Immagine che contiene oggetto

Descrizione generata automaticamente

ODD

Object

Design

Document

Gestione Prenotazione

**Borriello Martina 0512105144**

**Claudini Aldo 0512105486**

**Claro Andrea 0512105252**

**Marcantuono Veronica 051210542**

***SOMMARIO***

1. **Introduzione**

1.1 Object Design Trade-offs

1.2 Linee Guida per la Documentazione delle Interfacce

1.3 Definizioni, acronimi e abbreviazioni

1.4 Design Pattern

1.5 Riferimenti

1. **Packages**
2. **Class interfaces**
3. **Glossario**

**1. Introduzione**

Dopo aver stilato i documenti Requirements Analysis e System Design è necessario porre attenzione sugli aspetti implementativi. Questo documento ha l’obiettivo produrre un modello che integri in modo coerente tutte le informazioni collezionate nelle fasi precedenti. In particolar modo, in tale documento verranno definite le interfacce delle classi, le operazioni supportate, i tipi dei dati, i parametri delle procedure, i signatures dei sottosistemi definiti nel documento di System Design, i trade-offs e le linee guida.

**1.1 Object Design Trade-offs**

**Comprensibilità vs Tempo:**

Il codice del sistema deve essere comprensibile, in modo da facilitare la fase di testing ed eventuali future modifiche da apportare. Al fine di rispettare queste linee guida il codice sarà integrato da commenti volti a migliorarne la leggibilità; tuttavia questo richiederà una maggiore quantità di tempo necessario per lo sviluppo del nostro progetto.

**Interfaccia vs Usabilità:**

Verrà realizzata un’interfaccia grafica chiara e concisa, usando form e pulsanti predefiniti che hanno lo scopo di rendere semplice l’utilizzo del sistema da parte dell’utente finale.

**Sicurezza vs Efficienza:**

Il sistema mira ad essere chiuso per garantire la privacy, tuttavia ci limiteremo ad implementare sistemi di sicurezza basati su email e password a causa dei tempi limitati in modo da favorire l’efficienza.

**1.1 Linee Guida per la Documentazione delle Interfacce**

**Naming Convention:**

Alla luce di quanto analizzato, sarà necessario utilizzare nomi:

* Descrittivi;
* Pronunciabili;
* Di uso comune;
* Di lunghezza medio-corta;
* Utilizzando solo caratteri consentiti (a-z, A-Z, 0-9);
* Senza prefissi o suffissi.

**Variabili:**

* I nomi delle variabili dovranno iniziare con la lettera minuscola, e le parole successive con la lettera maiuscola.
* Le variabili locali non saranno dichiarate all’inizio del blocco che le contiene, ma verranno dichiarate in corrispondenza del punto in cui vengono usate per la prima volta: il tutto è volto alla limitazione del loro scope.
* In ogni riga dovrà esserci un’unica variabile dichiarata, eventualmente allineata con quelle del blocco dichiarativo.
* In determinati casi, è possibile utilizzare il carattere underscore “\_”: il caso principale previsto è quello relativo alla dichiarazione di costanti oppure di proprietà statiche.

**Metodi:**

* I nomi dei metodi dovranno iniziare con la lettera minuscola e le parole successive con la lettera maiuscola.
* Il nome del metodo sarà costituito da un verbo che ne identifica l’azione seguito da un sostantivo, eventualmente aggettivato.
* Il nome dei metodi accessori e modificatori seguirà, rispettivamente, i pattern getNomeVariabile e setNomeVariabile.
* Nel caso in cui vengano utilizzati costrutti “if”, “else”, “for”, “do” e “while” nei metodi dovranno essere utilizzate le parentesi graffe, anche se essi constano di una sola istruzione.
* Ai metodi verrà aggiunto un commento JavaDoc, il quale deve essere posizionata prima della dichiarazione del metodo, e deve descriverne lo scopo. Il commento dovrà comprendere anche le informazioni riguardanti i parametri, il valore di ritorno e le eccezioni lanciate.
* I metodi dovranno essere raggruppati in base alla loro funzionalità.

**Classi Java e pagine JSP:**

* I nomi delle classi e delle pagine dovranno iniziare con la lettera maiuscola, così come le parole successive all’interno del nome.
* I nomi delle classi e delle pagine dovranno corrispondere alle informazioni e le funzioni fornite da quest’ultime.
* Ogni file sorgente .class dovrà contenere una singola classe e dev’essere strutturato come segue:
* Breve introduzione alla classe; tale introduzione indica lo scopo della classe, l’autore, la versione e la data.

/\*\*

\* sommario dello scopo della classe.

\*

\* @author [nome dell’autore]

\* @version [numero di versione della classe]

\*/

* Le classi saranno strutturate prevedendo rispettivamente:

1. Dichiarazione della classe pubblica;

2. Dichiarazioni di costanti;

3. Dichiarazioni di variabili di classe;

4. Dichiarazioni di variabili d’istanza;

5. Costruttore;

6. Commento e dichiarazione metodi e variabili.

* Nel caso in cui una classe avesse costruttori o metodi multipli con lo stesso nome, questi saranno posizionati sequenzialmente senza codice posto fra essi.

