Laurea Triennale in informatica - Università di Salerno
Corso di Ingegneria del Software - Prof Carmine Gravino

Object Design Document

Movie Club

Riferimento	
Versione	1.0
Data	05/01/2024
Destinatario	Studenti di Ingegneria del Software 2023/24
Presentato da	NC7 Salurso, Paciello, Guida, Faraco
Approvato da	

Team Composition NC7

NOME	MATRICOLA	E-MAIL
Andrea Salurso	0512113694	A.SALURSO3@STUDENTI.UNISA.IT
Gaetano Vito Faraco	0512114147	G.FARACO1@STUDENTI.UNISA.IT
Costantino Paciello	0512113661	C.PACIELLO7@STUDENTI.UNISA.IT
Vittorio Guida	0512115293	V.GUIDA16@STUDENTI.UNISA.IT

Revision History

Data	Versione	Descrizione	Autori
22/12/2023	0.1	Introduzione	Salurso, Faraco
23/12/2023	0.2	Packages	TUTTI I MEMBRI
27/12/2023	0.3	Class Interfaces	Paciello, Guida
27/12/2023	0.3.1	Class Diagram Ristrutturato	TUTTI I MEMBRI
28/12/2023	0.4	Adapter	Paciello, Guida
28/12/2023	0.4.1	Glossario	Salurso, Guida
05/01/2024	1.0	Revisione	TUTTI I MEMBRI

Sommario

1. Introduzione

Lo scopo di MovieClub è fornire agli appassionati del cinema un'esperienza cinematografica completa ed interattiva. La piattaforma permette agli utenti di esplorare una vasta raccolta di film, suddivisi per categoria e genere. MovieClub è un luogo dove gli utenti possono lasciare recensioni dettagliate e valutazioni, contribuendo a creare una community di cinefili appassionati. MovieClub si propone di essere il punto di riferimento per chi ama il cinema, offrendo un ambiente coinvolgente ed organizzato.
In questo documento verranno evidenziate le linee guida per la scrittura del codice; inoltre, verranno mostrate le definizioni, gli acronimi e le abbreviazioni presenti nel documento, senza tralasciare riferimenti e link utili.

1.1 Linee guida per la documentazione dell'interfaccia

1.1.1 Nomi dei file

- Le classi devono avere nomi al singolare.
- I metodi devono essere scritti in inglese, la specifica sarà invece in italiano.
- I nomi dei file sorgente Java devono essere uguali al nome della classe top level.
- I nomi dei file generati da compilatori, sistemi di build, o altri tool compariranno con il loro nome di default.
- I nomi delle classi di test di unità devono avere il suffisso "Test".
- Per tutte le altre situazioni, i nomi dei file rifletteranno il loro contenuto e saranno composti utilizzando lettere minuscole, maiuscole, cifre e underscore.

1.1.2 Struttura dei file sorgente

La struttura dei file sorgente viene definita dal sistema di build "Maven".

- I file relativi all'implementazione del sistema avranno la seguente struttura:
  - *src/main/java/{package}/{file}.java*

- I file relativi alle VIEW avranno la seguente struttura:
  - *src/main/webapp/WEB-INF/gui/{file}.jsp*
  - *src/main/webapp/ WEB-INF/guiAdmin/{file}.jsp*
  - *src/main/webapp/ WEB-INF/navbar/{file}.jsp*

- I file di stile CSS avranno la seguente struttura:
  - *src/main/webapp/css/{file}.css*

- I file JAVASCRIPT avranno la seguente struttura:
  - *src/main/webapp/js/{file}.js*

- I file relativi al TESTING avranno la seguente struttura:
  - *src/test/java/{package}/{fileTest}.java*

1.1.3 Formattazione

Per la formattazione dei file Java, si adotteranno le convenzioni di stile della Sun di Java.
Per i file XML, HTML5, CSS e JS si farà uso del formatter di default di IntelliJ.

1.1.4 Dichiarazioni

È consentito che ogni dichiarazione di variabile locale definisca più di una variabile, mentre ogni dichiarazione di variabile di istanza deve definire soltanto una variabile.
Le variabili di istanza devono essere dichiarate come private.
Ogni dichiarazione di variabile locale deve essere seguita da un'inizializzazione, che può avvenire nella stessa linea o nella successiva.

