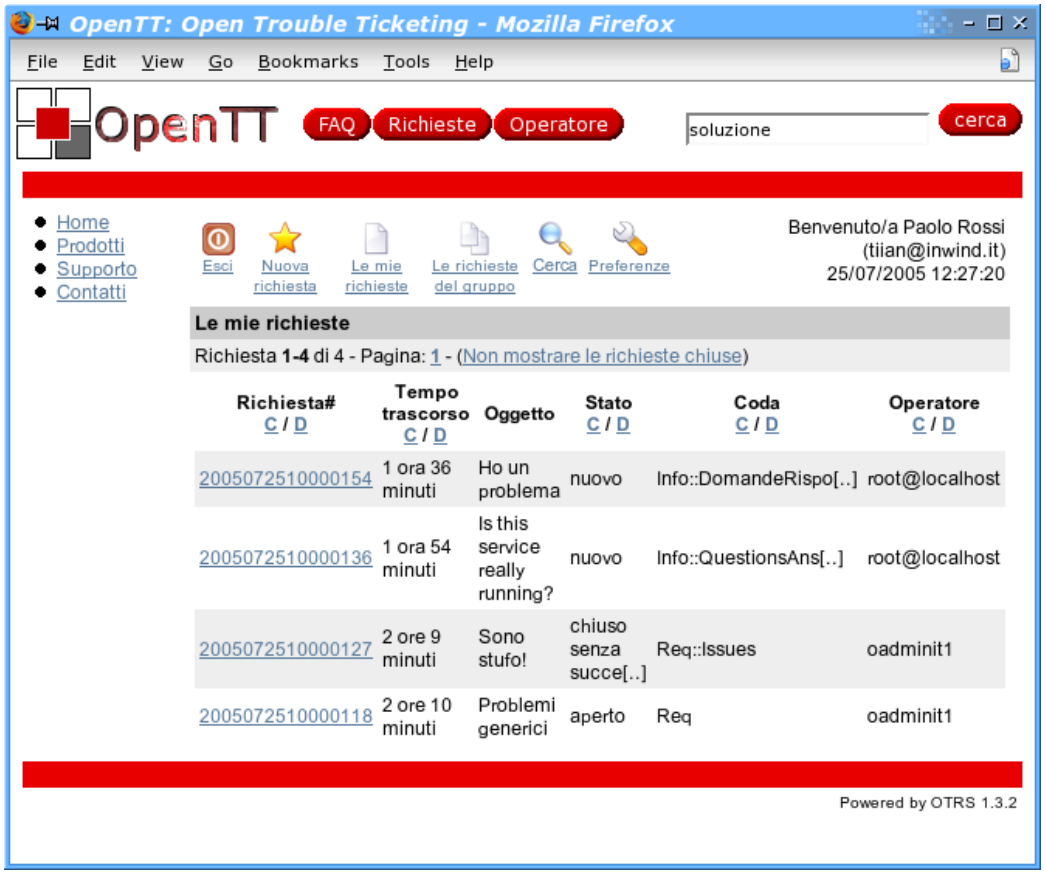

Figura 2.1: Interfaccia utente OpenTT

- un raffinato strumento di reportistica grafica per tenere sotto controllo la situazione.

OpenTT è un'applicazione completamente web based. Non richiede né l'installazione di software client né il download di applet Java o l'utilizzo di componenti ActiveX. E' compatibile con i browser più diffusi sul mercato attuale (Mozilla, Internet Explorer, Konqueror...) e può essere eseguita sotto tutti i principali sistemi operativi: Microsoft Windows, GNU/Linux, Mac OS X. Il software che compone questa soluzione – sistema operativo, database e application server, moduli applicativi originali, integrazioni ed estensioni – è realizzato dalla Primeur srl e distribuito con licenza approvata dall'Open Source Initiative. La risoluzione è certificata per volumi di 1.000 richieste

giornaliere ed è realizzata per garantire i livelli di affidabilità e scalabilità desiderati dal cliente.

2.1.1 Funzionalità

- Apertura segnalazioni via web
- Apertura segnalazioni via e-mail
- Apertura segnalazioni via telefono, tramite operatore (disponibile l'integrazione per CosmoCall ed i principali software di multimedia contact center)
- Gestione di utenti, gruppi e politiche di sicurezza (con database locale o server LDAP)
- Gestione gerarchica delle code di richiesta
- Gestione priorità, allarmi, risposte preconfezionate
- Accounting risorse
- Supporto multilingua
- Grafici sull'andamento generale delle segnalazioni aperte / chiuse
- FAQ.

2.2 SitePanel

SitePanel è un sistema di supporto per le aziende di grandi dimensioni nell'ambito della catalogazione dei problemi. Fornisce tutte le principali caratteristiche che un sistema di questo tipo deve avere. SitePanel è facile da installare e configurare. Il sistema fa uso comune di webserver e moduli PHP per assicurare il corretto funzionamento su qualsiasi configurazione del server web. Dal pannello di configurazione è possibile personalizzare l'intero sistema:

Figura 2.2: Interfaccia utente SitePanel

- creare account e-mail
- creare nuovi dipartimenti
- personalizzare la user interface
- definire i template per le e-mail.

Questo sistema punta molto sulla semplicità della sua Gui (Graphical user interface) sia dal lato della clientela, rendendo il sistema molto facile e a portata anche dell'utente meno esperto, sia dal lato rivolto al personale tecnico, che in questo modo è in grado di adattarsi ad esso in poco tempo.

2.2.1 Funzionalità

- Sistema proprietario di template – Con questo sistema è possibile personalizzare l'interfaccia dell'utente finale con il marchio dell'azienda e scegliere la combinazione di colori. Il sistema di template è facile da usare, ma, allo stesso tempo, offre molte opzioni avanzate.

- SitePanel Unlimited Gui sets – Se l'azienda possiede vari marchi è possibile creare Gui per ognuno di essi. SitePanel dà la possibilità di assegnare aree dell'help desk ai singoli gruppi Gui. Conoscenze di base, annunci e dipartimenti possono essere assegnati a uno o a tutti i set Gui creati.

- Database – Il sistema possiede un avanzato database integrato che fornisce ai clienti un elenco rapido dei problemi risolti, in modo da consentire loro di risolvere quelli semplici senza dover passare dal servizio clienti. In questo modo si riduce il carico di lavoro di chi deve lavorare i ticket (l'azienda che fornisce il supporto).

- Apertura ticket via e-mail – Con SitePanel si possono aprire ticket da un vasto numero di fonti diverse, anche via e-mail. Ciò permette ai clienti di utilizzare il proprio client di posta elettronica per l'apertura dei ticket. Il motore di analisi delle e-mail pervenute è completamente personalizzabile ed è in grado di gestire messaggi con allegati e messaggi in formato html. Il sistema possiede anche un potente filtro antispam.

- SitePanel Api – Include anche un set completo di Api che consente di integrare in modo rapido il sistema nel sito web esistente.

- Applicazione TicketCheck - Incluso in SitePanel c'è TicketCheck2, l'applicazione desktop scritta in Java che consente di tenere traccia del centro di supporto, senza dover avere anche un browser web aperto

2.3 Kayako Fusion

E' un sistema di ticketing usato da aziende di un certo rilevo, quali Acer, Worner Bros, Symantec, FedEx, DHL, Nasa e Peugeot. Kayako offre tre soluzioni di helpdesk: Resolve (è un sistema di mail e ticket management), Engage (offre vari metodi di contatto diretto con i clienti, integrando chat in tempo reale, chiamate voip e servizi di desktop remoto) e Fusion (la verisone completa, composta dall'unione delle funzionalità delle due versione precedenti). Kayako Fusion è il leader mondiale dei sistemi di helpdesk e offre alle organizzazioni che ne fanno uso una customer experience ottima rendendo migliore il lavoro di ogni team, indipendentemente dalla dimensione aziendale.

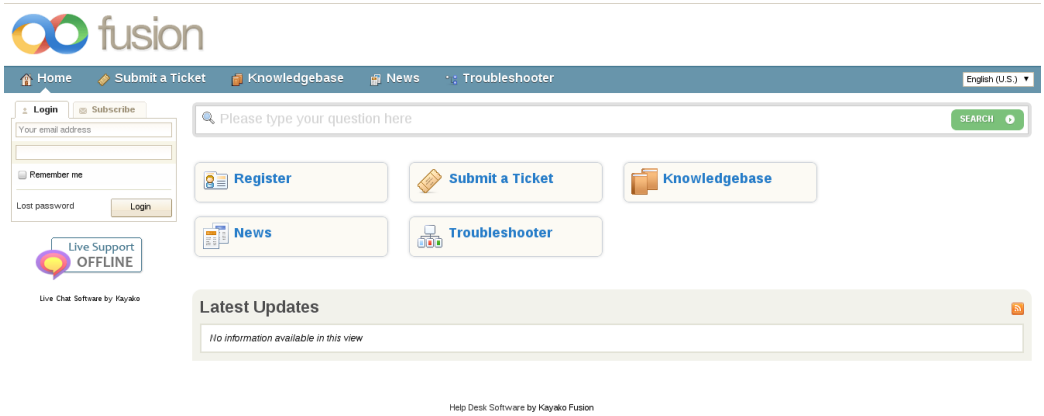

Figura 2.3: Interfaccia utente Kayako

2.3.1 Funzionalità

- Ricezione del ticket dalla piattaforma web based, dalle e-mail e tramite supporto helpdesk
- Chiamate voip e live chat per velocizzare il rapporto staff-cliente
- Alla chiusura di un ticket o di una sessione di chat tra operatore e cliente, il sistema offre a quest'ultimo la possibilità di compilare

un modulo (configurabile da parte dell'azienda) indicante il grado di soddisfazione per la prestazione ricevuta

- Supporto tecnico remoto, attraverso la condivisione remota del desktop
- I ticket registrati nel sistema sono visibili da tutto lo staff tecnico e non solo dall'owner, in modo da favorire il lavoro di team ed evitare, tramite Collision Locks, che più operatori modifichino lo stesso ticket contemporaneamente
- Calcolo dei tempi: data una previsione di ore lavorative per la risoluzione di un ticket, il sistema attribuisce alla segnalazione il giorno previsto di chiusura, escludendo i giorni festivi o di ferie.
- Possibilità di costruire una Knowledgebase con articoli di cui i clienti possono valutarne l'utilità o aggiungere commenti propri
- Offerta di tantissime Api per facilitare l'integrazione del sistema con applicazioni di terze parti
- Sistema di automazione con regole configurabili per filtrare, ordinare e assegnare i ticket in arrivo.

