Impact Analysis

Gestione Prenotazioni

|  |  |
| --- | --- |
| Riferimento |  |
| Versione | 1.2 |
| Data | 22/05/2021 |
| Destinatario | Prof.re De Lucia |
| Presentato da | Aldo Claudini e Emilio Schiavo |

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | autore |
| 04/05/2021 | 1.0 | Creazione del documento con:  capitolo 1  capitolo 2  capitolo 3 | Aldo Claudini, Emilio Schiavo |
| 05/05/2021 | 1.1 | Studio di fattibilità capitolo 4 | Aldo Claudini,  Emilio Schiavo |
| 22/05/2021 | 1.2 | Definizone del actual impact set | Aldo Claudini,  Emilio Schiavo |

**Indice**

[1. Scopo del documento 3](#_Toc73185418)

[2. Panoramica del sistema attuale 3](#_Toc73185419)

[2.1 Attori 4](#_Toc73185420)

[2.2 Design e implementazione e testing del sistema attuale 4](#_Toc73185421)

[Implementazione 4](#_Toc73185422)

[Testing 4](#_Toc73185423)

[3. Analisi della modifica e impact set 5](#_Toc73185424)

[3.1 Starting Impact Set 5](#_Toc73185425)

[3.2 Candidate Impact Set 5](#_Toc73185426)

[Aggiunta del requisito “Assegna/rimuovi aula per i Tutor” 6](#_Toc73185427)

[Aggiunta del requisito “Assegna/rimuovi Tutor” 6](#_Toc73185428)

[Aggiunta del requisito “Visualizza elenco dei tutor” 6](#_Toc73185429)

[4. Studio di fattibilità 7](#_Toc73185430)

[4.1 Identificazione, descrizione e valutazione dei costi 7](#_Toc73185431)

[4.2 Identificazione, descrizione e valutazione dei benefici 8](#_Toc73185432)

[4.3 Identificazione, descrizione e valutazione dei rischi 8](#_Toc73185433)

[4.5 Risultati previsti 9](#_Toc73185434)

# Scopo del documento

In riferimento al documento “Change Request” verranno descritte tutte le nuove funzionalità implementate e verrà stilata un’analisi d’impatto dei cambiamenti sul sistema, evidenziando i rischi, i costi e i benefici.

# Panoramica del sistema attuale

“Gestione Prenotazioni” è un piattaforma di prenotazione online che, come facilmente intuibile dal nome, ha come obiettivo la prenotazione delle aule di Ateneo, in particolare di quello dell’ università degli studi di Salerno. Le funzionalità offerte all’utente, oltre alle classiche funzionalità di accesso come login e signup sono: Visualizzare il calendario , Inserire una prenotazione , amministrare la parte di prenotazione, amministrazione delle aule per gli studenti e delle infrastrutture, gestione degli amministratori di dipartimento.

## 2.1 Attori

Il sistema tre tipi d attori principali:

L’utente non iscritto riesce solo a visualizzare il calendario delle lezioni.

Lo studente iscritto che è in grado di visualizzare gli orari e le aule prenotate e può prenotare le aule che gli sono state assegnate dall’amminsitratore di dipartimento.

Il Docente che può prenotare tutte le aule a disposizione, può visualizzare le prenotazioni fatte, accedere al calendario ed eliminare una richiesta di prenotazione.  
L’amministratore di dipartimento amministra le richieste di prenotazione del proprio dipartimento e gestisce l’assegnazione delle aule studio.

L’amministratore di Ateneo gestisce gli edifici dell’Università, i dipartimenti e l’assegnazione degli amministratori di dipartimento, ha pieno controllo della piattaforma.

## 2.2 Design e implementazione e testing del sistema attuale

Il sistema attuale è stato concepito come un’architettura three-tier. Lo strato di presentation, ovvero lo strato software che si occupa di interagire con l’utente, è stato sviluppato utilizzando con JSP . Lo strato di business logic è stato sviluppato con un approccio object oriented utilizzando il linguaggio Java Lo storage, quindi la gestione dei dati persistenti, è basato su un DB MySql.

### Implementazione

L’implementazione del sistema prevede, MVC, lato server-side per il database usiamo il pattern strutturale Data Access Object per la gestione della persistenza.