1.1.5 Nomenclatura

Di seguito sono elencati i vincoli di nomi relativi ai nomi delle componenti software del sistema:

- Package: solo lettere in < *lowerCamelCase* >
- Classi: solo lettere in < *UpperCamelCase* >
- Metodi: solo lettere in < *lowerCamelCase* >, devono contenere nel nome solo verbi e nomi degli attributi della classe
- Costanti: solo lettere e underscore in < CONSTANT_CASE >
- Variabili: solo lettere in < *lowerCamelCase* >
- Parametri: solo lettere in < *lowerCamelCase* >. In particolare, dello stesso nome delle relative variabili di istanza nei metodi setter e nei costruttori, possono essere solo sostantivi.

1.1.6 Convenzioni

Convenzioni per la scrittura del codice:

- Le condizioni di errore lanciano eccezioni e non valori di ritorno.
- Uso del for-each loop quando bisogna iterare per intero una collezione iterabile.
- Gli IF sui boolean non devono avere == true / == false.

1.1.7 Documentazione del codice

I commenti di implementazione sono stati effettuati all'interno del codice da parte dei Team Member.

1.2 Definizioni, acronimi e abbreviazioni

DEFINIZIONE	DESCRIZIONE
Package	raggruppamento di classi, interfacce o file correlati.
Design Pattern	template di soluzioni a problemi comuni, raffinate nel tempo dagli sviluppatori.
UpperCamelCase	pratica che prevede la scrittura di parole composte o frasi unendole tra loro, lasciando le loro iniziali maiuscole.
lowerCamelCase	pratica che prevede la scrittura di parole composte o frasi unendole tra loro, in modo tale che le parole in mezzo alla frase abbiano l'iniziale maiuscola.
Application Layer	nel pattern Three-layer rappresenta la parte che si occupa della logica di business dell'applicazione.
Presentation Layer	nel pattern Three-layer rappresenta la parte che si occupa della logica di visualizzazione dell'applicazione.
Storage Layer	nel pattern Three-layer rappresenta la parte che si occupa dell'interazione con il database.
CONSTANT_CASE	convenzione utilizzata per migliorare la leggibilità delle costanti e per distinguerle dalle variabili dove tutte le lettere di una parola sono in maiuscolo e le parole sono separate da underscore.

1.3 Riferimenti

Bernd Bruegge, Allen H. Dutoit – Object-Oriented Software Engineering: Using UML, Patterns, and Java.

- Documento di Statement of Work relativo al progetto MovieClub: **SOW**
- Requirements Analysis Document relativo al progetto MovieClub: **RAD**
- System Design Document relativo al progetto MovieClub: **SDD**

