Ingegneria, Gestione ed Evoluzione del Software
Maintenance Report & Impact Analisys
SocialNotes

Luigi Allocca	**0522501559**	l.allocca8@studenti.unisa.it
Rocco Iuliano	**0522501533**	r.iuliano13@studenti.unisa.it
Simone Della Porta	**0522501534**	s.dellaporta6@studenti.unisa.it

Approccio alla manutenzione

Per poter realizzare le change request definite e approvate, abbiamo eseguito un re-engineering del progetto. È stata effettuata una fase di **pianificazione** in cui sono stati analizzati i documenti esistenti per poter definire le modifiche da apportare sia alla documentazione che al codice. Successivamente abbiamo applicato una fase di **forward engineering** seguendo quanto pianificato e per ogni change request sono state descritte le modifiche apportate per poi calcolare le metriche di information retrieval.
Per definire la **matrice di connettività** abbiamo usato la **distance based approach** e la distanza è definita come *"A modifica B"*. La soglia entro la quale consideriamo le componenti da modificare è pari a 3.

1 Miglioramento sicurezza accessi

1.1 Starting Impact Set (SIS)

Modifiche previste:

- Pagina *login* per mostare un messaggio di eccesso di tentativi di login;
- Servlet *Login* per aggiungere il controllo del numero di tentativi di accesso falliti;
- Classe *LoginTest* per test di sistema della classe Login.
- Schema del database *SocialNotes_Schema* per l'aggiunta del campo *bloccato* all'utente;
- Classe *UserBean* per l'aggiunta dell'attributo `bloccato`;
- Classe *UserModelDS* per la modifica delle query e l'aggiunta dell'aggiornamento del campo *bloccato*.
- Classe *UserModelDSTest* per test di unità della classe UserModelDs.

1.2 Candidate Impact Set (CIS)

Dopo un'attenta analisi del database, Requirement Analysis Document (per i sequence diagram), Object Design Document e sulla base del SIS, presumiamo di modificare le componenti mostrate nella connectivity matrix 1. Per una maggiore leggibilità della matrice abbiamo assegnato i seguenti nomi alla componenti:

- *login.jsp* corrisponde a JSP_1
- *UsermodelDS.java* corrisponde a $Java_1$
- *Login.java* corrisponde a $Java_2$
- *SocialNotes_Schema.sql* corrisponde a SQL_1
- *UserBean.java* corrisponde a $Java_3$

Inoltre, il CIS include i file di test LoginTest.java e UserModelDSTest.java

Table 1: Distance based approach per identificare il Candidate Impact Set

	JSP_1	$Java_1$	$Java_2$	SQL_1	$Java_3$
JSP_1	-	2	1	3	3
$Java_1$	2	-	1	1	1
$Java_2$	1	1	-	2	1
SQL_1	3	1	2	-	1
$Java_3$	3	1	2	1	-

1.3 Actual Impact Set (AIS)

Dopo aver applicato le opportune modifiche al sistema, le componenti impattate sono:

- *login.jsp*
- *Login.java*
- *SocialNotes_Schema.sql*
- *UserBean.java*
- *UserModelDS.java*
- *LoginTest.java*
- *UserModelDSTest.java*
- *UtenteExpectedImageUserNotPresent.xml*
- *UtenteExpectedImage.xml*
- *UtenteExpectedEmailUsNotPresent.xml*
- *UtenteExpectedBloccatoUpdate.xml*
- *UtenteExpectedBan.xml*
- *UtenteExpected.xml*
- *Utente.xml*
- *Utente.sql*

1.4 False Positive Impact Set (FPIS)

Non sono stati riscontrati falsi positivi.

1.5 Discovered Impact Set (DIS)

- *UtenteExpectedImageUserNotPresent.xml*
- *UtenteExpectedImage.xml*
- *UtenteExpectedEmailUsNotPresent.xml*
- *UtenteExpectedBloccatoUpdate.xml*
- *UtenteExpectedBan.xml*
- *UtenteExpected.xml*
- *Utente.xml*
- *Utente.sql*

1.6 Metriche Impcat Analysis

A seguito delle operazioni di modifica, sono state calcolate Recall, Precision, Adequacy ed Effectiveness, al fine di valutare la bontà dell'impact analysis effettuato:
Recall : $|CIS \cap AIS| \div |AIS| = 7 \div 15 = 46.6\%$;
Precision: $|CIS \cap AIS| \div |CIS| = 7 \div 7 = 100\%$;
Inclusiveness = 0

1.7 Descrizione della modifica

Per poter implementare la change request 1 abbiamo modificato la funzionalità di login del sistema, quindi abbiamo ottenuto i seguenti sequence diagram e navigational path:

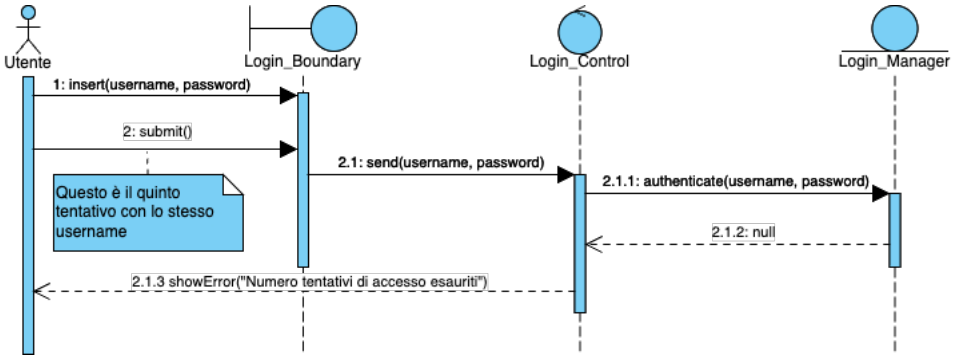


Figure 1: S.D. Login Fallito per eccessivo numero di tentativi falliti

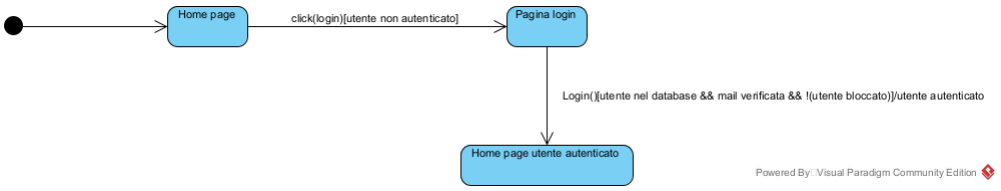


Figure 2: N.P. Login Fallito per eccessivo numero di tentativi falliti

Successivamente abbiamo apportato modifiche al codice attraverso il Test driven development per le modifiche di unità, aggiornando e creando i casi di test per poi apportare le modifiche alle classi bean e DS. Per quanto riguarda le modifiche di sistema, abbiamo aggiornato la pagina di login, e modificato la medesima servlet, per poi eseguire i test con Selenium. Durante lo sviluppo e la modifica dei test abbiamo scoperto che alcuni file XML necessitavano di un aggiornamento per essere consistenti con il nuovo DB. Gli aggiornamenti apportati sono i seguenti:

- La login.jsp ora mostra due nuovi messaggi di errore, uno per il raggiungimento del limite di tentativi di accesso falliti, ed uno in caso l'utente tentasse di accedere mentre è bloccato.
- È stato aggiornato lo stato della classe Login con l'aggiunta di un nuovo attributo calendar, ed è stato aggiornato il metodo doPost che ora aggiunge nella sessione HTTP un hashmap con username e numero di tentativi di accesso falliti da parte di quell'utente.
- La classe LoginTest appunto va a testare le modifiche apportate alla servlet
- SocialNotes_Schema per l'aggiunta del campo bloccato alla tabella Utente.
- La classe UserBean ha un nuovo attributo bloccato di tipo Timestamp, con annessi metodi getter e setter.
- UserModelDS per l'aggiunta delle query relative all' aggiiornamento del campo bloccato.
- UserModelDSTest per i test di unità della classe UserModelDS

Infine sono stati lanciati tutti i casi di test creati e modificati per verificare il corretto funzionamento del sistema dopo aver modificato il codice, con risultato positivo.

2 Invio link di verifica email nella mail di benvenuto

2.1 Starting Impact Set (SIS)

Componenti da modificare:

- Tabella *Utente* del database (*SocialNotes_Schema*) per aggiungere il campo *verificato*;
- Servlet *SignupControl* per includere il link di verifica nella mail di registrazione;
- Servlet *Login* per verificare se l'utente che sta cercando di effettuare l'accesso non ha ancora confermato la sua mail;
- Classe *UserBean* per aggiungere il campo booleano *verificato* che rappresenta la validità della mail dell'utente;
- Classe *UserModelDS* per aggiungere un metodo che permetta di aggiornare lo stato del campo *verificato*, modificare il metodo *doSave* in modo da memorizzare correttamente l'utente (per aggiungere il campo *verificato*) e i metodi *doRetrieve* per poter ottenere il campo verificato;
- Classe *ChangeProfile* per inviare un link di verifica mail nel caso l'utente modifichi la mail con la quale si è registrato;
- Classe *UserModelDSTest* per il testing di unità della classe UserModelDs;
- Classe *LoginTest* per il testing funzionale, in cui si verifica la funzionalità di autenticazione;
- Classe *RegistrazioneTest* per il testing funzionale, in cui si verifica la funzionalità di registrazione;
- Classe *ModificaEmailTest* per il testing funzionale, in cui si verifica la funzionalità di cambio mail.