**Packages:**

* I nomi dei packages dovranno essere scritti in minuscolo concatenando insieme diversi sostantivi o sigle, separate dal carattere “.” .
* Non saranno ammessi caratteri speciali.
* Import statici e non statici in saranno specificati in blocchi singoli e separati;
* Gli import statici non saranno usati per le classi annidate ma verranno importate con import tradizionali

**1.3 Definizioni, acronimi e abbreviazioni**

**Acronimi:**

RAD: Requirements Analysis Document

SDD: System Design Document

ODD: Object Design Document

Abbreviazioni;

NC: Nessuna Condizione

NA: Nessun Attributo

**1.4 Design Pattern**

**1.4.1 Design Pattern DAO (Data Access Object)**

Il DAO è un pattern architetturale finalizzato alla gestione dei dati persistenti. L’uso del DAO permette di gestire la connessione con il database per estrarre e/o immagazzinare i dati. Il DAO nasconde i dettagli implementativi dall’applicazione e tutte le complessità coinvolte nell’esecuzione delle operazioni inerenti all’accesso al database. Il suo scopo è:

Il design pattern DAO ha come scopo:

* Separare le operazioni di accesso ai dati dalle operazioni di business;
* Avere un’interfaccia comune di accesso ai dati.

Fornisce diversi vantaggi, tra cui:

* Manutenibilità
* Facilità di utilizzo
* Riuso del codice

**1.4.2 Singleton Pattern**

Si è scelto di utilizzare il Singleton Pattern che deve assicurarsi che esista una singola istanza della classe Singleton, che nel nostro caso si chiama *DriverManagerConnectionPool*, e fornire un accesso globale a tale istanza. L’istanza viene dichiarata come variabile privata statica e creata quando viene inizializzata.   
Per realizzare il singleton pattern occorre avere:

* una variabile privata statica della classe che rappresenta l'unica istanza creata;
* un metodo pubblico getInstance che torna l’istanza.

Il suo scopo è :

* Avere un accesso controllato all’unica istanza della classe
* Ridurre il numero di oggetti condivisi
* Centralizzare informazioni e comportamenti in un'unica entità condivisa dagli utilizzatori

Il principale vantaggio offerto è:

* Mutua esclusione

**1.5 Riferimenti**

* Object-Oriented Software Engineering Using UML, Patterns, and Java, Bernd Bruegge & Allen H. Dutoit.
* GestionePrenotazione\_RAD
* GestionePrenotazione\_SDD
* Google Java Style: <https://goo.gl/KRYr9s>

**2. Packages**

Il sistema di Gestione Prenotazioni sarà distribuito in packages:

Il package “controller” conterrà le Java Servlet

Il package “model” conterrà le risorse per l’accesso alle risorse e il relativo formato per la realizzazione dei medesimi.

Il package “front-end” conterrà le pagine html, css, jsp e js che si occuperanno dell’interazione con l’utente.