2. Packages

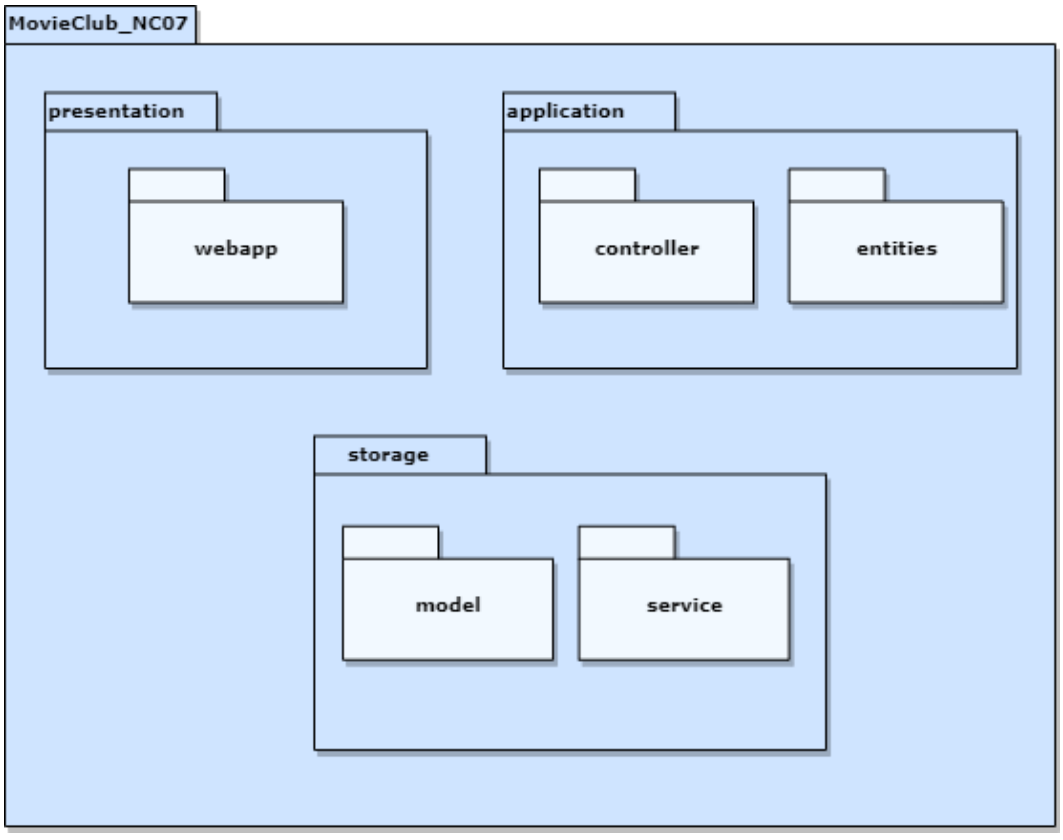
In questa sezione del documento verranno mostrate le divisioni del progetto in package in base all'architettura Three Layer definita nel SD. Il lato Server ha una struttura di directory standard definita da Maven contenente:

- **.idea**
- **.mvn** che contiene tutti i file di configurazione per Maven
- **src** contenente tutti i file sorgente:
    - **main**
        - **java:** contiene le classi Java relative alle componenti Control e Models
        - **resources** contenente tutte le immagini utilizzate
        - **webapp**
            - **css,** contiene tutti i file di stile
            - **js,** contiene i file javaScript
            - **WEB-INF**
                - **gui,** contiene tutte le pagine dinamiche relative alla View
                - **guiAdmin,** contiene tutte le pagine dinamiche relative alla View dell'Admin.
                - **navbar,** contiene tutte le pagine dinamiche relative alla navbar
    - **test,** contiene tutti i file necessari per il Testing
        - **java** contiene le classi Java per l'implementazione del testing.

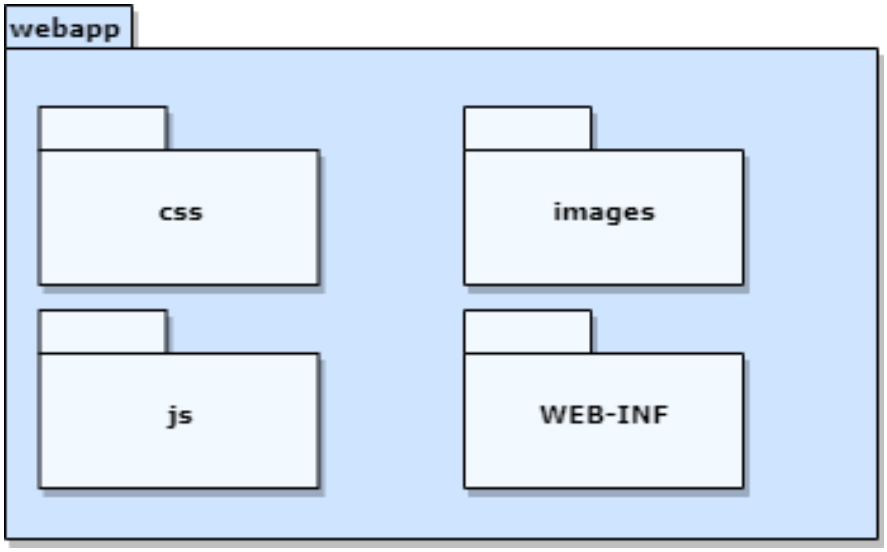
2.1 Package MovieClub

In questa sezione, illustreremo la struttura dei pacchetti del sistema. Tale suddivisione è stata adottata principalmente per due motivi:

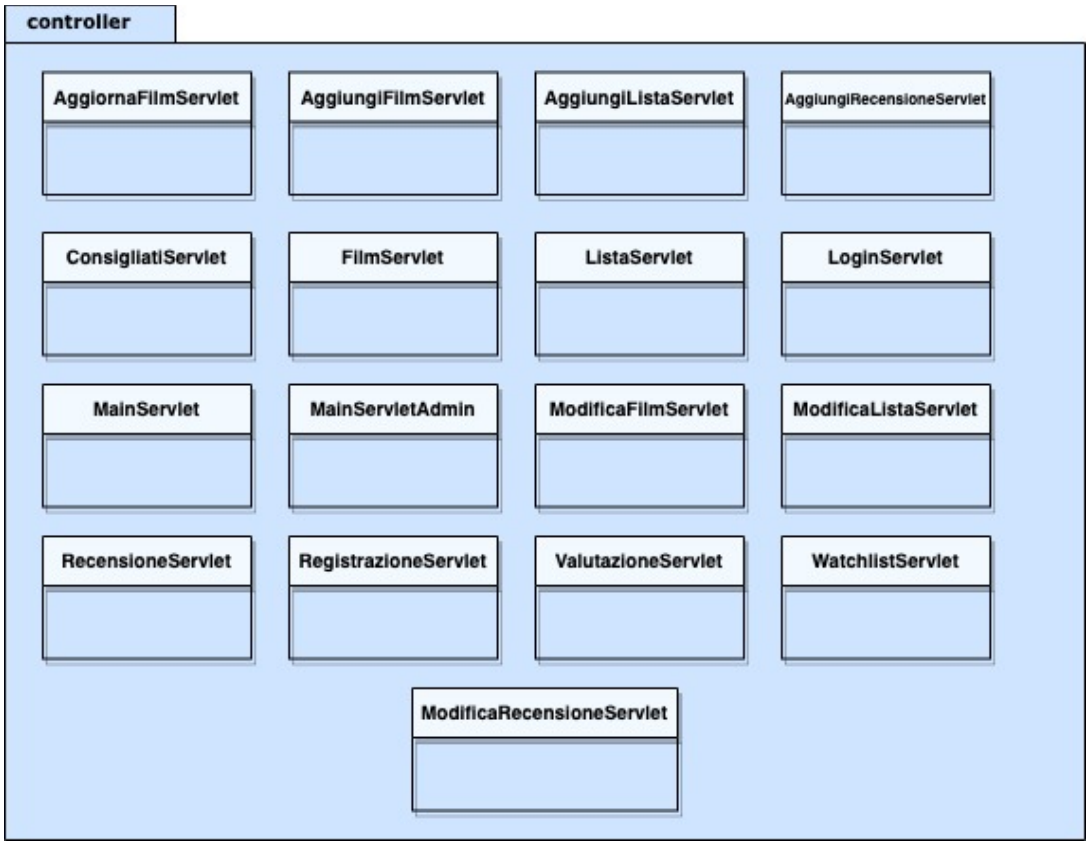
1. Separare le classi in conformità all'architettura definita nel System Design, al fine di raggruppare le classi con responsabilità simili nella stessa posizione;
2. Isolare il collegamento al database dalle query effettive eseguite su di esso, garantendo così che il codice non dipenda direttamente dal tipo di driver utilizzato per il collegamento.