Componenti da creare:

- Creare una nuova servlet (*VerificaMail.java*) che viene invocata quando l'utente cliccherà il link nella mail di benvenuto in modo da poter attivare il corrispettivo campo della tabella *Utente*;
- Creare una pagina JSP (*mailVerificata.jsp*) per notificare l'utente che la sua mail è stata verificata;
- Creare una nuova classe di test (*VerificaMailTest*) per effettuare il testing funzionale che riguarda la verifica della mail.

2.2 Candidate Impact Set (CIS)

Dopo un'attenta analisi del database, Requirement Analysis Document (per i sequence diagram), Object Design Document e sulla base del SIS, presumiamo di modificare le componenti mostrate nella connectivity matrix 2. Per una maggiore leggibilità della matrice abbiamo assegnato i seguenti nomi alla componenti:

- *SocialNotes_Schema.sql* è denominata con SQL_1;
- *UserBean.java* è denominata con $Java_1$;
- *UserModelDS.java* è denominata con $Java_2$;
- *SignupControl.java* è denominata con $Java_3$;
- *Login.java* è denominata con $Java_4$;
- *VerificaMail.java* è denominata con $Java_5$;
- *login.jsp* è denominata con JSP_1;
- *mailVerificata.jsp* è denominata con JSP_2;
- *ChangeProfile* è denominata con $Java_6$.

Inoltre, il CIS include i file di test *UserModelDSTest.java*, *LoginTest.java*, *RegistrazioneTest.java*, *ModificaEmailTest.java* e *VerificaMailTest.java*.

Table 2: Connectivity matrix per CR2

	SQL_1	$Java_1$	$Java_2$	$Java_3$	$Java_4$	$Java_5$	JSP_1	JSP_2	$Java_6$
SQL_1	-	1	1	2	2	-	3	-	2
$Java_1$	1	-	1	2	2	-	3	-	2
$Java_2$	1	1	-	1	1	-	2	-	1
$Java_3$	2	1	1	-	-	-	-	-	-
$Java_4$	2	1	1	-	-	-	1	-	-
$Java_5$	2	1	1	-	-	-	-	1	-

2.3 Actual Impact Set (AIS)

Dopo aver applicato le opportune modifiche al sistema, le componenti impattate sono:

- *SocialNotes_Schema.sql*;
- *UserBean.java*;
- *UserModelDS.java*;
- *SignupControl.java*;
- *Login.java*;
- *VerificaMail.java*;
- *login.jsp*;
- *mailVerificata.jsp*;

- *ChangeProfile.java*;
- *UserModelDSTest.java*;
- *LoginTest.java*;
- *RegistrazioneTest.java*;
- *ModificaEmailTest.java*;
- *VerificaMailTest.java*;
- *SendEmail.java*;
- *Success.jsp.*
- *UtenteExpected.xml*;
- *UtenteExpectedVerificato.xml*;
- *Utente.xml*;
- *Utente.sql.*

2.4 False Positive Impact Set (FPIS)

Durante la modifica non sono stati riscontrati falsi positivi.

2.5 Discovered Impact Set (DIS)

Durante le modifiche apportate al sistema sono state individuate queste altre componenti da modificare ma non previste nel CIS:

- *SendEmail.java*;
- *Success.jsp*;
- *UtenteExpected.xml*;
- *UtenteExpectedVerificato.xml*;
- *Utente.xml*;
- *Utente.sql.*

2.6 Metriche Impact Analysis

A seguito delle operazioni di modifica, sono state calcolate Recall, Precision, Adequacy ed Effectiveness, al fine di valutare la bontà dell'impact analysis effettuato:
Recall : $|CIS \cap AIS| \div |AIS| = 14 \div 20 = 70\%$;
Precision: $|CIS \cap AIS| \div |CIS| = 14 \div 14 = 100\%$;
Inclusiveness = 0

2.7 Descrizione della modifica

Per poter implementare la change request 2 abbiamo modificato i seguenti aspetti del sistema: schema del database, funzionalità di registrazione e di login. Quindi, prima di applicare le varie modifiche abbiamo prodotto i seguenti sequence diagram e navigational path:

Sequence diagram verifica mail

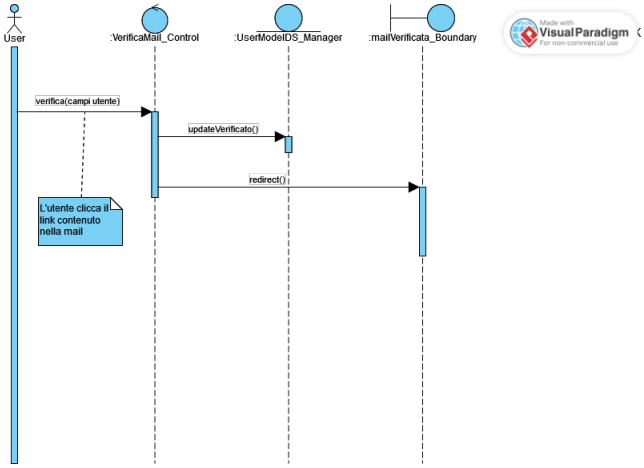
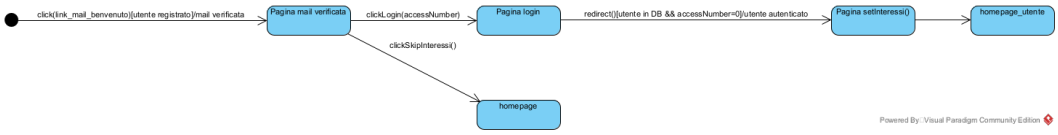


Figure 3: S.D. verifica mail

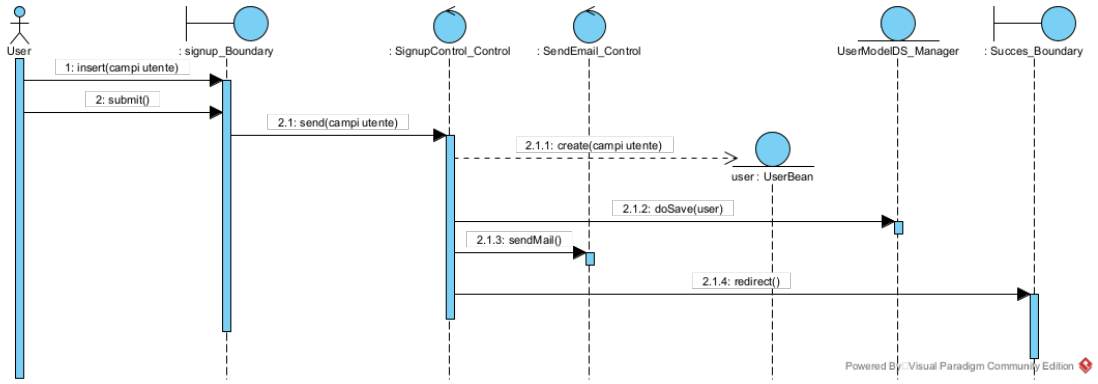
La figura seguente, invece, mostra il relativo navigational path.



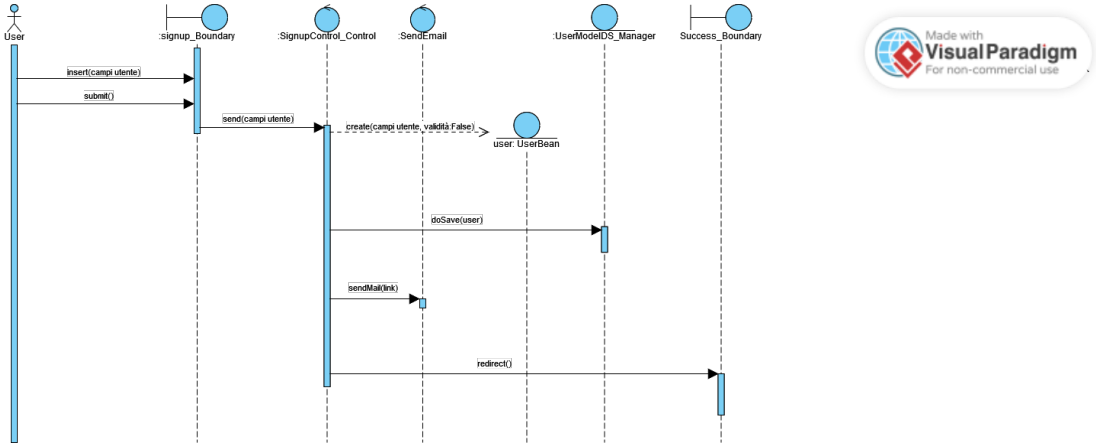
(a) N.P. verifica mail (1° accesso)

(b) N.P. verifica cambio mail

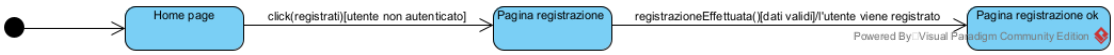
Diagrammi registrazione:



(a) S.D. registrazione (prima)

