Progetto di Ingegneria del Software 2

SWIMv2: Small World Hypothesis Machine v2

Autori: Professore:   
Bulla Jacopo Mottola Luca  
Caio Davide  
Cappa Stefano

Questo documento rappresenta la quinta ed ultima deliverable.

Lo scopo è di eseguire il testing del software realizzato da un altro team di sviluppo. Ovviamente, non consiste solo nel verificare il corretto funzionamento del prodotto, ma anche di esaminare la documentazione individuando eventuali incongruenze.

Indice

1. Introduzione 4

2. Testing degli scenari 5

2.1. Scenario 1 5

2.2. Scenario 2 7

2.3. Scenario 3 9

2.4. Scenario 4 10

2.5. Scenario 5 13

2.6. Scenario 6 17

2.7. Scenario 7 21

2.8. Scenario 8 22

2.9. Scenario 9 23

3. Test aggiuntivi 24

Indice delle figure 29

# Introduzione

Questo documento rappresenta la fase di testing sul progetto (<http://code.google.com/p/bernasconi-calabrese-hessel/>) di un altro team di sviluppo.

Essa è composta da:

* Anna Bernasconi
* Chiara Calabrese
* Matteo Hessel

La fase di testing, descritta nel Project Planning 1.4, è stata sviluppata e suddivisa in nuove sotto-fasi.

1. Verifica del manuale d’installazione.
2. Ricerca di eventuali incongruenze nel RASD e nel DD.
3. Sviluppo di alcuni casi di test, specificandone il risultato.
4. Aggiunta di alcuni consigli verso il team di sviluppo, per migliorare il prodotto.

## Definizioni ed abbreviazioni

* Team di testing: gruppo composto da Bulla, Caio e Cappa
* Team di sviluppo: gruppo composta da Bernasconi, Calabrese e Hessel.
* Repository: è un ambiente di un sistema informativo (di tipo ERP), in cui sono gestiti i metadati, attraverso tabelle relazionali; l'insieme di tabelle, regole e motori di calcolo tramite cui si gestiscono i metadati prendono il nome di metabase. (fonte: wikipedia.org)
* Deliverable: sinonimo di “consegna”

# Installazione

**Il team di sviluppo non ha fornito il prodotto come file eseguibili** (.jar, .war oppure un .ear che li ingloba), ma solamente un collegamento http al “repository” SVN di Google Code.

Poiché nei documenti forniti come deliverable **non sono state fatte considerazioni in merito ai destinatari del prodotto**, si presume che essi abbiano le conoscenze necessarie per installarlo.

Nel caso il committente sia una società/gruppo esterno tale ipotesi potrebbe essere una forzatura e potenzialmente improbabile, poiché solitamente il committente richiede un lavoro ad un gruppo di sviluppatori, proprio perché non ha le conoscenze necessarie.

Nel caso in cui il prodotto sia sviluppato da studenti per un tutor o professore, allora tali ipotesi sono certamente valide, ma in ogni caso **sarebbe stato più corretto precisarlo.**

**Tutte le procedure descritte nel documento d’installazione sono molto generiche e poco approfondite.**

Nel caso in cui il team di sviluppo avesse fornito gli eseguibili, **le procedure legate ad Eclipse** sarebbero state secondarie. Però nel caso in esame **costituiscono l’intero manuale di installazione e per tale motivo sarebbero dovute esser descritte in modo molto più approfondito.**

Per esempio,nel capitolo 3 **la procedura d’installazione del plug-in SVN è poco dettagliata**.

Nel capitolo 4, quando è descritta la procedura di configurazione di MySQL, è presente la seguente frase: *“Inoltre per inserire il* ***connettore*** *abbiamo usato MySQL Workbench.* ***Dopo averla installata*** *abbiamo* ***selezionato New Connection e creato in questo modo la connessione chiamandola swim\_project****”*

A cosa si riferisce la parola “connettore”?

Il team di testing ha ipotizzato che si trattasse del file **“mysql-connector-java-*versione*-bin.jar”**. Però, in tal caso la frase citata non avrebbe senso, poiché questo file .jar **sarebbe importato direttamente nel server e non avrebbe nulla a che vedere con MySQL Workbench.**

Inoltre, anche **la frase successiva è errata, perché dice di creare la connessione come “swim\_project”,** quando in realtà quel nome non è rilevante. Invece, **è importante che il nome del database (“schema”) da creare sia “swim\_project”, poiché configurato nel file persistence.xml all’interno del progetto.**

**Si noti inoltre una grave mancanza: non è stato spiegato che, dopo aver creato la connessione al database, bisogna anche creare il nuovo “schema” (nome del database) tramite MySQL Workbench, chiamandolo “swim\_project”.**

**Nota bene: il prodotto è impostato in modo che ad ogni avvio generi un nuovo database, poiché il file “persistence.xml” contiene la voce “create-drop”. Sarebbe stato più corretto fornire il software con la voce “update” ed un file .sql contenente le query per generare la base di dati.**

Tra l’altro la generazione di queste query può essere facilmente realizzata tramite MySQL Workbench, scegliendo addirittura di inserire alcuni dati predefiniti.