2.4 Bugzilla

E' un sistema di bug tracking per il monitoraggio dei problemi che affliggono i software. E' in grado di tenere traccia dei bug riscontrati. I più popolari sistemi di questo tipo offrono il servizio a pagamento; Bugzilla, invece, è un progetto open source. Grazie a questa politica è diventato uno dei sistemi più apprezzati dalle organizzazioni di tutto il mondo. Bugzilla è un bugtracker (programma per tenere traccia di errori di programmazione) general purpose (cioè non dedicato ad un solo possibile utilizzo), inizialmente sviluppato e usato dal team che ha prodotto Mozilla. Rilasciato come software open source da Netscape Communications nel 1998, Bugzilla è stato

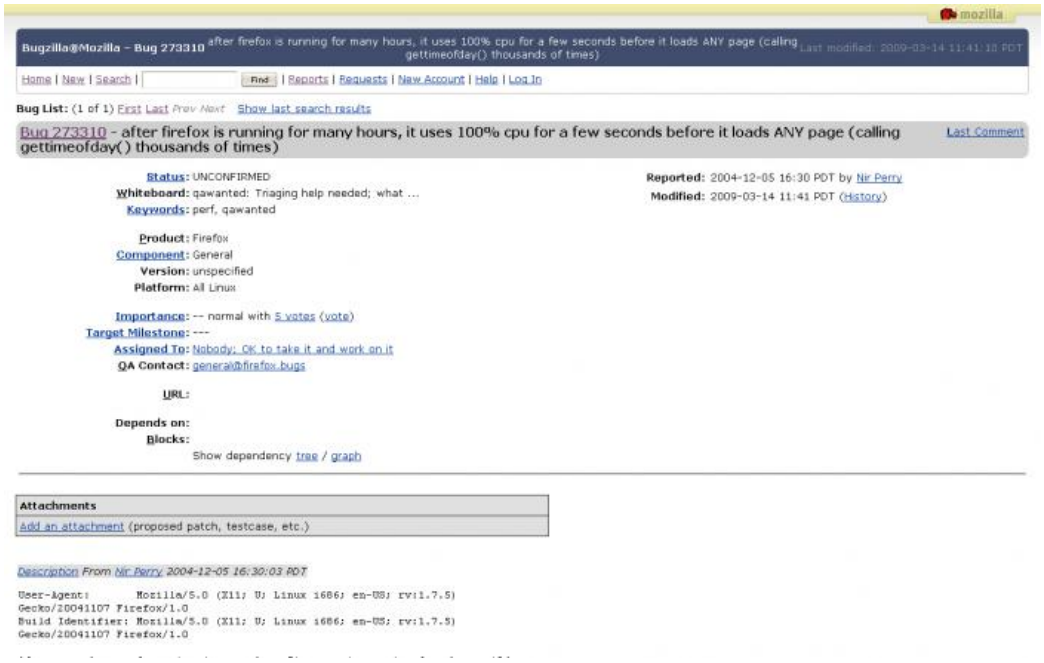

Figura 2.4: Interfaccia utente Bugzilla

adottato da varie organizzazioni per essere utilizzato come strumento per rintracciare errori (bug) nei progetti sia open source sia proprietari. Bugzilla è licenziato sotto Mozilla Public License. Bugzilla necessita di un web server, tipicamente Apache, e di un database management system, tipicamente MySQL o PostgreSQL. I ticket possono essere inseriti nel sistema da chiunque e vengono assegnati allo sviluppatore che è stato associato al bug stesso. Ai ticket possono essere associati vari status, assieme a commenti degli utenti e ad esempi dell'errore stesso. La nozione bugzilliana di bug software è molto generale: ad esempio, mozilla.org lo usa anche per tenere traccia delle richieste di modifica delle funzionalità. Il 31 gennaio 2012 è stata rilasciata l'ultima release stabile che ha portato Bugzilla alla versione 4.0.4, ma è stata anche rilasciata la versione 4.2 rc2.

2.4.1 Funzionalità

- Lato utente:

  - funzionalità di ricerca avanzata
  - notifiche e-mail
  - elenchi bug in più formati (Atom, iCal...)
  - rapporti di linea (giornaliero, settimanale, orari...) via e-mail
  - report e grafici
  - rilevamento automatico bug duplicati
  - modifica bug via e-mail
  - tempo di monitoraggio
  - richiesta di sistema
  - allegati privati e commenti
  - patch Viewer
  - "guarda" altri utenti
  - salva e condividi ricerche

- Lato amministratore:
  - meccanismo di estensione per le installazioni altamente personalizzabile
  - campi personalizzati
  - flusso di lavoro personalizzato
  - pieno supporto Unicode
  - localizzazione
  - supporto Perl
  - webservices (XML-RPC)
  - controllo della visibilità bug
  - metodi di autenticazione multipli
  - supporto per i motori di database multipli

2.5 Trac

Trac è fondamentalmente un wiki per il tracciamento dei problemi nei progetti di sviluppo dei software. Utilizza un approccio minimalista, con una semplice pagina web. Il motto è "Aiutare gli sviluppatori a scrivere software senza perdere la retta via". Trac è un sistema leggero di gestione ticket rilasciato sotto licenza open source ed è implementato sotto forma di applicazione web-based. E' scritto in linguaggio Phyton ed è molto facile da utilizzare.

2.5.1 Funzionalità

- Open source project management tool – Il sistema è stato sviluppato per potersi adattare a diversi ambiti applicativi, ma, dato che il progetto è open source, se non risponde alle esigenze dell'utente finale è possibile modificarlo e scrivere nuovi plugin.

- Ticket system – Si possono monitorare i progressi individuali di ogni risoluzione di bug, problemi o richieste di aggiunta di nuove funzionalità. Ogni ticket è identificato da un suo id univoco. Si possono facilmente unire i ticket che riguardano lo stesso argomento ed è possibile fare ricerche impostando filtri in base alla gravità del problema, al progetto, alla versione o a colui che ha aperto il ticket.

- View progress – Fornisce una serie di metodi per seguire l'andamento di una segnalazione. E' possibile impostare milestone e visualizzare un grafico con la tabella di marcia del progresso; ma anche consultare la cronologia dei cambiamenti scegliendone l'ordine di visualizzazione e ricevere mail sullo stato di avanzamento della risoluzione del problema o iscrivendosi al feed rss.

- Online repository viewing – Trac fornisce un moderno sistema di subversion open-source e offre una visualizzazione chiara del codice, in modo da poter facilmente vedere come i file differiscono tra le varie

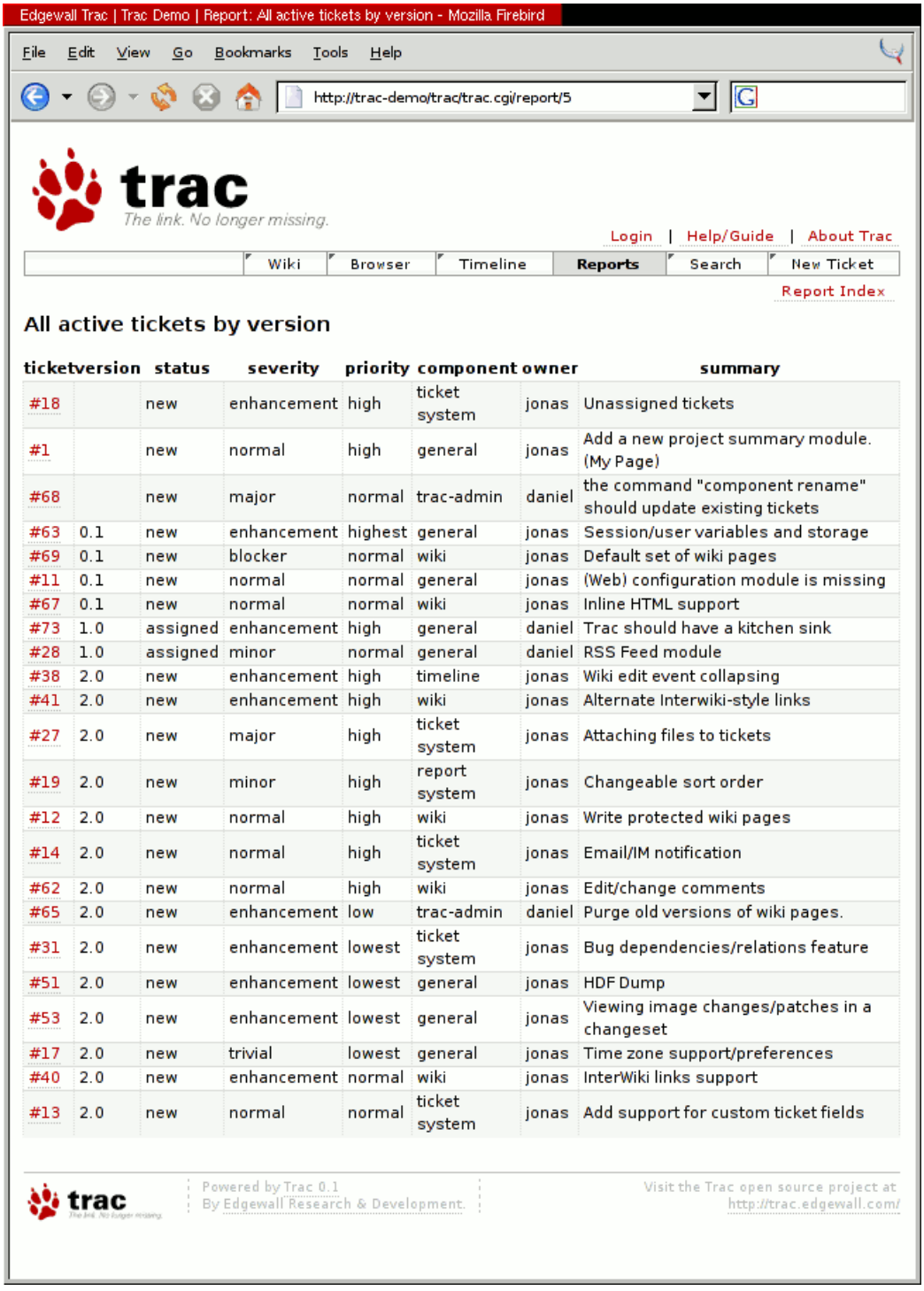

Figura 2.5: Interfaccia utente Bugzilla

versioni rilasciate. Utilizzando i plugin si possono installare anche altri software di subversion.

- User management – Trac ha un sistema semplice di accesso per controllare quali utenti possono o non possono accedere. Può essere migliorato, ad esempio, utilizzando il plugin "AccountManagerPlugin" o altri.

- Wiki – Contiene un server integrato per la gestione della documentazione che può essere consultato dagli sviluppatori o dagli utenti. Essendo un wiki è consentita la modifica dei contenuti da parte di tutti gli utenti. Usa la sintassi MoinMoin e magic link per i ticket, i sorgenti e i report.

- Feature disponibili attraverso i plugin – E' disponibile una serie di plugin che fornisce funzionalità aggiuntive di sostegno, dai filtri antispam ai diagrammi di Gantt e al monitoraggio del tempo.

2.6 Tabelle comparative

2.6.1 Funzionalità di administrator / staff

SOFTWARE	DISTINZIONE ADMIN E STAFF	STATISTICHE	RISPOSTE AUTOMATICHE	AUTO ASSEGNAMENTO TICKET	TEMPO LAVORATO SU UN TICKET	LASCIARE UN TICKET ASSEGNATO	SEGUIRE UN TICKET NON ASSEGNATO	AUTOCHIUSURA TICKET A TEMPO
openTT	X	V	V	X	X	X	X	X
sitePanel	X	V	V	V	X	X	V	X
kayako	V	V	V	X	V	V	V	V
bugzilla	n.p.	X	V	X	X	X	V	X
trac	n.p.	X	V	X	X	X	X	X

SOFTWARE	AGGIUNGERE NOTE AI TICKET	STORICO COMUNICAZIONI CON IL CLIENTE OUT PORTALE	ASSEGNARSI LA GESTIONE DI UN TICKET	EDIT TICKET	RICERCA TICKET	ULTIMA MODIFICA IN SENSO TEMPORALE	TEMPO RIMANENTE ALLA CHIUSURA	STIMA SUL TEMPO DI RISOLUZIONE
openTT	X	X	V	V	V	X	X	X
sitePanel	V	X	V	V	V	V	V	V
kayako	V	V	V	V	V	V	X	X
bugzilla	X	X	V	V	V	X	X	X
trac	X	X	V	V	V	X	X	X

Figura 2.6: Comparativa funzionalità administrator / staff

2.6.2 Funzionalità di utente

SOFTWARE	APERTURA TICKET VIA MAIL	APERTURA TICKET VIA INTERFACCIA WEB	APERTURA TICKET VIA TELEFONO (TRAMITE OPERATORE)	APERTURA TICKET VIA SCRIPT (LOCALE /REMOTO)	LISTA DI TUTTI I TICKET	LISTA DEI PROPRI TICKET	INDICE SODDISFAZIONE CLIENTE SULLA RISOLUZIONE	CAPTCHA ANTIBOT	SEGNALAZIONI DA UTENTI NON REGISTRATI
openTT	V	V	V	V	V	V	X	X	X
sitePanel	V	V	V	X	X	V	X	V	V
kayako	V	V	V	X	X	V	V	V	V
bugzilla	X	V	X	X	V	V	X	X	X
trac	X	V	X	X	V	V	X	X	X

SOFTWARE	SERVIZIO CHAT INTEGRATO	POSSIBILITÀ DI ALLEGARE FILE AL TICKET	POSSIBILITÀ DI SCEGLIERE L'OPERATORE CHE RICEVERÀ LA SEGNALAZIONE	AGGIORNAMENTI VIA MAIL SULLE SEGNALAZIONI APERTE
openTT	X	V	X	X
sitePanel	V	X	X	X
kayako	V	V	X	V
bugzilla	X	V	V	X
trac	X	V	V	X

Figura 2.7: Comparativa funzionalità utente

Legenda: X - Funzionalità non presente; V - Funzionalità presente

Capitolo 3

Tecnologie utilizzate

3.1 Liferay

Liferay Portal è un portale gratuito ed open source scritto in Java. E' distribuito sotto la licenza GNU[1] ed è utilizzato dalle aziende nelle intranet e nelle extranet. Il portale rispetta lo standard portlet JSR-286 [2](Figura 3.1).