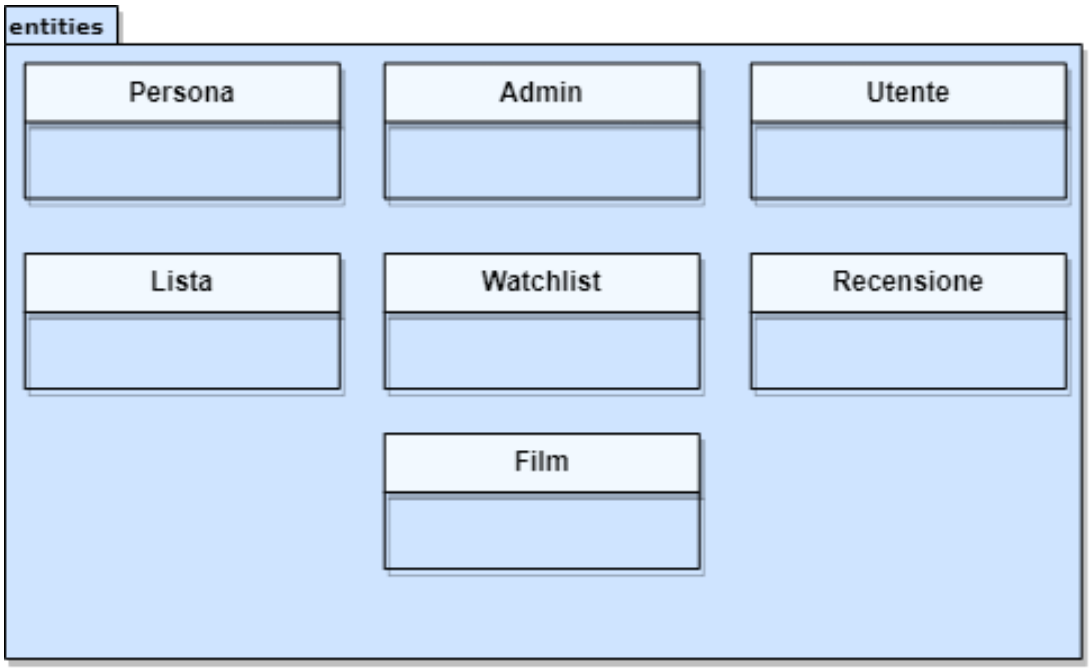
2.2 Package WebApp



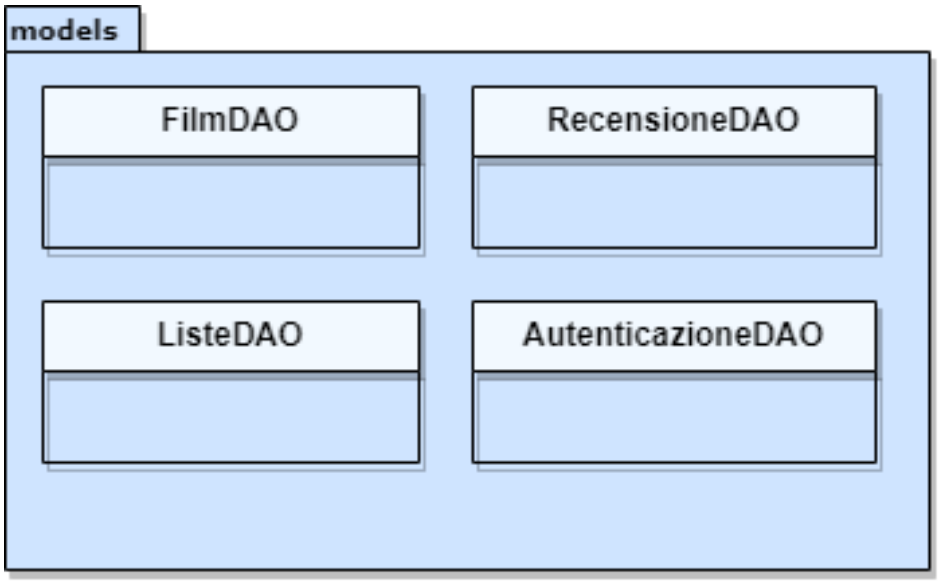
2.3 Package Controller



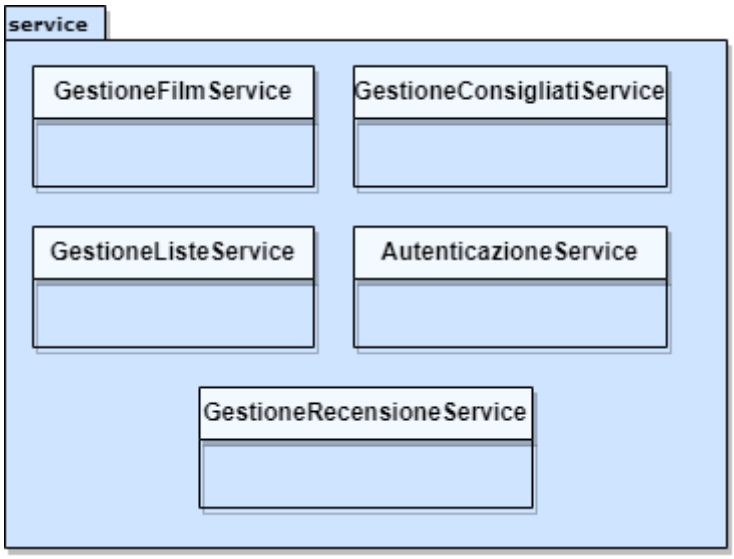
2.4 Package Entities



2.5 Package Models



2.6 Package Service



3 Class Interfaces

All'interno di questa sezione, verranno mostrate le interfacce del sistema, relative ai package sopra definiti.

Package controller

ListaServlet

Nome Classe	ListaServlet
Descrizione	Questa classe gestisce le varie funzionalità delle liste e permette il collegamento tra back-end e il front-end
Metodi	+ doPost(HttpServletRequest request, HttpServletResponse response) + doGet(HttpServletRequest request, HttpServletResponse response)
Invariante di classe	/

Nome Metodo	+ doPost(HttpServletRequest request, HttpServletResponse response)
Descrizione	Questo metodo gestisce le richieste POST in base al parametro "action" ed esegue le azioni per visualizzazione lista, informazioni lista, gestione lista, creazione lista e aggiunta di un film alla lista, reindirizzando l'utente alla pagina corretta.
Pre-condizione	/
Post-condizione	/

Nome Metodo	+ doGet(HttpServletRequest request, HttpServletResponse response)
Descrizione	Questo metodo richiama il doPost
Pre-condizione	/
Post-condizione	/

FilmServlet

Nome Classe	FilmServlet
Descrizione	Questa classe gestisce le varie funzionalità dei film e permette il collegamento tra back-end e il front-end
