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

1.1. Definizioni ed abbreviazioni 4

2. Installazione 5

3. Verifica della documentazione 8

4. Casi di test 9

4.1. Utente non registrato 9

4.1.1. Registrazione 9

4.1.2. Ricerca persone per nome e cognome 10

4.1.3. Ricerca persone per competenza 10

4.2. Utente registrato 10

4.2.1. Log-in 10

4.2.2. Modifica dati (dati personali e password) 11

4.2.3. Modifica dati (aggiungi competenze dichiarate) 12

4.2.4. Modifica dati (eliminazione competenze dichiarate) 12

4.2.5. Richiesta aggiunta competenza 12

4.2.6. Cancellazione account 13

4.2.7. Funzione di ricerca 13

4.2.8. Richiesta amicizia 13

4.2.9. Risposta ad una richiesta d’amicizia 14

4.2.10. Eliminazione di un’amicizia 14

4.2.11. Invio richiesta di aiuto 14

4.2.12. Accettazione di una richiesta di aiuto 15

4.2.13. Rifiuto di una richiesta di aiuto 15

4.2.14. Inserisci una valutazione 15

4.2.15. Logout 15

4.3. Amministratore 16

4.3.1. Login 16

4.3.2. Gestione richieste utente 16

4.3.3. Gestione archivio competenze 17

4.3.4. Gestione utenti 17

5. Consigli aggiuntivi 18

5.1. Pagina profilo utente 18

Indice delle figure 19

# 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:

* \*\*\*\*\*\* Rimosso per non fare nomi \*\*\*\*\*\*
* \*\*\*\*\*\* Rimosso per non fare nomi \*\*\*\*\*\*
* \*\*\*\*\*\* Rimosso per non fare nomi \*\*\*\*\*\*

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 al 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
* Input: condizioni/dati d’ingresso
* Output: risultato ottenuto

# 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 **è da importare direttamente nel server e non ha 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, dato che 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.