**2.1 Package Controller**

-controller

AccettaRichiesta.java

AssegnaAulaDipartimento.java

AulaGetter.java

AulaStudenti.java

CalGetter.java

CreaAula.java

CreaDipartimento.java

CreaEdificio.java

DipGetter.java

DomandaPrenotazione.java

EditAmministratore.java

EliminaPrenotazione.java

EliminaRichiesta.java

Login.java

Logout.java

ModificaPassword.java

NavAreaPersonale.java

NavAula.java

NavCreaAula.java

NavCreaDipartimento.java

NavCreaEdificio.java

NavDipartimento.java

NavEdifici.java

NavEditAmm.java

NavElencoUtenti.java

NavEliminaAula.java

NavEliminaDipartimento.java

NavLogin.java

NavModificaPassword.java

NavRegistrazione.java

NavRichiestePrenotazioni.java

NavRimuoviEdificio.java

NavScegliAmministratore.java

NavScegliDipartimento.java

NavStoricoPrenotazioni.java

Registrazione.java

RimAulaGetter.java

RimuoviAula.java

RimuoviDipartimento.java

RimuoviEdificio.java

ScegliAmministratore.java

ServletBasic.java

ServletHome.java

|  |  |
| --- | --- |
| Classe | Descrizione |
| AccettaRichiesta.java | Controller che permette l’accettazione di una richiesta di prenotazione da parte di un amministratore. |
| AssegnaDipartimentoAula.java | Controller che gestisce l’assegnazione di un dipartimento ad un’aula da parte dell’amministratore dell’Ateneo. |
| AulaGetter.java | Controller che permette la gestione delle aule all’interno dell’edificio di cui ne fanno parte. |
| AulaStudenti.java | Controller che permette l’assegnazione dell’aula prenotata da uno studente. |
| CalGetter.java | Controller che gestisce le prenotazioni riguardanti una determinata aula. |
| CreaAula.java | Controller che gestisce la creazione di un’aula da parte dell’amministratore dell’Ateneo. |
| CreaDipartimento.java | Controller che gestisce la creazione di un dipartimento da parte dell’amministratore dell’Ateneo. |
| CreaEdifico.java | Controller che gestisce la creazione di un edificio da parte dell’amministratore dell’Ateneo. |
| DipGetter.java | andrea |
| DomandaPrenotazione.java | Controller che gestisce la creazione di una domanda di prenotazione di uno studente. |
| EditAmministratore.java | Gestisce la modifica del dipartimento ad un amministratore. |
| EliminaPrenotazione.java | Controller che permette l’eliminazione di una prenotazione dallo storico prenotazioni che sia stata accettata o meno. |
| EliminaRichiesta.java | Controller che permette ad un amministratore l’eliminazione effettiva di una prenotazione ricevuta da un utente. |
| Login.java | Controller che permette l’accesso da parte di un utente registrato nel sistema Gestione Prenotazioni. |
| Logout.java | Controller che conferma l’uscita di un utente dal sistema Gestione Prenotazioni. |
| ModificaPassword.java | Controller che permette la modifica della password di un utente registrato al sistema Gestione Prenotazioni. |
| NavAreaPersonale.java | Controller che gestisce il reindirizzamento all’area personale. |
| NavAula.java | Controller che gestisce il reindirizzamento alla pagina del calendario relativo all’aula. |
| NavCreaAula.java | Controller che gestisce il reindirizzamento alla pagina di creazione dell’aula. |
| NavCreaDipartimento.java | Controller che gestisce il reindirizzamento alla pagina per la creazione del dipartimento. |
| NavCreaEdificio.java | Controller che gestisce il reindirizzamento alla pagina per la creazione dell’edificio. |
| NavDipartimento.java | Controller che gestisce il reindirizzamento alla pagina dei dipartimenti per la loro visualizzazione. |
| NavEdifici.java | Controller che gestisce il reindirizzamento alla pagina degli edifici per la loro visualizzazione. |
| NavEditAmm.java | Controller che gestisce il reindirizzamento alle modifiche delle funzionalità dell’amministratore |
| NavElencoUtenti.java | Controller che gestisce il reindirizzamento alla pagina elenco tenti per la loro visualizzazione. |
| NavEliminaAula.java | Controller che gestisce il reindirizzamento alla pagina per la rimozione dell’aula. |
| NavEliminaDipartimento.java | Controller che gestisce il reindirizzamento alla pagina per la rimozione del dipartimento. |
| NavLogin.java | Controller che gestisce il reindirizzamento alla pagina di Login. |
| NavModificaPassword.java | Controller che gestisce il reindirizzamento alla sezione modifica password. |
| NavRegistrazione.java | Controller che gestisce il reindirizzamento alla pagina di Registrazione. |
| NavRichiestePrenotazioni.java | Controller che gestisce il reindirizzamento alla pagina per la visualizzazione delle richieste di prenotazione. |
| NavRimuoviEdificio.java | Controller che gestisce il reindirizzamento alla pagina per la rimozione di un edificio. |
| NavScegliAmministratore.java | Controller che gestisce il reindirizzamento alla sezione per l’assegnazione di un amministratore. |
| NavScegliDipartimento.java | Controller che gestisce il reindirizzamento alla pagina per la scelta del dipartimento una volta selezionati aula ed edificio. |
| NavStoricoPrenotazioni.java | Controller che gestisce il reindirizzamento alla pagina dello storico prenotazioni. |
| Registrazione.java | Controller che gestisce la registrazione di un nuovo utente. |
| RimAulaGetter.java |  |
| RimuoviAula.java | Controller che permette l’eliminazione di un’aula ed una volta eliminata gestisce il reindirizzamento alla pagina degli edifici. |
| RimuoviDipartimento.java | Controller che permette la rimozione di un dipartimentoed una volta eliminato gestisce il reindirizzamento alla pagina dei dipartimenti. |
| RimuoviEdificio.java | Controller che permette la rimozione di un edificio ed una volta eliminato gestisce il reindirizzamento alla pagina degli edifici. |
| ScegliAmministratore.java | Controller che permette l’assegnazione del titolo di un amministratore ad un utente ed una volta assegnato gestisce il reindirizzamento alla pagina dell’elenco degli utenti. |
| ServletBasic.java | Controller che gestisce la data. |
| ServletHome.java | Controller che permette lo start del sistema e gestisce il reindirizzamento alla homepage. |

