Gestione Prenotazioni



ODD

Object

Design

Document

**Borriello Martina 0512105144**

**Claudini Aldo 051210**

**Claro Andrea 051210**

**Marcantuono Veronica 0512105426**

***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;
* Non abbreviati;
* 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

**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**

* 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

Aula.java

AulaGetter.java

AulaStudenti.java

CalGetter.java

CambiaDipartimento.java

CreaAula.java

CreaDipartimento.java

CreaEdificio.java

DipGetter.java

Dipartimento.java

DomandaPrenotazione.java

Edifici.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

NavDomandaPrenotazione.java

NavEdifici.java

NavEditAmm.java

NavElencoUtenti.java

NavEliminaAula.java

NavEliminaDipartimento.java

NavLogin.java

NavModificaPassword.java

NavRegistrazione.java

NavRichiestePrenotazioni.java

NavRimuoviEdificio.java

NavScegliAmministratore.java

NavStoricoPrenotazioni.java

NavScegliDipartimento.java

Registrazione.java

RimuoviAula.java

RimuoviDipartimento.java

RimuoviEdificio.java

ScegliAmministratore.java

ServletBasic.java

ServletHome.java

0 prenotabile dagli studenti 1 prenotabile dagli studenti

|  |  |
| --- | --- |
| Classe | Descrizione |
| AccettaRichiesta.java | Gestisce l’accettazione di una richiesta di prenotazione |
| AssegnaAulaDipartimento.java | Gestisce l’assegnazione di un aula ad un dipartimento esistente |
| Aula.java | Gestisce la visualizzazione dell’aula |
| AulaGetter.java | andrea |
| AulaStudenti.java | Gestisce l’assegnazione dell’aula prenotata da uno studente. |
| CalGetter.java | andrea |
| CambiaDipartimento.java | Gestisce l’amministratore per il dipartimento |
| CreaAula.java | Gestisce la creazione di un’aula. |
| CreaDipartimento.java | Gestisce la creazione di un dipartimento. |
| CreaEdifico.java | Gestisce la creazione di un edifico |
| DipGetter.java | andrea |
| Dipartimento.java | Gestisce la visualizzazione del dipartimento |
| DomandaPrenotazione.java | Gestisce la richiesta di prenotazione di un’aula. |
| Edifici.java | Gestisce la visualizzazione dell’edificio |
| EditAmministratore.java | Gestisce la modifica del dipartimento ad un amministratore |
| EliminaPrenotazione.java | Gestisce l’eliminazione di una prenotazione dallo storico prenotazioni |
| EliminaRichiesta.java | Gestisce l’eliminazione di una richiesta |
| Login.java | Gestisce le operazioni di Login |
| Logout.java | Gestisce l’uscita dal sistema. |
| ModificaPassword.java | Gestisce la modifica della password di un account |
| NavAreaPersonale.java | Gestisce il reindirizzamento all’area personale |
| NavAula.java | Gestisce il reindirizzamento all’aula |
| NavCreaAula.java | Gestisce il reindirizzamento all’aula |
| NavCreaDipartimento.java | Gestisce il reindirizzamento all’aula |
| NavCreaEdificio.java | Gestisce il reindirizzamento all’aula |
| NavDipartimento.java | Gestisce il reindirizzamento all’aula |
| NavDomandaPrenotazione.java | Gestisce il reindirizzamento all’aula |
| NavEdifici.java | Gestisce la visualizzazione degli edifici per dipartimento |
| NavEditAmm.java | Gestisce il reindirizzamento alle modifiche delle funzionalità dell’amministratore |
| NavElencoUtenti.java | Gestisce il reindirizzamento alla jsp |
| NavEliminaAula.java | Gestisce il reindirizzamento alla jsp |
| NavEliminaDipartimento.java | Gestisce il reindirizzamento jsp |
| NavLogin.java | Gestisce il reindirizzamento alla pagina di Login |
| NavModificaPassword.java | Gestisce il reindirizzamento alla sezione modifica password |
| NavRegistrazione.java | Gestisce il reindirizzamento alla sezione Registrazione |
| NavRichiestePrenotazioni.java | Gestisce il reindirizzamento alla sezione delle richieste di prenotazione |
| NavRimuoviEdificio.java | Gestisce il reindirizzamento alla sezione per la rimozione di un edificio |
| NavScegliAmministratore.java | Gestisce il reindirizzamento alla sezione per scegliere un amministrazione |
| NavStoricoPrenotazioni.java | Gestisce il reindirizzamento allo storico prenotazioni |
| NavScegliDipartimento.java | Gestisce il reindirizzamento per la scelta del dipartimento |
| Registrazione.java | Gestisce la registrazione di un nuovo utente. |
| RimuoviAula.java | Gestisce la sezione per la rimozione dell’aula |
| RimuoviDipartimento.java | Gestisce la sezione per la rimozione del dipartimento |
| RimuoviEdificio.java | Gestisce la sezione per la rimozione dell’edificio |
| ScegliAmministratore.java | Gestisce la sezione per la scelta dell’aministratore |
| ServletBasic.java | Gestisce la data |
| ServletHome.java | Gestisce le funzionalità di visualizzazione dei dipartimenti, delle aule e degli edifici. |