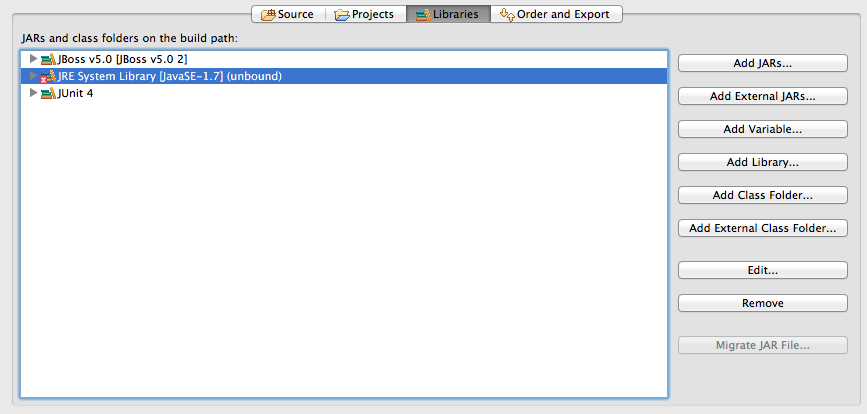


Fig. 2.1 - JRE 7.1 unbound

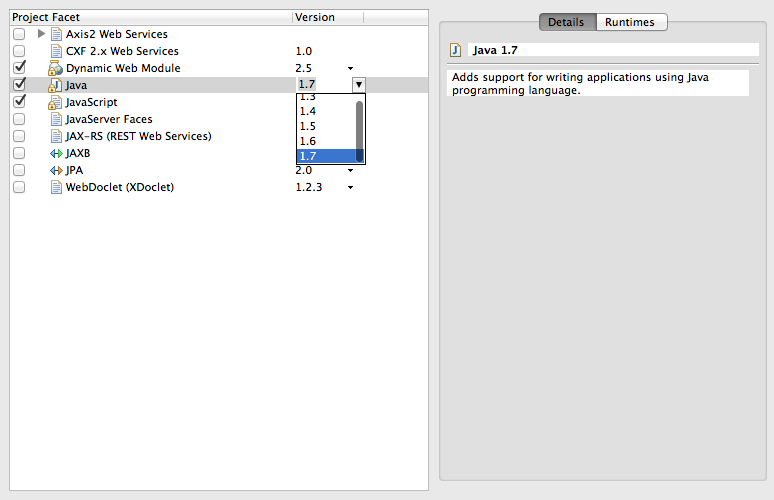


Fig. 2.2 – Problema Project Facets

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.**

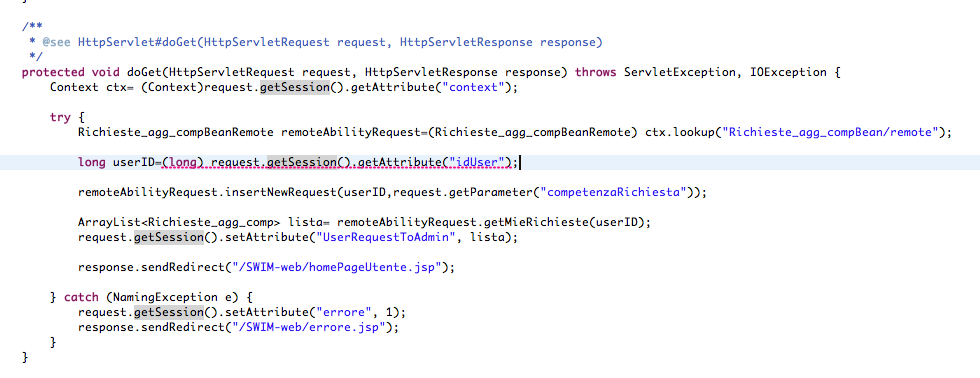


Fig. 2.3 - Problema di casting

**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:**

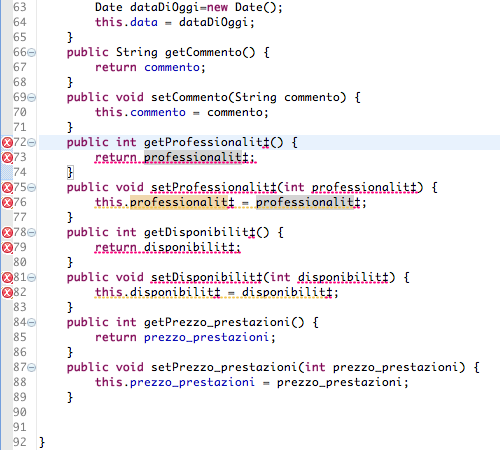


Fig. 2.4 - Problema codifica caratteri

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 RASD e delle successive rettifiche, il team di testing ha individuato le seguenti incongruenze:

* Come descritta nel RASD, la funzione del rilascio del feedback sembra facoltativa, mentre nell’implementazione è un passaggio obbligatorio, durante la terminazione di una collaborazione.
* La funzione del suggerimento di amicizia è inaccessibile, anche seguendo passo per passo il manuale utente, dove questa parte è descritta comunque in modo vago e superficiale (fare riferimento al capitolo 4 per una spiegazione più dettagliata).

Tutte le altre funzionalità descritte nel RASD sono congruenti con l’implementazione.

Inoltre, nel capitolo successivo saranno testate le varie funzionalità in modo approfondito, provando non solo i casi di test individuati dal team di sviluppo, ma anche quelli limite, cioè con “ingresso” (input) in condizioni particolari.

# Casi di test

In questo capitolo sono riportati i test eseguiti sul sistema, indicando il risultato ottenuto ed eventuali commenti aggiuntivi.