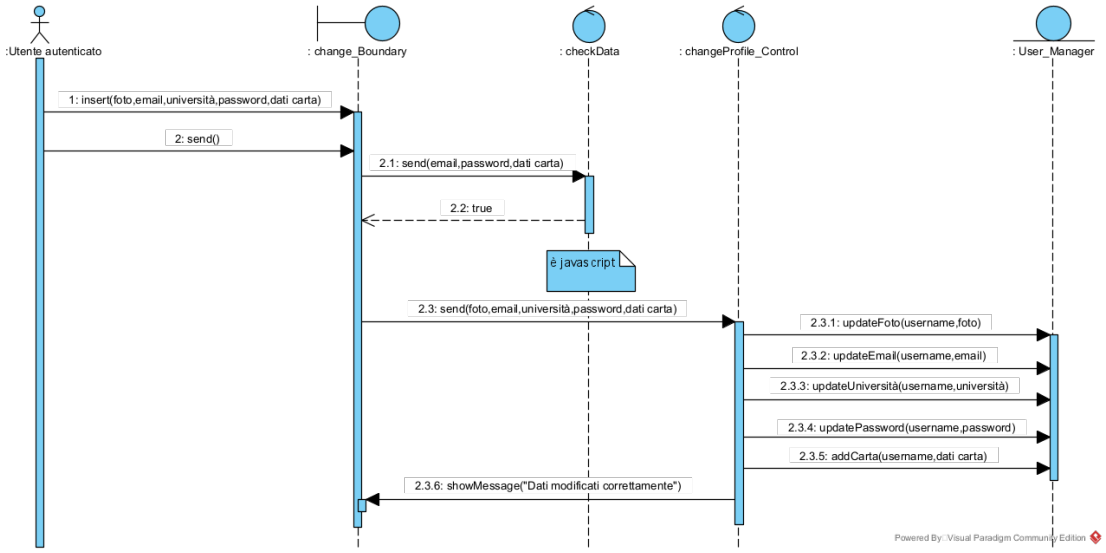


(b) S.D. registrazione (dopo)

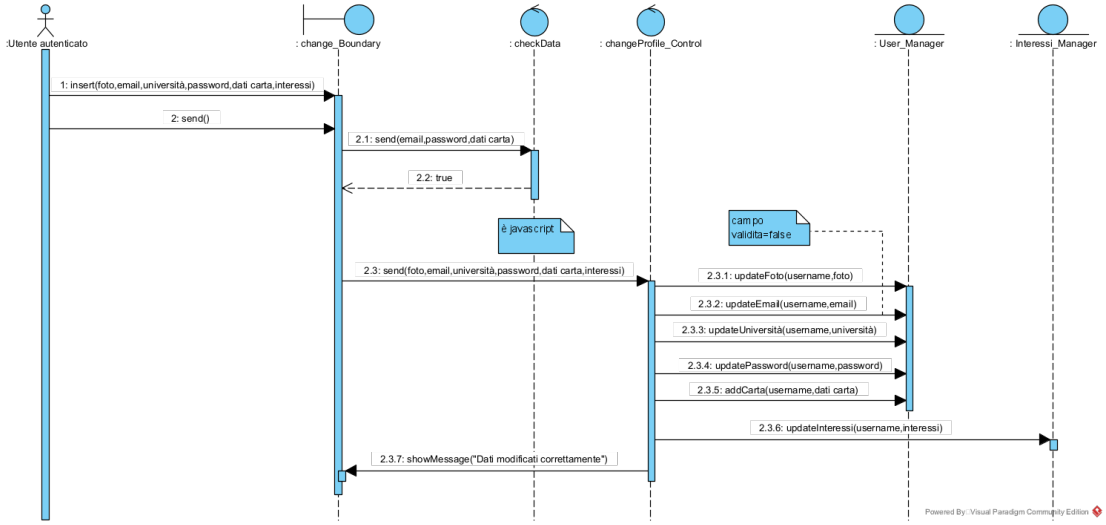


(c) N.P. registrazione (nuova)

Sequence diagram modifica dati personali

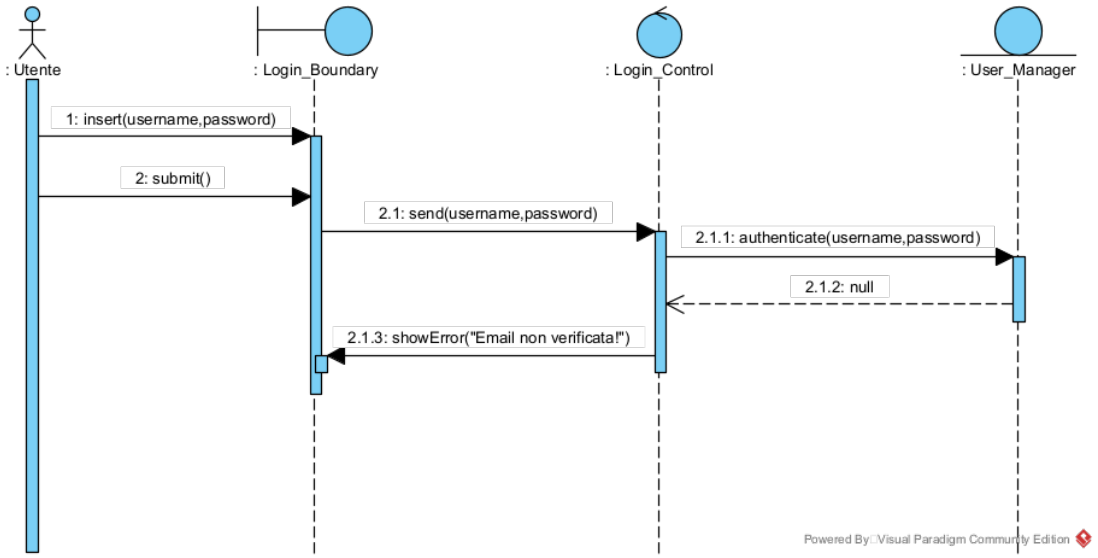


(a) S.D. modifica dati personali (prima)

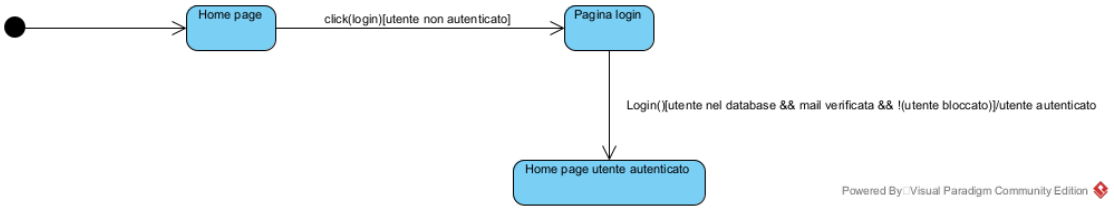


(b) S.D. modifica dati personali (dopo)

Come indicato precedentemente, abbiamo modificato anche la funzionalità di login, in quanto l'utente non può autenticarsi senza aver prima verificato la mail. Quindi abbiamo definito i seguenti sequence diagram e navigational path:



(a) S.D. Login fallita per mail non verificata



(b) N.P. login

Il navigational path iniziale per la registrazione e il login era il seguente:

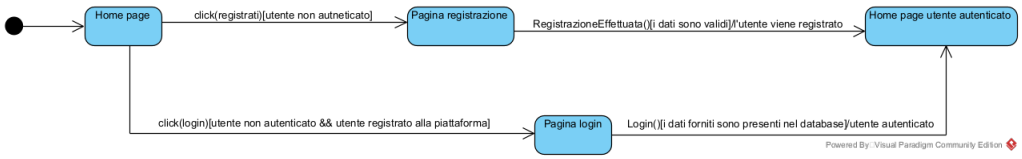


Figure 8: N.P. registrazione e login (prima)

Una volta sviluppato i sequence diagram e i navigational path, siamo passati alla modifica del software. Per poter applicare questa change request abbiamo modificato lo schema del database aggiungendo il campo *verificato* all'entità *Utente*. La query eseguita per aggiungere questo campo è la seguente:

```
ALTER TABLE utente
ADD COLUMN Verificato BOOLEAN NOT NULL DEFAULT FALSE;
```

Una volta modificato il database, abbiamo modificato allo stesso modo il bean che rappresenta l'entità utente. Come riportato nel Test plan di unità, abbiamo applicato il *Test Driven Development* che consiste nel sviluppare prima i casi di test e successivamente re-

alizzare o modificare la componente. Quindi, sulla base delle direttive di quest'approccio, abbiamo sviluppato prima i test di unità per la classe *UserModelDS* in quanto bisognava:

- modificare il metodo *doSave()* per poter salvare l'utente con lo stato verificato a false;
- aggiungere il metodo *getVerificato()* che consente di ottenere lo stato verificato dell'utente;
- aggiungere il metodo *doUpdateVerificato()* per poter aggiornare lo stato dell'attributo verificato.