**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 | Classe dove viene generato l’oggetto Dipartimento tramite get e set. |
| DipartimentoDAO.java | Classe che definisce le operazioni effettuabili sull’entità Dipartimento e dove l’oggetto interagisce con il DB. |
| DrivenManagerConnectionPool.java | Classe che permette la connessione al DBMS. |
| MyCalendar.java | Classe dove viene generato l’oggetto Calendario tramite get e set e le relative funzionalità |
| Prenotazione.java | Classe dove viene generato l’oggetto Prenotazione tramite get e set. |
| PrenotazioneDAO.java | Classe che definisce le operazioni effettuabili sull’entità Prenotazione e dove l’oggetto interagisce con il DB. |
| Struttura.java | Classe dove viene generato l’oggetto Struttura tramite get e set. |
| StrutturaDao.java | Classe che definisce le operazioni effettuabili sull’entità Struttura e dove l’oggetto interagisce con il DB. |
| Utente.java | Classe dove viene generato l’oggetto Utente tramite get e set. |
| UtenteDao.java | Classe che definisce le operazioni effettuabili sull’entità Utente e dove l’oggetto interagisce con il DB. |

**2.3 Package Front-end**

-front-end

AreaPersonale.html

CalendarioAula.jsp

AulaDipartimento.jsp

CreaAula.html

CreaDipartimento.html

CreaEdificio.html

Dipartimenti.html

Edifici.html

EditAmministratore.jsp

ElencoUtenti.html

Header.jsp

Footer.jsp

HomePage.html

Login.html

ModificaPassword.html

Registrazione.html

RichiestePrenotazioni.html

RimuoviAula.html

RimuoviDIpartimento.html

RimuoviEdificio.html

ScegliAmministratore.html

StroricoPrenotazioni.html

|  |  |
| --- | --- |
| Classe | Descrizione |
| AreaPersonale.jsp | Sezione che permette all’utente di visualizzare la propria area personale |
| CalendarioAula.jsp | Sezione che permette la visualizzazione delle richieste di prenotazione ad un amministratore con la conseguente possibilità di accettare o rifiuta |
| CreaAula.jsp | Sezione che permette all’amministratore dell’ateneo di creare una nuova aula. |
| CreaDipartimento.jsp | Sezione che permette all’amministratore dell’ateneo di creare un nuovo dipartimento. |
| CreaEdificio.jsp | Sezione che permette all’amministratore dell’ateneo di creare un nuovo edificio. |
| Dipartimenti.jsp | Sezione che permette la visualizzazione dei dipartimenti |
| Edifici.jsp | Sezione che permette la visualizzazione degli edifici |
| EditAmministratore.jsp | Sezione che permette la visualizzazione per la scelta dell’amministratore |
| ElencoUtenti.jsp | Sezione che permette all’amministratore di visualizzare l’elenco di tutti gli utenti iscritti |
| Header.jsp | Header del sistema |
| Footer.jsp | Foote del sistema |
| HomePage.jsp | Pagina principale di sistema da cui si può effettuare Login e Registrazione |
| Login.jsp | Sezione che permette ad un utente già registrato la propria autenticazione |
| ModificaPassword.jsp | Sezione che permette di modificare la password del proprio profilo |
| Registrazione.jsp | Sezione in cui lo studente o il docente può effettuare la registrazione compilando i campi richiesti. |
| RichiestaPrenotazione.jsp | Sezione che permette la visualizzazione delle richieste di prenotazione ad un amministratore con la conseguente possibilità di accettare o rifiuta |
| RimuoviAula.jsp | Sezione che permette all’amministratore dell’ateneo di rimuovere un’aula. |
| RimuoviDipartimento.jsp | Sezione che permette all’amministratore dell’ateno di rimuovere un dipartimento. |
| RimuoviEdificio.jsp | Sezione che permette all’amministratore dell’ateneo di rimuovere un’aula |
| ScegliAmministratore.jsp | Sezione che permette all’amministratore dell’ateneo la visualizzazione e la conseguente assegnazione per il titolo di amministratore ad un docente |
| ScegliDipartimento.jsp | Sezione che permette all’amministratore dell’ateneo la visualizzazione e la conseguente assegnazione del dipartimento ad un docente. |
| StroricoPrenotazioni.jsp | Sezione che permette ad un utente di visualizzare il proprio storico delle prenotazioni |