Il portale Liferay permette agli utenti di avere le più comuni feature di un portale web. Fondamentalmente è formato da unità funzionali chiamate portlet. Liferay è spesso descritto come un content managment framework o come una web application framework. Il supporto dei plugin di Liferay è esteso a più linguaggi di programmazione, incluso anche il supporto ai plugin PHP e Ruby. Nonostante Liferay offra una sofisticata interfaccia per gli sviluppatori, non sono richieste conoscenze di programmazione per la

[1]La GNU Lesser General Public Licence è una licenza di software libero creata dalla Free Software Foundation e studiata quale compromesso tra la GNU General Public License e altre licenze non-copyleft. Fu scritta nel 1991 (aggiornata nel 1999 e nel 2007) da Richard Stallman, con l'ausilio legale di Eben Moglen.

[2]JSR 286 è la Java Portlet Specification v2.0 sviluppata sotto licenza JCP e creata in allineamento con la versione 2.0 di WSRP. E' stata sviluppata per migliorare la versione 1.0 (JSR-168). Tra le maggiori feature incluse vi sono: consentire intra portlet comunication attraverso eventi e parametri pubblici di rendering; servire in modo dinamico le risorse generate direttamente attraverso le portlet; servire data AJAX o JSON attraverso le portlet; introdurre listeners e filters per le portlet.

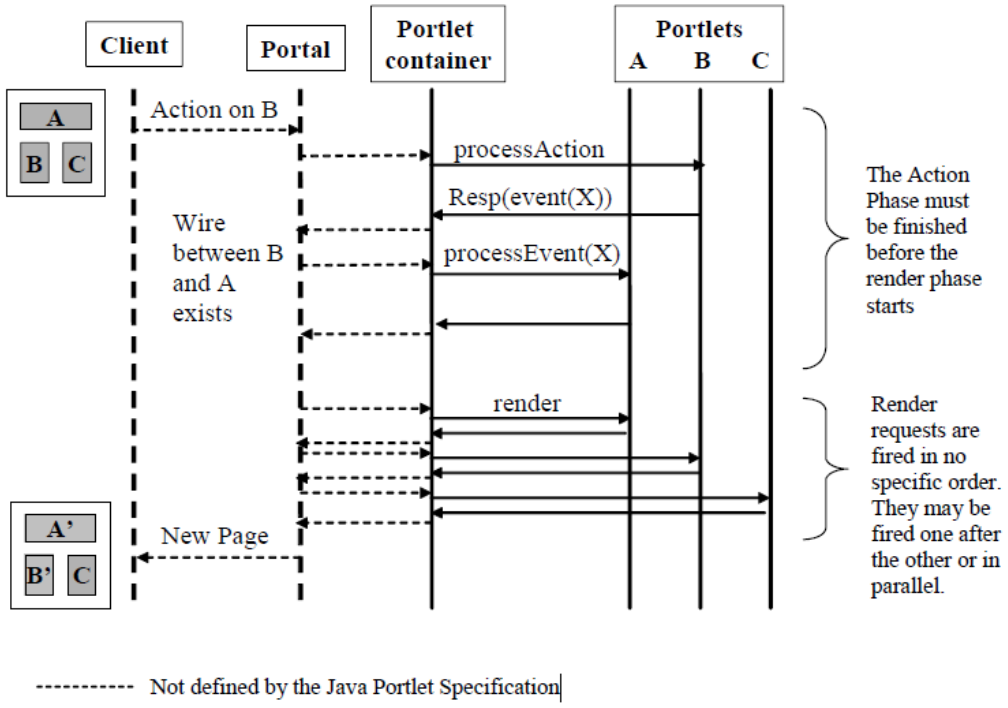

Figura 3.1: Schema comunicazione tra portlet Jsr286

costruzione, installazione e amministrazione di un portale base. Una portlet per il portale Liferay è scritta in Java ed è eseguita su ogni piattaforma che possa a sua volta eseguire la Java Virtual Machine e una application server. E' ottenibile in bundle con un web container come Apache Tomcat.

3.1.1 La storia

La Liferay Inc. è una software house che crea software a pagamento ampliando e modificando software open source già esistenti. Il core business aziendale è rappresentato dall'enterprise portal technology. Il quartiere generale della società è a Los Angeles, in California (USA). Liferay è fondata nel 2000 da Brian Chan per dare vita ad un portale enterprise per le organizzazioni no profit. Nel 2004 la compagnia sceglie di fondersi con la compagnia tedesca Liferay GmbH sotto l'unico nome Liferay Inc. Nel 2007

apre nuove sedi in Cina, in Spagna e a Dalian, in Asia. Nel 2009 inaugura un nuovo ufficio a Bangalore, in India. Il prodotto portale enterprise della Liferay riceve molti apprezzamenti da importanti aziende internazionali. Ottiene riconoscimenti da EContent magazine[3] in EContent100, la lista delle industrie leader del settore nel 2007, e InfoWorld la elegge tecnologia dell'anno. Nel luglio 2007, Liferay Inc. annuncia una nuova partnership con ICEsoft Technologies[4], distributore della libreria ICEfaces usata per lo sviluppo su tecnologia Ajax del software del portale enterprise. Nel gennaio 2008 la compagnia assume un ingegnere capo per JQuery UI, perché lavori full-time sulla libreria JavaScript. Sun Microsystem e Liferay siglano un accordo di scambio tecnologico nel maggio 2008. La versione di Liferay Community Edition attualmente disponibile è la 6.1, mentre la versione di Liferay Enterprise è la 6.0.

3.1.2 I prodotti

Liferay Portal viene distribuito in due differenti edizioni:

- Liferay Portal Community Edition: è la versione che contiene le ultime future e ha il supporto attivo della community.

- Liferay Portal Enterprise Edition: si tratta di una versione commerciale, quindi a pagamento, ed usufruisce del pieno supporto da parte dell'azienda. A differenza della versione Community, prima del rilascio deve sottostare a – e superare – ripetuti cicli di controllo della qualità. Di solito è rilasciata due o tre mesi dopo la versione Community.

[3]EContent è una autorità nel businnes delle pubblicazioni digitali. Usando le ultime tecnologie e strategie nell'ecosistema dei contenuti digitali, l'EContent Magazine e il sito web EContentmag.com aiutano i professionisti a massimizzare i loro investimenti costruendo al tempo stesso un modello di business redditizio

[4]ICEsoft Technologies Inc. è leader globale delle tecnologie Java enterprise open source. Fondata nel 2001, la ICESoft è una società con sede a Calgary (Canada) in prima linea nello sviluppo del web 2.0. Conta oltre trecento clienti in più di trenta Paesi.

3.1.3 Core portlet

Liferay è distribuito con alcune portlet preinstallate, che formano il cuore del portale, tra le quali:

- Web Content
- Asset Publishing
- Tags and Categories
- Document and Image management
- Document Library Manager, Recent Documents
- Alfresco, Documentum, and other document library integration
- Image Gallery
- WebDAV Integration
- Website Tools
- Web Form Builder
- Breadcrumbs
- Site Navigation
- Nested Portlets
- Site Map
- Page Ratings and Flags
- User Directory
- LDAP Integration
- Software Catalog

- Blogs and blog aggregation
- Calendar
- Chat
- Mail
- Message Boards
- Polls
- Wiki (supports Creole as well as MediaWiki syntax)
- Alerts and Announcements
- Knowledge Base
- Social Equity
- Themes, supporting Velocity and FreeMarker markup

3.2 PostgreSQL

PostgreSQL è un completo database relazionale ad oggetti rilasciato con licenza libera (stile Licenza BSD). Spesso è abbreviato come "Postgres", sebbene questo sia il nome vecchio dello stesso progetto. PostgreSQL è una reale alternativa rispetto sia ad altri prodotti open source – come MySQL, Firebird SQL e MaxDB – sia a quelli proprietari – come Oracle, Informix o DB2 – ed offre caratteristiche uniche nel suo genere che lo pongono, per alcuni aspetti, all'avanguardia nel settore dei database.

3.2.1 La storia

Inizialmente il DBMS (Database Management System) si chiamava Ingres ed era un progetto dell'Università di Berkeley. Nel 1982 il capo progetto, Michael Stonebraker, lascia la Berkeley per commercializzare il prodotto,

ma in seguito torna nel contesto accademico. Nel 1985 lascia nuovamente l'università per dare vita a un progetto post-Ingres (Postgres) che superi gli evidenti limiti dei prodotti concorrenti dell'epoca. Le basi dei sorgenti di Ingres e di Postgres rimangono ben distinte nel tempo. Il nuovo progetto punta a fornire un supporto completo ai tipi di dati, in particolare la possibilità di definirne di nuovi, introducendo UDF (User Defined Function) e UDT (User Defined Types). Vi è anche la possibilità di descrivere la relazione tra le entità (tabelle), che in precedenza era lasciata completamente all'utente. Perciò non solo Postgres preserva l'integrità dei dati, ma è in grado di leggere informazioni da tabelle relazionate in modo naturale, seguendo le regole definite dall'utente. Dal 1986 gli sviluppatori diffondono un gran numero di articoli che descrivono il nuovo sistema e nel 1988 è rilasciato un primo prototipo funzionante. La versione 1 è rilasciata a giugno 1989 per un numero di utenti contenuto. Un anno più tardi segue la versione 2, in cui il sistema è completamente riscritto. Nella versione 3, del 1991, questo sistema è riscritto per l'ennesima volta ed è aggiunto anche il supporto a gestori multipli di immagazzinamento dei dati e un motore di query migliorato. Nel 1993 vi è già un numero di utenti notevole che inonda il team di sviluppo con richieste di supporto e di nuove caratteristiche. Dopo aver rilasciato la versione 4, che è principalmente una rivisitazione del codice, il progetto termina. Sebbene il progetto Postgres sia ufficialmente abbandonato, la licenza BSD dà modo agli sviluppatori open source di ottenere una copia del software per poi migliorarlo a loro discrezione. Nel 1994 due studenti del Berkeley, Andrew Yu e Jolly Chen, aggiungono a Postgres un interprete SQL per rimpiazzare il vecchio QUEL che risaliva ai tempi di Ingres. Il nuovo software è rilasciato sul web col nome di Postgres95. Nel 1996 cambia nome di nuovo: per evidenziare il supporto al linguaggio SQL, è chiamato PostgreSQL. Il primo rilascio di PostgreSQL è la versione 6. Da allora, ad occuparsi del progetto è una comunità di sviluppatori volontari provenienti da tutto il mondo che si coordina attraverso Internet. Alla versione 6 ne seguono altre, ognuna delle quali porta nuovi miglioramenti. Nel gennaio 2005 è rilasciata la 8, la prima

nativa per Windows. Sebbene la licenza permetta la commercializzazione del software, il codice di Postgres non è sviluppato commercialmente con la stessa rapidità di Ingres. Ad un certo punto, Paula Hawthorn, membro originale del team di Ingres, e Michael Stonebraker fondano una società – la Illustra Information Technologies – finalizzata alla commercializzazione del software.