Nell'andare a realizzare i casi di test abbiamo scoperto di dover modificare e creare i file XML affinchè potessimo verificare l'esito dei test. Dopo aver sviluppato i test, essi ovviamente fallivano in quanto le modifiche non erano state ancora apportate (red phase). Una volta modificato la componente, abbiamo subito eseguito i test precedentemente definiti e verificato il loro corretto funzionamento (green phase). Una volta modificato e testato la classe UserModelDS, siamo passati a modificare la servlet *SignupControl* che è responsabile della funzionalità di registrazione dell'utente. Abbiamo modificato questa classe per eliminare il codice che autenticava l'utente dopo la registrazione, per poter inviare una mail di verifica alla mail indicata dall'utente e per poter memorizzare correttamente l'utente nel database. Inoltre, abbiamo modificato la classe *SendEmail* in quanto è responsabile dell'invio delle mail e abbiamo dovuto modificare il token associato alla mail del sito in modo da poter inviare in modo automatico una mail. Infine, abbiamo modificato la pagina *Success.jsp* la quale mostra il messaggio di successo dopo la registrazione. La modifica è stata necessaria perchè era presente un errore nel controllo degli accessi, quindi abbiamo inserito un controllo in modo tale che se l'utente è già autenticato non può accedere a questa pagina. Dopodichè abbiamo modificato la servlet *ChangeProfile* in quanto è responsabile della modifica dei dati dell'utente e quindi abbiamo modificato la funzionalità della modifica della mail. Questa modifica prevedeva l'aggiornamento della mail dell'utente, l'invio di una mail per verificare la nuova mail, trasformare lo stato verificato a false ed effettuare il logout dell'utente. Inoltre, abbiamo modificato la servlet *Login* in modo da controllare lo stato verificato dell'utente che sta effettuando l'autenticazione. Questa modifica prevedeva di autenticare l'utente solo se lo stato è true altrimenti viene mostrato un messaggio d'errore. Come ultima modifica al sistema, abbiamo creato una nuova servlet, ovvero *VerificaMail* che viene invocata quando l'utente clicca il link di verifica incluso nella mail di benvenuto o di modifica mail ed abbiamo creato la pagina *mailVerificata* che mostra il messaggio di avvenuta verifica della mail. Il compito di questa classe è quello di aggiornare lo stato verificato dell'utente e inoltrare un parametro alla servlet di login in modo da stabilire se è il primo accesso dell'utente. Questo servirà per mostrare o meno la pagina di inserimento interessi per la change request 5. Dopo aver modificato queste componenti del sistema, abbiamo modificato le relative classi di test tramite Selenium per poter verificare il loro corretto funzionamento.

3 Migrazione salvataggio dei file pdf caricati sul sito

3.1 Starting Impact Set (SIS)

Le componenti da modificare per apportare questo miglioramento al sistema sono:

- Eliminazione dell'associazione 1:1 tra l'entità **materiale** e l'entità *file* del database (*SocialNotes_Schema*) ed aggiungere un campo stringa (*nomeFile*) all'entità *materiale* che rappresenta il nome del file nella cartella del server in cui memorizziamo i file;
- Classe *MaterialBean* per la sostituzione di *idFile* con *nomeFile* ed eliminazione dell'attributo *anteprima*;
- Classe *MaterialModelDS* per la sostituzione di *idFile* con *nomeFile* e l'eliminazione di *anteprima*;
- Servlet *FileUploadServlet* che si occupa della gestione del caricamento del materiale;
- Pagina *documentPreview* che si occupa della visualizzazione dell'anteprima del materiale;
- Servlet *DownloadZip* per la rimozione del codice relativo al retrieve del file dal database e sostituirlo con il retrieve del file dalla cartella del server;
- Pagina *viewFile* per la rimozione del codice relativo al retrieve del file dal database e sostituirlo con il retrieve del file dalla cartella del server;
- Classe *MaterialModelDSTest* per il testing di unità della classe MaterialModelDS;
- Classe *CaricaMaterialeTest* per il testing funzionale, in cui si verifica la funzionalità di caricamento materiale.

Inoltre comporta la definizione di un path di salvataggio sul server locale dei file relativi agli appunti utilizzate sulla piattaforma.

3.2 Candidate Impact Set (CIS)

Dopo un'attenta analisi del database, Requirement Analysis Document (per i sequence diagram), Object Design Document e sulla base del SIS, presumiamo di modificare le componenti mostrate nella connectivity matrix 3. Per una maggiore leggibilità della matrice abbiamo assegnato i seguenti nomi alla componenti:

- *SocialNotes_Schema.sql* corrisponde a SQL_1;
- *MaterialBean.java* corrisponde a $Java_1$;
- *MaterialModelDS.java* corrisponde a $Java_2$;

- *FileUploadServlet.java* corrisponde a $Java_3$;
- *DownloadZip.java* corrisponde a $Java_4$;
- *viewFile.jsp* corrisponde a JSP_1;
- *documentPreview.jsp* corrisponde a JSP_2;
- *homepageNotesManager.jsp* corrisponde a JSP_3;
- *PrintAnteprima.java* corrisponde a $Java_5$;
- *homepage_user.jsp* corrisponde a JSP_4.

Inoltre, il CIS include il file per il test di unità *MaterialModelDSTest.java* e il file per il test funzionale *CaricaMaterialeTest.java*

	SQL_1	$Java_1$	$Java_2$	$Java_3$	$Java_4$	JSP_1	JSP_2	JSP_3	$Java_5$	JSP_4
SQL_1	-	1	1	2	2	3	3	3	2	2
$Java_1$	1	-	1	2	2	3	1	1	1	1
$Java_2$	1	1	-	1	1	2	1	1	1	1
$Java_3$	2	1	1	-	-	-	-	-	-	-
$Java_4$	2	1	1	-	-	-	-	-	-	-
JSP_1	2	1	1	-	-	-	-	-	-	-
JSP_2	2	1	1	-	-	-	-	-	-	-
JSP_3	-	-	-	-	-	-	-	-	-	-
$Java_5$	2	1	1	-	-	1	1	1	-	1
JSP_4	-	-	-	-	-	-	-	-	-	-

Table 3: Connectivity Matrics CR3

3.3 Actual Impact Set (AIS)

Dopo aver apportato le opportune modifiche al sistema le componenti impattate sono:

- *SocialNotes_Schema.sql*;
- *MaterialBean.java*;
- *MaterialModelDS.java*;
- *FileUploadServlet.java*;
- *DownloadZip.java*;
- *viewFile.jsp*;
- *documentPreview.jsp*;

- *homepageNotesManager.jsp*;
- *PrintAnteprima.java*;
- *homepage_user.jsp.*
- *MaterialModelDSTest.java*
- *CaricaMaterialeTest.java*
- *Materiale.sql*
- *Materiale.xml*
- *MaterialeExpected.xml*
- *MaterialeExpectedUpdate.xml*
- *MaterialeExpectedDelete.xml*
- *ReportMaterial.java*
- *report.jsp*

3.4 False Positive Impact Set (FPIS)

Durante le modifiche non sono stati riscontrati falsi positivi.

3.5 Discovered Impact Set (DIS)

Durante le modifiche sono state individuate ulteriori componenti da modificare non previste nel CIS:

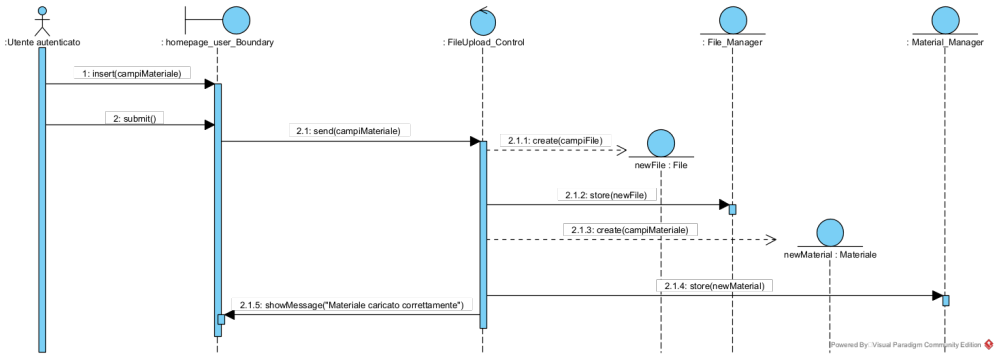
- *Materiale.sql*
- *Materiale.xml*
- *MaterialeExpected.xml*
- *MaterialeExpectedUpdate.xml*
- *MaterialeExpectedDelete.xml*
- *ReportMaterial.java*
- *report.jsp*

3.6 Metriche Impact Analysis

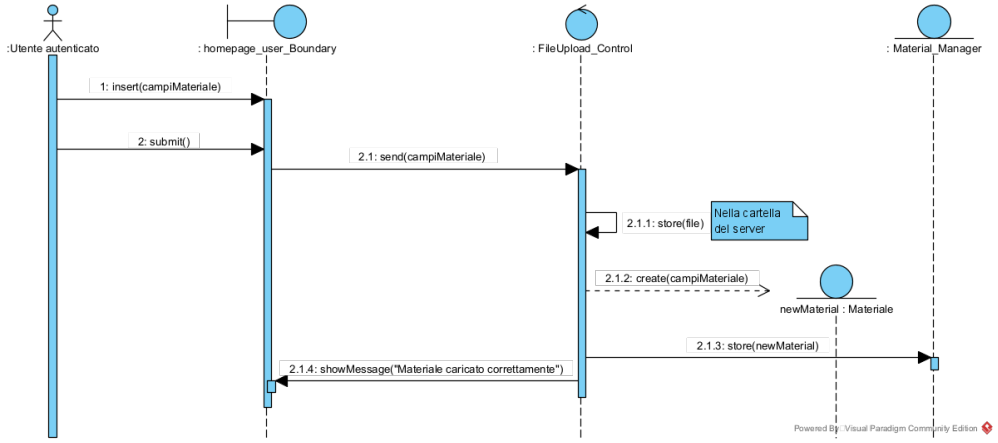
A seguito delle operazioni di modifica, sono state calcolate Precision, Recall, Adequacy ed Effectiveness, al fine di valutare la bontà dell'impact analysis effettuato. Sono stati ottenuti i seguenti risultati:
Recall : $|CIS \cap AIS| \div |AIS| = 12 \div 19 = 63\%$;
Precision: $|CIS \cap AIS| \div |CIS| = 12 \div 12 = 100\%$;
Inclusiveness = 0

3.7 Descrizione della modifica

Per poter implementare la change request 3 abbiamo modificato il modo in cui vengono memorizzati i file relativi agli appunti caricati sulla piattaforma, in particolare abbiamo migrato il loro salvataggio dal database ad una cartella del server. Da questa modifica abbiamo ottenuto i seguenti sequence diagram:

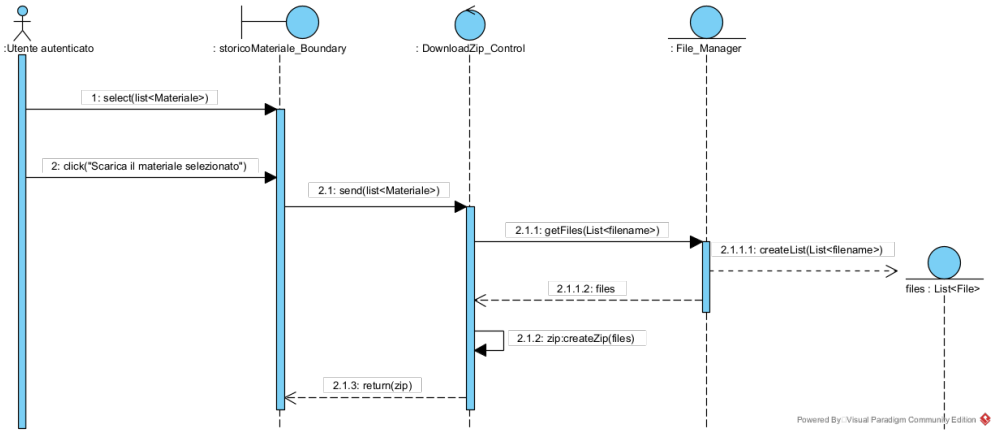


(a) S.D. caricamento materiale prima

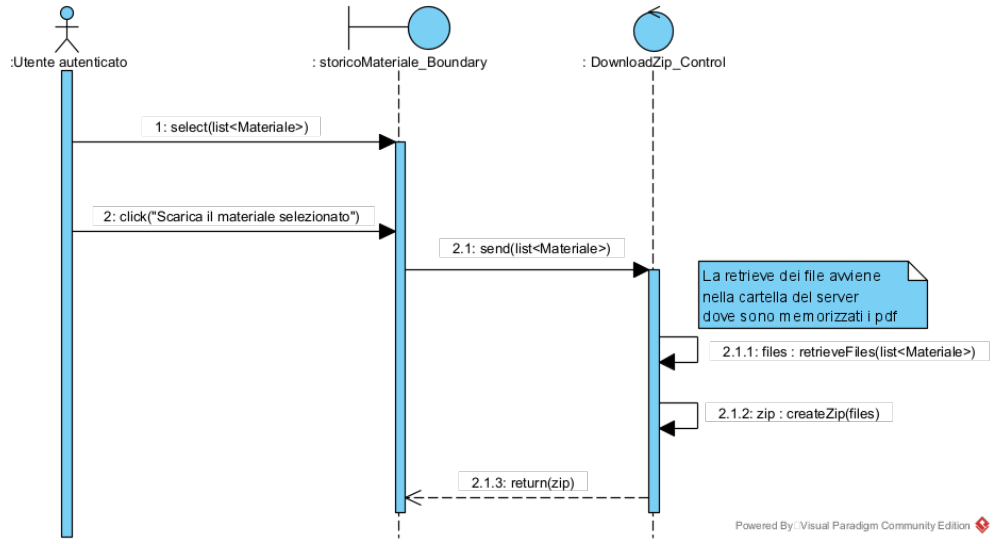


(b) S.D. caricamento materiale dopo

Figure 9: S.D. caricamento materiale prima e dopo la modifica



(a) S.D. download materiale prima



(b) S.D. download materiale dopo

Figure 10: S.D. download materiale prima e dopo la modifica

Una volta ridefiniti i sequence diagram riportati in precedenza, siamo passati alla modifica del software. Per poter sviluppare questa change request abbiamo apportato le seguenti modifiche al database:

- É stata rimossa l'associazione 1:1 tra le entità *File* e *Materiale* e di conseguenza è stata rimossa la chiave esterna dell'entità *File* (*IdFile*) dall'entità *Materiale*;
- É stato eliminato l'attributo *Anteprima* dall'entità materiale utilizzando la seguente query:

```
ALTER TABLE Materiale
DROP COLUMN Anteprima;
```

- É stato aggiunto l'attributo *FileName* all'entità *Materiale* attraverso la seguente query:

```
ALTER TABLE Materiale
ADD COLUMN NomeFile VARCHAR(255) DEFAULT NULL;
```

Una volta modificato il database, abbiamo modificato allo stesso modo il bean che rappresenta l'entità *Materiale*. Come riportato nel test plan di unità, abbiamo applicato il *Test Driven Development*. Sulla base delle direttive di questo approccio di sviluppo abbiamo sviluppato prima i test di unità per la classe *MaterialModelDS* in quanto bisognava:

- Modificare il metodo *doSave()* per poter salvare correttamente un nuovo materiale, non salvando più il file pdf direttamente nel database ma in una cartella del server e mantenendo nella riga associata al materiale soltanto il riferimento al file;
- Modificare i metodi *doRetrieveByKey(String code)*, *notValidated()*, *doRetrieveByString(String str)*, *doRetrieveByUsername(String username)*, *doRetrieveByParameters(String str,String ratingOrder, int rating)* e *doRetrieveByOrderDate()* per gestire correttamente il retrieve del materiale andando a rispecchiare le modifiche apportate al bean.

Nell'andare a sviluppare i test case di unità abbiamo scoperto di dover modificare il file sql che contiene la struttura della tabella i file XML che contengono l'init della tabella al fine di verificare l'esito dei test. Dopo aver sviluppato i test essi ovviamente fallivano in quanto le modifiche non erano ancora state apportate al software. Dopodiché abbiamo modificato la componente e abbiamo rieseguito i test precedentemente definiti verificandone il loro corretto funzionamento. Una volta modificata e testata la classe MaterialModelDS, siamo passati a modificare la servlet *FileUploadServlet* che è responsabile dell'aggiunta di un nuovo materiale. Abbiamo modificato questa classe per rimuovere il vecchio codice che si occupava del salvataggio di un nuovo materiale per sostituirlo con il nuovo codice che salva il file pdf relativo al materiale sulla cartella del server e aggiunge un riferimento a tale file nella riga della tabella relativa a quel materiale. É stata modificata la jsp *homepage_user* per la rimozione dell'input anteprima dal form di inserimento di nuovo materiale. Successivamente abbiamo modificato la servlet *PrintAnteprima* per gestire correttamente la visualizzazione dell'anteprima di un materiale, questa classe è stata modificata perché è stato rimosso l'attributo *Anteprima* dall'entità *Materiale* ed è stata utilizzata la libreria *PDFBox* di *Apache* per estrarre la prima pagina del file pdf associato a quel materiale e visualizzarla come un immagine. Quindi la modifica a questa classe è consistita nella rimozione del codice per il retrieve del file di anteprima del materiale e nella sostituzione di tale codice con il codice per estrarre la prima pagina del file pdf e visualizzarla come anteprima. In seguito sono state modificate le jsp *documentPreview* e *report* per la rimozione del codice relativo alla retrieve del file associato al materiale per stamparne il nome, la jsp *homepageNotesManager* per passare il corretto parametro alla jsp *viewFile*. La jsp *viewFile* è stata a sua volta modificata in quanto

il suo compito è quello di visualizzare il file pdf associato al materiale per permettere al notes manager di verificarne l'idoneità del materiale, di conseguenza abbiamo dovuro sostituire il codice della retieve del file dal database con quello relativo alla retrieve del file dalla cartella del server dove sono attualmente memorizzati i file. Successivamente è stata modificata la servlet *ReportMaterial* per gestire la corretta eliminazione del materiale nel caso in cui viene segnalato dal notes manager, in particolare è stato aggiunto il codice per eliminare il file relativo al materiale segnalato dalla cartella del server. Infine abbiamo modificato la servlet *DownloadZip* per gestire il corretto retrieve dei file da aggiungere allo zip che verrà scaricato dall'utente, il retrieve non viene più fatto dal database ma dalla cartella del server che contiene i file. Dopo aver modificato queste componenti del sistema, abbiamo modificato le relative classi di test tramite l'ausilio di Selenium per poterne verificare il loro corretto funzionamento.

4 Modifica gestione recupero password

4.1 Starting Impact Set (SIS)

Modifiche previste:

- Classe *RecoveryPass* il metodo *protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException* deve inviare un nuovo corpo nell' email per ripristinare la password.
- Classe *RecuperoPasswordTest* per testing della funzionalità di recupero password.

Creazioni previste:

- Classe *setNewPassword* per far si che l'utente posso scegliere la nuova password da utilizzare;
- Pagina *setNewPassword* per poter inserire la nuova password.

Eliminazioni previste:

- Classe *RandomString* non serve più in quanto non generiamo più in modo casuale la nuova password.

4.2 Candidate Impact Set (CIS)

Dopo un'attenta analisi del database, Requirement Analysis Document (per i sequence diagram), Object Design Document e sulla base del SIS, presumiamo di modificare le componenti mostrate nella connectivity matrix 4. Per una maggiore leggibilità della matrice abbiamo assegnato i seguenti nomi alla componenti:

- *RecoveryPass.java* corrisponde a $Java_1$
- *Random_String.java* corrisponde a $Java_2$

- *SetNewPassword.java* corrisponde a $Java_3$
- *setNewPassword.jsp* corrisponde a JSP_1

Inoltre, il CIS include il file per test di unità RecuperoPasswordTest.java

Table 4: Distance based approach per identificare il Candidate Impact Set

	$Java_1$	$Java_2$	$Java_3$	JSP_1
$Java_1$	-	1	-	-
$Java_2$	1	-	-	-
$Java_3$	-	-	-	1
JSP_1	-	-	1	-

4.3 Actual Impact Set (AIS)

Dopo aver applicato le opportune modifiche al sistema, le componenti impattate sono:

- *RecoveryPass.java*;
- *SetNewPassword.java*
- *setNewPassword.jsp*;
- *RandomString.java*;
- *RecuperoPasswordTest.java*;
- *RecoveryPassword.jsp*.