**2.2 Package model**

-model

Dipartimento.java

DipartimentoDAO.java

DriverManagerConnectionPool.java

MyCalendar.java

Prenotazione.java

PrenotazioneDAO.java

Struttura.java

StrutturaDAO.java

Utente.java

UtenteDAO.java

|  |  |
| --- | --- |
| Classe | Descrizione |
| Dipartimento.java | Model che rappresenta le informazioni riguardanti il dipartimento. |
| DipartimentoDAO.java | Model che permette di eseguire le operazioni sul DB per l’entità Dipartimento. |
| DrivenManagerConnectionPool.java | Model la cui istanza rappresenta la connessione con il database. |
| MyCalendar.java | Model che rappresenta le informazioni riguardanti il calendario. |
| Prenotazione.java | Model che rappresenta le informazioni riguardanti la prenotazione di un’aula. |
| PrenotazioneDAO.java | Model che permette di eseguire le operazioni sul DB per l’entità Prenotazione. |
| Struttura.java | Model che rappresenta le informazioni riguardanti la struttura. |
| StrutturaDao.java | Model che permette di eseguire le operazioni sul DB per l’entità Struttura. |
| Utente.java | Model che rappresenta tutti gli utenti che possono utilizzare i servizi inerenti alla piattaforma Gestione Prenotazioni. |
| UtenteDao.java | Model che permette di eseguire le operazioni sul DB per l’entità Utente. |

**2.3 Package Front-end**

-front-end

AreaPersonale.jsp

CalendarioAula.jsp

CreaAula. jsp

CreaDipartimento. jsp

CreaEdificio. jsp

Dipartimenti. jsp

Edifici. jsp

EditAmministratore.jsp

ElencoUtenti. jsp

Header.jsp

Footer.jsp

HomePage.jsp

Login.jsp

ModificaPassword.jsp

Registrazione.jsp

RichiestePrenotazioni.jsp

RimuoviAula.jsp

RimuoviDipartimento.jsp

RimuoviEdificio.jsp

ScegliAmministratore. jsp

ScegliDipartimento.jsp

StroricoPrenotazioni. jsp

|  |  |
| --- | --- |
| Classe | Descrizione |
| AreaPersonale.jsp | Pagina che permette ad un utente loggato di visualizzare la propria area personale. |
| CalendarioAula.jsp | Pagina che permette la visualizzazione del calendario relativo all’aula. |
| CreaAula.jsp | Pagina che permette la creazione di un’aula da parte dell’amministratore dell’Ateneo. |
| CreaDipartimento.jsp | Pagina che permette la creazione di un dipartimento da parte dell’amministratore dell’Ateneo. |
| CreaEdificio.jsp | Pagina che permette la creazione di un edifico da parte dell’amministratore dell’Ateneo. |
| Dipartimenti.jsp | Pagina che permette la visualizzazione dell’elenco relativo ai dipartimenti esistenti. |
| Edifici.jsp | Pagina che permette la visualizzazione dell’elenco relativo agli edifici esistenti. |
| EditAmministratore.jsp | Pagina che permette la visualizzazione dell’elenco degli utenti e la conseguente scelta o modifica di un amministratore. |
| ElencoUtenti.jsp | Pagina che permette la visualizzazione dell’elenco degli utenti iscritti al sistema. |
| Header.jsp | Header del sistema. |
| Footer.jsp | Foote del sistema. |
| HomePage.jsp | Pagina principale del sistema da cui si può effettuare Login, Registrazione, visualizzazione degli edifici e dei dipartimenti presenti. |
| Login.jsp | Pagina che permette l’accesso da parte di un utente registrato nel sistema Gestione Prenotazioni. |
| ModificaPassword.jsp | Pagina che permette di modificare la password del proprio profilo. |
| Registrazione.jsp | Pagina in cui lo studente o il docente può effettuare la registrazione al sistema Gestione Prenotazioni compilando i campi richiesti. |
| RichiestaPrenotazione.jsp | Pagina che permette la visualizzazione delle richieste di prenotazione ad un amministratore con la conseguente possibilità di accettare o rifiutare. |
| RimuoviAula.jsp | Pagina visualizzabile solo dall’amministratore dell’ateneo e permette la rimozione di un’aula. |
| RimuoviDipartimento.jsp | Pagina visualizzabile solo dall’amministratore dell’ateneo e permette la rimozione di un dipartimento. |
| RimuoviEdificio.jsp | Pagina visualizzabile solo dall’amministratore dell’ateneo e permette la rimozione di un edificio. |
| ScegliAmministratore.jsp | Pagina che permette all’amministratore dell’ateneo la visualizzazione e la conseguente assegnazione per il titolo di amministratore ad un docente. |
| ScegliDipartimento.jsp | Pagina che permette all’amministratore dell’ateneo la visualizzazione e la conseguente assegnazione del dipartimento ad un docente. |
| StroricoPrenotazioni.jsp | Pagina che permette ad un utente di visualizzare il proprio storico delle prenotazioni. |

