Laurea in informatica-Università di Salerno
Corso di *Ingegneria del Software*- Prof. C. Gravino

Object Design Progetto GuardaTV

Riferimento	
Versione	1.2
Data	12/02/2022
Destinatario	Studenti di Ingegneria del Software 2021/22
Presentato da	Gruppo 16: N. Cacace S. Pastore A. Prezioso A. Ricchetti
Approvato da	

Revision History

Data	Versione	Descrizione	Autori
22/12/2021	0.1	Prima stesura	N. Cacace S. Pastore A. Prezioso A. Ricchetti
20/01/2022	0.2	Aggiunta contenuto	A. Prezioso
24/01/2022	0.3	Descrizione design pattern	N. Cacace
24/01/2022	0.4	ODD trade-off	N. Cacace S. Pastore A. Prezioso A. Ricchetti
25/01/2022	1.0	Aggiunta design goal Creazione tabelle Package GuardaTv	N. Cacace S. Pastore
10/01/2022	1.1	Formattazione, Indice e modifiche minori	S. Pastore
12/02/2022	1.2	Aggiunta link per documentazione Javadoc	S.Pastore N.Cacace

Sommario

1. Introduzione

Il sistema GuardaTV è stato creato con lo scopo di aiutare gli utenti del sito riguardo la scelta di un film o una serie TV da guardare. Infatti il sistema comprende un sistema di recensioni grazie al quale l'utente può informarsi riguardo un contenuto da usufruire. Inoltre GuardaTV permette la gestione di liste personalizzate.

1.1. Object design goal

Robustezza: il sistema deve risultare robusto, reagendo correttamente a situazioni impreviste attraverso la gestione delle eccezioni e il controllo degli errori che sarà ,dove possibile, implementato sia lato client che server.
Costo: il tempo per lo sviluppo di GuardaTv non deve superare le 50 ore di lavoro a persona .

1.2. Object design Trade-off

Durante la fase di analisi e di progettazione del sistema abbiamo individuato diversi compromessi per lo sviluppo del sistema. Anche durante la fase di Object Design sorgono diversi compromessi che andremo ad analizzare in questo paragrafo:

Criteri di manutenzione / Criteri di performance: Il sistema sarà implementato preferendo la manutenibilità alla performance in modo da facilitare gli sviluppatori nel processo di aggiornamento del software a discapito delle performance del sistema.

Interfaccia vs Usabilità: Il sistema verrà sviluppato con un interfaccia grafica realizzata in modo da poter essere molto semplice, chiara ed intuitiva. Nell'interfaccia saranno presenti form, menu e pulsanti, disposti in maniera da Rendere semplice l'utilizzo del sistema da parte dell'utente finale.
I font che abbiamo usato sono di tipo Sans Serif ,specificatamente Trebuchet e Tahoma come consigliato dalla British Dysleixia Association. Sono stati utilizzati pochi colori con alto contrasto in modo da essere percepiti in modo chiaro.

Sicurezza vs Efficienza: La sicurezza, come descritto nei requisiti non funzionali, rappresenta uno degli aspetti importanti del sistema. A causa dei tempi di sviluppo molto limitati, ci limiteremo ad implementare un sistema di sicurezza basato sull'utilizzo di username e password degli utenti, memorizzando l'hash della password con aggiunta di salting per aumentare la sicurezza dei dati sensibili degli utenti nel database. Inoltre per

l'esecuzione delle query sono stati utilizzati oggetti PreparedSatement per la protezione da attacchi SQL Injection.

