|  |
| --- |
| **Università degli Studi**  **di Salerno  Corso di Ingegneria del Software** |

Object Design Document

Movie Desk

Versione: 1.9

Immagine che contiene testo, clipart

Descrizione generata automaticamente

**Coordinatore del progetto:**

|  |  |
| --- | --- |
| **Nome** | **Matricola** |
|  |  |
| **none** | **none** |

**Partecipanti:**

|  |  |
| --- | --- |
| **Nome** | **Matricola** |
|  |  |
| Caliendo Gioacchino | 05706 |
| Puglia Chiara | 06018 |
| Vitillo Ilaria | 05918 |

|  |  |
| --- | --- |
| **Scritto da:** | Caliendo Gioacchino, Puglia Chiara, Vitillo Ilaria |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versione** | **Descrizione** | **Autore** |
| 15/12/2020 | 1.0 | Prima bozza del documento | Chiara Puglia |
| 16/12/2020  18/12/2020  20/12/2020  22/12/2020  23/12/2020  27/12/2020  28/12/2020  08/01/2021  14/01/2021 | 1.1  1.2  1.3  1.4  1.5  1.6  1.7  1.8  1.9 | Prima stesura del documento  Aggiunta dell’Object design trade-offs  Aggiunta dell'Interface Documentation guidelines  Accorgimenti vari  Aggiunta dei Packages  Modifiche dei punti 2 e 3  Aggiunta Class Interfaces  Modifica Class Diagram  Revisione di accorgimenti | Gioacchino Caliendo  Chiara Puglia  Ilaria Vitillo  Chiara Puglia  Ilaria Vitillo  Chiara Puglia  Gioacchino Caliendo  Gioacchino Caliendo  Ilaria Vitillo |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Sommario**

[***1. Introduction*** *4*](#_Toc62147649)

[***1.1 Object Design Trade-offs*** *4*](#_Toc62147650)

[***1.2 Componenti off-the-shelf*** *4*](#_Toc62147651)

[***1.3 Interface Documentation guidelines*** *5*](#_Toc62147652)

[***1.3.1 Classi e interfacce Java*** *5*](#_Toc62147653)

[***1.3.2 Pagine lato server e JSP*** *9*](#_Toc62147654)

[***1.3.3 Script Javascript*** *9*](#_Toc62147655)

[***1.3.4 Fogli di stile CSS*** *10*](#_Toc62147656)

[***1.3.5 Database SQL*** *11*](#_Toc62147657)

[***1.4 Design Pattern*** *11*](#_Toc62147658)

[***1.4.1 MVC*** *11*](#_Toc62147659)

[***1.4.2 Singleton Pattern*** *12*](#_Toc62147660)

[***1.4.3 Façade Pattern*** *13*](#_Toc62147661)

[***1.5 Definitions, acronyms, and abbreviations*** *16*](#_Toc62147662)

[***2. Packages*** *16*](#_Toc62147663)

[***2.1 View*** *16*](#_Toc62147664)

[***2.2 Model*** *17*](#_Toc62147665)

[***2.3 Control*** *19*](#_Toc62147666)

[***3. Class Interfaces*** *21*](#_Toc62147667)

[***4. Class Diagram*** *26*](#_Toc62147668)

[***5. Glossary*** *27*](#_Toc62147669)

# **1. Introduction**

## **1.1 Object Design Trade-offs**

Per quanto riguarda la realizzazione del sistema sono stati individuati i seguenti trade off:

**Comprensibilità vs Adattabilità:** Uno degli obiettivi relativi all’implementazione sarà quello di garantire un’elevata comprensibilità riguardo sia lo stile implementativo specificato nei capitoli successivi e sia nella nomenclatura delle componenti trascurando l’adattabilità del sistema verso altri domini applicativi.

**Memoria vs Usabilità:** Il sistema deve permettere un alto livello di usabilità per l’utente che lo utilizza a discapito della memoria utilizzata per memorizzare i dati presenti al suo interno.

# **1.2 Componenti off-the-shelf**

Per il nostro sistema utilizzeremo componenti off-the-shelf che sono componenti software già disponibili ed utilizzati per facilitare la realizzazione del software.

Il framework open source adatto a creare la grafica di siti web che utilizzeremo è bootstrap.

Esso contiene tools basati su HTML e CSS con estensioni opzionali realizzate attraverso Javascript.

Affinché l’interfaccia del sistema possa rispondere alle azioni dell’utente agilmente, si utilizzerà JQuery e tecniche basate su Ajax.

Per il lato grafico ci si è ispirati a diversi template modificati in base alle esigenze del nostro sistema.