### Testing

Per il sistema Gestione Prenotazioni, il testing si compone tre fasi. Nella prima fase, sono stati eseguiti i test di unità dei singoli componenti, in modo da testare nello specifico la correttezza di ciascuna unità andando a constatare il corretto funzionamento..

Nella seconda fase, è stato effettuato il testing di integrazione in cui si è andato a testare l’integrazione dei vari sottosistemi.

Infine, verrà eseguito il testing di sistema dove si è testato l’intero sistema assemblato nei suoi componenti.

Nella sessione di testing del sistema è stato utilizzato un approccio “BLACK BOX” ed una strategia top down: tale strategia prevede che i sottosistemi nel layer più in alto della gerarchia vengono testati individualmente, successivamente vengono integrati i layer più in basso. Si ripete questo passo finché tutti i sottosistemi sono testati. Tale metodologia fa uso dei test stub. Un test stub è una implementazione parziale di una componente chiamata dall’unità testata.

# Analisi della modifica e impact set

La change request effettuata prevede di migliorare i requisiti funzionali del sistema attuale e aggiungerne degli altri. Come prima cosa abbiamo preso familiarità col sistema facendo Reverse Engineering.   
Siamo quindi partiti dal codice con l’obiettivo di testare l’applicazione e analizzare le varie componenti, cercando di capirne l’intera logica di business.

## 3.1 Starting Impact Set

Analizzando la change request più approfonditamente e i dati raccolti [1] abbiamo individuato il nostro Starting Impact Set, che rappresenta l’insieme delle componenti che verranno direttamente impattate da queste modifiche, tali componenti sono facilmente intuibili anche dal testo in chiaro della change request [2]. Nella tabella seguente verranno indicate le componenti presenti all’interno del nostro Starting impact set, l’impatto della modifica verrà valutato su tre livelli.

* FORTE: nel caso in cui sia necessario riscrivere completamente l’artefatto.
* MEDIO: nel caso in cui sia necessario una rivisitazione e modifica di un artefatto.
* DEBOLE: nel caso in cui siano necessarie solo modifiche marginali.
* NULLO: nel caso che non sono presenti modifiche

|  |  |  |
| --- | --- | --- |
| Artefatto | Impatto | Descrizione |
| Requisiti funzionali | MEDIO | Aggiunta dei requisiti:   * Assegna/Rimuovi aula per i tutor * Assegna/rimuovi tutor * Visualizza elenco dei tutor   **Modifica dei requisiti:** Visualizza post degli altri [2] |
| Casi d’uso | MEDIO | Aggiunta casi d’uso:   * Assegna/Rimuovi aula per i tutor * Assegna/rimuovi tutor * Visualizza elenco dei tutor   Modifica caso d’uso:  modifica gestione utente  modifica eventi amministratore di dipartimento |
| DB | MEDIO | Aggiunta tabella corso |

## 3.2 Candidate Impact Set

In seguito a questa prima fase è stato effettuato un ulteriore lavoro di reverse engineering [1] dove sono stati esaminati i Sequence Diagram per valutare la dinamicità del sistema relativa ai requisiti di sistema e i Class Diagram relativi al nostro sistema. Questo lavoro è stato fatto per individuare le componenti impattate indirettamente da quelle prese in considerazione nel SIS descritto sopra. Partendo dal nostro SIS, abbiamo analizzato più approfonditamente il class diagram, evidenziando le varie dipendenze per poi farne un’analisi ed estrapolare le componenti che faranno parte del nostro CIS.

### Aggiunta del requisito “Assegna/rimuovi aula per i Tutor”

Di seguito prenderemo in considerazione la modifica del requisito “Assegna/Rimuovi aula per i Tutor” il quale è implementato tramite la JSP **CalendarioAula.jsp** dove sarà inserito un nuovo pulsante per la rimozione o l’ assegnazione dei permessi dei tutor.Per tanto analizzeremo quest’ultima per comprendere quali componenti dipendono da essa e quindi quali potrebbero essere realmente impattate, direttamente o indirettamente.

|  |  |  |
| --- | --- | --- |
| Classe | Impatto | N° |
| AulaStudenti.java | DIRETTO | 1 |
| CalendarioAula.jsp | DIRETTO | 2 |
| strutturaDAO.java | INDIRETTO | 1.1 |
| Struttura.java | INDIRETTO | 1.2 |

### Aggiunta del requisito “Assegna/rimuovi Tutor”

Di seguito prenderemo in considerazione l’aggiunta del nuovo requisito “Assegna/Rimuovi Tutor”, che andrà ad impattare direttamente la JSP **Header.jsp** nella quale sarà inserito un nuovo pulsante che, grazie a NavGestisciTutor reindirizza alla servlet EditTutor che avrà come compito le funzionalità di Assegnazione e Rimozione dei tutor in base al corso. Da qui bisogna creare un ulteriore tabella nel DB per i corsi, perché ogni tutor deve essere associato ad un corso.