**3. Class Interfaces**

**CONTROLLER**

|  |  |
| --- | --- |
| **Nome Classe** | AccettazioneRichiesta |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | Aula |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | AulaGetter |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | CalGetter |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | CambiaDipartimento |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | CercaPrenotazioni |
| **Descrizione** | Gestisce la visualizzazione delle prenotazioni delle aule. |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | CreaAula |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | CreaDipartimento |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | CreaEdificio |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | Dipartimento |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | DipGetter |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | DomandaPrenotazione |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | Edifici |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | EditAmministratore |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | EliminaPrenotazione |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | EliminaRichiesta |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | Login |
| **Descrizione** | Consente di gestire le richieste di autenticazione degli utenti attraverso la procedura di login. |
| **Pre-Condizione** | **context** Login:: doGet(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null && request.getParameter(“email”)!=null && request.getParameter(“password”)!=null  **context** Login:: doPost(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | Logout |
| **Descrizione** | Consente di gestire le richieste di chiusura della sessione attraverso la procedura di logout. |
| **Pre-Condizione** | **context** Logout:: doGet(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null  **context** Logout:: doPost(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null |
| **Post-Condizione** |  |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | ModificaPassword |
| **Descrizione** | Consente di modificare la password. |
| **Pre-Condizione** | **context** ModificaPassword:: doGet(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null && request.getParameter(“email”)!=null && request.getParameter(“newPassw”)!=null  **context** ModificaPassword:: doPost(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null |
| **Post-Condizione** | **context** UtenteDAO::doUpdate(email:String, newPassw:String):int **post:** (???) |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | NavAreaPersonale |
| **Descrizione** | Gestisce la visualizzazione dell’area personale. |
| **Pre-Condizione** | **context** NavAreaPersonale:: doGet(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null  **context** NavAreaPersonale:: doPost(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | NavAula |
| **Descrizione** | Gestisce la visualizzazione dell’aule. |
| **Pre-Condizione** | **context** NavAula:: doGet(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null  **context** NavAula:: doPost(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | NavCreaAula |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | NavCreaDipartimento |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | NavCreaEdificio |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | NavDipartimento |
| **Descrizione** | Gestisce la visualizzazione delle aule per ogni dipartimento. |
| **Pre-Condizione** | **context** NavDipartimento:: doGet(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null  **context** NavDipartimento:: doPost(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | NavDomandaPrenotazione |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | NavEdifici |
| **Descrizione** | Gestisce la visualizzazione degli per ogni dipartimento. |
| **Pre-Condizione** | **context** NavEdifici:: doGet(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null  **context** NavEdifici:: doPost(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | NavEditAmm |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | NavElencoUtenti |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | NavEliminaAula |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | NavEliminaDipartimento |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | NavEliminaEdificio |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | NavLogin |
| **Descrizione** | Consente di gestire le richieste di autenticazione degli utenti attraverso la procedura di login. |
| **Pre-Condizione** | **context** NavLogin:: doGet(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null  **context** NavLogin:: doPost(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | NavModificaPassword |
| **Descrizione** | Consente di gestire le richieste di modifica password. |
| **Pre-Condizione** | **context** NavModificaPassword:: doGet(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null  **context** NavModificaPassoword:: doPost(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | NavRegistrazione |
| **Descrizione** | Consente di gestire le richieste di registrazione. |
| **Pre-Condizione** | **context** NavRegistrazione:: doGet(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null  **context** NavRegistrazione:: doPost(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | NavRichiestePrenotazione |
| **Descrizione** | Consente di gestire le riichieste di prenotazione |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | NavRimuoviEdificio |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | NavScegliAmministratore |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | NavStoricoPrenotazioni |
| **Descrizione** | Consente di gestire le richieste di visualizzazione dello storico. |
| **Pre-Condizione** | **context** NavStoricoPrenotazioni:: doGet(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null  **context** NavStoricoPrenotazioni:: doPost(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | Registrazione |
| **Descrizione** | Consente di gestire la registrazione di un nuovo utente. |
| **Pre-Condizione** | **context** Registrazione:: doGet(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null  **context** Registrazione:: doPost(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | RimuoviAula |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | RimuoviDipartimento |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | RimuoviEdificio |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | ScegliAmministratore |
| **Descrizione** |  |
| **Pre-Condizione** |  |
| **Post-Condizione** |  |
| **Invarianti** |  |

|  |  |
| --- | --- |
| **Nome Classe** | ServletBasic |
| **Descrizione** | Consente di mantenere il calendario sempre aggiornato. |
| **Pre-Condizione** | **context** ServletBasic:: doGet(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null  **context** ServletBasic:: doPost(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | ServletHome |
| **Descrizione** | Consente di gestire le funzionalità di visualizzazione dei dipartimenti, degli edifici e delle aule. |
| **Pre-Condizione** | **context** ServletBasic:: doGet(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null  **context** ServletBasic:: doPost(request:HttpServletRequest,response:HttpServletResponse) **pre:** request!=null && response!=null |
| **Post-Condizione** | - |
| **Invarianti** | - |