1.3. Object design Trade-off

Gli sviluppatori seguiranno alcune linee guida per la scrittura del codice:

1.3.1. Naming convention

- **E' buona norma utilizzare nomi:**
  - Descrittivi
  - Pronunciabili
  - Di uso comune
  - Di lunghezza medio-corta
  - Non abbreviati
  - Evitando la notazione ungherese
  - Utilizzando solo caratteri consentiti (a-z, A-Z, 0-9)

- **Variabili**
  - I nomi delle variabili devono cominciare con una lettera minuscola, e le parole seguenti con la lettera maiuscola. Quest'ultime devono essere dichiarate ad inizio blocco, solamente una per riga e devono essere tutte allineate e facilitarne la leggibilità. Esse possono essere annotate con dei commenti.
  - E' inoltre possibile, in alcuni casi, utilizzare il carattere underscore ("_") per la definizione del nome.

- **Metodi**
  - I nomi dei metodi devono cominciare con una lettera minuscola, e le parole seguenti con la lettera maiuscola. Il nome del metodo tipicamente consiste in un verbo che identifica una azione, seguito dal nome di un oggetto.
  - I commenti dei metodi devono essere raggruppati in base alla loro funzionalità, la descrizione dei metodi deve apparire prima di ogni dichiarazione di metodo, e deve descriverne lo scopo. Deve includere anche informazioni sugli argomenti, sul valore di ritorno, e se applicabile, sulle eccezioni.

- **Classi e pagine**
  - I nomi delle classi e delle pagine devono cominciare con una lettera maiuscola, e anche le parole seguenti all'interno del nome devono cominciare con una lettera maiuscola. I nomi di quest'ultime devono fornire informazioni sul loro scopo.
  - La dichiarazione di classe deve essere caratterizzata da:
    - Dichiarazione della classe pubblica
    - Dichiarazioni di costanti
    - Dichiarazioni di variabili di classe
    - Dichiarazione di variabili d'istanza
    - Costruttore
    - Commento e dichiarazione dei metodi

1.4. Definizioni, acronimi e abbreviazioni

Package: raggruppamento di classi, interfacce o file correlati;

Design pattern: template di soluzioni a problemi ricorrenti impiegati per ottenere riuso e flessibilità

Javadoc: sistema di documentazione offerto da Java, che viene generato sottoforma di interfaccia in modo da rendere la documentazione accessibile e facilmente leggibile.

lowerCamelCase: è la pratica di scrivere frasi in modo tale che ogni parola o abbreviazione nel mezzo della frase inizi con una lettera maiuscola, senza spazi o punteggiatura intermedi;

UpperCamelCase: è la pratica di scrivere frasi in modo tale che ogni parola o abbreviazione inizi con una lettera maiuscola, senza spazi o punteggiatura intermedi;