In oltre dato che queste componenti non sono state ancora impementate non riusciamo a sapere l’impatto effettivo che potrebbe avere questa modifica, cominciamo a vedere però l’ impatto che ha sui componenti già esistenti.

Nella seguente tabella verranno mostrate le componenti che potrebbero essere impattate direttamente o indirettamente da tale modifica.

|  |  |  |
| --- | --- | --- |
| Classe | Impatto | N° |
| Header.jsp | DIRETTO | 1 |
| Areapersonale.jsp | DIRETTO | 2 |
| Homepage.jsp | DIRETTO | 3 |
| ElencoUtenti.jsp | DIRETTO | 4 |
| NAVareapersonale.java | INDIRETTO | 1.1 |
| NavRichiestePrenotazioni.java | INDIRETTO | 1.2 |
| Utente.java | INDIRETTO | 2.1 |
| AulaStudenti.java | INDIRETTO | 2.2.1 |
| UtenteDAO.java | INDIRETTO | 2.1.1 |
| ServletBasic.java | INDIRETTO | 2.1.1.1 |

### Aggiunta del requisito “Visualizza elenco dei tutor”

In questo paragrafo prendiamo in esame l’aggiunta del requisito “Visualizza elenco dei tutor”. Sarà la nuova funzionalità implementata grazie alla componente **GestisciTutor** che permetterà di visualizzare l’elenco di tutti gli studenti che possono fare i tutor, tramite la classe già esistente **Utente** visualizzeranno tutte le informazioni**.** Dato che **GestisciTutor** non è ancora implementata non possiamo vedere le probabili componenti impattate da essa, ma sappiamo che sicuramente verrà utilizzata la componente **Header.jsp**  di conseguenza andiamo a considerare nella seguente tabella le componenti che possono essere impattate da essa.

|  |  |  |
| --- | --- | --- |
| Classe | Impatto | N° |
| Header.jsp | DIRETTO | 1 |
| UtenteDAO | INDIRETTO | 1.1 |

Al temine di questa analisi siamo giunti alla conclusione che il Candidate Impact Set conterrà le seguenti componenti: |CIS|=13

|  |  |
| --- | --- |
| Classe | Impatto |
| AulaStudenti.java | DIRETTO |
| CalendarioAula.jsp | DIRETTO |
| strutturaDAO.java | INDIRETTO |
| Struttura.java | INDIRETTO |
| Header.jsp | DIRETTO |
| Areapersonale.jsp | DIRETTO |
| Homepage.jsp | DIRETTO |
| ElencoUtenti.jsp | DIRETTO |
| NAVareapersonale.java | INDIRETTO |
| NavRichiestePrenotazioni.java | INDIRETTO |
| Utente.java | INDIRETTO |
| UtenteDAO.java | INDIRETTO |
| ServletBasic.java | INDIRETTO |

# Studio di fattibilità

## Identificazione, descrizione e valutazione dei costi

|  |  |  |
| --- | --- | --- |
| Identificatore | Valutazione | Motivazione |
| Integrazione dei requisiti funzionali | FORTE | L’aggiunta dei requisiti avverrà in modo incrementale. |
| Modifica DB | MEDIA | Verrà aggiunto un nuovo attributo all’entità utente del db in questione e verrà creata una nuova tabella “corsi” inrelazione agli utenti che diventeranno tutor |
| Implementazione con Java e jsp | DEBOLE | Lo sviluppatore ha sufficienti conoscenze della tecnologia |
| Testing funzionale | MEDIA | Le componenti del sistema sono già state testare, bisogna testare le nuove funzionalità implementate insieme alle vecchie |