4.4 False Positive Impact Set (FPIS)

Durante la modifica non sono stati riscontrati falsi positivi.

4.5 Discovered Impact Set (DIS)

Durante le modifiche apportate al sistema si è notato di dover modificare la pagina *RecoveryPassword.jsp*.

4.6 Metriche Impcat Analysis

A seguito delle operazioni di modifica, sono state calcolate Recall, Precision, Adequacy ed Effectiveness, al fine di valutare la bontà dell'impact analysis effettuato:
Recall : $|CIS \cap AIS| \div |AIS| = 5 \div 6 = 83\%$;
Precision: $|CIS \cap AIS| \div |CIS| = 5 \div 5 = 100\%$;
Inclusiveness = 0

4.7 Descrizione della modifica

Per poter implementare la change request 4 abbiamo modificato la funzionalità di recupero password del sistema, quindi abbiamo ottenuto i seguenti sequence diagram e navigational path:

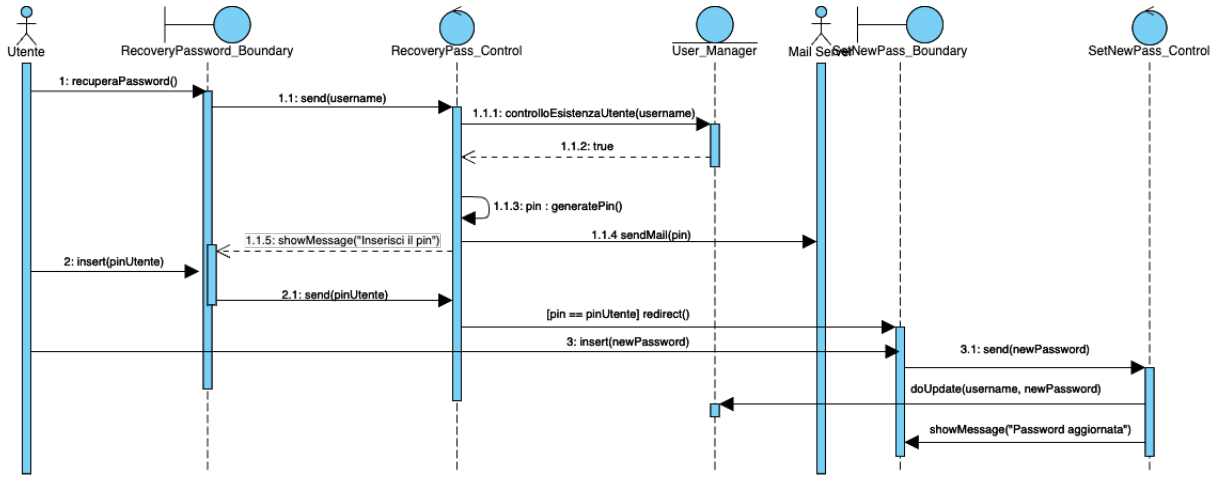


Figure 11: S.D. Recupero password

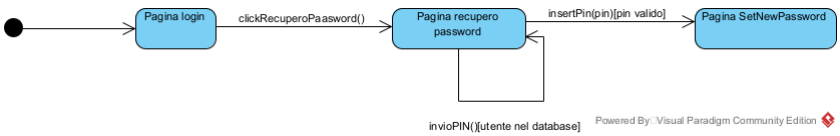


Figure 12: N.P. Recupero password

Che andranno a sostituire i vecchi diagrammi:

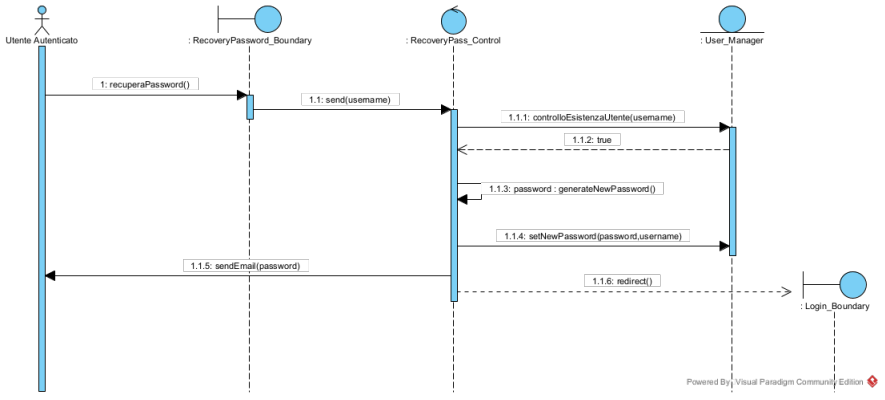


Figure 13: S.D. Recupero password precedente

Figure 14: N.P. Recupero password precedente

Una volta sviluppato i sequence diagram e i navigational path, siamo passati alla modifica del software. Per poter applicare questa change request abbiamo modificato la servlet *RecoveryPass* in quanto è responsabile del recupero password. In particolare, le modifiche riguardavano l'aggiornamento del token associata alla mail del sito per poter inviare automaticamente il pin all'utente tramite mail, gestione delle eccezioni in caso di errore, eliminazione della porzione di codice che generava automaticamente una nuova password casuale per l'utente se il pin inserito è corretto (in questo modo l'utente non aveva la possibilità di scegliere la nuova password) e direzionare l'utente nella nuova JSP per inserire la nuova password se il pin che ha inserito è corretto. Di conseguenza, abbiamo eliminato la classe *RandomString* in quanto veniva utilizzata per la generazione della nuova password e che ora non è più utilizzata ed abbiamo modificato la pagina *RecoveryPassword.jsp* per mostrare i messaggi di errore. Dopodichè abbiamo creato due nuovi file, ovvero *setNewPassword.jsp* per consentire all'utente di definire la nuova password e la servlet *SetNewPassword* per aggiornare la password dell'utente con quella indicata nel form della JSP precedentemente citata e direzionarlo alla pagina di login. Infine, abbiamo modificato e generato nuovi casi di test per la classe *RecuperoPasswordTest* tramite Selenium in modo da verificare il corretto funzionamento delle componenti modificate e create.

5 Suggerire materiale in base agli interessi degli utenti

5.1 Starting Impact Set (SIS)

Componenti da modificare:

- Classe *CourseBean* per aggiungere un campo stringa che rappresenta la chiave esterna che fa riferimento al dipartimento di appartenenza.
- Classe *CourseModelDS* per modificare il metodo *doSave* in modo da memorizzare correttamente il corso (per aggiungere il campo *dipartimento*);
- Pagina *homepage_user* per aggiungere una sezione in cui vengono mostrati i materiali consigliati;
- Modificare lo schema del database per aggiungere una relazione 1:N tra l'entità *dipartimento* e l'entità *corso* per memorizzare l'appartenenza del corso;
- Modificare lo schema del database per aggiungere una relazione N:N tra l'entità *studente* e l'entità *corso* in modo da rappresentare gli interessi degli utenti;
- Classe *ChangeProfile* per modificare gli interessi dell'utente;

- Pagina *change* per dare la possibilità all'utente di modificare i suoi interessi in futuro;
- Classe *CourseModelDSTest* perl il testing di unità della classe CourseModelDS;
- Classe Login per consentire all'utente di inserire i propri interessi solo al suo primo accesso;
- Classe *LoginTest* per il testing funzionale, in cui dopo la prima autenticazione l'utente può inserire i propri interessi;
- Classe *CaricaMaterialeTest* per il testing funzionale nel caso l'utente carica un materiale per un corso non esistente nel database.

Componenti da creare:

- Creare un bean (*InteresseBean*) che rappresenti la relazione N:N tra l'entità *studente* e l'entità *corso*;
- Creare un DAO (*InteresseModelDS*) che permette di effettuare le operazioni CRUD sulla tabella *Interesse*;
- Creare una pagina JSP (*setInteressi.jsp*) che permette all'utente di inserire i suoi interessi. L'utente visiterà questa pagina subito dopo aver verificato la sua mail;
- Creare una Servlet (*SetInteressiServlet.java*) che consente di memorizzare gli interessi dell'utente indicati nella pagina *setPreference.jsp*;
- Creare una classe di test (*InteresseModelDSTest*) per effettuare il test unitario di InteresseModelDS;
- Creare una classe di test (*InserimentoInteressiTest*) per effettuare il testing funzionale per quanto riguarda l'inserimento degli interessi;
- Creare una classe di test (*ModificaInteressiTest*) per effettuare il testing funzionale per quanto riguarda la modifica degli interessi;

5.2 Candidate Impact Set (CIS)

Dopo un'attenta analisi del database, Requirement Analysis Document (per i sequence diagram), Object Design Document e sulla base del SIS, presumiamo di modificare le componenti mostrate nella connectivity matrix 5. Per una maggiore leggibilità della matrice abbiamo assegnato i seguenti nomi alla componenti:

- *SocialNotes_Schema.sql* è denominata con SQL_1;
- *InteresseBean.java* è denominata con $Java_1$;
- *InteresseModelDS.java* è denominata con $Java_2$;

- *CourseBean.java* è denominata con $Java_3$;
- *CourseModelDS.java* è denominata con $Java_4$;
- *homepage_user.jsp* è denominata con JSP_1;
- *Login.java* è denominata con $Java_5$;
- *FileUploadServlet.java* è denominata con $Java_6$;
- *change.jsp* è denominata con JSP_2
- *setInteressi.jsp* è denominato con JSP_3;
- *SetInteressiServlet.java* è denominata con $Java_7$;
- *ChangeProfile.java* è denominata con $Java_8$.