Metodi	+ doPost(HttpServletRequest request, HttpServletResponse response) + doGet(HttpServletRequest request, HttpServletResponse response)
Invariante di classe	/

Nome Metodo	+ doPost(HttpServletRequest request, HttpServletResponse response)
Descrizione	Reindirizza la richiesta alla pagina web infoFilm.jsp.
Pre-condizione	/
Post-condizione	/

Nome Metodo	+ doGet(HttpServletRequest request, HttpServletResponse response)
Descrizione	Questo metodo recupera le informazioni su un film, e le relative recensioni ad esso associate, le inserisce nella request e reindirizza l'utente alla pagina corrente.
Pre-condizione	/
Post-condizione	/

Package model

FilmDAO

Nome Classe	FilmDAO
Descrizione	Questa classe permette l'accesso al database relativo al modulo gestione film
Metodi	+doRetrieveByID(Int id): Film +doRetrieveAllFromYear2022(): List<Film> +doRetrieveAll(): List<Film> +doInsert(tutti i parametri del film): int +doUpdate(tutti i parametri del film): int +doDelete(int id): int
Invariante di classe	/

Nome Metodo	+doRetrieveByID(Int id): Film
Descrizione	Questo metodo restituisce l'oggetto Film con il campo 'ID' corrispondente all'identificativo fornito come intero.
Pre-condizione	idFilm != null
Post-condizione	/

Nome Metodo	+doRetrieveAllFromYear2022(): List<Film>
Descrizione	Questo metodo restituisce una lista contente tutti i Film usciti dal 2022 presenti sul database.
Pre-condizione	/
Post-condizione	/