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

**Attenzione: il team di sviluppo non ha effettuato nessun controllo sulle lunghezze dei campi di testo in tutto il software. Ogni inserimento superiore ai 255 caratteri per le stringhe o 11 cifre per i numeri genera un’eccezione simile a auesta: javax.persistence.PersistenceException: org.hibernate.exception.DataException: could not insert: [swim.entitybeans.User].**

## Utente non registrato

* + 1. Registrazione

1. Input: campi tutti vuoti

Output: nickname non inserito

Risultato: superato

1. Input: campo nickname vuoto

Output: nickname non inserito

Risultato: superato

1. Input: campo password vuoto

Output: password non inserita

Risultato: superato

1. Input: campo conferma password vuoto

Output: conferma della password mancante

Risultato: superato

1. Input: campi password e conferma password uguali (parola di meno di 6 caratteri)

Output: password troppo corta, almeno 6 caratteri

Risultato: superato

1. Input: campi password e conferma password diversi (parola con più di 6 caratteri)

Output: le due password non coincidono

Risultato: superato

1. Input: campo nome vuoto

Output: nome non inserito

Risultato: superato

1. Input: campo cognome vuoto

Output: cognome non inserito

Risultato: superato

1. Input: campo città vuoto

Output: città non inserita

Risultato: superato

1. Input: campo mail vuoto

Output: mail non inserita

Risultato: superato

1. Input: campo mail = “prova.prova.it”

Output: email considerata come valida

Risultato: migliorabile, nessun controllo sulla validità dell’e-mail

1. Input: campo anno nascita vuoto

Output: nickname non inserito

Risultato: superato

1. Input: campo anno nascita = “2001”oppure “1899”

Output: anno di nascita impossibile

Risultato: migliorabile. Il controllo della data è statico e non dipende dall’anno attuale, quindi ogni anno sarà necessario aggiornare il sistema, cambiando l’intervallo temporale oppure eliminandolo completamente.

1. Input: campo opzionali vuoti

Output: avviene la registrazione, se i campi obbligatori sono compilati correttamente

Risultato: superato

* + 1. Ricerca persone per nome e cognome

1. Input: solo il nome o solo il cognome dell’utente da ricercare

Output: nessun risultato

Risultato: migliorabile, nonostante sia stato scelto di limitare la ricerca all’inserimento di nome e cognome, sarebbe più corretto mostrare un errore nel caso d’inserimento di uno solo dei due, invece di mostrare una lista vuota di risultati.

1. Input: nome e cognome dell’utente da ricercare

Output: profili trovati con la possibilità di visualizzarli

Risultato: migliorabile, l’email è mostrata pubblicamente anche senza eseguire il login

* + 1. Ricerca persone per competenza

1. Input: la competenza da ricercare

Output: lista degli utenti con la competenza scelta

Risultato: superato

## Utente registrato

* + 1. Log-in

1. Input: nickname e password

Output: viene visualizzato il profilo

Risultato: superato

* + 1. Modifica dati (dati personali e password)

1. Input: uno o più dei campi di testo nome, cognome, città, mail, sesso, diploma, laurea, altro

Output: i dati sono modificati correttamente ed è mostrato il profilo

Risultato: superato

1. Input: nickname nuovo (inutilizzato) oppure quello già in uso dall’utente stesso

Output: nickname modificato correttamente e la pagina del profilo viene visualizzata

Risultato: superato

1. Input: nickname nuovo (già usato da un altro utente)

Output: nickname modificato, nonostante sia già in uso

Risultato: fallito. Si genera un’inconsistenza nel database, poiché vi potranno essere più utenti con gli stessi nickname. Questo potrebbe non costituire un problema nel solo caso in cui tutti gli utenti con lo stesso nickname abbiano una password differente. Ovviamente, tale situazione è imprevedibile e comunque inaccettabile.

Per completezza il team di testing ha deciso di aggiungere la seguente condizione che mostra un particolare errore: almeno due utenti non lo stesso nickname e con la stessa password presenti nel database causano il seguente errore durante il login (effettuato da uno qualunque di essi):

javax.ejb.EJBException: javax.persistence.NonUniqueResultException: result returns more than one elements

org.jboss.ejb3.tx.Ejb3TxPolicy.handleExceptionInOurTx(Ejb3TxPolicy.java:77)

org.jboss.aspects.tx.TxPolicy.invokeInOurTx(TxPolicy.java:83)

org.jboss.aspects.tx.TxInterceptor$Required.invoke(TxInterceptor.java:190)