Inoltre, per la gestione delle connessioni, si è pensato di utilizzare un connection-pool, fornita da Tomcat, con riferimento: <https://tomcat.apache.org/tomcat-9.0-doc/jdbc-pool.html>.

# **1.3 Interface Documentation guidelines**

Nell’implementazione del sistema ci si dovrà attenere alle linee guida riportate di seguito.

# **1.3.1 Classi e interfacce Java**

Nella scrittura del codice per le classi Java ci si atterrà allo standard Google Java ([https://google.github.io/styleguide/javaguide.html#s4.6-whitespace](https://google.github.io/styleguide/javaguide.html%23s4.6-whitespace)) tranne ciò che concerne la presenza delle parentesi graffe per una riga d’instruzioni singola e il rientro per l’intendazione che viene effettuato non con 4 spazi ma con un tab.

Questo standard fornisce le regole da seguire come quello relativo ai nomi delle variabili, i quali vengono definiti secondo la nota lowerCamelCase, che consiste nello scrivere parole composte o frasi unendo tutte le parole fra loro mantenendo l’iniziale della prima parola in minuscolo per poi mantenere in maiuscolo le iniziali delle parole successive indicanti la variabile stessa.

Inoltre, si utilizzerà un quality tool come Checkstyle con una configurazione personalizzata dal nome Movie D. Configuration come è riportata di seguito:

* **Javadoc Comments**

|  |  |
| --- | --- |
| **Module** | **Properties** |
| Method Javadoc | **allowedAnnotations:** @param @return  **tokens:** Method declaration  **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Javadoc Missing Whitespace After Asterisk | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Style Javadoc | **severity:** warning  **tokens:**   * Enum declaration * Method declaration   *Tutto il resto deve essere lasciato di default.* |
| Missing Javadoc Method | **severity:** warning  **tokens:** Method declaration  *Tutto il resto deve essere lasciato di default.* |
| Class Type Parameter Name | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Constant Names | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Local Final Variable Names | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Method Names | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Package Names | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Parameter Names | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Static Variable Names | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Type Names | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Catch Parameter Name | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Method Type Parameter Name | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |

* **Naming Conventions**

|  |  |
| --- | --- |
| **Module** | **Properties** |
| Class Type Parameter Name | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Constant Names | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Local Final Variable Names | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Method Names | **severity:** warning  **Properties format**: ^[a-z][a-zA-Z0-9\_]\*$  *Tutto il resto deve essere lasciato di default.* |
| Package Names | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Parameter Names | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Static Variable Names | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Type Names | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Catch Parameter Name | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Method Type Parameter Name | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |

* **Imports**

|  |  |
| --- | --- |
| **Module** | **Properties** |
| Illegal Imports | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Redundant Imports | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Unused Imports | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |

* **Coding Problems**

|  |  |
| --- | --- |
| **Module** | **Properties** |
| Empty Statement | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Illegal Instantiation | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Illegal Catch | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Illegal Throws | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| Illegal Type | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| One Statement Per Line | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |

* **Miscellaneous**

|  |  |
| --- | --- |
| **Module** | **Properties** |
| Descendent Token Check | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |

* **Other**

|  |  |
| --- | --- |
| **Module** | **Properties** |
| Checker | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |
| TreeWalker | **severity:** warning  *Tutto il resto deve essere lasciato di default.* |

Nel caso ci si trovi di fronte ad istruzioni semplici, ogni linea deve contenere al massimo una sola istruzione.

Mentre, nel caso ci si trovi di fronte ad istruzioni composte vanno rispettate le seguenti regole:

1. Le istruzioni racchiuse all’interno di un blocco (esempio: for), devono essere indentate di un’unità all’interno dell’istruzione composta;
2. La parentesi di apertura del blocco deve trovarsi alla fine della riga dell’istruzione composta.

NOTE: I nomi delle classi dovrebbero essere semplici, descrittivi e che rispettino il dominio applicativo. Non andrebbero usati underscore per legare nomi. I nomi dei metodi non devono contenere caratteri speciali.