3.2.2 Descrizione

Un rapido esame di PostgreSQL potrebbe far pensare che sia simile ad altri database. PostgreSQL usa il linguaggio SQL per eseguire le query sui dati; questi sono conservati come una serie di tabelle con chiavi esterne che servono a collegare i dati correlati. La programmabilità di PostgreSQL è il suo principale punto di forza ed il principale vantaggio rispetto alla concorrenza: PostgreSQL rende più semplice costruire applicazioni per il mondo reale, utilizzando i dati prelevati dal database. I database SQL conservano dati semplici in "flat table", richiedendo che sia l'utente a prelevare e raggruppare le informazioni correlate utilizzando le query. La conversione delle informazioni dal mondo SQL a quello della programmazione orientata agli oggetti presenta difficoltà dovute principalmente al fatto che i due mondi utilizzano differenti modelli di organizzazione dei dati. L'industria chiama tale problema "impedance mismatch" (discrepanza di impedenza): mappare i dati da un modello all'altro può assorbire fino al 40 per cento del tempo di sviluppo di un progetto. Un certo numero di soluzioni di mappatura, normalmente dette "object-relational mapping", possono risolvere il problema, ma tendono ad essere costose e a presentare problemi, causando scarse prestazioni o forzando tutti gli accessi ai dati ad aver luogo attraverso il solo linguaggio che supporta la mappatura stessa. PostgreSQL può risolvere molti di questi problemi direttamente nel database, permettendo agli utenti di definire nuovi tipi basati sui normali tipi di dato SQL e al database stesso di comprendere dati complessi. Per esempio, si può definire un indirizzo come un insieme di diverse stringhe di testo per rappresentare il numero civico, la città... Si possono poi creare facilmente tabelle che contengono tutti i campi

necessari a memorizzare un indirizzo con una sola linea di codice. PostgreSQL, inoltre, permette l'ereditarietà dei tipi, uno dei principali concetti della programmazione orientata agli oggetti. Ad esempio, si può definire un tipo codice_postale, quindi creare un tipo cap (codice di avviamento postale) o un tipo us_zip_code basato su di esso. Gli indirizzi nel database potrebbero, quindi, accettare entrambi i tipi mentre regole specifiche potrebbero validare i dati in entrambi i casi. Nelle prime versioni di PostgreSQL, implementare nuovi tipi richiede la scrittura di estensioni in C e la loro compilazione nel server di database. Dalla versione 7.4 diventa molto più semplice creare ed usare tipi personalizzati attraverso il comando "create domain". La programmazione del database può ottenere grandi vantaggi dall'uso delle funzioni. La maggior parte dei sistemi SQL permette agli utenti di scrivere una procedura, un blocco di codice SQL che le altre istruzioni SQL possono richiamare. SQL è inadatto come linguaggio di programmazione; pertanto gli utenti possono incontrare grandi difficoltà nel costruire logiche complesse. Non supporta neppure molti dei principali operatori di base dei linguaggi di programmazione, come le strutture di controllo di ciclo e condizionale. Pertanto, i programmatori possono scrivere le proprie estensioni al linguaggio SQL per aggiungere nuove caratteristiche. In PostgresSQL i programmatori implementano la logica in uno dei molti linguaggi supportati:

- un linguaggio nativo chiamato PL/pgSQL, simile al linguaggio procedurale di Oracle PL/SQL, che offre particolari vantaggi nelle procedure che fanno uso intensivo di query

- wrapper per i più diffusi linguaggi di scripting – come Perl, Python, Tcl, e Ruby – che permettono di utilizzare la loro potenza nella manipolazione delle stringhe e nel link ad estese librerie di funzioni esterne

- C e C++ utilizzate per le procedure che richiedono prestazioni maggiori e logiche di programmazione complesse

Punti di forza della programmabilità di PostgreSQL:

- incremento delle prestazioni, in quanto la logica è applicata direttamente dal server di database in un'unica volta, riducendo il passaggio di informazioni tra il client ed il server
- incremento dell'affidabilità, dovuto alla centralizzazione del codice di controllo sul server, non dovendo gestire la sincronizzazione della logica tra molteplici client e i dati memorizzati sul server
- il codice del client può essere snellito e semplificato inserendo livelli di astrazione dei dati direttamente nel server.

Questi vantaggi fanno di PostgreSQL il più avanzato sistema database dal punto di vista della programmabilità; il che aiuta a spiegarne il successo. Utilizzare PostgreSQL riduce il tempo totale di programmazione, con vantaggi che crescono in base alla complessità di ogni singolo progetto.

3.3 Eclipse

Eclipse è un ambiente di sviluppo integrato multi-linguaggio e multipiattaforma. Ideato da un consorzio di grandi società – quali Ericsson, HP, IBM, Intel, MontaVista Software, QNX, SAP e Serena Software – è chiamato Eclipse Foundation sullo stile dell'open source. Eclipse può essere utilizzato per la produzione di software di vario genere: si passa da un completo IDE per il linguaggio Java (JDT, "Java Development Tools") a un ambiente di sviluppo per il linguaggio C++ (CDT, "C/C++ Development Tools") e a plug-in che permettono di gestire XML, Javascript, PHP e persino di progettare graficamente una GUI per un'applicazione Java (Eclipse VE, "Visual Editor"), rendendo di fatto Eclipse un ambiente RAD (Rapid Application Development). Il programma è scritto in linguaggio Java, ma anziché basare la propria GUI su Swing, il toolkit grafico di Sun Microsystems, si appoggia a SWT, librerie di nuova concezione che conferiscono ad Eclipse un'elevata reattività. La piattaforma di sviluppo è incentrata sull'uso di plug-in, componenti software ideati per uno specifico scopo, per esempio la generazione

di diagrammi UML. In effetti tutta la piattaforma è un insieme di plug-in, versione base compresa, e chiunque può svilupparli e modificarli. Essendo scritto in Java, Eclipse è disponibile per le piattaforme Linux, HP-UX, AIX, Mac OS X e Windows.

3.4 Ant

Apache Ant (Another Neat Tool) è un tool implementato in linguaggio Java, questo tool viene usato per l'automazione dei processi di build dei software Java e può essere visto come il Make per il linguaggio C. La differenza maggiore tra i tool Make e Ant è che quest'ultimo usa la struttura XML per descrivere il processo build e le sue dipendenze, mentre Make usa il formato makefile. Ant è un progetto sviluppato da Apache, open source e rilasciato sotto licenza ASL (Apache Software Licence).

3.4.1 La storia

Ant è ideato da James Duncan Davidson mentre sta trasformando un prodotto Sun in un progetto open source. Questo progetto, il motore delle JSP/Servlet, diventa poi Apache Tomcat. Ant è creato come un semplice tool indipendente dalla piattaforma per compilare Tomcat tramite le direttive scritte nel file buld.xml. Ant (versione 1.1) è rilasciata ufficialmente il 19 luglio 2000. Vengono fatte varie proposte per una versione 2, come AntEater di James Duncan Davidson, Myrmidon di Peter Donald e Mutant di Conor MacNeill, però nessuna di queste si dimostrra capace di ottenere il consenso della community degli sviluppatori. Oggi, Ant è il buid tool più usato dagli sviluppatori di progetti Java.

Capitolo 4

Gli obiettivi

La gestione informatizzata dei servizi di segnalazione dei problemi è un tema noto in ambiti quali i call center e le grandi aziende. Il nostro software è articolato sullo sviluppo di una portlet per la gestione dei ticket installabile sulla piattaforma Liferay, portlet che attualmente non è disponibile nel marketplace del portale. Non avrebbe avuto senso sviluppare l'ennesima "normale" web application, dato che il mercato ne offre già una vasta gamma, molte delle quali sono open source e quindi adattabili alle singole esigenze. Sviluppare il nostro sistema all'interno di una portlet ci dà quindi la possibilità di imparare ad usare in modo approfondito le metodologie di sviluppo messe a disposizione da Liferay. Oggi, il portale Liferay viene usato tantissimo in molte intranet aziendali. La portlet che andiamo a sviluppare darà la possibilità alle aziende che ne vorranno fare uso di avere un ottimo sistema di gestione dei problemi di prodotto o di servizio. Ciò potrà essere utilizzato su due diversi fronti:

- per segnalare i problemi al portale: l'azienda invia i problemi riscontrati direttamente a chi si occupa dello sviluppo del portale

- ricezione problemi segnalati dal cliente: l'azienda riceve le segnalazioni dei problemi riscontrati dal cliente sui prodotti o servizi forniti

Tutto questo integrato all'interno del portale, aggiungendo solamente la portlet e configurandola come meglio si desidera. Gli utenti che possono accedere al sistema sono riconducibili a tre tipologie:

- user
- staff
- administrator

Ogni tipo di utente ha accesso a funzionalità specifiche per il proprio ruolo.

4.1 Lo user

Lo user è un cliente che necessita di assistenza in quanto ha riscontrato un problema in un prodotto o in un servizio acquistati. Ne porta perciò a conoscenza l'azienda fornitrice attraverso il sistema di ticketing. La portlet dal lato dell'utente consente di compiere le seguenti operazioni:

- registrazione – permette la registrazione di un nuovo utente di tipo user, che in questo modo può accedere al portale
- login – lo user effettua il login nel portale attraverso lo username e la password scelti al momento della registrazione. In tal modo accede a tutte le funzioni del portale
- logout – disconnessione dal portale
- modifica dati personali – attraverso la specifica form, lo user edita i propri dati (nome, cognome, indirizzo e-mail...)
- apertura di un nuovo ticket attraverso l'interfaccia web, così composta
  - priorità: lo user assegna quella che ritiene giusta per il proprio ticket, anche se potrà essere comunque modificata dallo staff
  - ambito del problema: scelta tra le varie tipologie del problema

  - titolo: identificazione del problema con una definizione
  - descrizione: dettagliata spiegazione del problema
  - file allegato (opzionale): per una migliore e dettagliata descrizione del problema è possibile caricare file che siano di aiuto nella fase di diagnosi. Il sistema accetta file di immagini, di archivi o testuali. La dimensione massima scelta è 5MB
  - keywords: parole che aiutano la ricerca del ticket da parte di chi fa ricerche
  - captcha: per evitare problemi di DoS (Denial of Service) causati da botnet

- visualizzazione dei ticket – tabella che permette allo user di visualizzare tutti i ticket che ha aperto nel sistema
- visualizzazione di tutti i ticket presenti nel sistema – tabella che permette di visualizzare tutti i ticket che sono presenti nel sistema tra i quali poter rintracciare la soluzione fornita ad un problema simile
- aggiunta di note ad un ticket già aperto – per avere comunicazione con i membri dello staff è possibile aggiungere alcune note al ticket aperto
- valutazione finale sulla risoluzione del problema a ticket chiuso – una volta che il ticket viene chiuso, l'utente può assegnare un livello di gradimento all'intervento svolto da parte dello staff.