## Identificazione, descrizione e valutazione dei benefici

|  |  |  |
| --- | --- | --- |
| Identificatore | Valutazione | Motivazione |
| Facilità nella gestione delle aule per i tutor | FORTE | L’amministratore di dipartimento può gestire e riservare con facilità le aule solo per i tutor |
| Facilità nella gestione dei Tutor per i corsi | FORTE | L’ amministratore di dipartimento può gestire i tutor in maniera estremamente comoda e veloce tramite un interfaccia apposita. |

## Identificazione, descrizione e valutazione dei rischi

|  |  |  |
| --- | --- | --- |
| Identificatore | Valutazione | Motivazione |
| Introduzione di nuovi fault | FORTE | L’attività di testing da effettuare potrà avere un’alta probabilità di riscontrare fault, questo è dovuto alle nuove modifiche. |
| Carico di lavoro eccessivo per il client | DEBOLE | Il client dovrà solo elaborare dei dati provenienti da richieste a servizi forniti dal server |
| Carico di lavoro eccessivo per il server | DEBOLE | Al server saranno attuate lievi modifiche e la maggior parte delle azioni saranno eseguite utilizzando i metodi già esistenti che effettuano operazioni CRUD. |

## Risultati previsti

La realizzazione della modifica terrà conto di tale studio di fattibilità.

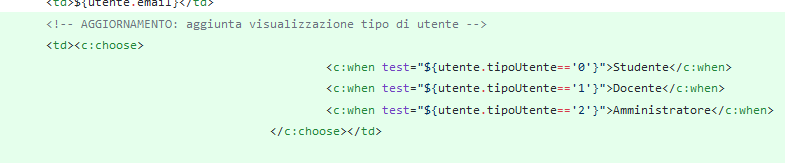
# Actual Impact set

Di seguito saranno elencante le componenti realmente modificate, tali componenti faranno parte del nostro **AIS**.

## Lato client-side

Aggiunta visualizzazione tipo utente

1. In AreaPersonale.jsp aggiunto campo tipoutente



### Aggiunta del requisito “Assegna/rimuovi aula per i tutor”

Per l’inserimento di questo nuovo requisito sono state modificate le seguenti componenti:

1. CalendarioAula.jsp inserimento del nuovo bottone aggiungi o rimuovi aula tutor.



### Aggiunta del requisito “Aggiungi/Rimuovi Tutor”

1. In Header.jsp Aggiunto pulsante “Gestisci Tutor”

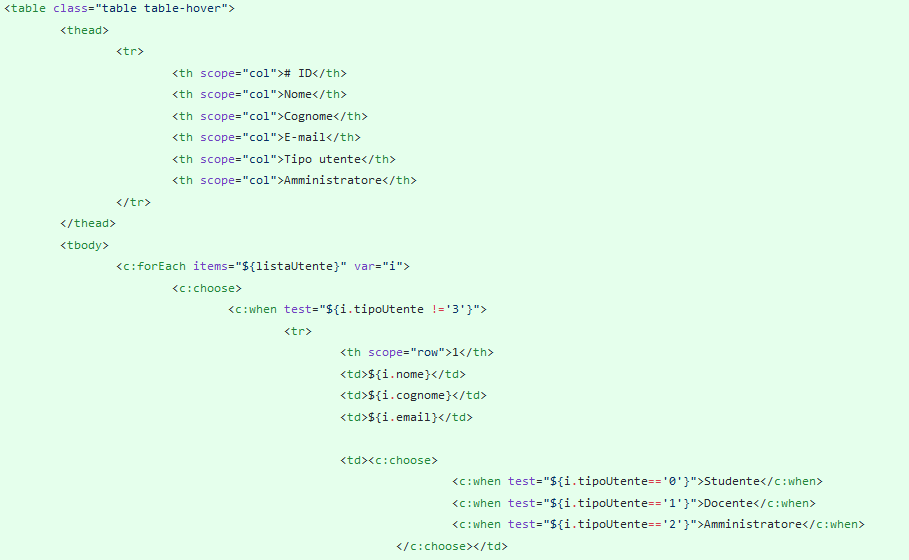


1. Creazione dell’elenco di studenti che sono tutor e non in GestisciTutor.jsp



### Aggiunta del requisito “Visualizza lista Tutor”

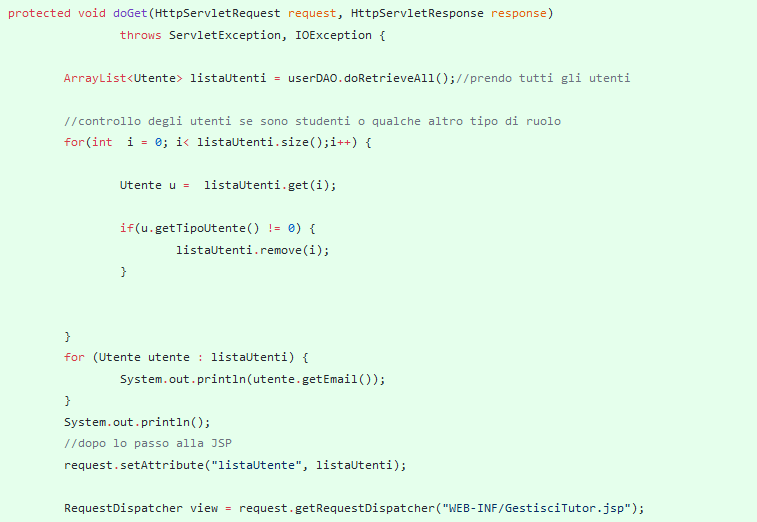
1. In GestisciTutor.jsp , implementata la lista per la visualizzazione dei tutor



## Lato server-side

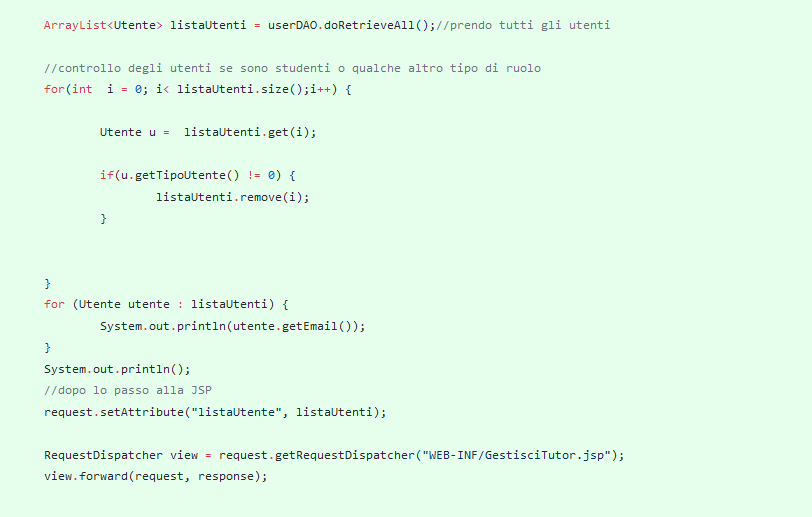
### Visualizza lista tutor

1. In NavGestisciTutor.java, creo la lista degli utenti da passare al front-end

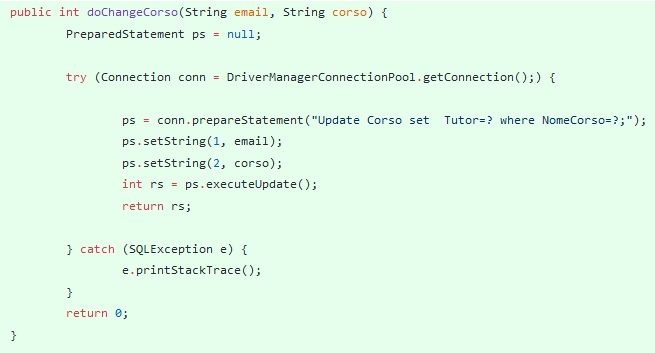


### Aggiunta del requisito “Aggiungi /rimuovi tutor ”

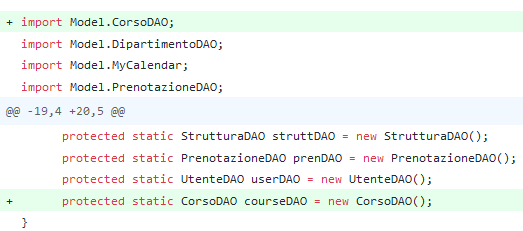
1. Creazione della servlet NavGestisciTutor.java per il reindirizzamento alla pagina