# **1.3.2 Pagine lato server e JSP**

Le pagine JSP devono, quando eseguite, produrre un documento conforme allo standard HTML5. Il codice Java delle pagine deve aderire alle convenzioni per la codifica in Java, con le seguenti puntualizzazioni:

1. Il tag di apertura (<%) è seguito immediatamente dalla fine della riga;

2. Il tag di chiusura (%>) si trova all'inizio di una riga;

3. È possibile evitare le due regole precedenti, se il corpo del codice Java consiste in una singola istruzione (<%= %>)

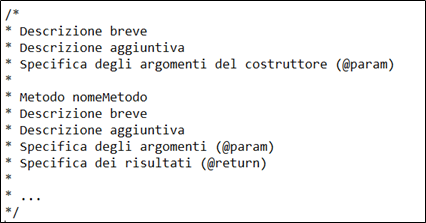
Le pagine HTML, sia in forma statica che dinamica, devono essere conformi allo standard HTML5. Inoltre, il codice HTML statico deve utilizzare l'indentazione, per facilitare la lettura, secondo le seguenti regole:

1. Un'indentazione consiste in una tabulazione;
2. Ogni tag deve avere un'indentazione maggiore del tag che lo contiene;
3. Ogni tag di chiusura deve avere lo stesso livello di indentazione del corrispondente tag di apertura;
4. I tag di commento devono seguire le stesse regole che si applicano ai tag normali.

# **1.3.3 Script Javascript**

Gli Script in JavaScript devono rispettare le seguenti convenzioni:

1. Il codice JavaScript deve seguire le stesse convenzioni per il layout e i nomi del codice Java;
2. Gli oggetti JavaScript devono essere preceduti da un commento in stile JavaDoc, che segue il seguente formato:



# **1.3.4 Fogli di stile CSS**

I fogli di stile (CSS) devono seguire le seguenti convenzioni:

1. Lo stile delle varie pagine.jsp deve far capo ad un foglio di stile denominato style.css, il quale rappresenterà nel suo insieme lo stile per ogni componente presente all’interno di tutte le pagine;
2. Al capo di ogni componente specificata all’interno del foglio di stile denominato style.css, ci sarà un relativo commento adatto a specificare il nome della componente di riferimento nella pagina.jsp;
3. Le pagine che richiedono uno stile personalizzato rispetto a quello realizzato per l’intero progetto, devono essere specificate all’interno della pagina fra i tag <head></head>;
4. Al fine di garantire un sistema responsive e adattabile sui diversi dispositivi si potrà fare riferimento ad un file responsive.css, il quale, al suo interno, avrà il codice specifico per ogni componente presente all’interno del sistema;
5. Tutte le pagine che visualizzeranno uno slider di immagini dovranno far capo ad un file nivo-slider-theme.css, il quale, al suo interno, avrà il codice specifico per formattare lo slider.

# **1.3.5 Database SQL**

I nomi delle tabelle devono seguire le seguenti regole:

1. Devono essere costituiti da: nomeDatabase.NomeTabella
2. Il nome deve essere un sostantivo singolare tratto dal dominio del problema ed esplicativo del contenuto;

I nomi dei campi devono seguire le seguenti regole:

1. Devono indicare lo scopo per il quale essi esistono;
2. I nomi dei campi devono contenere l’iniziale maiuscola e se essi sono composti da più di una parola, le parole interne devono avere l’iniziale maiuscola;
3. Tutti i comandi e le funzioni SQL devono essere scritte in maiuscolo;

I nomi dei campi che fungono da chiave esterna devono seguire la seguente regola:

1. I nomi dei campi devono seguire la seguente convenzione:

NomeTabellaDiRiferimento\_NomeCampo

# **1.4 Design Pattern**

# **1.4.1 MVC**

Il design pattern MVC consente la suddivisione del sistema in tre blocchi principali: Model, View e Controller.

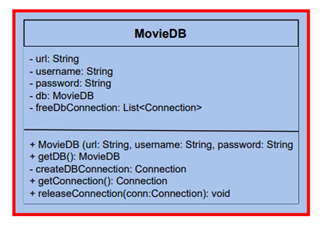
* Il Model fornisce i metodi di accesso ai dati persistenti;
* Il View si occupa dell’interazione con l’utente e della presentazione dei dati prelevati dal Model;
* Il Controller riceve i comandi dell’utente attraverso il View e modifica lo stato di quest’ultimo e del Model.

Nel nostro sistema le classi sono state divise in package avente nome richiamante il blocco di appartenenza nel design pattern.

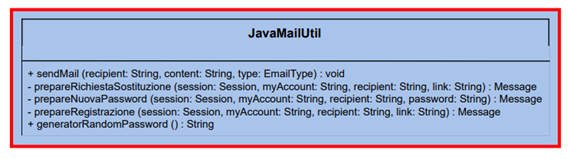
# **1.4.2 Singleton Pattern**

Abbiamo progettato una classe (MovieDB) per consentire l’esecuzione di tutte le operazioni con il Database.