Nel capitolo 5, il problema è che **il team di sviluppo ha realizzato il software tramite Java7 (1.7), nonostante JBoss AS 5.1 sia stato sviluppato per funzionare con la versione precedente o addirittura Java5 (http://anonsvn.jboss.org/repos/jbossas/tags/JBoss\_5\_1\_0\_GA/build/docs/readme.html).** Ovviamente, **tutto ciò ha richiesto la conversione dei progetti da Java7 a Java6**, applicando i seguenti cambiamenti prima per il progetto EJB e dopo per quello WEB:

1. Tasto destro sul progetto->”Properties”->”Java Compiler” e scegliere dal menu a tendina “1.6”, dopodiché salvare ed attendere che Eclipse aggiorni il progetto.
2. Tasto destro sul progetto->”Properties”->”Project Facets” e scegliere la versione “1.6” nella riga denominata “Java”.
3. Tasto destro sul progetto->”Properties”->”Java Build path” rimuovere JRE 1.7 ed importare la 1.6 tramite il pulsante “Add Library”->”JRE System Library”->”Workspace default JRE” e confermare l’operazione.

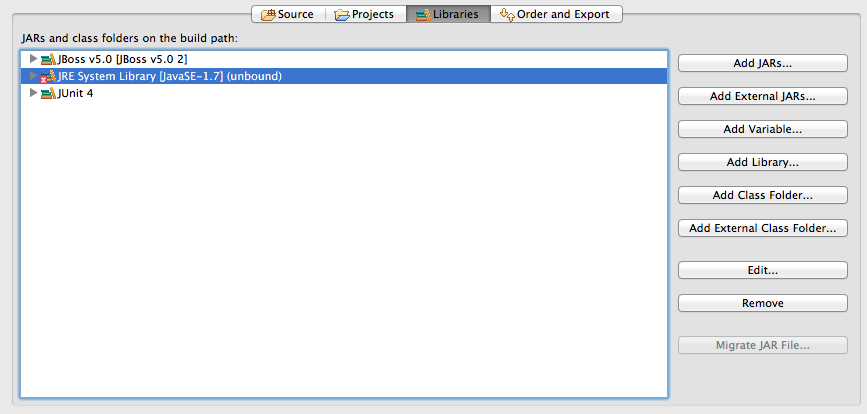


Fig. 2.1 - JRE 7.1 unbound

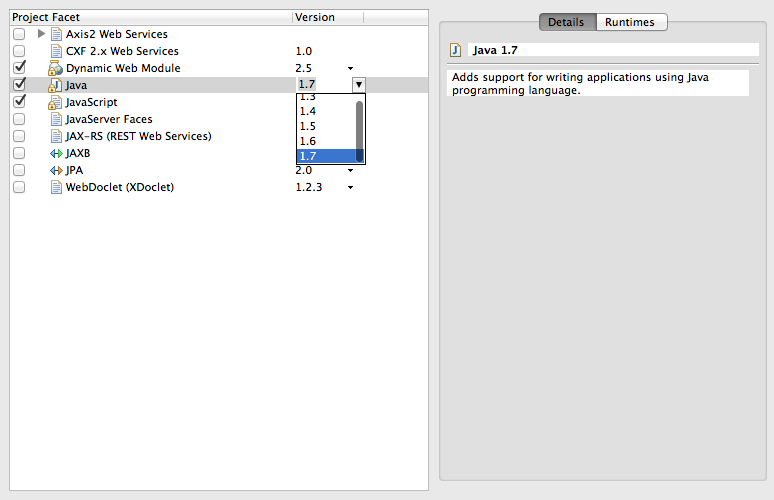


Fig. 2.2 – Problema Project Facets

E’ importante far notare che non è stato spiegato come impostare il server e cioè come aggiungerlo alle “Libraries” (Fig. 2.1).