**3. Class Interfaces**

**3.1 Class Interface Controller**

|  |
| --- |
| **Nome Classe** |
| AccettaRichiesta |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  la richiesta di prenotazione deve essere presente nel database. |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  la modifica del flag all’interno del database viene effettuata. |

|  |
| --- |
| **Nome Classe** |
| AssegnaDipartimentoAula |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’aula deve essere presente nel database e non deve appartenere a nessun dipartimento. |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’assegnazione del dipartimento all’aula avviene con successo. |

|  |
| --- |
| **Nome Classe** |
| AulaGetter |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’aula deve essere presente nel database. |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  il sistema mostra l’elenco delle aule nell’edificio corrispondente. |

|  |
| --- |
| **Nome Classe** |
| AulaStudenti |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’aula non deve essere già prenotata. |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’aula viene assegnata allo studente con successo e viene cambiato lo stato dell’aula nel database. |

|  |
| --- |
| **Nome Classe** |
| CalGetter |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  la prenotazione deve essere nello stato “accettata” |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  il sistema elenca le prenotazioni con le relative informazioni. |

|  |
| --- |
| **Nome Classe** |
| CreaAula |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’aula non deve essere presente all’interno del database. |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’inserimento della nuova aula all’interno del database è andato a buon fine. |

|  |
| --- |
| **Nome Classe** |
| CreaDipartimento |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  il dipartimento non deve essere presente all’interno del database. |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’inserimento del nuovo dipartimento è andato a buon fine. |

|  |
| --- |
| **Nome Classe** |
| CreaEdificio |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’edificio non deve essere presente all’interno del database. |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’inserimento del nuovo edificio è andato a buon fine. |

|  |
| --- |
| **Nome Classe** |
| DipGetter |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  il dipartimento deve essere assegnato ad un’aula e deve avere un edifico. |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  il sistema elenca i dipartimenti con relative aule ed edifici. |

|  |
| --- |
| **Nome Classe** |
| DomandaPrenotazione |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’aula che si intende prenotare deve essere libera. |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  il sistema invia la domanda per la prenotazione all’amministratore del relativo dipartimento. |

|  |
| --- |
| **Nome Classe** |
| EditAmministratore |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente deve essere un docente dell’Ateneo |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  il flag relativo al tipo di utente viene aggiornato nel database. |

|  |
| --- |
| **Nome Classe** |
| EliminaPrenotazione |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  le proprie prenotazione devono essere presenti nel database. |
| Post-Condizione |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  rimozione della prenotazione dallo storico prenotazioni andata a buon fine |

|  |
| --- |
| **Nome Classe** |
| EliminaRichiesta |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  la richiesta di prenotazione deve essere presente nel database. |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  la richiesta viene eliminata dal database. |

|  |
| --- |
| **Nome Classe** |
| Login |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente non deve essere loggato nella piattaforma “Gestione Prenotazioni”. L’utente deve essere presente sul database affinché il login vada a buon fine. |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente si è loggato al sistema. |

|  |
| --- |
| **Nome Classe** |
| Logout |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente deve essere loggato alla piattaforma “Gestione Prenotazioni”. |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente risulta sconnesso dal sistema. |

|  |
| --- |
| **Nome Classe** |
| ModificaPassword |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente deve essere loggato alla piattaforma “Gestione Prenotazioni” |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  la modifica della password nel database è andata a buon fine. |

|  |
| --- |
| **Nome Classe** |
| NavAreaPersonale |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente è loggato alla piattaforma “Gestione Prenotazioni”. |

|  |
| --- |
| **Nome Classe** |
| NavAula |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente è loggato alla piattaforma “Gestione Prenotazioni”. |

|  |
| --- |
| **Nome Classe** |
| NavCreaAula |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente è loggato alla piattaforma “Gestione Prenotazioni”. |

|  |
| --- |
| **Nome Classe** |
| NavCreaDipartimento |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente è loggato alla piattaforma “Gestione Prenotazioni”. |

|  |
| --- |
| **Nome Classe** |
| NavCreaEdificio |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente è loggato alla piattaforma “Gestione Prenotazioni”. |

|  |
| --- |
| **Nome Classe** |
| NavDipartimento |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente è loggato alla piattaforma “Gestione Prenotazioni”. |

|  |
| --- |
| **Nome Classe** |
| NavEdifici |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente è loggato alla piattaforma “Gestione Prenotazioni”. |

|  |
| --- |
| **Nome Classe** |
| NavEditAmm |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente è loggato alla piattaforma “Gestione Prenotazioni”. |

|  |
| --- |
| **Nome Classe** |
| NavElencoUtenti |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente è loggato alla piattaforma “Gestione Prenotazioni”. |

|  |
| --- |
| **Nome Classe** |
| NavEliminaAula |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente è loggato alla piattaforma “Gestione Prenotazioni”. |

|  |
| --- |
| **Nome Classe** |
| NavEliminaDiparimento |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente è loggato alla piattaforma “Gestione Prenotazioni”. |

|  |
| --- |
| **Nome Classe** |
| NavLogin |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente deve essere presente nel sistema. |

|  |
| --- |
| **Nome Classe** |
| NavModificaPassword |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente è loggato alla piattaforma “Gestione Prenotazioni”. |

|  |
| --- |
| **Nome Classe** |
| NavRegistrazione |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente non deve essere presente nel database. |

|  |
| --- |
| **Nome Classe** |
| NavRichiestaPrenotazione |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente è loggato alla piattaforma “Gestione Prenotazioni”.”. |

|  |
| --- |
| **Nome Classe** |
| NavRimuoviEdificio |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente è loggato alla piattaforma “Gestione Prenotazioni”. |