4.2 Lo staff

Per staff si intendono coloro che, all'interno dell'azienda fornitrice del prodotto o del servizio, si occupano della risoluzione del problema posto dal cliente. Lo staff ha in carico le seguenti operazioni:

- visualizzare tutti i ticket – tabella che contiene tutti i ticket che sono presenti nel sistema

- visualizzare solo alcuni ticket – è possibile, in base all'organizzazione dello staff, prevedere una tabella per la visualizzazione strutturata dei ticket. In automatico il sistema visualizza solo i ticket che riguardano un determinato ambito (quello al quale è adibito ciascun operatore dello staff)

- visualizzare i ticket non assegnati né presi ancora in carico

- prendere la ownership di un ticket – permette ad ogni singolo operatore di prendersi carico della risoluzione del ticket

- lasciare la ownership del ticket – permette all'operatore di abbandonare la soluzione del problema lasciando che siano altri ad occuparsene

- modificare il ticket, con la possibilità di

  - aggiungere note: per avere una comunicazione tra user e operatore dello staff

  - cambiare priorità: ovvero se l'utente ha inserito una priorità sbagliata lo staff la può modificare

  - aggiornare le ore lavorative impiegate al fine di conoscere il tempo dedicato dallo staff alla risoluzione del problema

  - aggiungere keyword: parole che aiutano la ricerca del ticket da parte dello staff

  - assegnare uno stato al ticket: aperto (in lavorazione), chiuso (problema risolto), non assegnato (nessun owner)

- chiudere il ticket – nel momento in cui il problema viene risolto

- inserire un nuovo ticket – in casi di necessità, anche lo staff può inserire un nuovo ticket nello stesso modo in cui lo fa lo user (ad esempio, se il problema è segnalato per telefono o e-mail)

- seguire lo svolgimento di un ticket assegnato ad un altro collega – il ticket assegnato ad un altro operatore rimane comunque visibile tra i ticket ed è consultabile anche da un operatore non addetto alla lavorazione
- visualizzare le statistiche generali sui ticket – è prevista una statistica sui ticket aperti, su quelli chiusi risolti, sui chiusi irrisolti, sui tempi medi di lavorazione

4.3 L'administrator

Oltre ad avere tutti i poteri assegnati allo staff, ha compiti di mantenimento del sistema e amministrazione del portale. L'administrator, tra l'altro, può:

- personalizzare la form di inserimento del ticket, alla quale accede lo user per aprire un ticket (in base al prodotto utilizzato). E' possibile che in base al prodotto acquistato dal cliente ci sia bisogno di informazioni diverse. Il software garantisce all'administrator la possibilità di modificare l'interfaccia presente nella portlet per l'inserimento del ticket
- gestire gli utenti del sistema attraverso la creazione, la modificazione, l'aggiunta o la cancellazione
- avere accesso diretto al database, attraverso portlet dedicate all'interno del portale

Capitolo 5

Progettazione ed implementazione del database

Il primo passo per realizzare il nostro sistema di ticketing è la progettazione e implementazione del database. Durante la fase di analisi stiliamo un elenco di tutti i tipi di dati che servono al corretto funzionamento del programma e li traduciamo in tabelle, mentre durante quella implementativa, costruiamo le tabelle vere e proprie nel database.

5.1 Analisi dei requisiti del database

La corretta ed efficiente esecuzione del software passa attraverso un'accurata analisi iniziale, a cui segue la conseguente specifica implementazione del database. Effettuiamo l'analisi iniziale individuando le componenti fondamentali del nostro database – Utente e Ticket – per poi approfondirne proprietà e relazioni. Un qualunque utente registrato è caratterizzato, oltre che dai dati personali, dal ruolo utilizzato per distinguere le tre figure presenti nel sistema:

- normale utente, ovvero un cliente che ha la sola possibilità di effettuare segnalazioni

- operatore dello staff, adibito alla lavorazione e risoluzione dei problemi segnalati

- administrator, che, oltre ai poteri dello staff, può lavorare direttamente alla funzionalità del portale.

I ruoli di operatore dello staff e administrator sono assegnati ad un dipartimento di competenza. Qualunque categoria di utente può inserire all'interno del sistema un ticket che è assegnato ad un operatore dello staff per provvedere alla risoluzione del problema segnalato. In ogni caso, un differente operatore o lo stesso administrator possono seguire lo svolgimento della lavorazione anche se in carico ad un collega. Ogni utente di qualunque categoria può aggiungere ad un qualsiasi ticket a lui visibile alcune note che, fornendo ulteriori dettagli, danno un contributo alla risoluzione del problema. Può, altresì, allegare file sia alle note sia al ticket stesso: non sono salvati all'interno del database (per ovvi motivi di performance a fronte di record numerosi), bensì viene utilizzato un riferimento per poterli recuperare in un secondo momento. Ogni riferimento è creato in automatico dal sistema aggiungendo un timestamp al nome originale con cui è stato caricato.

5.2 Tabelle del database

Al termine della fase di analisi, definiamo lo schema Entità-Relazione (Figura 5.1) con le relative tabelle. Per comodità del lettore, presentiamo una sintetica descrizione dei contenuti delle tabelle.

- Tabella TICKET
  Contenuti: il sistema registra tutte le segnalazioni ricevute da un qualsiasi utente

  - ticketid: campo di tipo INTEGER auto incrementale. Associa ad ogni ticket un numero intero univoco utilizzato come chiave primaria della tabella

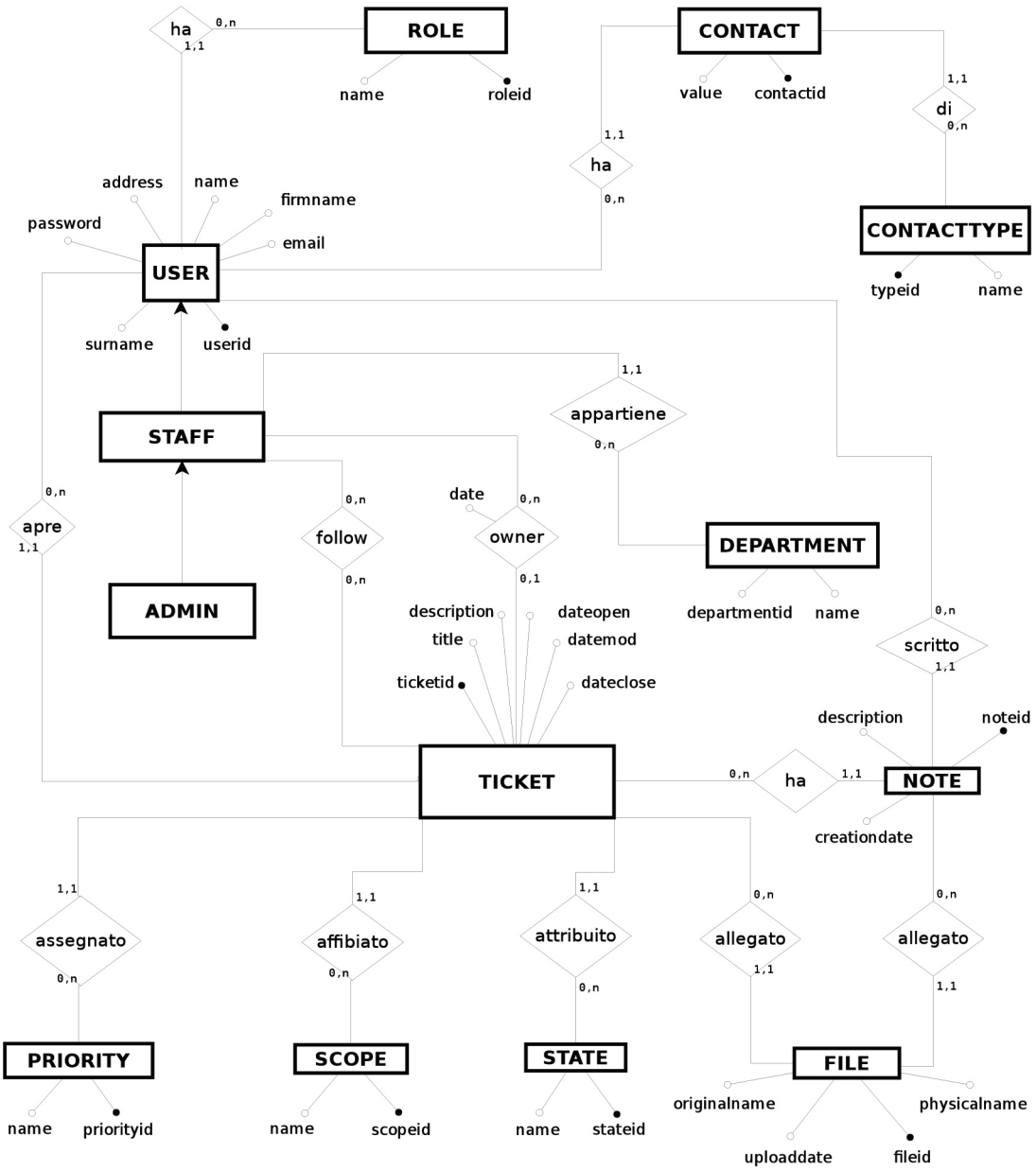

Figura 5.1: Schema E-R

  - title: campo di tipo TEXT. Breve descrizione riguardante l'oggetto del contenuto del ticket
  - description: campo di tipo TEXT. Descrizione dettagliata del problema riscontrato da parte dell'utente
  - dateopen: campo di tipo TIMESTAMP. Data (assegnata dal sistema) in cui è stato creato il record del ticket in questione sul database
  - datemod: campo di tipo TIMESTAMP. Data (aggiornata dal sistema) in cui è effettuata una qualunque modifica al ticket
  - dateclose: campo di tipo TIMESTAMP. Data (assegnata dal sistema) in cui il ticket è chiuso da un operatore

- Tabella USER
  Contenuti: la tabella User non è creata da noi, ma, dato che il portale Liferay la genera in modo automatico, ci limitiamo a segnalare gli unici campi da noi utilizzati per la registrazione di tutti gli utenti del sistema (utenti comuni, operatori dello staff e administrator)

  - userid: campo di tipo INTEGER auto incrementale. Associa ad ogni utente un numero intero univoco utilizzato come chiave primaria della tabella
  - name: campo di tipo TEXT. Nome reale dell'utente
  - surname: campo di tipo TEXT. Cognome dell'utente
  - firmname: campo di tipo TEXT. Nome (opzionale) della società di appartenenza
  - address: campo di tipo TEXT. Indirizzo dell'azienda o dell'utente
  - e-mail: campo di tipo TEXT. Indirizzo e-mail dove poter contattare l'utente. Campo utilizzato dal sistema come nominativo per effettuare il login
  - password: campo di tipo TEXT. Password dell'utente da usare in combinazione al campo e-mail per effettuare il login al sistema

- Tabella ROLE
  Contenuti: identifica le tipologie di utenti che possono operare nel sistema. Ogni user ha un riferimento per il ruolo che ricopre.
  - roleid: campo di tipo INTEGER auto incrementale. Associa ad ogni ruolo un numero intero univoco utilizzato come chiave primaria della tabella
  - name: campo di tipo TEXT. Nome del ruolo

- Tabella CONTACT
  Contenuti: riporta i valori (numeri di telefono, contatto Skype...) dei vari contatti dell'utente.
  - contactid: campo di tipo INTEGER auto incrementale. Associa ad ogni contatto un numero intero univoco utilizzato come chiave primaria della tabella
  - value: campo di tipo TEXT. Contiene i valori dei possibili contatti dell'utente.

- Tabella CONTACTTYPE
  Contenuti: offre la possibilità di aggiungere e rimuovere dal sistema tipologie di contatti assegnabili ad ogni utente.
  - typeid: campo di tipo INTEGER auto incrementale. Associa ad ogni tipo un numero intero univoco utilizzato come chiave primaria della tabella
  - name: campo di tipo TEXT. Contiene i nomi delle possibili tipologie di contatti assegnabili all'utente

- Tabella DEPARTMENT
  Contenuti: assegna il dipartimento di competenza agli utenti che ricoprono i ruoli staff e administrator.

  - departmentid: campo di tipo INTEGER auto incrementale. Associa ad ogni dipartimento un numero intero univoco utilizzato come chiave primaria della tabella
  - name: campo di tipo TEXT. Contiene i nomi delle possibili tipologie di dipartimenti assegnabili agli operatori presenti nell'azienda

- Tabella NOTE
  Contenuti: qualunque utente del sistema utilizza la seguente tabella per la gestione delle note aggiunte, in qualsiasi momento, ad un ticket.