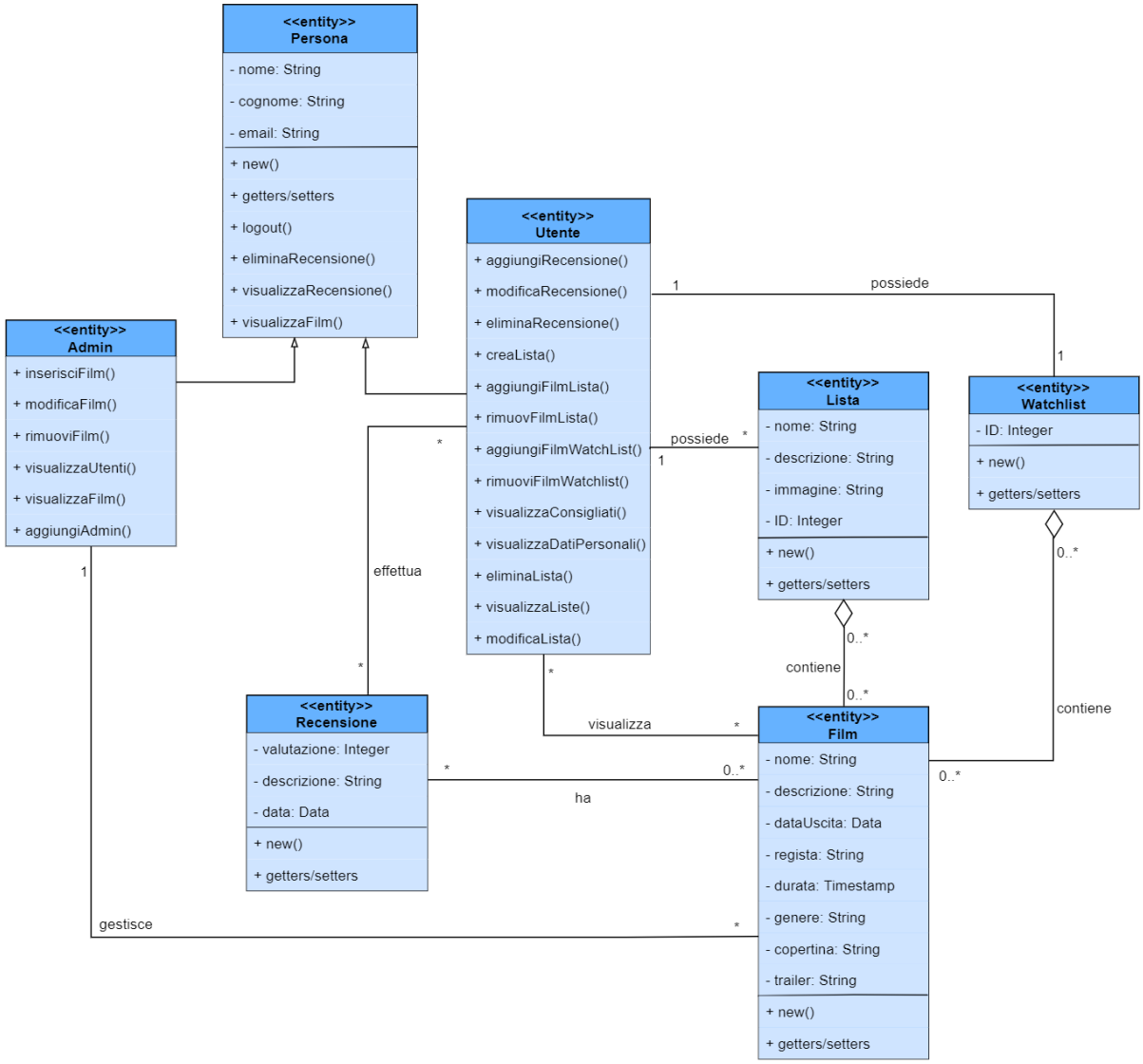
Nome Metodo	+doRetrieveAll(): List<Film>
Descrizione	Questo metodo restituisce una lista contente tutti i Film presenti sul database.
Pre-condizione	/
Post-condizione	/

Nome Metodo	+doInsert(Film f): int
Descrizione	Questo metodo permette l'inserimento di un nuovo film all'interno del database.
Pre-condizione	Non devono esistere film con lo stesso nome
Post-condizione	/