**Inoltre, in Java6 il casting di un oggetto in (long) non è possibile, bensì è necessario utilizzare l’oggetto Long, cioè “(Long)”. Di conseguenza, il team di testing ha dovuto apportare tale modifica.**

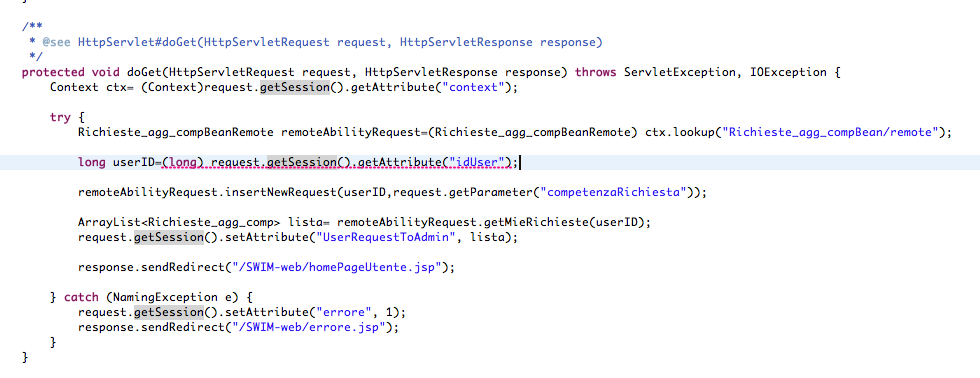


Fig. 2.3 - Problema di casting

**Per concludere, il team di testing ha messo in evidenza il fatto che i caratteri accentati nei nomi dei metodi e delle variabili, creano degli enormi problemi di funzionamento su piattaforme Mac OSX e Linux. Infatti, la codifica dei caratteri dei sistemi Microsoft è diversa e su Mac OSX appare così:**

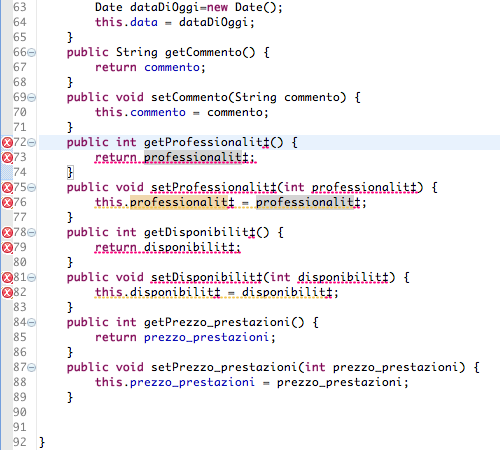


Fig. 2.4 - Problema codifica caratteri

Anche in questo caso un membro del team di testing ha dovuto sostituire manualmente tutti i caratteri errati con la “à” in tutti i file .java e .jsp (procedura tutt’altro che breve).

# Verifica della documentazione

Dopo un’attenta lettura del documento RASD, e delle successive rettifiche, abbiamo notato le seguenti incongruenze:

* La funzionalità del rilascio del feedback, come descritta nel RASD, sembrava facoltativa, mentre nell’implementazione è un passaggio obbligatorio durante la terminazione di una collaborazione.
* La funzionalità del suggerimento di amicizia, descritta nel RASD, è risultata a noi inaccessibile, anche seguendo passo per passo il manuale utente, dove questa parte è comunque descritta in modo vago e superficiale. (Nel prossimo capitolo ci sarà un paragrafo dettagliato su questo problema).

Tutte le altre funzionalità sono congruenti nell’implementazione e nel RASD.

Nel capitolo successivo tuttavia, saranno testate le varie funzionalità in maniera approfondita, provando non solo gli input standard, ma anche casi limite, o addirittura input errati, per vedere come si comporta il sistema in queste condizioni.

# Casi di test

Vengono riportati i test eseguiti sul sistema, corredati dal risultato ottenuto ed eventuali commenti relativi alle discrepanze rispetto all’output atteso.

I test verranno contrassegnati utilizzando un colore diverso: superato, in caso di successo del test, fallito, in caso di fallimento, migliorabile, nel caso in cui il test viene superato, ma con un risultato non del tutto corretto.

## CANCELLAZIONE ACCOUNT SWIMv2

Input: cancellazione account appena creato;

Output: utente cancellato ;

Risultato: superato.

Input: cancellazione account con diversi rapporti di amicizia;

Output: utente cancellato, e utente cancellato dalla lista degli amici;

Risultato: superato.

Input: cancellazione account con richieste non ancora accettate;

Output: utente cancellato, ma le richieste di aiuto non vengono cancellate, e viene sollevata un’eccezione quando l’utente richiesto dall’utente cancellato cerca di accedere alle richieste di aiuto;

Risultato: fallito.