  - noteid: campo di tipo INTEGER auto incrementale. Associa ad ogni nota un numero intero univoco utilizzato come chiave primaria della tabella
  - description: campo di tipo TEXT. Contiene il testo della nota
  - creationdate: campo di tipo TIMESTAMP. Data (autoassegnata) in cui viene scritta la nota

- Tabella FILE
  Contenuti: raccoglie i riferimenti ai file che possono essere allegati ad una nota o a un ticket.

  - fileid: campo di tipo INTEGER auto incrementale. Associa ad ogni file un numero intero univoco utilizzato come chiave primaria della tabella
  - originalname: campo di tipo TEXT. Nome del file allegato dall'utente
  - physicalname: campo di tipo TEXT. Nome (automaticamente modificato) con cui il file allegato è salvato nel sistema
  - uploaddate: campo di tipo TIMESTAMP. Data (autoassegnata) in cui è caricato il file

- Tabella PRIORITY
  Contenuti: contiene l'elenco delle priorità che possono essere assegnate ad un ticket.

  - priorityid: campo di tipo INTEGER auto incrementale. Associa ad ogni priorità un numero intero univoco utilizzato come chiave primaria della tabella
  - name: campo di tipo TEXT. Contiene i nomi delle possibili tipologie di priorità assegnabili al ticket

- Tabella SCOPE
  Contenuti: raccoglie le tipologie di problema su cui possono essere effettuate segnalazioni.

  - scopeid: campo di tipo INTEGER auto incrementale. Associa ad ogni tipologia di problema un numero intero univoco utilizzato come chiave primaria della tabella
  - name: campo di tipo TEXT. Contiene i nomi delle possibili tipologie di problema assegnabili al ticket

- Tabella STATE
  Contenuti: identifica i possibili stati in cui si può trovare un determinato ticket (aperto, chiuso, in lavorazione, irrisolto...).

  - stateid: campo di tipo INTEGER auto incrementale. Associa ad ogni stato un numero intero univoco utilizzato come chiave primaria della tabella
  - name: campo di tipo TEXT. Contiene i nomi delle possibili tipologie di stato assegnabili al ticket

5.3 Implementazione del database

Il portale Liferay offre in bundle l'RDBMS (Relational Database Management System) chiamato HSQLDB (Hypersonic SQL Database)[1]. HSQLDB offre diversi vantaggi, ma, per motivi di performance a fronte di possibili database di grandi dimensioni, preferiamo integrare il DBMS PostgreSQL.

5.3.1 Integrazione di PostgreSQL in Liferay

Il primo passo è predisporre il portale Liferay ad accettare il DBMS PostgreSQL. Per fare ciò dobbiamo creare il file "portal-ext.properties" (Codice 5.1); questo serve per sovrascrivere la property di default del portale e consentirgli di recuperare la connessione dal jndi name jdbc/LiferayPooL. Il file in questione si trova in ".../liferay-portal/tomcat-6.0.29/webapps/ROOT/WEB-INF/classes" e va aggiunta la seguente riga di codice:

```
1 jdbc.default.jndi.name=jdbc/LiferayPool
```

Codice 5.1: portal-ext.properties

Ora è necessario configurare Tomcat per restituire la connessione al database per mezzo del jndi name specificato precedentemente. Bisogna modificare il file ROOT.xml (Codice 5.2) che si trova nel percorso ".../liferay-portal/tomcat-6.0.29/conf/Catalina/localhost" e definire al suo interno i parametri necessari ad effettuare la connessione al database:

```
1 <Resource
2         name="jdbc/LiferayPool"
3         auth="Container"
4         type="javax.sql.DataSource"
```

[1] E' un progetto open source stand-alone nato nel 2001. Questo DBMS è molto leggero (circa 600 kb) e ad ogni avvio carica tutte le sue tabelle in ram. Questo fa sì che sia molto efficiente per database di piccole e medie dimensioni, ma inadeguato per database di grandi dimensioni. Può risultare anche molto utile in fase di debug. Essendo scritto totalmente in Java ci sono tanti vantaggi nell'integrarlo con le Java application anche grazie al fatto che si basa sui driver jdbc, facili da integrare ed utilizzare in qualunque progetto.

```
5         description="Datasource␣for␣Liferay␣webapp"
6         username="<username>"
7         password="<password>"
8         url="jdbc:postgresql://postgres83:5432/ticketing"
9         driverClassName="org.postgresql.Driver"
10        maxActive="20" maxIdle="5" maxWait="50"
11        removeAbandoned="true"/>
```

Codice 5.2: ROOT.xml

5.3.2 Creazione delle tabelle

Le tabelle del database sono create in automatico dal portale Liferay al momento del deploy. Il portale crea di default tutte le tabelle che servono al corretto funzionamento, mentre noi dobbiamo specificare le tabelle custom attraverso il file service.xml, da creare all'interno della cartella WEB-INF del Portlet. In questo file, tramite la sintassi espressa nella DTD, dichiariamo le tabelle e le colonne di cui sono composte. A titolo di esempio, descriviamo come abbiamo creato la tabella Ticket:

```
6  ...
7  <entity name="Ticket" local-service="true"
8          remote-service="false">
9          <!-- PK fields -->
10         <column name="ticketId" type="long"
11                 primary="true" />
12         <!-- Audit fields -->
13         <column name="companyId" type="long" />
14         <column name="userId" type="long" />
15         <column name="userName" type="String" />
16         <column name="createDate" type="Date" />
17         <column name="modifiedDate" type="Date" />
18
19         <!-- Other fields -->
```

```
20          <column name="title" type="String" />
21          <column name="description" type="String" />
22          <column name="dateopen" type="String" />
23          <column name="datemod" type="String" />
24          <column name="dateclose" type="String" />
25          <column name="priorityId" type="long" />
26          <column name="scopeId" type="long" />
27          <column name="stateId" type="long" />
28          <column
29              name="file"
30              type="Collection"
31              entity="File"
32              mapping-key="ticketId"/> <!--1 to n-->
33          <column
34              name="note"
35              type="Collection"
36              entity="Note"
37              mapping-key="ticketId"/> <!--1 to n-->
38          <reference package-path="com.liferay.portal.model"
39                  entity="User" />
40  </entity>
41  ...
```

Codice 5.3: Frammento service.xml

Analizziamo nel dettaglio i vari tag utilizzati per la creazione della tabella:

- entity: rappresenta la nuova tabella
  - name: nome della tabella nel database
  - local-service: genera il codice per accedere ai servizi localmente alla portlet
  - remote-service: istruisce il service builder a generare anche i file di interfaccia per l'invocazione remota. A noi non servono, quindi lo impostiamo a false

- column: crea una nuova colonna nella tabella attuale
  - name: nome della nuova colonna
  - type: tipo di dato accettato nella colonna
  - primary-key: indentifica la colonna come chiave primaria.

La column "file" (vale anche per quella "note") non è creata nella tabella Ticket; serve, invece, a specificare una relazione 1-n tra la tabella Ticket e quella File. In questo modo il service builder genera automaticamente i metodi per recuperare una lista dei "file" dato l'id di un ticket, poiché è la colonna che usiamo per la relazione nell'attributo mapping-key. Naturalmente è possibile esprimere anche relazioni n-n; in questo caso invece di mapping-key si usa l'attributo mapping-table. Una volta creato il file service.xml non ci rimane che invocare il target build-service del nostro Ant. Facendo un refresh del progetto dentro Eclipse, è possibile controllare il risultato dell'operazione appena fatta: dentro la cartella src ci sono nuovi package e nuove classi, mentre dentro la cartella lib, sotto WEB-INF, c'è il JAR che contiene altre classi e interfacce. A questo punto facciamo deploy e riavviamo il portale: sarà il portale stesso a creare le tabelle definite nel file service.xml nel database.

Capitolo 6

Implementazione del software

Analizzando i più diffusi software di gestione ticket presenti sul mercato, ci siamo potuti fare un'idea di quali fossero le principali caratteristiche da implementare. Dopo aver redatto le specifiche sulle quali ci saremmo basati, abbiamo iniziato lo sviluppo dell'applicativo.

6.1 Il nostro sistema

Il sistema che abbiamo sviluppato è scritto in Java ed è installabile nel portale Liferay. Rispetto ad una normale applicazione web-based, la nostra può essere installata solo all'interno del portale Liferay con la seguente procdura:

- copiare il file .war che contiene le portlet dentro la cartella "deploy" del portale
- caricare il portale
- aggiungere le portlet al portale (Figura 6.1)

In questo modo la nostra portlet può essere installata in qualunque portale Liferay esistente seguendo passo passo la procedura descritta in precedenza, che non richiede alcuna competenza specifica e può essere eseguita dal cliente

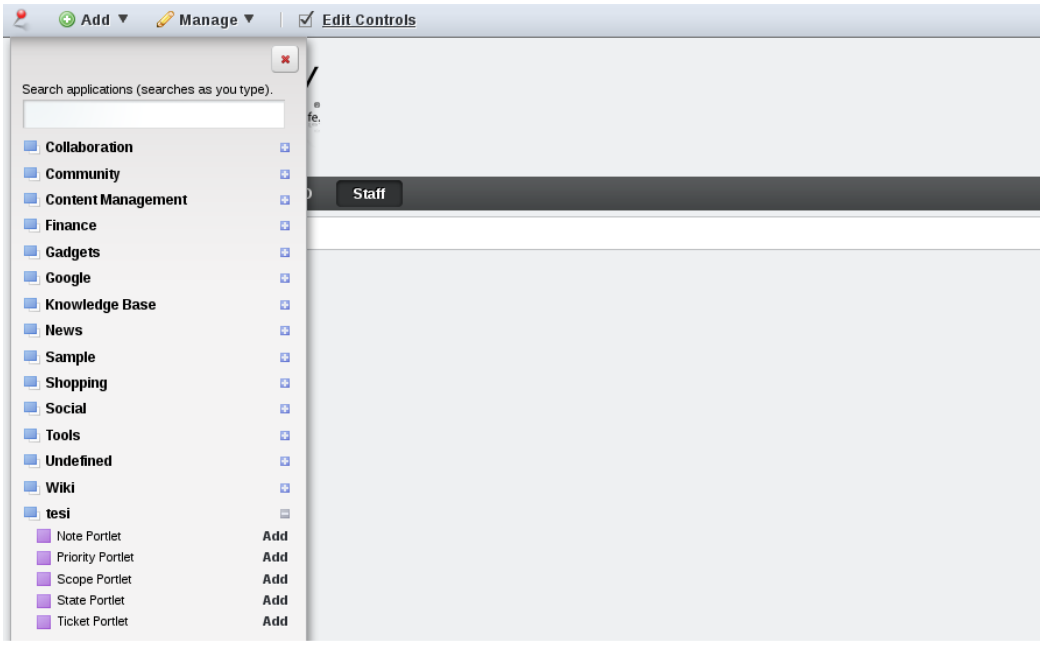

Figura 6.1: Aggiunta di portlet al Portale

stesso, dal momento che è Tomcat che si occupa di scompattare il file .war e caricarlo nel portale. Per quanto riguarda la parte grafica del software, abbiamo utilizzato i CSS messi a disposizione da Liferay. Sotto questo aspetto il portale è completamente personalizzabile, in quanto è possibile creare layout e CSS custom ed aggiungerli in modo molto semplice.

6.2 Generazione tabelle e portlet di CRUD

Come si è visto nel capitolo precedente, per la creazione delle tabelle del database occorre editare il file service.xml, per poi implementare tutte le funzioni CRUD[1]. Dato che si tratta di un'operazione ripetitiva (vale a dire è sempre la stessa, cambiano solo i dati da inserire) abbiamo utilizzato un tool chiamato XMLPortletFactory che, tramite un singolo file xml, si occupa della generazione del file service.xml e di tutti i sorgenti che servono alla

[1]Le funzioni di CRUD sono le quattro operazioni fondamentali per la gestione di un database: Create, Read, Update e Delete.