**MODEL**

|  |  |
| --- | --- |
| **Nome Classe** | Dipartimento |
| **Descrizione** | Definisce la struttura di un oggetto Dipartimento. |
| **Pre-Condizione** | - |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | DipartimentoDAO |
| **Descrizione** | Definisce le operazioni effettuabili sull’oggetto Dipartimento che interagisce con il DB. |
| **Pre-Condizione** | **context** DipartimentoDAO:: doSave(ammDip:String, dip:String):int  **context** DipartimentoDAO:: doDelete(dip:int)  **context** DipartimentoDAO::doRetrieveAll(): ArrayList<Dipartimento>  **context** DipartimentoDAO:: doRetrieveByKey(nome:String): Dipartimento |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | DriverManagerConnectionPool |
| **Descrizione** | Consente la connessione al DBMS. |
| **Pre-Condizione** | **context** DriverManagerConnectionPool:: getConnection(): Connection |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | MyCalendar |
| **Descrizione** | Definisce la struttura di un oggetto Calendario. |
| **Pre-Condizione** | **context** MyCalendar:: getDate(): String  **context** MyCalendar:: getToday(): String  **context** MyCalendar:: splitDate(temp:String) |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | Prenotazione |
| **Descrizione** | Definisce la struttura di un oggetto Prenotazione. |
| **Pre-Condizione** | - |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | PrenotazioneDAO |
| **Descrizione** | Definisce le operazioni effettuabili sull’oggetto Prenotazione che interagisce con il DB. |
| **Pre-Condizione** | **context** PrenotazioneDAO:: doSave(titolo:String, data: String, oraInizio: int, oraFine: int, descrizione: String, nomeUtente: String, aula: String, edificio: String): int  **context** PrenotazioneDAO:: doDelete(id: int)  **context** PrenotazioneDAO:: doRetrieveAll(): ArrayList<Prenotazione>  **context** PrenotazioneDAO:: doRetrieveByUtente(nomeUtente: String): ArrayList<Prenotazione>  **context** PrenotazioneDAO:: doRetrieveByDate(email: String, data: String): ArrayList<Prenotazione>  **context** PrenotazioneDAO:: SplitData(temp: String): MyCalendar |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | Struttura |
| **Descrizione** | Definisce la struttura di un oggetto Struttura. |
| **Pre-Condizione** | - |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | StrutturaDAO |
| **Descrizione** | Definisce le operazioni effettuabili sull’oggetto Struttura che interagisce con il DB. |
| **Pre-Condizione** | **context** StrutturaDAO:: doSave(aula: String, edificio: String, tipoAula: int): int  **context** StrutturaDAO:: doDelete(aula: String, edificio: String): int  **context** StrutturaDAO:: doRetrieveAll(): ArrayList<Struttura>  **context** StrutturaDAO:: doRetrieveByKey(nome: String): Dipartimento  **context** StrutturaDAO:: doRetrieveAllEdifici(): ArrayList<String>  **context** StrutturaDAO:: doAulabyEdificio( edificio: String): ArrayList<Struttura>  **context** StrutturaDAO:: doStrutturabyName (aula: String, edificio: String): Struttura  **context** StrutturaDAO:: doStrutturabyDip (dip: String): ArrayList<Struttura> |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | Utente |
| **Descrizione** | Definisce la struttura di un oggetto Utente. |
| **Pre-Condizione** | - |
| **Post-Condizione** | - |
| **Invarianti** | - |

|  |  |
| --- | --- |
| **Nome Classe** | UtenteDAO |
| **Descrizione** | Definisce le operazioni effettuabili sull’oggetto Utente che interagisce con il DB. |
| **Pre-Condizione** | **context** UtenteDAO:: doSave(email: String, password: String, nome: String, cognome: String, tipoUtente: int): int  **context** UtenteDAO:: doRetrieveByKey(email: String, passoword: String): Utente  **context** UtenteDAO:: doUpdate(email: String, password: String): int |
| **Post-Condizione** | - |
| **Invarianti** | - |