ODD: Object Design Document

DBMS: DataBase Management System

CD: Class diagram

REQ: Requisito

SC: Scenario

1.5. Riferimenti

Object-Oriented Software Engineering Using UML, Patterns, and Java - 3rd Edition

Statement of work

Requirements analysis document

System design document

2. Package

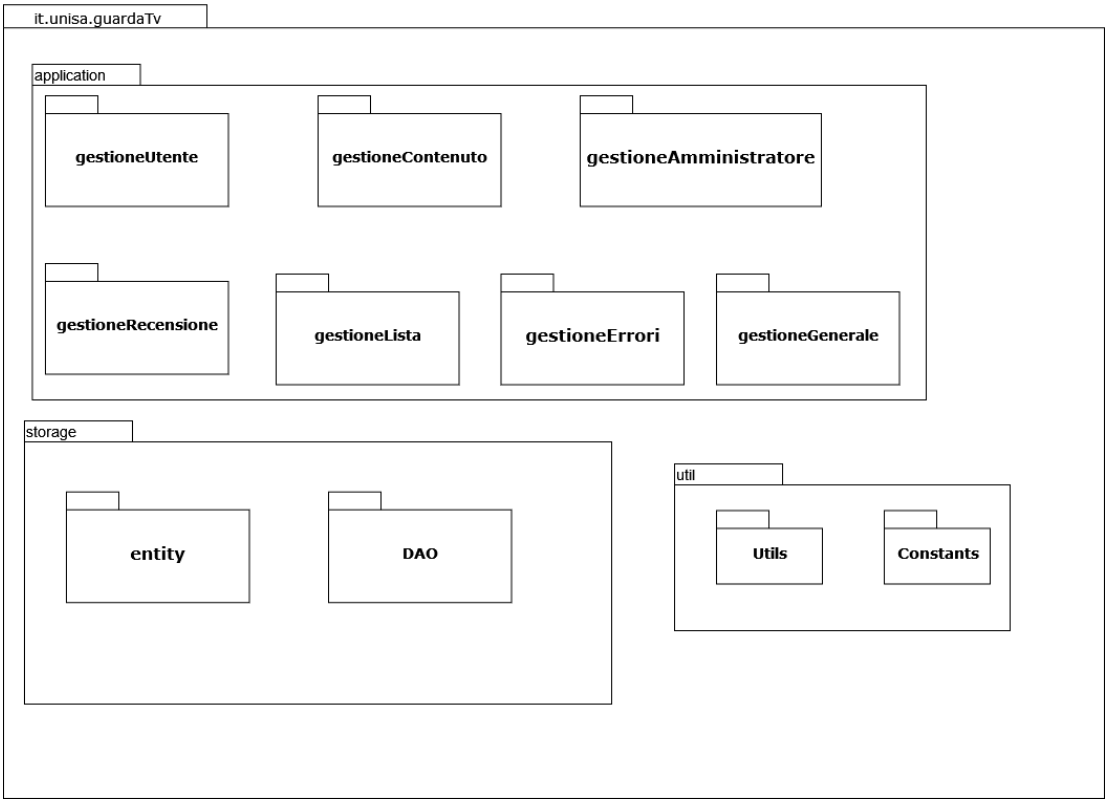
In questa sezione viene mostrata la suddivisione del sistema in package, in base a quanto definito nel documento di System Design. Tale suddivisione è motivata dalle scelte architetturali prese e ricalca la struttura di directory standard definita da Maven.

- .idea
- src
    - main
        - java, contiene le classi java relatie ai layer di application e storage
        - webapp, contiene i file relativi al layer di presentation
            - css, contiene i fogli di stile css
            - jsp, contiene i file jsp e html da inviare al client da mostrare all'utente
    - test, contiene tutto il necessario per il testing
        - java, contiene le classi java per l'implementazione del testing

2.1. Package Guardatv

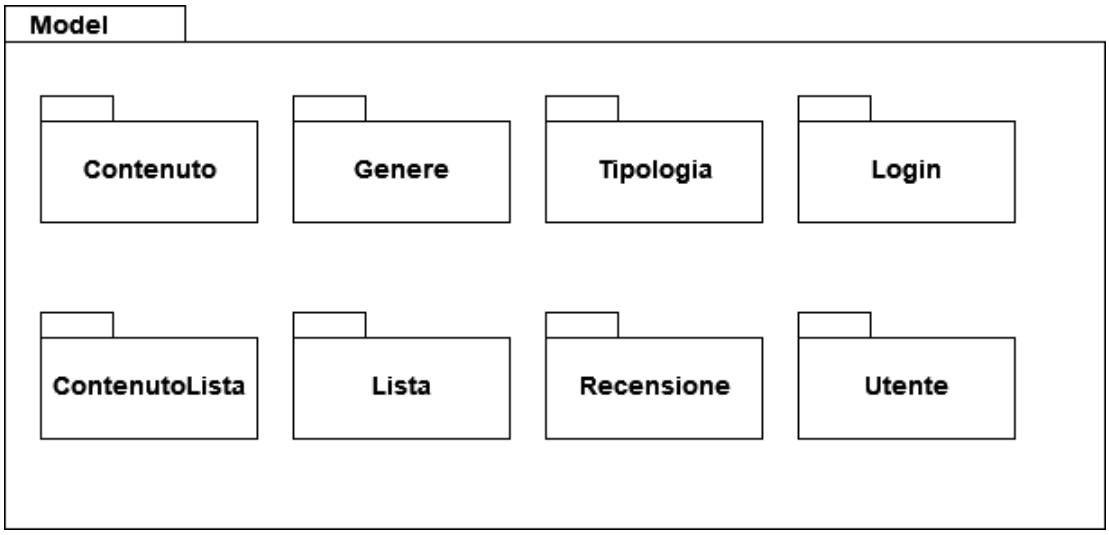
Nella presente sezione si mostra la struttura del package principale di Guardatv. La struttura generale è stata ottenuta a partire da tre principali scelte:

1. Creare un package separato per ogni sottosistema, contenente le classi service e controller del sottosistema, ed eventuali classi di utilità usate unicamente da esso
2. Creare un package separato per le classi del model, contenente le classi entity e i DAO per l'accesso al DB
3. Creare un package chiamato utils in cui inserire eventuali classi di utilità per il sistema e usabili da più sottosistemi.



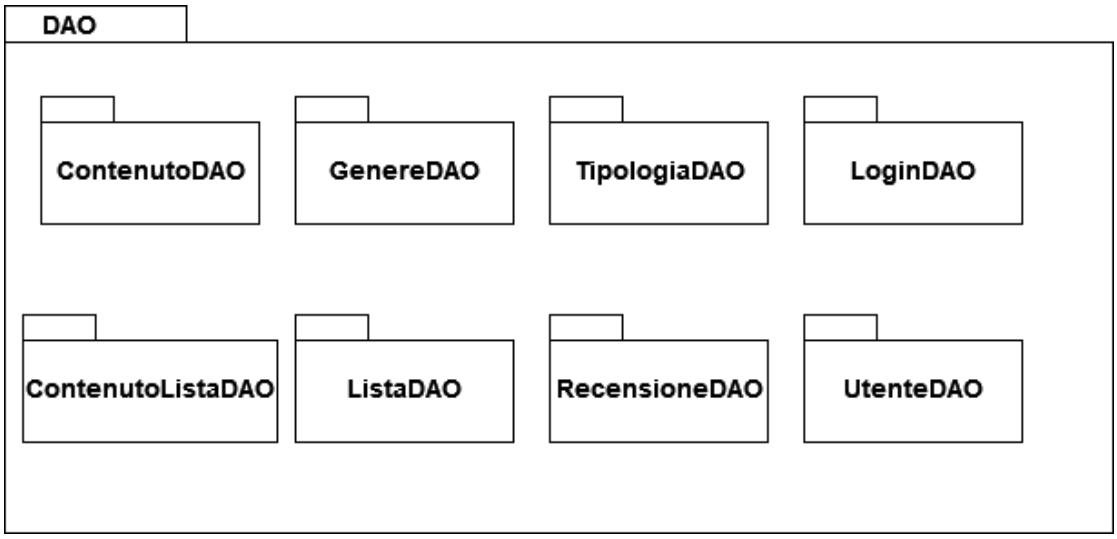
Storage Layer

Package model



Contenuto	Questa classe rappresenta il Contenuto digitale; che può essere un film o una serie.
ContenutoLista	Questa classe rappresenta il Contenuto presente in una determinata Lista.
Genere	Questa classe rappresenta il Genere.
Lista	Questa classe rappresenta la Lista di contenuti di un Utente.
Login	Questa classe rappresenta l'accesso a GuardaTv di un Utente.
Recensione	Questa classe rappresenta la Recensione di un Contenuto da parte di un Utente.
Tipologia	Questa classe rappresenta lo specifico Genere di un determinato Contenuto.
Utente	Questa classe rappresenta l'Utente registrato.