Si è deciso di rendere questa classe una classe di tipo Singleton, poiché occorre avere un controllo centralizzato delle connessioni al Database in modo da ridurre le risorse utilizzate nella creazione di nuove connessioni.

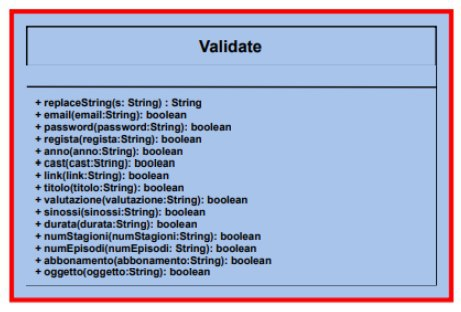


Abbiamo progettato una classe (JavaMailUtil) per consentire l’invio di mail quando si effettuano operazioni che, implicano una notifica informativa come la registrazione di un nuovo account o un link cliccabile per il ripristino della password. Per evitare ridondanze e/o mancata coerenza tra più istanze si è deciso di sviluppare questa classe come Singleton.



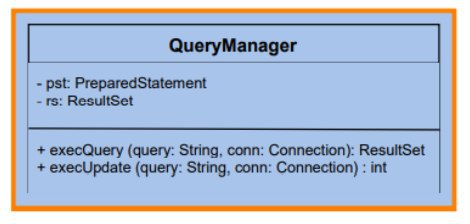
Abbiamo progettato una classe (Validate) per consentire la validazione degli input inseriti dall’Utente Generico controllando che siano conformi alle specifiche di formato. Per evitare di avere più istanze di questa classe, abbiamo deciso di renderla Singleton.

//Cambiare con int

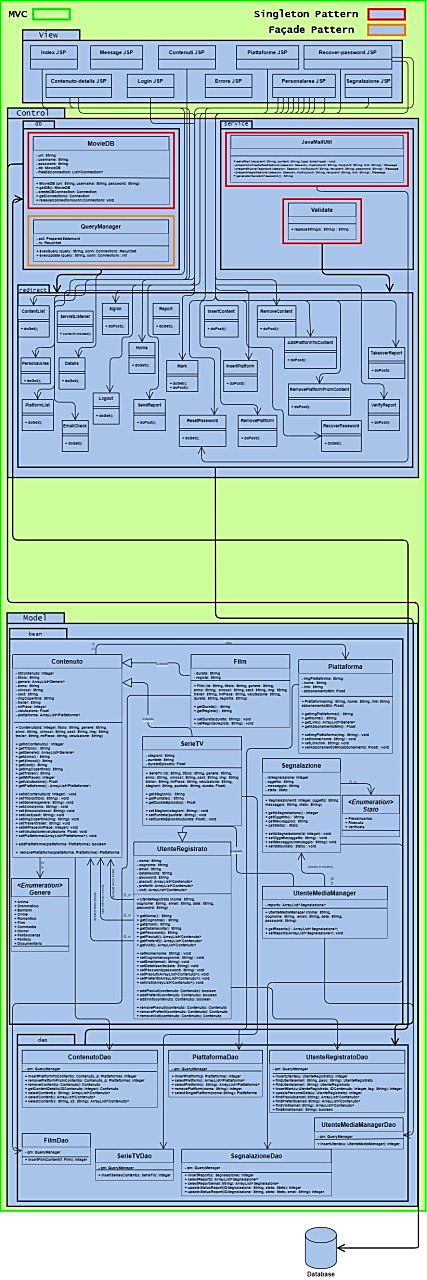


# **1.4.3 Façade Pattern**

Abbiamo progettato una classe (QueyManager) per consentire l’esecuzione delle operazioni all’interno del Database. Al fine di semplificare la struttura delle singole classi Dao, abbiamo deciso di rendere la classe QueryManager un Façade Pattern. Essa si comporta come un’interfaccia semplice che consente ai Dao di eseguire operazioni sul Database utilizzando i metodi messi a disposizione dalla stessa



Vista completa del sistema:



# **1.5 Definitions, acronyms, and abbreviations**

Bootstrap: è una raccolta di strumenti liberi per la creazione di siti e applicazioni per il web.

Tomcat: Apache Tomcat è un web server open source. Implementa le specifiche JavaServer Pages (JSP) e servlet, fornendo quindi una piattaforma software per l'esecuzione di applicazioni Web sviluppate in linguaggio Java.

JSP: acronimo di Java Scripting Preprocessor, è una tecnologia di programmazione web in java per lo sviluppo della logica di presentazione (tipicamente secondo il pattern MVC) di applicazioni web.