|  |
| --- |
| **Nome Classe** |
| NavScegliAmministratore |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente è loggato alla piattaforma “Gestione Prenotazioni”. |

|  |
| --- |
| **Nome Classe** |
| NavScegliDipartimento |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente è loggato alla piattaforma “Gestione Prenotazioni”. |

|  |
| --- |
| **Nome Classe** |
| NavStoricoPrenotazioni |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente è loggato alla piattaforma “Gestione Prenotazioni”. |

|  |
| --- |
| **Nome Classe** |
| Registrazione |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utente è loggato alla piattaforma “Gestione Prenotazioni”. |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’account appena creato è inserito con successo nel database. |

|  |
| --- |
| **Nome Classe** |
| RimAulaGetter |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’aula deve essere presente nel database. |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  il sistema mostra l’elenco degli edifci con la relativa aula rimossa |

|  |
| --- |
| **Nome Classe** |
| RimuoviAula |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’aula deve essere presente nel database. |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’aula è rimossa dal database. |

|  |
| --- |
| **Nome Classe** |
| RimuoviDipartimento |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  il dipartimento deve essere presente nel database. |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  il dipartimento è rimosso dal database. |

|  |
| --- |
| **Nome Classe** |
| RimuoviEdificio |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’edificio deve essere presente nel database. |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’edificio è rimosso dal database. |

|  |
| --- |
| **Nome Classe** |
| ScegliAmministratore |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  l’utenete deve essere un docente |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  lo stato dell’utente è modificato nel database. |

|  |
| --- |
| **Nome Classe** |
| ServletBasic |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  gestione calendario. |

|  |
| --- |
| **Nome Classe** |
| ServletHome |
| **Attributi** |
| - |
| **Metodi** |
| +doGet(HttpServletRequest request, HttpServletResponse response) : void  +doPost(HttpServletRequest request, HttpServletResponse response) :void |
| **Pre-Condizione** |
| - |
| **Post-Condizione** |
| doGet(HttpServletRequest request, HttpServletResponse response)/doPost(HttpServletRequest request, HttpServletResponse response) :  start sistema. |

**3.2 Class Interface Model**

|  |
| --- |
| **Nome Classe** |
| Dipartimento |
| **Attributi** |
| -dip : String  -ammDip : String |
| **Metodi** |
| +getters() & setters()  +toString() |
| **Pre-Condizione** |
| NC |
| **Post-Condizione** |
| NC |

|  |
| --- |
| **Nome Classe** |
| DipartimentoDAO |
| **Attributi** |
| NA |
| **Metodi** |
| +doSave(String dip) : int  +doDelete(String dip) : int  +doDeleteAmm(String email) : int  +doUpdate(String email) : int  +doRetriveAll() : ArrayList<Dipartimento>  +doRetriveAllDip() : ArrayList<Dipartimento>  +doRetriveAllDipartimenti() : ArrayList<String>  +doRetriveByKey(String email) : String  +doDipartimentoByStruttura(String aula,String edificio) : Dipartimento |
| **Pre-Condizione** |
| doSave(String) : dip non deve essere gia presente nel database.  doDelete(String) : dip deve essere presente nel database.  doDeleteAmm(String) : email identificativo di un utente che è un amministratore  presente nel database.  doUpdate(String) : email non deve essere presente nel database.  doRetriveAll() : tutte le informazioni relative al dipartimento e all’amministratore devono essere presenti nel database.  doRetriveAllDip() : le informazioni relative al dipartimento devono essere presenti nel database.  doRetriveAllDipartimenti() : i dipartimenti devono essere presenti nel database.  doRetriveByKey(String) : l’utente deve essere amministratore di un dipartimento.  doDipartimentoByStruttura(String,String) : l’aula e l’edificio devono appartenere ad un dipartimento. |
| **Post-Condizione** |
| doSave(String) : dip è il nome associato al dipartimento.  doDelete(String) : dip viene eliminato definitivamente da database.  doDeleteAmm(String) : email identificativo di un utente che è un amministratore a cui il titolo di amministratore è rimosso.  doUpdate(String) : email è presente nel database viene modificata.  doRetriveAll() : mostra l’elenco delle informazioni relative al dipartimento e ai suoi amministratori.  doRetriveAllDip() : mostra le informazioni relative al dipartimento.  doRetriveAllDipartimenti() : mostra i dipartimenti.  doRetriveByKey(String) : mostra l’amministratore del dipartimento.  doDipartimentoByStruttura(String,String) : mostra il dipartimento con relative aule ed edifici |

|  |
| --- |
| **Nome Classe** |
| DriverManagerConnectionPool |
| **Attributi** |
| -datasource : DataSource |
| **Metodi** |
| +getConnection : Connection |
| **Pre-Condizione** |
| getConnection() : Il database deve essere disponibile alla porta 3306 e le credenziali di accesso al database devono essere le stesse indicate. |
| **Post-Condizione** |
| getConnection(): Connection deve essere una connessione attiva al database. |