Inoltre, il CIS include i file di test *CourseModelDSTest.java*, *LoginTest.java*, *CaricaMaterialeTest.java*, *InteresseModelDSTest.java*, *InserimentoInteressiTest.java* e *ModificaInteressiTest.java*.

Table 5: Connectivity matrix per CR4

	SQL_1	$Java_1$	$Java_2$	$Java_3$	$Java_4$	JSP_1	$Java_5$	$Java_6$	JSP_2	JSP_3	$Java_7$	$Java_8$
SQL_1	-	-	-	1	1	3	3	3	3	3	2	3
$Java_1$	-	-	-	-	-	3	2	-	3	3	1	1
$Java_2$	-	-	-	-	-	2	1	-	2	2	1	1
$Java_3$	1	-	-	-	1	3	2	2	-	3	1	-
$Java_4$	1	-	-	1	-	2	1	1	-	2	1	-
$Java_8$	2	-	-	-	-	-	-	-	-	-	-	-
JSP_1	3	-	-	3	2	-	1	-	-	-	1	-
JSP_2	3	-	-	-	-	-	-	-	-	-	-	1
JSP_3	3	3	2	3	2	-	1	-	-	-	1	-
$Java_7$	2	1	1	1	1	1	-	-	-	1	-	-
$Java_8$	3	1	1	-	-	-	-	-	1	-	-	-

5.3 Actual Impact Set (AIS)

Dopo aver applicato le opportune modifiche al sistema, le componenti impattate sono:

- *SocialNotes_Schema.sql*;
- *InteresseBean.java*;
- *InteresseModelDS.java*;
- *CourseBean.java*;
- *CourseModelDS.java*;

- *homepage_user.jsp*;
- *ChangeProfile.java*;
- *change.jsp*
- *Login.java*;
- *FileUploadServlet.java*;
- *setInteressi.jsp*;
- *SetInteressiServlet.java*;
- *CourseModelDSTest.java*;
- *CaricaMaterialeTest.java*;
- *InteresseModelDSTest.java*;
- *InserimentoInteressiTest.java*;
- *ModificaInteressiTest.java*;
- *Corso.sql*;
- *Corso.xml*;
- *CorsoExpected.xml*;
- *Materiale.sql*;
- *Materiale.xml*;
- *MaterialeExpected.xml*;
- *Interesse.sql*;
- *Interesse.xml*;
- *InteresseExpected.xml*;
- *InteresseExpectedDelete.xml.*

5.4 False Positive Impact Set (FPIS)

Durante le modifiche è stata indiviuata una componente da non modificare ma prevista nel CIS:

- *LoginTest.java*;

5.5 Discovered Impact Set (DIS)

Durante le modifiche sono state individuate ulteriori componenti da modificare non previste nel CIS:

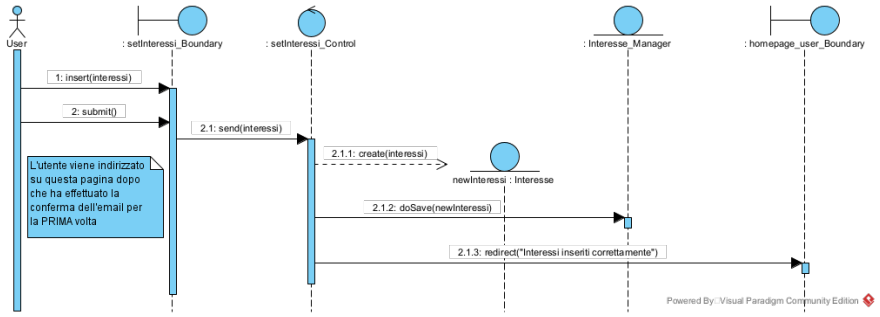
- *Corso.sql*;
- *Corso.xml*;
- *CorsoExpected.xml*;
- *Materiale.sql*;
- *Materiale.xml*;
- *MaterialeExpected.xml*;
- *Interesse.sql*;
- *Interesse.xml*;
- *InteresseExpected.xml*;
- *InteresseExpectedDelete.xml.*

5.6 Metriche Impact Analysis

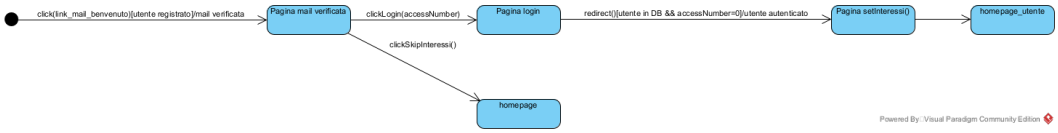
A seguito delle operazioni di modifica, sono state calcolate Recall, Precision, Adequacy ed Effectiveness, al fine di valutare la bontà dell'impact analysis effettuato:
Recall : $|CIS \cap AIS| \div |AIS| = 17 \div 27 = 63\%$;
Precision: $|CIS \cap AIS| \div |CIS| = 17 \div 18 = 94\%$;
Inclusiveness = 0

5.6.1 Descrizione della modifica

Per poter implementare la change request 5 abbiamo modificato lo schema del database, la funzionalità di modifica dati personali, di login, di caricamento del materiale e alcune JSP del sistema. Quindi, abbiamo prodotto i seguenti sequence diagram e navigational path:
Diagrammi inserimento interessi

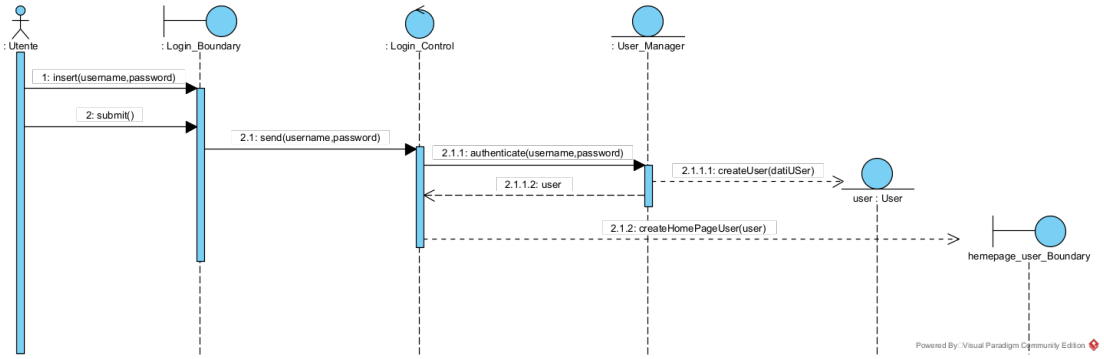


(a) S.D. inserimento interessi

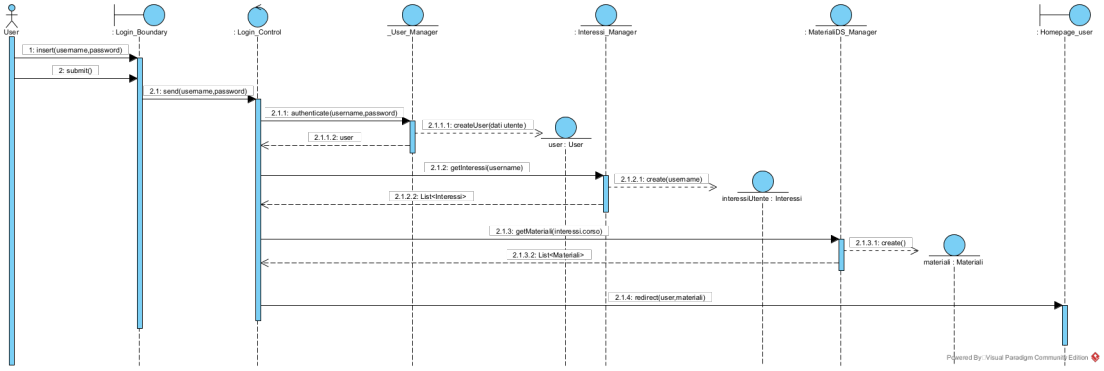


(b) N.P. inserimento interessi

Sequence diagram modifica dati personali sono mostrati nelle Figure 6a e 6b.
Sequence diagram caricamento materiale sono mostrati nella Figura 9.
Sequence diagram login



(a) S.D. login (prima)



(b) S.D. login (dopo)

Una volta ridefiniti i sequence diagram e navigational path riportati in precedenza, siamo passati alla modifica del software. Per poter sviluppare questa change request abbiamo apportato le seguenti modifiche al database:

- É stata aggiunta una relazione 1:N tra le entità *Corso* e *Dipartimento* e di conseguenza è stata aggiunta la chiave esterna di *Dipartimento* in *Corso*;
- É stata aggiunta una relazione N:N tra le entità *Utente* e *Corso*;
- É stata creata la tabella *Interesse* risultante dall'associazione N:N tra *Utente* e *Corso*.