Nome Metodo	+doUpdate(Film f): int
Descrizione	Questo metodo permette la modifica di un film all'interno del database, tramite l'utilizzo del campo 'ID'.
Pre-condizione	idFilm != null
Post-condizione	/

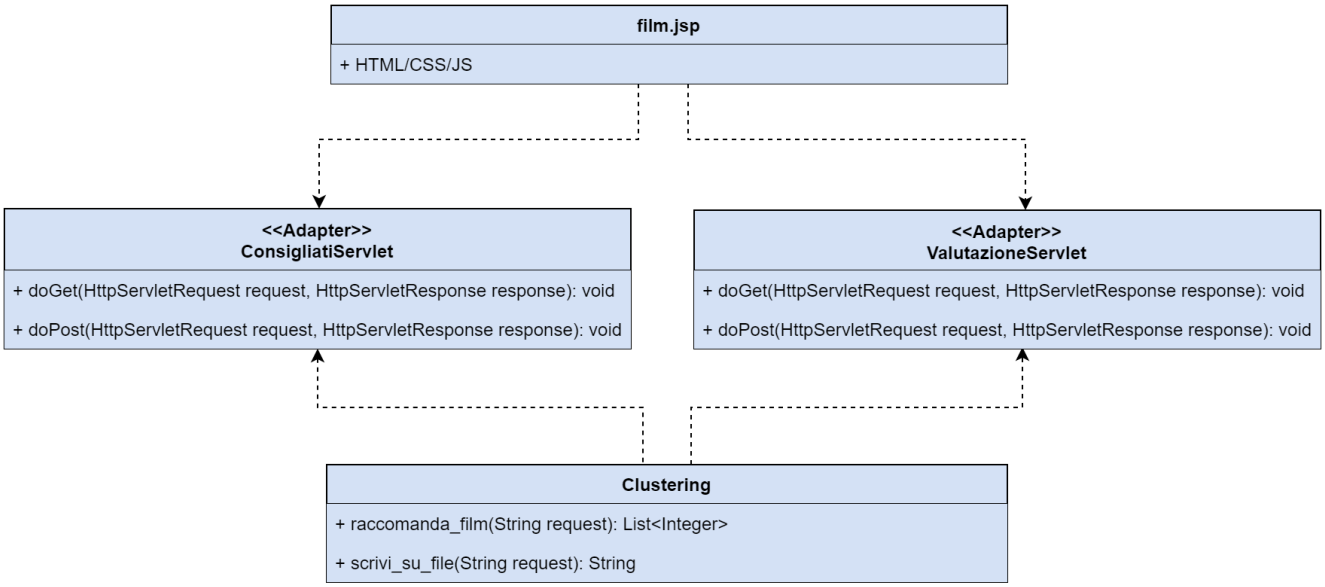
Nome Metodo	+doDelete(int id): int
Descrizione	Questo metodo permette l'eliminazione di un film dal database, tramite l'utilizzo del campo 'ID'.
Pre-condizione	idFilm != null
Post-condizione	/

4 Class Diagram Ristrutturato



5 Elementi di Riuso – Design Patterns [FIA]

Il design pattern strutturale "Adapter" ci offre la possibilità di agevolare la collaborazione tra oggetti con interfacce differenti mediante la creazione di classi Adapter, le quali saranno responsabili della manipolazione dei dati. L'Adapter seguente sarà impiegato nella trasformazione dei risultati generati dal modulo di intelligenza artificiale in dati manipolabili dal sistema, con l'obiettivo di ottimizzare questa fase di elaborazione.



6 Glossario

SIGLA/TERMINE	DEFINIZIONE
Package	Raggruppamento di classi ed interfacce
DAO	Data Access Object. Pattern per la gestione della persistenza
Controller	Classe che gestisce le richieste del Client
Model	Insieme dei metodi che permettono l'accesso ai dati.
Three – Layer	Modello di organizzazione del codice applicativo basato sulla separazione delle funzionalità logiche.
Adapter	Design Pattern che permette di collegare tra loro librerie con interfacce non compatibili.