# **2. Packages**

# **2.1 View**

Il package view è composto dalle seguenti classi:

Index JSP, Message JSP, Contenuti JSP, Piattaforme JSP, Contenuto-details JSP, Login JSP, Errore JSP, Personalarea JSP, Segnalazione JSP, Recover-password JSP.

**Index JSP:** Accoglie l’utenza sul sito web.

**Message JSP:** Mostra all’utenza messaggi da parte del sistema.

**Contenuti JSP:** Mostra all’utenza un elenco di contenuti multimediali (Serie tv e film).

**Piattaforme JSP:** Mostra all’utenza l’elenco delle piattaforme di streaming presenti sul sito.

**Contenuto-details JSP:** Mostra all’utenza i dettagli di un contenuto da esso selezionato.

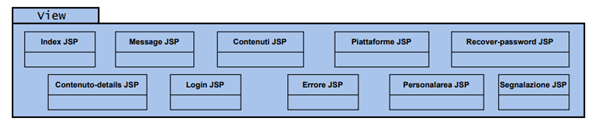
**Login JSP:** Consente all’utenza di accedere alla sua area personale, se in possesso delle credenziali, o di registrarsi nel caso in cui non sia in possesso delle credenziali.

**Errore JSP:** Mostra all’utenza gli errori di sistema.

**Personalarea JSP:** Mostra all’utente registrato le informazioni personali memorizzate all’interno del sistema.

**Segnalazione JSP:** Permette all’utente standard di inviare segnalazioni.

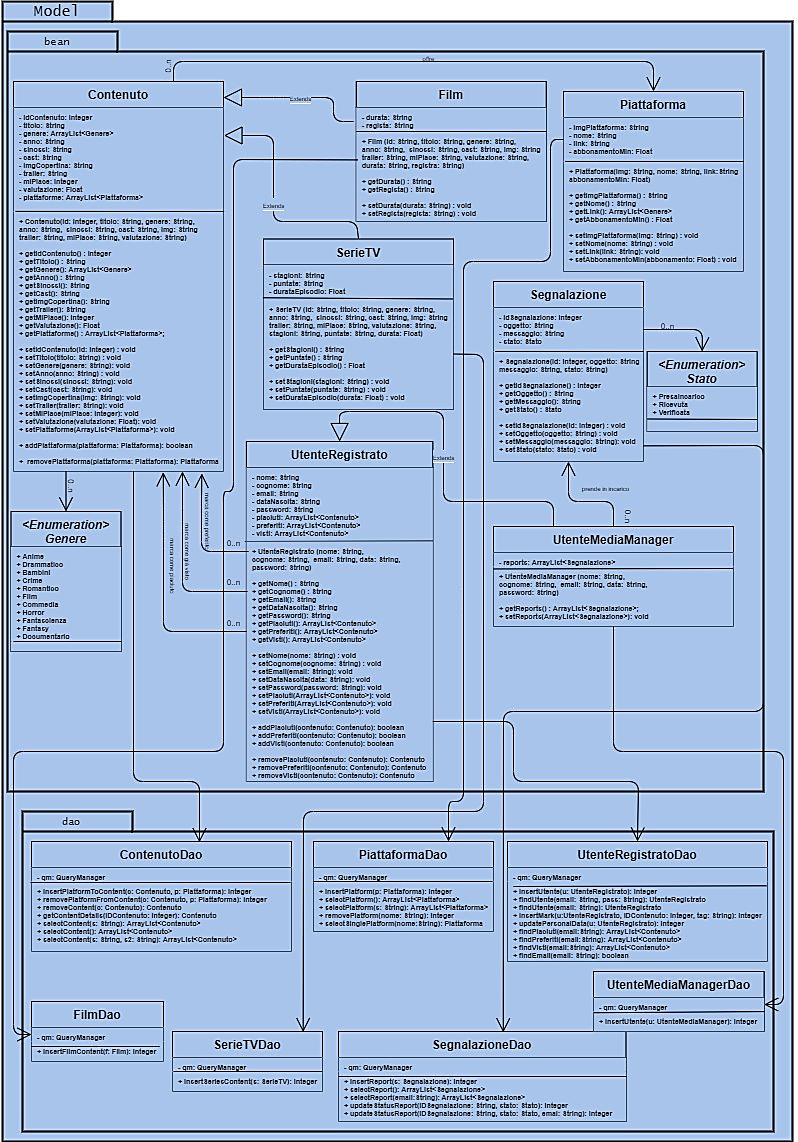
**Recover-password JSP:** Permette all’utente registrato di ricevere una nuova password in caso di smarrimento.