Una volta modificato il database, abbiamo modificato allo stesso modo il bean che rappresenta l'entità *Corso* ed abbiamo creato il bean che rappresenta l'entità *Interesse*. Come riportato nel test plan di unità, abbiamo applicato il Test Driven Development. Sulla base delle direttive di questo approccio di sviluppo abbiamo modificato prima i test di unità della classe *CourseModelDS* in quanto bisognava:

- Modificare il metodo *doSave()* per poter salvare correttamente un nuovo corso, salvando oltre al nome del corso anche la chiave esterna dell'entità *Dipartimento* al quale è associato il corso;

- Modificare il metodo *doRetrieveByKey()* per gestire correttamente il retrieve del corso andando a rispecchiare le modifiche apportate al bean;
- Modificare il metodo *doRetrieveByName()* per rimuovere la logica del salvataggio di un nuovo corso dal metodo di retrieve e spostarlo nella relativa servlet che si occupa del caricamento di materiale e associazione di quest'ultimo al relativo corso, che, se non già presente nel database viene inserito;
- Implementare il metodo *doRetrieveAll()* per poter ottenere tutti i corsi disponibili nel database da visualizzare al primo accesso per consentire l'inserimento degli interessi iniziale.

Dopo aver modificato i test della classe *CourseModelDS* siamo passati allo sviluppo dei test per la classe *InteresseModelDS* in quanto bisognava:

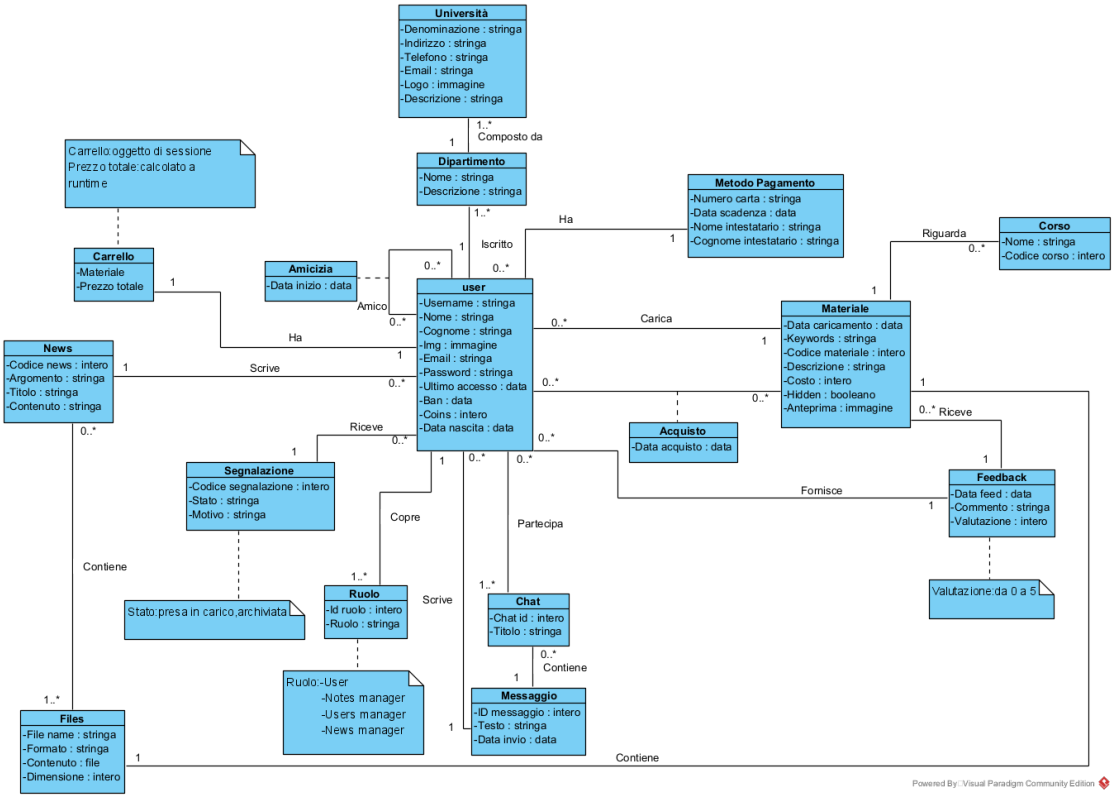
- Implementare il metodo *doSave()* per salvare un nuovo interesse;
- Implementare il metodo *doRetrieveByUsername()* per poter ottenere tutti gli interessi di un utente;
- Implementare il metodo *doRetrieveByCorso()* per poter ottenere tutti gli utenti interessati ad un dato corso;
- Implementare il metodo *doDelete()* per poter eliminare un dato interesse;
- Implementare il metodo *doRetrieveNewInteressi()* per poter ottenere i nuovi interessi che un utente può inserire;
- Implementare il metodo *getInteressi()* per poter ottenere la lista di interessi di un utente;
- Implementare il metodo *doRetrieveMaterialByInteressi()* per poter ottenere una lista di materiale di interesse per un dato utente.

Nell'andare a sviluppare i test case di unità abbiamo scoperto di dover modificare anche i file sql che contengono le strutture delle tabelle e i file XML che contengono l'init di tali tabelle al fine di verificare l'esito dei test. Dopo aver sviluppato i test essi ovviamente fallivano in quanto le modifiche non erano state ancora apportate al software. Quindi abbiamo modificato la componente *CourseModelDS* e creato la componente *InteresseModelDS* e abbiamo rieseguito i test precedentemente definiti verificandone il loro corretto funzionamento. Una volta modificata la classe *CourseModelDS*, creata la classe *InteressoModelDS* e testate entrambe le classi siamo passati allo sviluppo della jsp *setInteressi* che permette di selezionare i corsi di interesse dell'utente al primo accesso, dopo aver verificato la propria mail. Per poter salvare gli interessi selezionati nella jsp creata in precedenza abbiamo svilupparo la servlet *SetInteressi* che si occupa del salvataggio degli interessi dell'utente al primo accesso. Dopodiché siamo passati alla fase di modifica dei dati personali dell'utente e quindi anche l'inserimento/eliminazione di interessi, in

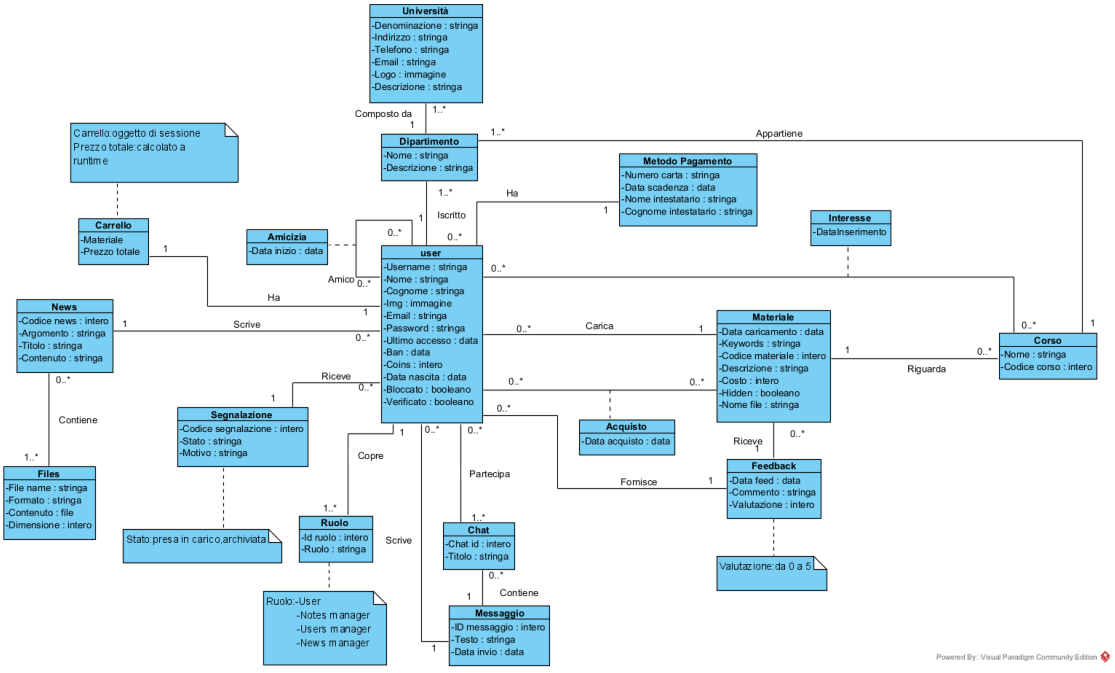
particolare abbiamo modificati la jsp *change* per aggiungere la sezione in cui si selezionano nuovi interessi o si eliminano interessi. Una volta modificata la jsp per aggiungere la sezione di modifica interessi siamo passati alla modifica della servlet *ChangeProfile* per aggiungere il codice per salvare nuovi interessi per un dato utente o eliminare interessi di un dato utente. Successivamente è stata modificata la servlet *Login* per aggiungere il controllo sull'attributo di sessione *accessNumber*, il quale, se vale 0 allora vuol dire che è il primo accesso di un utente da poco registrato il quale viene riportato sulla jsp *setInteressi* per inserire i propri interessi, altrimenti viene reindirizzato direttamente alla propria homepage. Infine è stata modificara la jsp *homepage_user* per poter visulalizzare il materiale di interesse per un utente, oltre a visualizzare il materiale caricato dai propri amici che veniva già mostrato in precedenza. Dopo aver modificato/creato queste componenti del sistema, abbiamo modificato/creato le relative classi di test tramite l'ausilio di Selenium per poterne verificare il corretto funzionamento.

6 Class diagram

Sulla base dell'impact analysis delle varie change requests, il class diagram subirà le seguenti modifiche:



(a) Class diagram (prima)



(b) Class diagram (dopo)