Package DAO

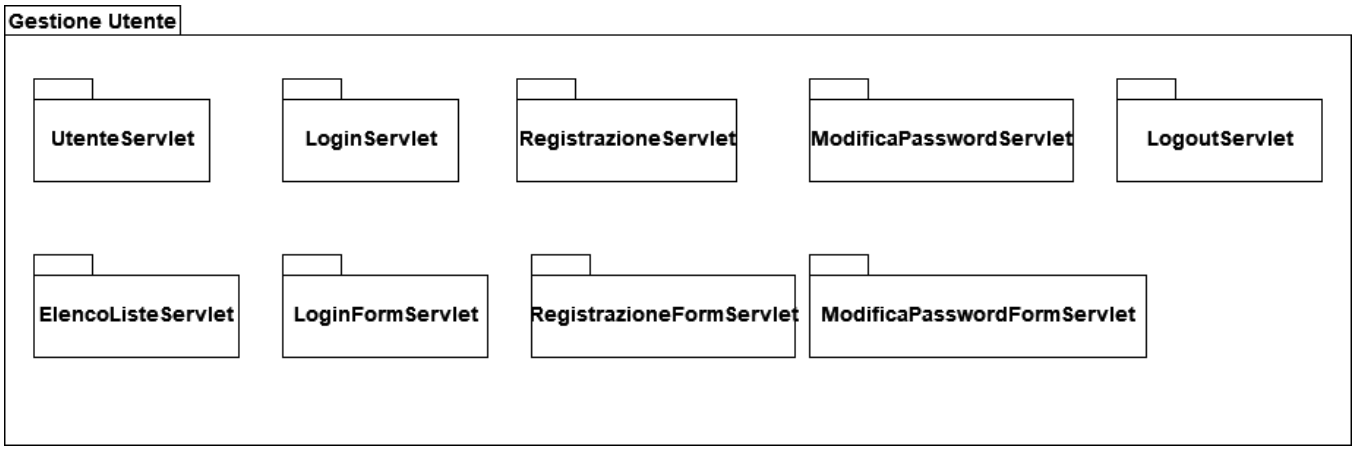


ContenutoDAO	Questa classe comunica con il Database per la gestione e raccolta degli oggetti "Contenuto" attraverso vari tipi di query.
ContenutoListaDAO	Questa classe comunica con il Database per la gestione e raccolta degli oggetti "ContenutoLista" attraverso vari tipi di query.
GenereDAO	Questa classe comunica con il Database per la gestione e raccolta degli oggetti "Genere" attraverso vari tipi di query.
ListaDAO	Questa classe comunica con il Database per la gestione e raccolta degli oggetti "Lista" attraverso vari tipi di query.
LoginDAO	Questa classe comunica con il Database per la gestione e raccolta degli oggetti "Login" attraverso vari tipi di query.
RecensioneDAO	Questa classe comunica con il Database per la gestione e raccolta degli oggetti "Recensione" attraverso vari tipi di query.

TipologiaDAO	Questa classe comunica con il Database per la gestione e raccolta degli oggetti "Tipologia", contenenti le liste dei generi degli specifici contenuti, attraverso vari tipi di query.
UtenteDAO	Questa classe comunica con il Database per la gestione e raccolta degli oggetti "Utente" attraverso vari tipi di query.