# **2.2 Model**

Il package model contiene tutti i dati e le classi per la gestione dei dati persistenti.

Le classi contenute all’interno di questo package fanno parte a loro volta di altri package come: bean (al cui interno figurano classi rappresentanti entità del Database) e dao (al cui interno figurano classi rappresentanti le operazioni CRUD sulle entità del Database).



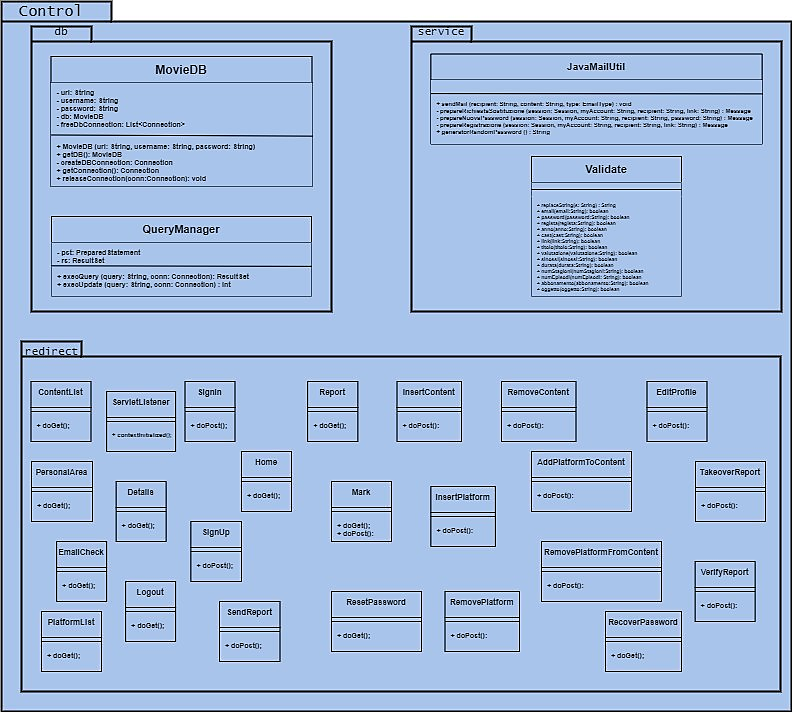
# **2.3 Control**

Il package Control riceve, tramite il pacchetto View, i comandi dell’Utente Generico. Esso è composto dal pacchetto redirect il quale riceve dati dal pacchetto db e service ed è formato da 23 Servlet riportate di seguito:

* ContentList (si occupa di mostrare un elenco di contenuti);
* PlatformList (si occupa di mostrare un elenco di piattaforme);
* ServletListener (si occupa di inizializzare il contesto per il Database);
* Details (si occupa di mostrare i dettagli di un contenuto);
* SignIn (si occupa di accedere ad un account);
* Report (si occupa di reindirizzare l’Utente Registrato verso la pagina per l’invio delle segnalazioni);
* Mark (si occupa di rendere persistenti sul Database le marcature dei contenuti);
* Home (si occupa di accogliere l’Utente Generico sull’applicazione web);
* ResetPassword (si occupa di gestire il ripristino della password);
* SendReport (si occupa di gestire l’invio delle segnalazioni);
* EmailCheck (verifica che l’email inserita dall’Utente Registrato sia presente nel sistema e in caso positivo invia una mail all’Utente Registrato per il ripristino della password);
* VerifyReport (si occupa di verificare una segnalazione);
* TakeoverReport (si occupa della presa in incarico una segnalazione);
* InsertContent (si occupa dell’inserimento di un contenuto);
* InsertPlatform (si occupa dell’inserimento di una piattaforma);
* AddPlatformToContent (si occupa dell’aggiunta di una piattaforma ad un contenuto);
* RemovePlatformFromContent (si occupa della rimozione di una piattaforma da un contenuto);
* RemovePlatform (si occupa della rimozione di una piattaforma);
* RemoveContent (si occupa della rimozione di un contenuto);
* PersonalArea(si occupa di reindirizzare l’Utente Registrato verso la propria area personale);
* RecoverPassword(si occupa di reindirizzare l’Utente Registrato verso la pagina riservata al ripristino della password);
* Logout (si occupa di disconnettere un account);

Il pacchetto db che figura all’interno del package Control contiene le due classi Singleton: MovieDB e QueryManager descritte nel paragrafo 1.4.2 del documento.