|  |
| --- |
| **Nome Classe** |
| MyCalendar |
| **Attributi** |
| -date : Calendar |
| **Metodi** |
| +getters() & setters()  +getToday() String  +splitData(String tmp) void  +setDayofWeek() int  +getDayofWeek(String data) int |
| **Pre-Condizione** |
| NC |
| **Post-Condizione** |
| getToday(): mostra la data corrente.  splitData(String): mostra anno, giorno, mese e giorno della settimana.  setDayofWeek(): setta la settimana corrente.  getDayofWeek(): mostra la settimana corrente. |

|  |
| --- |
| **Nome Classe** |
| Prenotazione |
| **Attributi** |
| -Idprenotazione : int  -titolo : String  -data : String  -oraInizio : int  -oraFine : int  -descrizione : String  -aulaPrenotata : String  -edificio : utente  -utente : String  -aula : String |
| **Metodi** |
| +getters() & setters()  +toString() |
| **Pre-Condizione** |
| NC |
| **Post-Condizione** |
| NC |

|  |
| --- |
| **Nome Classe** |
| PrenotazioneDAO |
| **Attributi** |
| NA |
| **Metodi** |
| +doSave(String titolo, String data, int oraInizio, int oraFine, int descrizione) : int  +doDelete(int id) : int  +doDeleteByStruttura(String edificio, String aula) : int  +doDeleteByEdificio(String edificio) : int  +doupdate(int id) : void  +doRetriveAll() : ArrayList<Prenotazione>  +doRetriveByUtente(String nomeUtente) : ArrayList<Prenotazione>  +doRetriveByDip(String dipartimento, String data) : ArrayList<Prenotazione>  +doRetriveByDate(String email,String data) : ArrayList<Prenotazione>  +doRetriveByCalendario(String aula,String edificio, String data) : ArrayList<Prenotazione>  +splitData(String temp) : String  +doAulabyDate(String data, String aula) : ArrayList<Prenotazione>  +controlloOra(String data, int oraInizio, int oraFine, String aula, String edificio) : int |
| **Pre-Condizione** |
| doSave(String,String, int, int, int) : la tupla per la prenotazione non deve essere uguale a quella che si vuole inserire  doDelete(int) : deve esistere nel database la prenotazione con questo id  doDeleteByStruttura(String,String) : deve esistere nel database l’edificio e l’aula con la medesima prenotazione.  doDeleteByEdificio(String) : deve esistere nel database l’edificio con la medesima. prenotazione  doupdate(int ) : deve essere presente nel database una richiesta di prenotazione.  doRetriveAll() : devono essere presenti delle prenotazione nel database.  doRetriveByUtente(String) : l’utente preso in input deve aver effettuato delle prenotazioni.  doRetriveByDip(String, String ) : devono essere presenti prenotazioni per il dipartimento preso in input.  doRetriveByDate(String, String) : devono essere presenti prenotazioni per la data presa in input  doRetriveByCalendario(String,String, String) : devono essere presenti prenotazioni  doAulabyDate(String, String) : la prenotazione deve essere presente nel database.  controlloOra(String, int, int, String, String) : deve essere presente una prenotazione nel database. |
| **Post-Condizione** |
| doSave(String,String, int, int, int) : viene creata una nuova tupla nel database corrispondente a quella richiesta.  doDelete(int) : cancellazione della tubla con corrispondente id dal database    doDeleteByStruttura(String,String) : cancellazione della prenotazione dal database con corrispondente aula ed edificio  doDeleteByEdificio(String) : cancellazione della prenotazione dal database con il corrispondente edificio.  doupdate(int ) : lo stato della prenotazione viene cambiato, 1 accettata.  doRetriveAll() : mostra tutte le prenotazioni.  doRetriveByUtente(String) : mostra le prenotazione effettuate dall’utente.  doRetriveByDip(String, String ) : mostra le prenotazioni per quel dipartimento e tutte le relative informazioni.  doRetriveByDate(String, String) : mostra le prenotazioni relative a quella data e tutte le relative informazioni.  doRetriveByCalendario(String,String, String) : mostra tutte le informazioni riguardanti le prenotazioni relative al calendario settimanale  doAulabyDate(String, String) : mostra la data e l’aula della prenotazione.  controlloOra(String, int, int, String, String) : mostra i dati relativi ad una prenotazione anche con relativo orario di inizio e fine. |

|  |
| --- |
| **Nome Classe** |
| Struttura |
| **Attributi** |
| -aula : String  -edificio : String  -tipoAula : String  -dipartimento : String  -descrizione : String |
| **Metodi** |
| +getters() & setters() |
| **Pre-Condizione** |
| NC |
| **Post-Condizione** |
| NC |