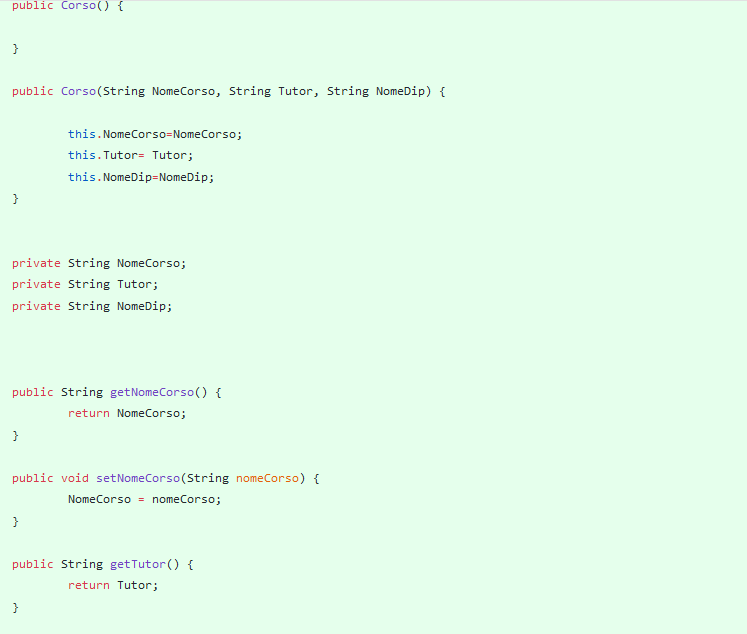
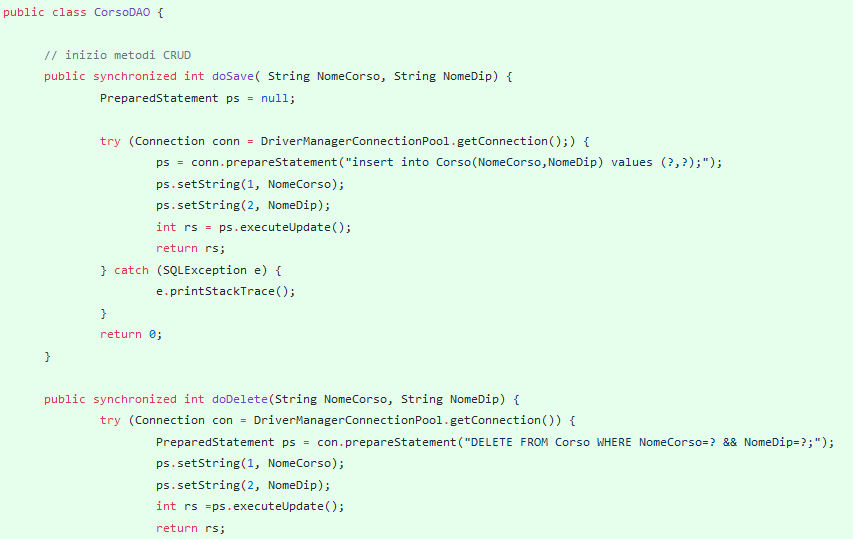


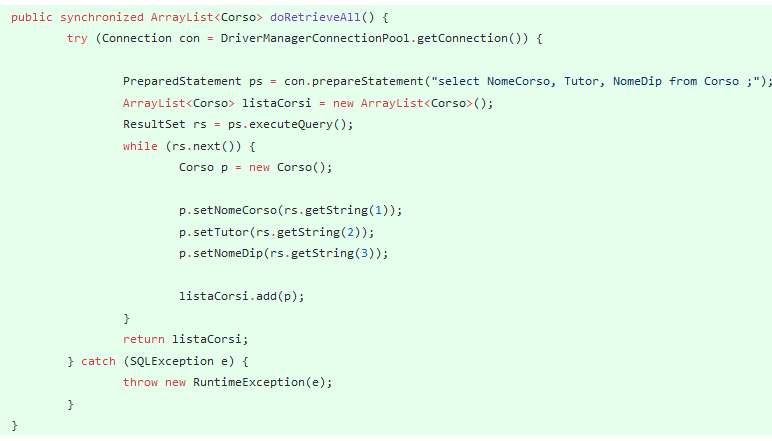
1. In UtenteDAO.java aggiunta funzione doChangeCorso