Il pacchetto service che figura all’interno del package Control contiene le due classi Singleton: JavaMailUtil e Validate descritte nel paragrafo 1.4.2 del documento.



# **3. Class Interfaces**

|  |  |
| --- | --- |
| **Nome classe** | **Index JSP** |
| Descrizione | Questa classe accoglie l’Utente Generico sul sito e mostra un carosello di immagini e le FAQ del sistema.  Nel caso di un Utente Generico essa reindirizza alle seguenti pagine:   * **contenuti.jsp** * **piattaforme.jsp** * **login.jsp**   Nel caso di un Utente Registrato che ha effettuato il login, oltre ai reindirizzamenti sopra descritti, può dirigersi in:   * **personalarea.jsp** |
| Pre-condizione | NA |
| Post-condizione | NA |
| Invarianti | NA |

|  |  |
| --- | --- |
| **Nome classe** | **Contenuti JSP** |
| Descrizione | Questa classe rappresenta il gestore della funzionalità relativa all’Utente Generico inerente al processo di visualizzazione di un elenco di contenuti.  Nel caso di un Utente Generico essa reindirizza alle seguenti pagine:   * **contenuto-details.jsp** * **piattaforme.jsp** * **login.jsp** * **index.jsp**   Nel caso di un Utente Registrato che ha effettuato il login, oltre ai reindirizzamenti sopra descritti, può dirigersi in:   * **segnalazione.jsp** * **personalarea.jsp** |
| Pre-condizione | context Contenuti JSP: selectContent(); |
| Post-condizione | NA |
| Invarianti | NA |

|  |  |
| --- | --- |
| **Nome classe** | **Piattaforme JSP** |
| Descrizione | Questa classe rappresenta il gestore della funzionalità relativa all’Utente Generico inerente al processo di visualizzazione di un elenco di piattaforme.  Nel caso di un Utente Generico essa reindirizza alle seguenti pagine:   * **login.jsp** * **index.jsp** * **contenuti.jsp** |
| Pre-condizione | context Piattaforma JSP: selectPlatform(); |
| Post-condizione | NA |
| Invarianti | NA |

|  |  |
| --- | --- |
| **Nome classe** | **Personalarea JSP** |
| Descrizione | Questa classe rappresenta il gestore della funzionalità relativa all’Utente Registrato, inerente al processo di visualizzazione della propria area personale.  Nel caso di un Utente Registrato essa reindirizza alle seguenti pagine:   * **contenuti.jsp** * **piattaforme.jsp** * **index.jsp** |
| Pre-condizione | context Personalarea JSP:  j\_security\_check(email, password); |
| Post-condizione | NA |
| Invarianti | NA |

|  |  |
| --- | --- |
| **Nome classe** | **Login JSP** |
| Descrizione | Questa classe rappresenta il gestore della funzionalità relativa all’Utente Generico, inerente al processo di visualizzazione di un form di registrazione e di un form relativo al login.  Nel caso di un Utente Generico essa reindirizza alle seguenti pagine:   * **contenuti.jsp** * **piattaforme.jsp** * **login.jsp** * **index.jsp** * **recover-password.jsp** |
| Pre-condizione | context Login JSP:  findUtente(email, password);  pre:  (email!=null && password!=null)  (email.equals(MovieDB.email) && (password.equals(MovieDB.password) |
| Post-condizione | NA |
| Invarianti | NA |

|  |  |
| --- | --- |
| **Nome classe** | **Recover-password JSP** |
| Descrizione | Questa classe rappresenta il gestore della funzionalità relativa all’Utente Generico inerente al recupero della password.  Nel caso di un Utente Generico essa reindirizza alle seguenti pagine:   * **contenuti.jsp** * **piattaforme.jsp** * **login.jsp** * **index.jsp** |
| Pre-condizione | pre:  (email!=null)  (email.equals(MovieDB.email) |
| Post-condizione | NA |
| Invarianti | NA |