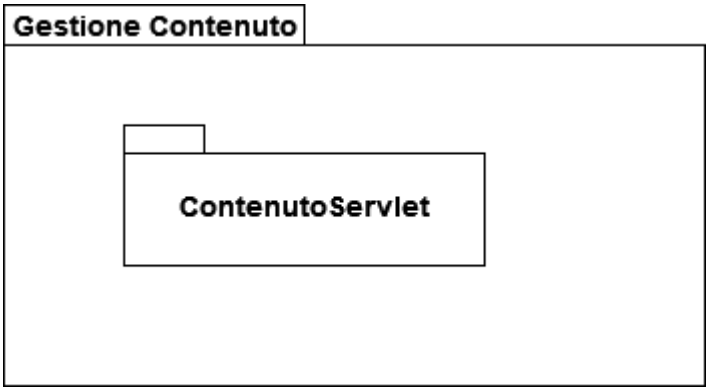
Application Layer

Gestione Utente



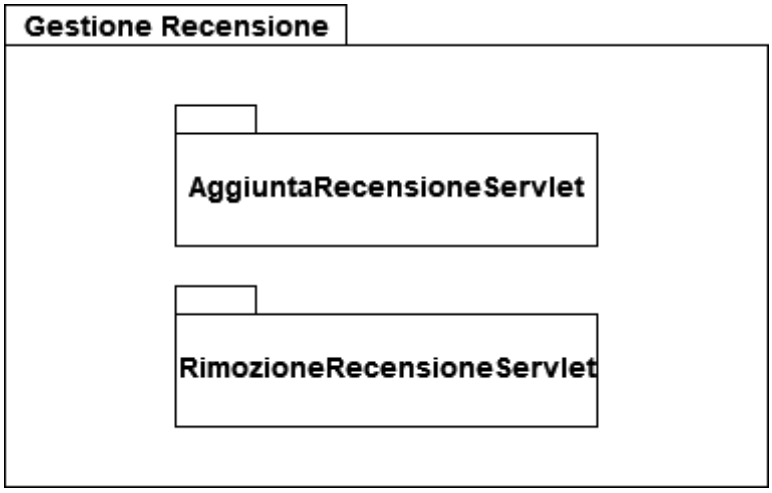
ElencoListeServlet	Questa Servlet invia al Presentation Layer l'elenco delle liste di un Utente.
LoginServlet	Questa Servlet effettua il Login con i dati ricevuti dall'Utente.
LoginFormServlet	Questa Servlet reindirizza alla pagina di Login.
LogoutServlet	Questa Servlet effettua il Logout dell'Utente.
ModificaPasswordServlet	Questa Servlet permette la modifica della password dell'Utente.
ModificaPasswordFormServlet	Questa Servlet reindirizza alla pagina per la modifica della password
RegistrazioneServlet	Questa Servlet effettua la Registrazione con i dati ricevuti dall'Utente.
RegistrazioneFormServlet	Questa Servlet reindirizza alla pagina di Registrazione.
UtenteServlet	Questa Servlet invia al Presentation Layer i dati dell'Utente.

Gestione Contenuto



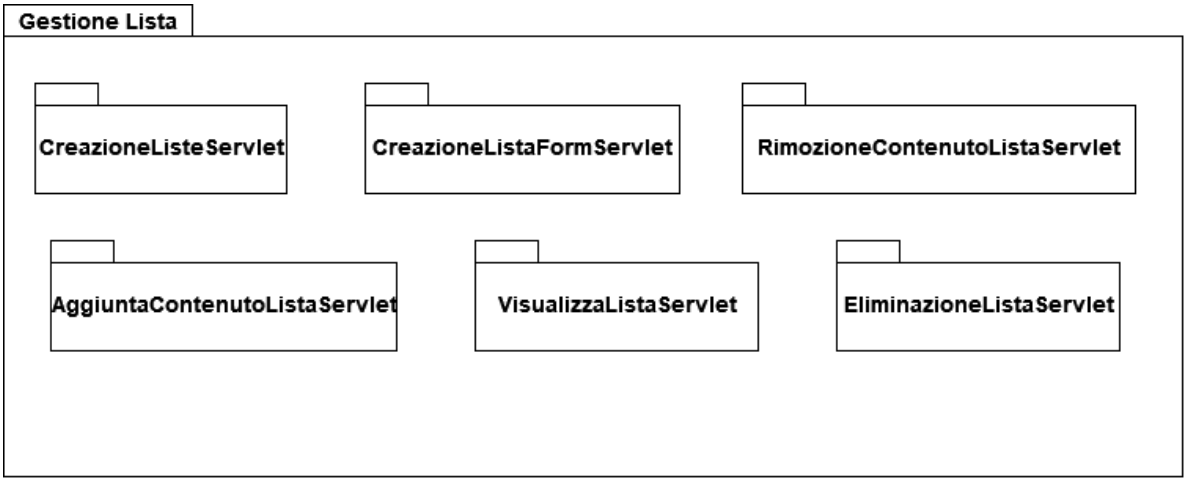
ContenutoServlet	Questa Servlet invia al Presentation Layer i dati del Contenuto richiesto.