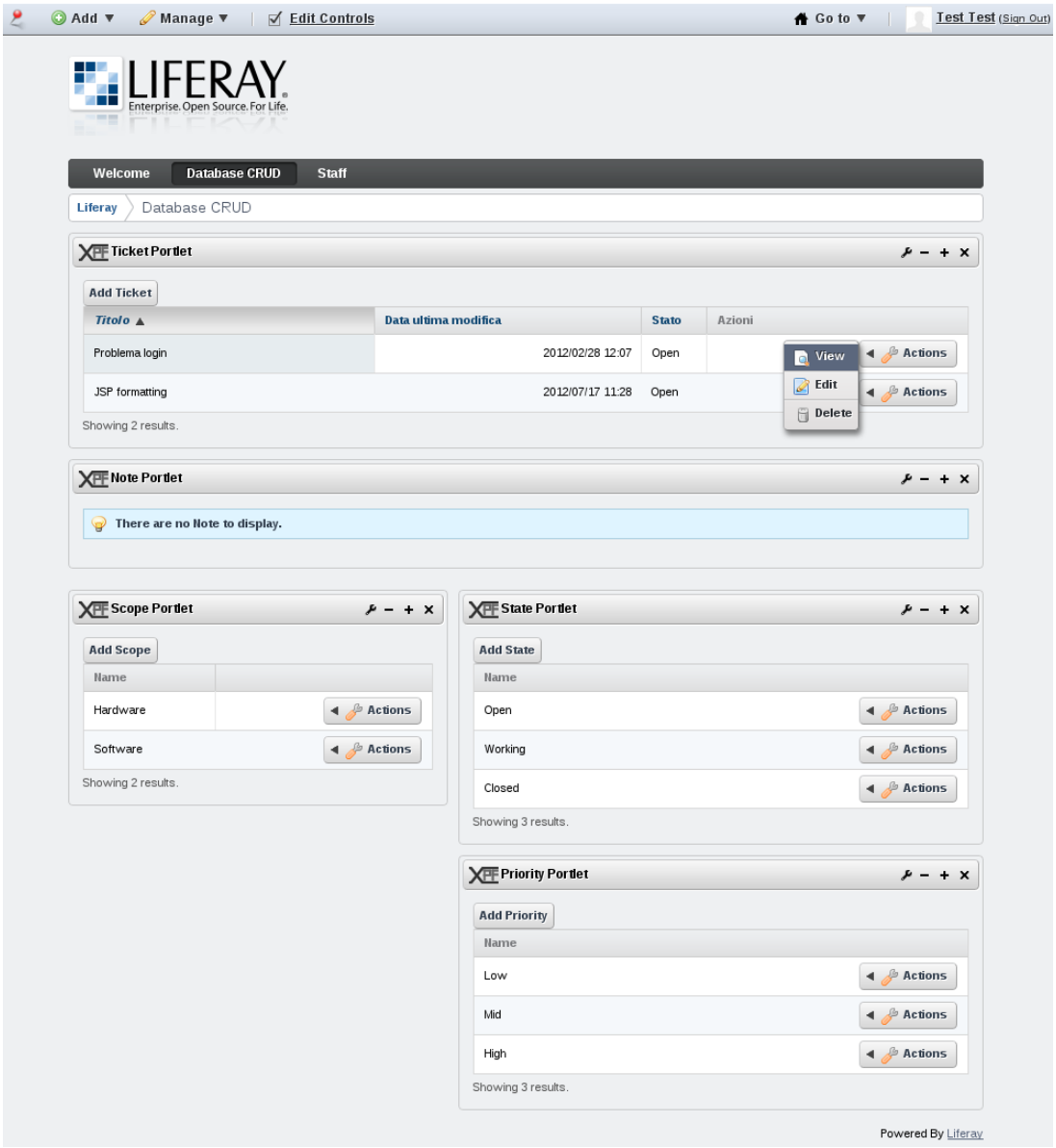

Figura 6.2: Portlet di CRUD

creazione delle portlet di CRUD. Questo tool funziona in modo molto semplice. Bisogna posizionarsi dentro la cartella del tool e lanciare da terminale uno dei seguenti comandi:

- sotto windows - create.bat tesi "tesi-portlet"
- sotto linux - ./create.sh tesi "tesi-portlet"

Il risultato dell'esecuzione di questo comando è

- la creazione di una cartella nominata tesi-portlet dentro la cartella "portlet" dell'SDK di Liferay, popolata con tutti i file necessari
- la generazione del file build.xml: questo file servirà per il successivo punto e non andrà toccato
- la generazione del file tesi.xml: questo sarà il file da editare inserendo tutte le tabelle del nostro database. Di seguito presentiamo un frammento (Codice 6.1) del file per la specifica della tabella State

```
230 ...
231 <application>
232    <classDef>
233       <name>State</name>
234       <title>State</title>
235       <restrictBy>
236          <userId>false</userId>
237       </restrictBy>
238    </classDef>
239    <fileDef>
240       <name>State</name>
241       <fields>
242          <field>
243             <name>stateId</name>
244             <title>stateId</title>
245             <type>
```

```
246                 <long>
247                    <length>2</length>
248                    <signed>false</signed>
249                    <nullable>false</nullable>
250                 </long>
251              </type>
252              <showFieldInView>false</showFieldInView>
253              <required>true</required>
254           </field>
255           <field>
256              <name>name</name>
257              <title>name</title>
258              <type>
259                 <varchar>
260                    <length>50</length>
261                 </varchar>
262              </type>
263              <showFieldInView>true</showFieldInView>
264              <required>true</required>
265           </field>
266        </fields>
267     </fileDef>
268 </application>
269 ...
```

Codice 6.1: Frammento tesi.xml

In seguito occorre lanciare il comando "Ant", che produce i seguenti risultati:

- esegue il build service per generare il Liferay service.xml e persistence
- esegue l'ant deploy per generare il file .war che contiene le portlet
- al momento del deploy saranno create le tabelle sul db.

6.3 Gestione utenti e permessi del portale

Per poter gestire le diverse funzionalità in base al tipo di utente al momento autenticato, abbiamo creato all'interno del portale tre diverse pagine dedicate a:

- clienti
- staff
- administrator

In ognuna delle suddette pagine, ogni categoria di utente potrà eseguire le operazioni di propria competenza (come specificato nel Capitolo 4).

Il passo successivo è la creazione dei ruoli: User, Staff e Administrator. Ad ogni utente abbiamo assegnato il proprio ruolo e di conseguenza, i permessi corretti. In questo modo ogni utente ha accesso esclusivamente alla pagina di propria competenza, al contrario dell'amministratore che ha accesso all'intero sistema.

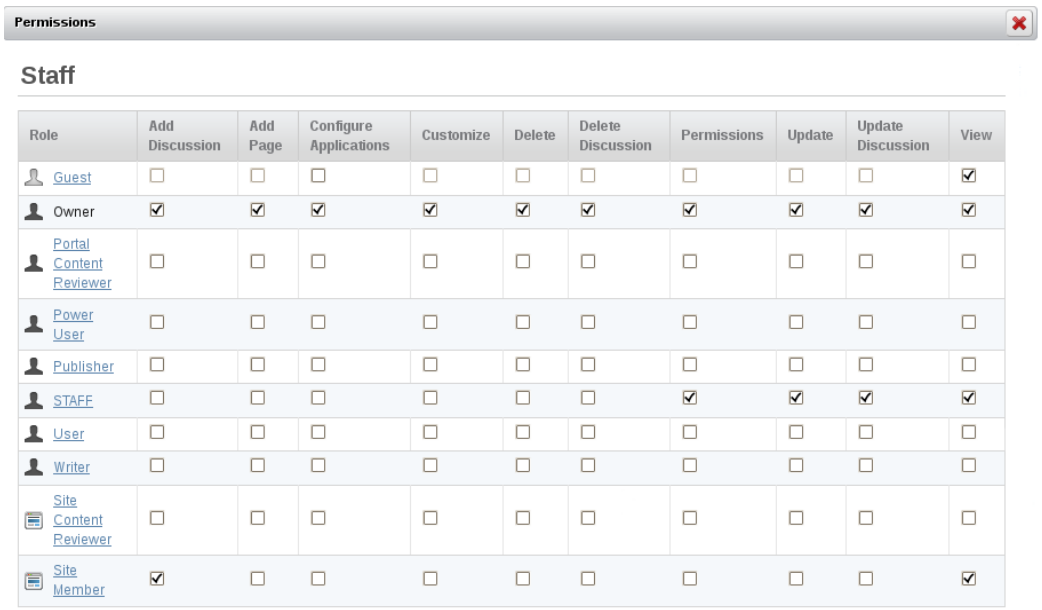

Figura 6.3: Settaggio permessi pagina Staff

Capitolo 7

Test case

Durante il tirocinio svolto presso l'azienda D'vel snc ci sono state mostrate alcune tipologie di lavoro tipicamente usate per lo sviluppo di una qualsiasi applicazione software. Una di queste, che vogliamo adottare anche per lo svolgimento della tesi, è la creazione di un test case, che viene realizzato tipicamente prima di iniziare a sviluppare l'applicazione e subito dopo la fase preliminare di analisi dei requisiti. Se, durante lo svolgimento del progetto avvengono cambiamenti significativi, il test case dovrà essere aggiornato. La realizzazione di un software di buona qualità (e quindi facilmente mantenibile) passa attraverso la corretta progettazione della fase di testing e, in particolar modo, dell'individuazione di test case, che devono riuscire a coprire il maggior numero possibile di componenti del software: funzioni, moduli, costrutti di controllo... Purtroppo, per progetti di grandi dimensioni, è impossibile riuscire ad avere una copertura completa di tutti i componenti. Perciò è indispensabile realizzare procedure che consentano di automatizzare l'individuazione dei test case. Lo possiamo considerare come il banco di prova finale che permette di determinare se ciò che è stato scritto è funzionate o meno: se il software passa tutti i test previsti allora è possibile considerarlo idoneo per essere rilasciato. Il nostro test case, data la ristretta grandezza del progetto, è una tabella che racchiude tutte le funzionalità del software. Al termine dello sviluppo della portlet provvediamo a testare, a mano, tutte

le funzionalità per controllare se siano implementate in modo corretto. Come detto in precedenza, per progetti di dimensioni maggiori vengono realizzati appositi script, solitamente raccolti in suite di test, utilizzati per testare tutte le aree del software.

7.1 Formal test case

Il formal test case è creato per applicazioni che hanno un insieme di requisiti formali. Al fine di testare a fondo che tutti i requisiti di un'applicazione siano funzionanti, ci devono essere almeno due test case per ogni funzionalità:

- un test positivo
- un test negativo.

Per ogni funzionalità è presente una sub-funzionalità, che deve essere sottoposta ad almeno due test case. Il formal test case contiene una descrizione delle funzionalità da testare e la preparazione necessaria per garantire che la prova possa essere effettuata. Un formal test case è caratterizzato da un insieme di dati in ingresso e da un risultato atteso.

7.2 Informal test case

Per le applicazioni o sistemi senza i requisiti formali, il test case può essere scritto sulla base del funzionamento di programmi simili. Alcuni test case non sono scritti in modo completo, in quanto certe funzionalità e i relativi risultati sono riportati solo dopo che il test è stato eseguito.

7.3 Il nostro formal test case

Di seguito presentiamo il test case utilizzato nella fase finale di debug per il controllo delle funzionalità implementate.