org.jboss.aop.joinpoint.MethodInvocation.invokeNext(MethodInvocation.java:102) org.jboss.aspects.tx.TxPropagationInterceptor.invoke(TxPropagationInterceptor.java:76) org.jboss.aop.joinpoint.MethodInvocation.invokeNext(MethodInvocation.java:102)

org.jboss.ejb3.tx.NullInterceptor.invoke(NullInterceptor.java:42)

org.jboss.aop.joinpoint.MethodInvocation.invokeNext(MethodInvocation.java:102) org.jboss.ejb3.security.Ejb3AuthenticationInterceptorv2.invoke(Ejb3AuthenticationInterceptorv2.java:186) org.jboss.aop.joinpoint.MethodInvocation.invokeNext(MethodInvocation.java:102) org.jboss.ejb3.ENCPropagationInterceptor.invoke(ENCPropagationInterceptor.java:41) org.jboss.aop.joinpoint.MethodInvocation.invokeNext(MethodInvocation.java:102) org.jboss.ejb3.BlockContainerShutdownInterceptor.invoke(BlockContainerShutdownInterceptor.java:67) org.jboss.aop.joinpoint.MethodInvocation.invokeNext(MethodInvocation.java:102) org.jboss.aspects.currentinvocation.CurrentInvocationInterceptor.invoke(CurrentInvocationInterceptor.java:67) org.jboss.aop.joinpoint.MethodInvocation.invokeNext(MethodInvocation.java:102) org.jboss.ejb3.stateless.StatelessContainer.dynamicInvoke(StatelessContainer.java:421) org.jboss.ejb3.remoting.IsLocalInterceptor.invokeLocal(IsLocalInterceptor.java:85)

org.jboss.ejb3.remoting.IsLocalInterceptor.invoke(IsLocalInterceptor.java:72) org.jboss.aop.joinpoint.MethodInvocation.invokeNext(MethodInvocation.java:102)

org.jboss.aspects.remoting.PojiProxy.invoke(PojiProxy.java:62)

$Proxy334.invoke(Unknown Source)

org.jboss.ejb3.proxy.impl.handler.session.SessionProxyInvocationHandlerBase.invoke(SessionProxyInvocationHandlerBase.java:207) org.jboss.ejb3.proxy.impl.handler.session.SessionProxyInvocationHandlerBase.invoke(SessionProxyInvocationHandlerBase.java:164) $Proxy333.verificaCredenziali(Unknown Source)

accessServlet.LoginServlet.doPost(LoginServlet.java:78)

javax.servlet.http.HttpServlet.service(HttpServlet.java:637)

javax.servlet.http.HttpServlet.service(HttpServlet.java:717)

org.jboss.web.tomcat.filters.ReplyHeaderFilter.doFilter(ReplyHeaderFilter.java:96)

Cioè si tratta di un errore in un entity bean, dovuto al fatto che per ottenere un Utente ci si aspetta un solo risultato dalla query, mentre in questo caso particolare i risultati potrebbero essere molti di più.

1. Input: anno di nascita numerico

Output: l’anno di nascita è stato modificato

Risultato: migliorabile, non è eseguito nessun controllo sull’anno di nascita, come invece avviene in fase di registrazione

1. Input: anno di nascita non numerico, per esempio “qweqwe”

Output: NumberFormatException

java.lang.NumberFormatException: For input string: "qweqwe" java.lang.NumberFormatException.forInputString(NumberFormatException.java:48)

java.lang.Integer.parseInt(Integer.java:449)

java.lang.Integer.parseInt(Integer.java:499) navigationServlet.DataModificationServlet.doPost(DataModificationServlet.java:96) javax.servlet.http.HttpServlet.service(HttpServlet.java:637) javax.servlet.http.HttpServlet.service(HttpServlet.java:717) org.jboss.web.tomcat.filters.ReplyHeaderFilter.doFilter(ReplyHeaderFilter.java:96)