Gestione Recensione



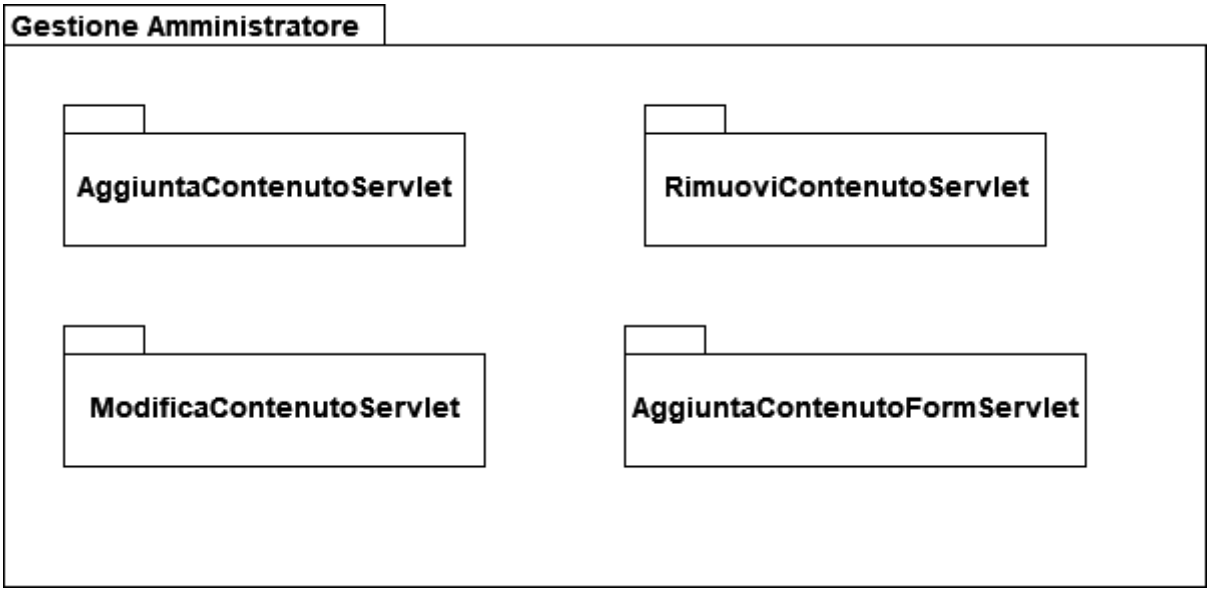
AggiuntaRecensioneServlet	Questa Servlet permette all'Utente di aggiungere una Recensione ad un Contenuto.
RimozioneRecensioneServlet	Questa Servlet permette la Rimozione di una Recensione