|  |
| --- |
| **Nome Classe** |
| StrutturaDao |
| **Attributi** |
| NA |
| **Metodi** |
| +doSave(String aula, String edificio, String descrizione) : int  +doupdate(String aula, String edificio, String tipoAula) : int  +doSaveEdificio(String aula, String edificio, int descrizione) : int  +doDelete(String aula, String edificio) : int  +doDeleteEdificio(String edificio) : int  +doUpdateDipartimento(String aula, String edificio, String dip) : int  +doRetriveAll() : ArrayList<Struttura>  +doRetriveByKey(String nome) : Dipartimento  +doRetriveAllEdifici() : ArrayList<String>  +doAulabyEdificio(String edificio) : ArrayList<Struttura>  +doStrutturabyName (String aula,String edificio) : Struttura  +doStrutturabyDip (String dip) : ArrayList<Struttura>  +doStrutturaByAula(String aula) : Struttura  +doStrutturabyDipartimenti(String dip) : ArrayList<Struttura>  +doRetrieveAllAule() : ArrayList<Struttura> |
| **Pre-Condizione** |
| doSave(String, String, String) : aula, edificio e descrizione non devono essere presenti nel database  doupdate(String, String, String) : deve essere presente la tupla all’interno del database  doSaveEdificio(String, String, int) : l’edifico non deve essere presente nel database  doDelete(String, String) : aula ed edifico devono essere presenti nel database  doDeleteEdificio(String) : l’edificio deve essere presente nel database  doUpdateDipartimento(String, String, String) : aula, edificio e dipartimento devono essere presenti nel database  doRetriveAll() : devono essere presenti delle strutture nel database  doRetriveByKey(String) : deve esistere almeno un dipartimento nel database  doRetriveAllEdifici() : non deve esistere già nel database  doAulabyEdificio(String) : un edificio deve avere almeno un’aula e deve essere presente nel database  doStrutturabyName (String,String) : la struttura con la relativa aula ed edificio presi da input deve essere presente ne database  doStrutturabyDip (String) : il dipartimento preso da input deve essere presente nel database  doStrutturaByAula(String) : l’ aula presa da input deve essere presente nel database  doStrutturabyDipartimenti(String) : la struttura deve essere presente nel database  doRetrieveAllAule() : almeno un’aula deve appartenere ad una struttura presente nel database |
| **Post-Condizione** |
| doSave(String, String, String) : aggiunta della tupla nel database  doupdate(String, String, String) : aggiornamento della tupla nel database  doSaveEdificio(String, String, int) : aggiunta dell’edificio al database  doDelete(String, String) : aula ed edificio vengono rimossi dal database  doDeleteEdificio(String) : l’edificio viene rimosso dal database  doUpdateDipartimento(String, String, String) : il dipartimento con relativa aula ed edificio viene aggiornato nel database  doRetriveAll() : mostra tutte le strutture  doRetriveByKey(String) : il dipartimento cercato in base al nome è presente nel database  doRetriveAllEdifici() : vengono aggiunte nuove tuple nel database relative agli edifici  doAulabyEdificio(String) : l’edificio cercato e le relative informazioni sono presentei nel database  doStrutturabyName (String,String) : mostra la struttura corrispondente  doStrutturabyDip (String) : mostra la struttura corrispondente  doStrutturaByAula(String) : mostra la struttura corrispondente  doStrutturabyDipartimenti(String) : mostra tutti i daprtimenti relativi ad una struttura  doRetrieveAllAule() : mostra tutte le aule appartenenti ad una struttura |

|  |
| --- |
| **Nome Classe** |
| Utente |
| **Attributi** |
| -email : String  -password : String  -nome : String  -cognome : String  -tipoUtente : int |
| **Metodi** |
| +getters() & setters()  +toString() |
| **Pre-Condizione** |
| NC |
| **Post-Condizione** |
| NC |

|  |
| --- |
| **Nome Classe** |
| UtenteDAO |
| **Attributi** |
| NA |
| **Metodi** |
| +doSave(String email, String password, String nome, String cognome, int tipoUtente) : int  +doUpdate(String email, int flag) : int  +doChangeDip(String email, String dip) : int  +doRetriveByKey(String email, String password) : int  +doRetriveAll() : ArrayList<Utente>  +doUpdate(String email, String password) : int |
| **Pre-Condizione** |
| doSave(String, String, String, String, int) : i parametri passati non devono essere presenti  doUpdate(String, int) : l’utente deve essere presente nel database e deve essere un docente  doChangeDip(String, String) : il dipartimento deve essere presente nel database e non deve avere assegnatovi nessun amministratore  doRetriveByKey(String, String) : l’utente deve essere presente nel database  doRetriveAll() : deve essere presente almeno un utente nel database  doUpdate(String, String) : l’utente deve essere presente nel database |
| **Post-Condizione** |
| doSave(String, String, String, String, int) : i parametri passati sono presenti nel database  doUpdate(String, int) : il flag dell’utente viene aggiornato, 1 docente, 2 amministratore di dipartimento  doChangeDip(String, String) : l’amministratore del dipartimento viene cambiato nel database  doRetriveByKey(String, String) : l’email passata come parametro è presente nel database  doRetriveAll() : viene mostrato l’elenco degli utenti presenti nel database  doUpdate(String, String) : viene modificata la password passata come parametro del relativo utente sempre passato come parametro |