Risultato: fallito

1. Input: anno di nascita numerico ma troppo lungo, per esempio “12121213123123124124124124”

Output: NumberFormatException

java.lang.NumberFormatException: For input string: "12121213123123124124124124"

java.lang.NumberFormatException.forInputString(NumberFormatException.java:48)

java.lang.Integer.parseInt(Integer.java:461)

java.lang.Integer.parseInt(Integer.java:499)

navigationServlet.DataModificationServlet.doPost(DataModificationServlet.java:96)

javax.servlet.http.HttpServlet.service(HttpServlet.java:637)

javax.servlet.http.HttpServlet.service(HttpServlet.java:717)

org.jboss.web.tomcat.filters.ReplyHeaderFilter.doFilter(ReplyHeaderFilter.java:96)

Risultato: fallito

1. Input: la nuova password e quella precedente errata (oppure lasciando il campo vuoto)

Output: la password vecchia non corrisponde

Risultato: superato

1. Input: la nuova password più lunga di 6 caratteri e quella precedente corretta

Output: la password è stata modificata correttamente ed è mostrato il profilo

Risultato: superato

1. Input: la nuova password più corta di 6 caratteri e quella precedente corretta

Output: la password è stata modificata nonostante non rispetti il vincolo sulla lunghezza deciso in fase di registrazione

Risultato: migliorabile, inserire un controllo sulla lunghezza come per la registrazione

* + 1. Modifica dati (aggiungi competenze dichiarate)

1. Input: la competenza da dichiarare tramite menu a tendina

Output: conferma dell’operazione in caso di successo, oppure un messaggio di errore nel caso in cui sia già stata dichiarata

Risultato: migliorabile, con un numero elevato di competenze il menu a tendina diventerebbe lunghissimo ed inoltre, sarebbe più sensato rimuovere dalla lista le competenze già dichiarate

* + 1. Modifica dati (eliminazione competenze dichiarate)

1. Input: la competenza da eliminare

Output: lista aggiornata delle competenze che si possono eliminare

Risultato: migliorabile, non è richiesta nessuna conferma di eliminazione. Questo potrebbe portare a cancellazioni accidentali dell’intero profilo.

* + 1. Richiesta aggiunta competenza

1. Input: la competenza da proporre

Output: la pagina del profilo dell’utente

Risultato: superato

1. Input: la competenza da proporre è un campo vuoto

Output: la pagina del profilo dell’utente

Risultato: fallito. Non vi è una vera eccezione, ma il team di testing ha deciso di considerlo fallito poiché l’amministratore riceverebbe una proposta vuota, senza alcun testo che permetterebbe di risalire alla competenza da aggiungere.

* + 1. Cancellazione account

Nota: è consigliabile chiedere conferma prima di eseguire un’operazione così importante e completamente irreversibile. Se un utente dovesse accidentalmente cliccare sul pulsante per la cancellazione, perderebbe il suo intero profilo.

1. Input: cancellazione account appena creato

Output: utente cancellato

Risultato: migliorabile

1. Input: cancellazione account con diversi rapporti di amicizia

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

Risultato: migliorabile

1. Input: cancellazione account con richieste non ancora accettate

Output: utente cancellato, ma le richieste di aiuto non vengono cancellate. Inoltre si solleva un’eccezione quando l’utente richiesto da quello cancellato cerca di accedere alle richieste di aiuto

Risultato: fallito

* + 1. Funzione di ricerca

1. Input: ricerca utenti con cognome e nome desiderato

Output: utente ricercato, con l’opzione di accedere al suo profilo

Risultato: superato

1. 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

1. 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

1. Input: ricerca utenti per competenza, ma solo tra gli amici

Output: lista degli utenti amici che hanno la competenza descritta

Risultato: superato

* + 1. Richiesta amicizia

1. 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

1. Input: ricerca utente già amico

Output: il pulsante richiedi amicizia non compare

Risultato: superato

1. Input: tentativo di richiedere amicizia a se stessi

Output: il pulsante richiedi amicizia non compare sul profilo dell’utente stesso

Risultato: superato

* + 1. Risposta ad una richiesta d’amicizia

1. Input: conferma amicizia

Output: i due utenti vengono inseriti nelle rispettive liste di amici

Risultato: superato

1. 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

1. 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. Nessun meccanismo di suggerimento viene sviluppato, o almeno non si riesce ad accedervi