Id Check List	ticketing_001_C1 _USER	Registrazione utente			
Modalità di Effettuazione	Unità				
Prog	Id Test	Scopo	Input	Risultati Attesi	Stato	Esito
1	ticketing_001_C1 _USERT1	Avviare una nuova registrazione	Avvio link registrazione	Caricamento e visualizzazione form registrazione		
2	ticketing_001_C1 _USERT2	Invio dati registrazione corretti	Form compilata in modo giusto	Utente registrato con successo (aggiunto al db)		
3	ticketing_001_C1 _USERT3	Invio dati parziali registrazione	Form compilata parzialmente con almeno un campo obbligatorio mancante	Errore di registrazione, utente non inserito nel db		
4	ticketing_001_C1 _USERT4	Registrazione con nickname o email già esistete a db	Form compilata in modo giusto ma nickname email già presente a db	Errore di registrazione, utente non inserito nel db		

Id Check List	ticketing_001_C2 _USER	Login / logout / modifica utente			
Modalità di Effettuazione	Unità				
Prog	Id Test	Scopo	Input	Risultati Attesi	Stato	Esito
1	ticketing_001_C2 _USERT1	Login utente corretto	Username, password utente	Login avvenuto con successo		
2	ticketing_001_C2 _USERT2	Login utente errato	Username o password errati	Login failed		
3	ticketing_001_C2 _USERT3	Logout utente	Avvio link logout	Pagina di logout avvenuto con successo		
4	ticketing_001_C2 _USERT4	Modifica dati utente corretti	Nuovi dati utente nella form prevalorizzata con valori vecchi	Pagina di avvenuta modifica		
5	ticketing_001_C2 _USERT5	Modifica dati utente errati	Inserimento nuovi dati incompleti utente nella form prevalorizzata con valori vecchi	Pagina di errore modifica		

Id Check List	ticketing_001_C3 _USER	Apertura ticket da utente loggato			
Modalità di Effettuazione	Unità				
Prog	Id Test	Scopo	Input	Risultati Attesi	Stato	Esito
1	ticketing_001_C3 _USERT1	Visualizzazione form invio ticket	Avvio link apri ticket	Form per invio ticket		
2	ticketing_001_C3 _USERT2	Invio ticket dati corretti	Valorizzazione corretta dati form ed invio ticket	Ticket inserito correttamente nel db		
3	ticketing_001_C3 _USERT3	Invio ticket dati errati	Valorizzazione errati dati form ed invio ticket	Ticket non inserito a db		

Figura 7.1: Test Case part A

Id Check List	ticketing_001_C4 _USER	Visualizzazione ticket da utente loggato			
Modalità di Effettuazione	Unità				
Prog	Id Test	Scopo	Input	Risultati Attesi	Stato	Esito
1	ticketing_001_C4 _USERT1	Visualizzare i propri ticket	Avvio link visualizza miei ticket	Elenco dei miei ticket		
2	ticketing_001_C4 _USERT2	Visualizzare i ticket presenti nel sistema	Avvio link visualizza ticket	Elenco dei ticket		

Id Check List	ticketing_001_C5 _USER	Informazioni aggiuntive sul ticket già aperto			
Modalità di Effettuazione	Unità				
Prog	Id Test	Scopo	Input	Risultati Attesi	Stato	Esito
1	ticketing_001_C5 _USERT1	Visualizzare il ticket da modificare	Id ticket da modificare	Form valorizzata con i dati del ticket selezionato		
2	ticketing_001_C5 _USERT2	Aggiunta informazioni al ticket	Informazioni da aggiungere	Aggiornamento sul db con le nuove informazioni		

Id Check List	ticketing_001_C6 _USER	Valutazione sulla risoluzione del tiket			
Modalità di Effettuazione	Unità				
Prog	Id Test	Scopo	Input	Risultati Attesi	Stato	Esito
1	ticketing_001_C6 _USERT1	Inserire una valutazione a ticket chiuso	Id del ticket sul quale assegnare una valutazione	Valorizzazione del campo valutazione nel db		

Id Check List	ticketing_001_C7 _STAFF	Visualizzazione ticket			
Modalità di Effettuazione	Unità				
Prog	Id Test	Scopo	Input	Risultati Attesi	Stato	Esito
1	ticketing_001_C7 _STAFFT1	Visualizzare dei ticket assegnati	Avvio link visualizza miei ticket – id membro	Elenco dei ticket assegnati		
2	ticketing_001_C7 _STAFFT2	Visualizzare i ticket presenti nel sistema	Avvio link visualizza ticket	Elenco dei ticket		
3	ticketing_001_C7 _STAFFT3	Visualizzazione dei ticket in vari ordini	Modo in cui si vuole ordinare (priorità, data, ecc...)	Elenco ticket ordinati		
4	ticketing_001_C7 _STAFFT4	Visualizzazione dei ticket non assegnati	Richiesta di ticket liberi	Elenco ticket non assegnati		

Figura 7.2: Test Case part B

Id Check List	ticketing_001_C8 _STAFF	Assegnarsi / abbandonare la lavorazione un ticket			
Modalità di Effettuazione	Unità				
Prog	Id Test	Scopo	Input	Risultati Attesi	Stato	Esito
1	ticketing_001_C8 _STAFFT1	Assegnamento ticket libero	Id ticket libero, id membro staff	Assegnamento corretto del ticket		
2	ticketing_001_C8 _STAFFT2	Assegnamento ticket già assegnato	Id ticket assegnato, id membro staff	Errore, nessun nuovo riassegnamento del ticket		
3	ticketing_001_C8 _STAFFT3	Abbandonare un ticket	Id ticket assegnato, id membro staff	Ticket assegnato a nessuno		

Id Check List	ticketing_001_C9 _STAFF	Modifica / chiusura Ticket			
Modalità di Effettuazione	Unità				
Prog	Id Test	Scopo	Input	Risultati Attesi	Stato	Esito
1	ticketing_001_C9 _STAFFT1	Aggiornare lo stato di lavorazione del ticket	Id ticket, nuova percentuale di avanzamento	Aggiornamento del ticket su db		
2	ticketing_001_C9 _STAFFT2	Aggiunta note al ticket	Id ticket, note da inserire	Aggiornamento del ticket su db		
3	ticketing_001_C9 _STAFFT3	Cambio priorità del ticket	Id ticket, nuova priorità	Aggiornamento del ticket su db		
4	ticketing_001_C9 _STAFFT4	Cambio ambito ticket	Id ticket, nuovo ambito	Aggiornamento del ticket su db		
5	ticketing_001_C9 _STAFFT5	Aggiornamento ore lavorate sul ticket	Id ticket, ore lavorate, da sommare alle precedenti	Aggiornamento del ticket su db		
6	ticketing_001_C9 _STAFFT6	Modifica / eliminazione delle keywords	Id ticket, nuove keywords	Aggiornamento del ticket su db		
7	ticketing_001_C9 _STAFFT7	Segnare come follower di un ticket	Id ticket, mio id	Aggiornamento del ticket su db		

Id Check List	ticketing_001_C11 _STAFF	Scabio comunicazioni con cliente			
Modalità di Effettuazione	Unità				
Prog	Id Test	Scopo	Input	Risultati Attesi	Stato	Esito
1	ticketing_001_C11 _STAFFT1	Aggiunta comunicazione al ticket	Id ticket, comunicazione	Inserimento in db della comunicazione relativa al ticket		

Figura 7.3: Test Case part C

Id Check List	ticketing_001_C14 _ADMIN	Gestione utenti sistema			
Modalità di Effettuazione	Unità				
Prog	Id Test	Scopo	Input	Risultati Attesi	Stato	Esito
1	ticketing_001_C14 _ADMINT1	Eliminazione utente	Id utente da eliminare	Eliminazione dell'utente dal db		
2	ticketing_001_C14 _ADMINT2	Aggiunta utente corretto	Campi per aggiunta utente corretti	Aggiunta utente al db		
3	ticketing_001_C14 _ADMINT3	Aggiunta utente sbagliato	Campi per aggiunta utente errati o parziali	No modifica al db, messaggio di errore		
4	ticketing_001_C14 _ADMINT4	Modifica utente corretta	Id utente, nuovi dati corretti	Modifica dell'utente a db		
5	ticketing_001_C14 _ADMINT5	Modifica utente errata	id utente, nuovi dati errati o parziali	No modifica al db, messaggio di errore		

Figura 7.4: Test Case part D

Conclusioni

Il lavoro di tesi ha avuto come obiettivo la realizzazione di una portlet che permettesse agli acquirenti di un qualunque prodotto o servizio di potersi avvalere di un sistema veloce e pratico per contattare l'azienda venditrice in caso di problemi e a quest'ultima di poter contare su un ottimo sistema di gestione e catalogazione delle segnalazioni ricevute. Il tutto è facilmente integrabile all'interno del portale Liferay aggiungendo e settando opportunamente la nostra portlet. Abbiamo deciso di sviluppare una portlet per il portale Liferay, che negli utlimi anni sta incontrando il consenso di molte aziende anche in Italia, piuttosto che dare vita all'ennesima web application, sopratutto per prendere dimestichezza con lo sviluppo di portlet custom per il portale. Siamo partiti da un'attenta analisi dei software presenti sul mercato, per capire come funzionano i sistemi di ticketing e per stabilire quali fossero le funzionalità più importanti da implementare e ci siamo messi al lavoro. Siamo poi passati attraverso varie fasi: redazione delle specifiche da seguire durante la progettazione, progettazione ed implementazione del database e, successivamente, implementazione della portlet. A conclusione del lavoro abbiamo controllato che tutte le funzionalità implementate fossero correttamente funzionati e rispettassero le specifiche che ci eravamo posti inizialmente. Questa portlet è stata sviluppata per gestire carichi di lavoro medio piccoli, ma nulla vieta di estenderla per un utilizzo in realtà ben più ampio. E' facilmente modificabile per integrare nuove funzionalità che attualmente non sono state implementate.

Bibliografia

[1] *OpenTT: Open source Trouble Ticketing*,
http://opentt.sourceforge.net/it/index.php

[2] *Professional help desk - Site Panel*,
http://www.thephpzone.com/

[3] *Help desk Software - Kayako Fusion*,
http://www.kayako.com/products/fusion/

[4] *Bugzilla*,
http://www.bugzilla.org/

[5] *The Trac Project*,
http://trac.edgewall.org/

[6] *Enterprice open source portal and collaboration software*,
http://www.liferay.com/

[7] *Wiki Liferay*,
http://www.liferay.com/community/wiki

[8] *Strumenti e concetti per sviluppare portlet*,
http://www2.mokabyte.it/cms/article.run?permalink=mb160_liferay-3

[9] *Gestire i dati con il Service Builder*,
http://www2.mokabyte.it/cms/article.run?permalink=mb161_liferay-4

[10] *Portlet, configurazioni, preferenze ed edit mode*, http://www2.mokabyte.it/cms/article.run?permalink=mb162_liferay-5

[11] *Utilizzare il captcha in Liferay*, http://blog.d-vel.com/web/blog/home/-/blogs/utilizzare-il-captcha-in-liferay

[12] *CRUD Liferay portlet development*, http://www.xmlportletfactory.org/

Ultimo controllo siti web 20 Febbraio 2012

Ringraziamenti

Desidero esprimere la mia gratitudine a tutti coloro che mi hanno accompagnato durante i miei studi universitari e nella stesura della tesi finale.

Dedico un particolare ringraziamento in primis al mio relatore, il Prof. Vittorio Ghini, che fin dall'inizio mi ha seguito con grande professionalità e disponibilità nello sviluppo e nella redazione di questa tesi.

Ringrazio la mia famiglia, sempre presente e di supporto nell'aiutarmi a portare a termine questo percorso, incoraggiandomi nei momenti di difficoltà e condividendo i miei successi.

Un sentito grazie va ai miei colleghi universitari. In modo speciale ringrazio Luca Comellini, amico e insostituibile compagno di studio e di progetti, e Alessandro Fogacci, perfetto collaboratore nell'implementazione e scrittura di questo lavoro.

Rivolgo ringraziamenti anche ai miei più cari amici e alla mia ragazza Laura, sempre pronti a sostenermi e a brindare ad ogni piccola o grande vittoria.