Gestione Lista



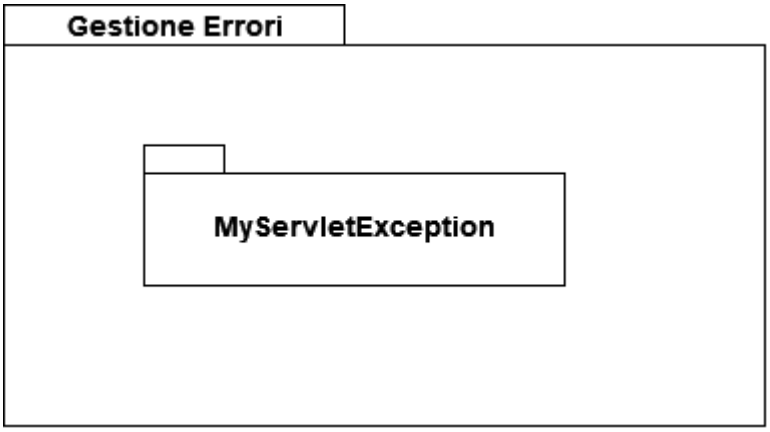
AggiuntaContenutoListaServlet	Questa Servlet aggiunge ad una Lista dell'Utente il Contenuto specificato.
CreazioneListaServlet	Questa Servlet permette all'Utente la creazione di una Lista.
CreazioneListaFormServlet	Questa Servlet reindirrizza alla pagina per la creazione di una nuova Lista.
EliminazioneListaServlet	Questa Servlet permette all'Utente l'eliminazione di una Lista.
RimozioneContenutoListaServlet	Questa Servlet di rimuovere un Contenuto da una Lista.
VisualizzaListaServlet	Questa Servlet invia al Presentation Layer l'elenco dei Contenuti all'interno della Lista selezionata dall'Utente.

Gestione Amministratore



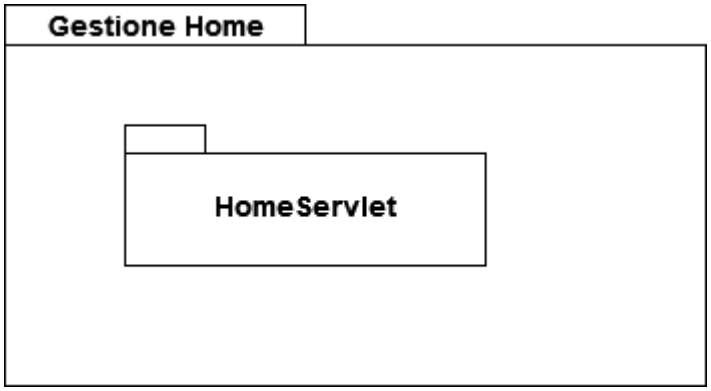
AggiuntaContenutoServlet	Questa Servlet permmette all'Amministratore di aggiungere un nuovo Contenuto.
AggiuntaContenutoFormServlet	Questa Servlet reindirizza alla pagina per l'aggiunta di un nuovo contenuto
ModificaContenutoServlet	Questa Servlet permmette all'Amministratore di modificare un Contenuto.
RimuoviContenutoServlet	Questa Servlet permmette all'Amministratore di rimuovere un Contenuto.

Gestione Errori



MyServletException	Questa Servlet gestisce i vari tipi di eccezione.

Gestione Home



HomeServlet	Questa Servlet invia al Presentation Layer i dati necessari a mostrare la Home del sito GuardaTv.

3. Class Interface

Javadoc di GuardaTV

Per motivi di leggibilità si è scelto di creare un sito, hostato tramite GitHub pages, contenente la JavaDoc di GuardaTV. In questo modo, chiunque può consultare la documentazione aggiornata dell'intero sistema. Di seguito, il link al sito in questione: https://ilselva.github.io/ProgettoIS/

4. Design Pattern

DAO (Data Access Object)

Per garantire un accesso alle informazioni persistenti senza prescindere dal sistema utilizzato per la persistenza, utilizzeremo un'interfaccia DAO che ci permetterà di disaccoppiare la logica del sistema dal gestore della persistenza (che potrà cambiare nel tempo in maniera completamente isolata).Un DAO (Data Access Object) è un pattern che offre un'interfaccia astratta per alcuni tipi di database. Mappando le chiamate dell'applicazione allo stato persistente, il DAO fornisce alcune operazioni specifiche sui dati senza esporre i dettagli del database.

Singleton

Per garantire che non esistano due istanze della stessa classe della classe Utils utilizziamo il design pattern Singleton. Esso è un design pattern creazionale, ossia un design pattern che si occupa dell'istanziazione degli oggetti, che ha lo scopo di garantire che di una determinata classe venga strutturata una sola istanza e di fornire un punto di accesso globale a tale istanza.