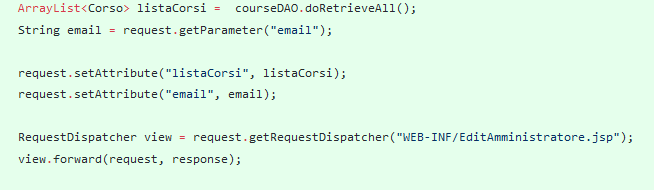
1. In ServletBasic.java Aggiunta di una nuova variabile d’ambinete courseDAO



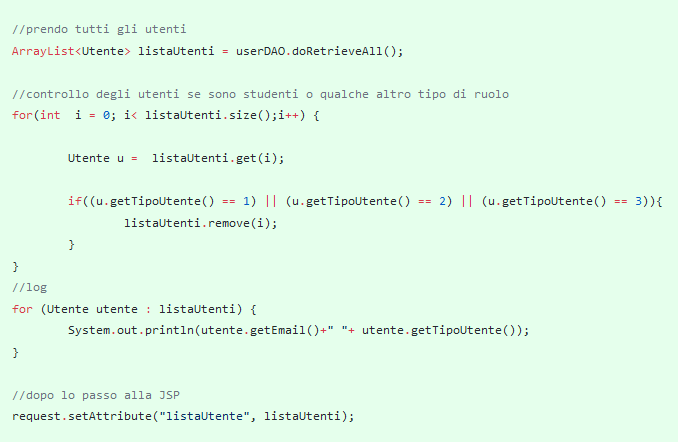
1. Creazione della classe Corso 
2. Creazione della classe CorsoDAO.java e aggiunta dei metodi CRUD 
3. In CorsoDAO. Aggiunta metodo DoretriveALL()



1. Creazione della classe EditTutor.java che assegna o rimuove il grado di tutor di dipartimento ad uno studente 
2. Creazione della classe NAvEditTutor per la reindirizzazione alla lista di tutti i corsi di ateneo

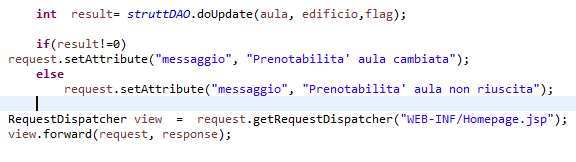


9)In DeleteTutor.java, andiamo ad eliminare il tutor



### Aggiunta del requisito “Assegna/Rimuovi aula per i tutor”

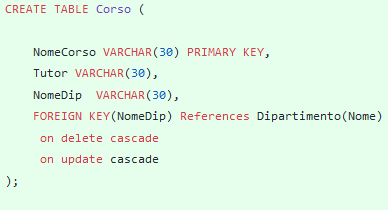
1. In AulaStudenti.java è stato usato il metodo doUpdate per salvare la modifica dell’ aula nel database



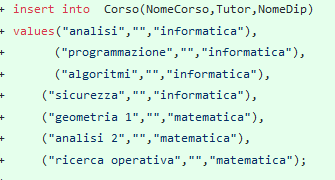
## Modifica MySQL

## 

1. Aggiunta tabella corso al database



1. Popolamento tabella corso



mostriamo la tabella delle componenti che fanno parte del nostro Actual impact set, e ne evidenzieremo quelle che fanno parte del DIS

|  |  |  |
| --- | --- | --- |
| Classe | Impatto | Azione |
| NavGestisciTutor.java | MEDIO-DIRETTO | Create |
| GestisciTutor.jas | DEBOLE-DIRETTO | Create |
| AreaPersonale.jsp | DEBOLE-DIRETTO | Update |
| CalendarioAula.jsp | DEBOLE-DIRETTO | Update |
| Header.jsp | FORTE-DIRETTO | Update |
| UtenteDAO.java | DEBOLE-INDIRETTO | Update |
| Utente | DEBOLE-INDIRETTO | Update |
| Corso.java | MEDIO-DIRETTO | Create |
| CorsoDAO.java | MEDIO-DIRETTO | Create |
| ServletBasic.java | DEBOLE-INDIRETTO | Update |
| EditTutor.java | MEDIO-DIRETTO | Create |
| NavEditTutor.java | MEDIO-DIRETTO | Create |
| EditTutor.jsp | FORTE-DIRETTO | Create |
| DeleteTutor.java | MEDIO-DIRETTO | Create |