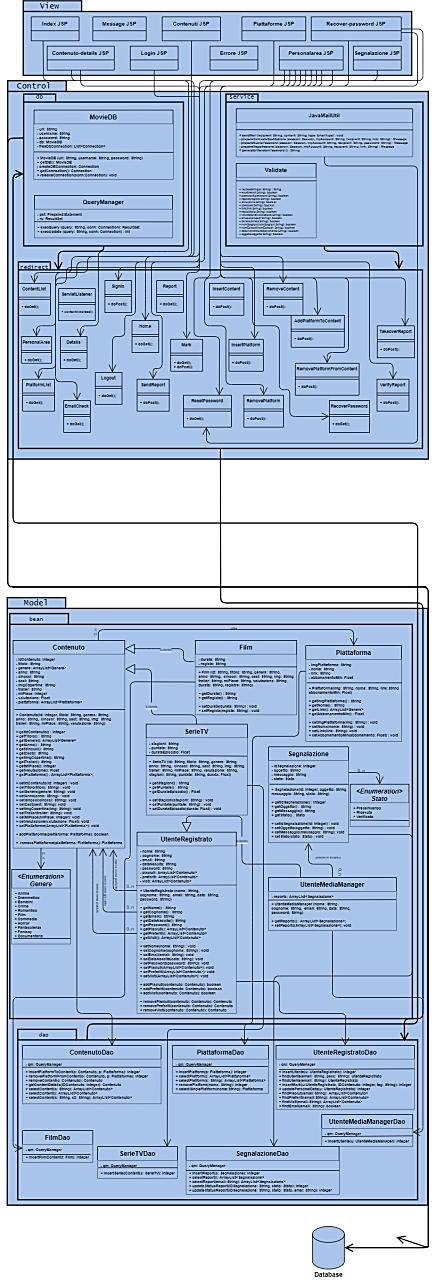
|  |  |
| --- | --- |
| **Nome classe** | **Message JSP** |
| Descrizione | Questa classe rappresenta il gestore della funzionalità relativa all’Utente Generico inerente al processo di visualizzazione di un messaggio di notifica.  Nel caso di un Utente Generico essa reindirizza alle seguenti pagine:   * **contenuti.jsp** * **piattaforme.jsp** * **login.jsp** * **index.jsp**   Nel caso di un Utente Registrato che ha effettuato il login, oltre ai reindirizzamenti sopra descritti, può dirigersi in:   * **personalarea.jsp** |
| Pre-condizione | context Message JSP:  request.getAttribute(); |
| Post-condizione | NA |
| Invarianti | NA |

|  |  |
| --- | --- |
| **Nome classe** | **Errore JSP** |
| Descrizione | Questa classe rappresenta il gestore della funzionalità relativa all’Utente Generico, inerente al processo di visualizzazione di un errore catturato dal sistema.  Nel caso di un Utente Generico essa reindirizza alle seguenti pagine:   * **contenuti.jsp** * **piattaforme.jsp** * **login.jsp** * **index.jsp**   Nel caso di un Utente Registrato che ha effettuato il login, oltre ai reindirizzamenti sopra descritti, può dirigersi in:   * **personalarea.jsp** |
| Pre-condizione | context Errore JSP:  requestScope(Exception); |
| Post-condizione | NA |
| Invarianti | NA |

|  |  |
| --- | --- |
| **Nome classe** | **Contenuto-details JSP** |
| Descrizione | Questa classe rappresenta il gestore della funzionalità relativa all’Utente Generico, inerente al processo di visualizzazione dei dettagli di un contenuto specifico.  Nel caso di un Utente Generico essa reindirizza alle seguenti pagine:   * **contenuti.jsp** * **piattaforme.jsp** * **login.jsp** * **index.jsp**   Nel caso di un Utente Registrato che ha effettuato il login, oltre ai reindirizzamenti sopra descritti, può dirigersi in:   * **personalarea.jsp** |
| Pre-condizione | context Contenuto-details JSP:  findContenuto(IDContenuto); |
| Post-condizione | NA |
| Invarianti | NA |

|  |  |
| --- | --- |
| **Nome classe** | **Segnalazione JSP** |
| Descrizione | Questa classe rappresenta il gestore della funzionalità relativa all’Utente Registrato inerente al processo per l’invio di una segnalazione.  Nel caso di un Utente Registrato che ha effettuato il login, essa reindirizza alle seguenti pagine:   * **contenuti.jsp** * **piattaforme.jsp** * **personalarea.jsp** * **index.jsp** |
| Pre-condizione | pre:  (oggetto!=null && messaggio!=null) |
| Post-condizione | La segnalazione è stata inviata al sistema. |
| Invarianti | NA |

# **4. Class Diagram**



# **5. Glossary**

lowerCamelCase: è una tecnica di naming delle variabili adottata dallo standard Google Java. Essa consiste nello scrivere più parole insieme delimitando la fine e l’inizio di una nuova parola con la lettera maiuscola.

Tomcat: Apache Tomcat è un web server open source. Implementa le specifiche JavaServer Pages (JSP) e servlet, fornendo quindi una piattaforma software per l'esecuzione di applicazioni Web sviluppate in linguaggio Java.

Off-The-Shelf: Servizi esterni al sistema di cui viene fatto utilizzo.