Risultato: fallito, 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

* + 1. Eliminazione di un’amicizia

1. 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: migliorabile, è meglio creare una conferma prima dell’eliminazione, in quanto un utente potrebbe erroneamente schiacciare il pulsante di eliminazione

* + 1. Invio richiesta di aiuto

1. Input: tutti i campi vuoti

Output: messaggio di errore in cui il sistema avverte la presenza di campi non compilati

Risultato: superato

1. 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

1. 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: migliorabile, si potrebbe mettere una lista di checkbox con cui selezionare le abilità, oppure un menù a tendina con tutte le abilità dell’utente, per favorire una rapida navigazione dell’utente.

1. Input: nessun messaggio di richiesta

Output: richiesta inviata

Risultato: superato

* + 1. Accettazione di una richiesta di aiuto

1. Input: rifiuto collaborazione

Output: la collaborazione di fatto viene annullata

Risultato: superato

1. Input: conferma collaborazione senza messaggio di risposta

Output: la collaborazione viene stipulata

Risultato: migliorabile, quando i campi non sono compilati, sarebbe meglio inserire una stringa di default, come ad esempio “nessun messaggio rilasciato”. Questo eviterebbe che nella descrizione di questi campi di testo, il loro nome sia seguito dalla stringa vuota.

Esempio nella descrizione di una collaborazione senza messaggio di risposta:

Messaggio di Risposta:

* + 1. Rifiuto di una richiesta di aiuto

1. Input: clic sul pulsante “rifiuta”

Output: nessuno

Risultato: superato

* + 1. Inserisci una valutazione

1. Input: campi punteggio default, campi testo vuoti

Output: valutazione inserita

Risultato: migliorabile, nella pagina dell’utente valutato, tra i commenti compare la scritta “Commento:” anche se non è stato lasciato alcun commento

1. Input: commento di lunghezza normale, punteggio

Output: valutazione inserita

Risultato: superato

Commento: vedi sezione consigli

1. Input: commento lungo, punteggio

Output: Eccezione non gestita

Risultato: fallito

* + 1. Logout

1. Input: nessuno

Output: pagina di login

Risultato: superato

## Amministratore

* + 1. Login

1. Input: nessuno

Output: login non effettuato

Risultato: superato

1. Input: nickname

Output: login non effettuato

Risultato: superato

1. Input: password

Output: login non effettuato

Risultato: superato

1. Input: nickname, password

Output: login effettuato

Risultato: superato

* + 1. Gestione richieste utente

1. Input: esiste già

Output: abilità non aggiunta

Risultato: superato

1. Input: rifiuta

Output: abilità non aggiunta

Risultato: superato

1. Input: nessuno

Output: abilità non aggiunta

Risultato: fallito

1. Input: codice competenza

Output: abilità non aggiunta

Risultato: fallito

1. Input: competenza

Output: abilità non aggiunta

Risultato: fallito

1. Input: codice competenza, competenza

Output: abilità aggiunta

Risultato: superato

1. Input: codice competenza già esistente o competenza già esistente

Output: abilità non aggiunta

Risultato: superato

* + 1. Gestione archivio competenze

1. Input: competenza

Output: abilità non aggiunta

Risultato: fallito. Si può aggiungere una competenza con uno o tutti e due i campi vuoti.

1. Input: codice competenza

Output: abilità non aggiunta

Risultato: fallito

1. Input: nessuno

Output: abilità non aggiunta

Risultato: fallito

1. Input: competenza, codice competenza

Output: abilità aggiunta

Risultato: superato

1. Input: competenza già aggiunta o codice competenza già aggiunto

Output: abilità non aggiunta

Risultato: superato

* + 1. Gestione utenti

1. Input: visualizza profilo

Output: pagina profilo

Risultato: superato

# Consigli aggiuntivi

## Pagina profilo utente

La sezione commenti dei feedback diventa illeggibile, una volta che iniziano ad esserci diversi commenti. Tutti i commenti sono attaccati senza spaziatura da un paragrafo all’altro, e non c’è nessun tipo di formattazione nel testo che faccia capire dove finisce un commento e ne inizia un altro.

# 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