## FUNZIONE DI RICERCA

* Input: ricerca utenti con cognome e nome desiderato;
* Output: utente ricercato, con l’opzione di accedere al suo profilo;
* Risultato: superato.
* Input: ricerca utenti con data competenza;
* Output: lista utenti con determinata competenza, se l’utente che sta effettuando la ricerca nelle proprie competenze ha la competenza cercata, risulta fra i risultati;
* Risultato: migliorabile.
* Input: ricerca utenti per cognome e nome desiderato, riempiendo anche la textbox delle competenze, e premendo il tasto submit per la ricerca di utenti per nominativo;
* Output: la lista degli utenti trovata non è quella per nome, ma quella per competenza, nonostante il pulsante schiacciato si riferisce alla ricerca per nominativo;
* Risultato: fallito.
* Input: ricerca utenti per competenza, ma solo tra gli amici;
* Output: lista degli utenti amici che hanno la competenza descritta;
* Risultato: superato;

## RICHIEDERE AMICIZIA

* Input: ricerca dell’utente con nome e cognome, e visualizzazione del profilo. Invio richiesta amicizia con l’apposito pulsante;
* Output: nel profilo dell’utente a cui si è richiesta l’amicizia compare la richiesta stessa;
* Risultato: superato;
* Input: ricerca utente già amico;
* Output: il pulsante richiedi amicizia non compare;
* Risultato: superato;
* Input: tentativo di richiedere amicizia a se stessi;
* Output: il pulsante richiedi amicizia non compare sul profilo dell’utente stesso;
* Risultato: superato;

## RISPONDERE A RICHIESTA DI AMICIZIA

* Input: conferma amicizia;
* Output: i due utenti vengono inseriti nelle rispettive liste di amici;
* Risultato: superato;
* Input: rifiuto amicizia;
* Output: nessuna notifica di accettazione, i due utenti non compaiono nelle rispettive liste di amici, e l’amicizia si può ora riproporre;
* Risultato: superato;
* Input: conferma amicizia diretta;
* Output: i due utenti divengono effettivamente amici, ma non si verifica la situazione descritta nelle note del TEST UR 12 del documento di testing degli sviluppatori. Nessuno meccanismo di suggerimento viene sviluppato, o almeno non si riesce ad accedervi;
* Risultato: fallito.

***NOTA***: sia simulando lo scenario, sia leggendo il manuale utente, con le informazioni ottenute non si riesce ad accedere alla funzionalità di suggerimento di amicizie. Nel manuale c’è scritto che basta premere il pulsante nuovi amici e confermare un’amicizia per ricevere suggerimenti, ma questo effettivamente non avviene, quando si accede a nuovi amici risulta che non ci sono utenti suggeriti, e quando si conferma un’amicizia si viene subito reindirizzati presso il proprio profilo, non potendo quindi osservare questa pagina successivamente alla conferma della richiesta. Una volta rientrati, la situazione è sempre la stessa, nessun suggerimento.

Effettivamente dopo una veloce lettura del codice, sembrerebbe ci sia un meccanismo di suggerimento, ma dopo averlo testato sembra non funzionare, oppure il funzionamento non è spiegato bene nel manuale dell’utente.

## ELIMINARE UN’AMICIZIA

* Input: eliminazione di un’amicizia, attraverso la lista dei propri amici;
* Output: l’amicizia tra i due utenti viene effettivamente cancellata, ed è ora possibile rifare una richiesta di amicizia;
* Risultato: superato

SUGGERIMENTO: meglio creare una conferma prima dell’eliminazione, in quanto un utente può erroneamente schiacciare il pulsante di eliminazione.

## INVIO RICHIESTA AIUTO

* Input: tutti i campi vuoti;
* Output: messaggio di errore in cui il sistema avverte la presenza di campi non compilati;
* Risultato: superato
* Input: abilità richiesta inesistente;
* Output: messaggio di errore in cui il sistema avverte che l’abilità non appartiene al set dell’utente in questione;
* Risultato: superato
* Input: abilità esistente nel set del sito, ma non appartenente al set di abilità dell’utente;
* Output: messaggio di errore in cui il sistema avverte che l’abilità non appartiene al set dell’utente in questione;
* Risultato: superato

SUGGERIMENTI: mettere una checkbox con cui selezionare le abilità, oppure un menù a tendina con tutte le abilità dell’utente, per favorire una rapida navigazione dell’utente.

* Input: nessun messaggio di richiesta;
* Output: richiesta inviata;
* Risultato: superato

SUGGERIMENTI: imporre un limite di lunghezza, per evitare problemi con il database.

## ACCETTZIONE DI UNA RICHIESTA DI AIUTO

* Input: rifiuto collaborazione;
* Output: la collaborazione di fatto viene annullata;
* Risultato: superato;
* Input: conferma collaborazione senza messaggio di risposta;
* Output: la collaborazione viene stipulata;
* Risultato: superato;

SUGGERIMENTI: quando i campi non vengono compilati, meglio inserire una stringa di default, come ad esempio “nessun messaggio rilasciato”, evitano così nella descrizione di questi campi, il loro nome seguito dalla stringa vuota.

Esempio nella descrizione di una collaborazione senza messaggio di risposta:

Messaggio di Risposta:

# Consigli aggiuntivi

# Indice delle figure

Fig. 2.1 - JRE 7.1 unbound 6

Fig. 2.2 – Problema Project Facets 6

Fig. 2.3 - Problema di casting 7

Fig. 2.4 - Problema codifica caratteri 7