Università degli Studi di Salerno
Corso di Ingegneria del Software

PlanetVideo
Object Design Document
Versione 1.2

Data: 27/03/2020

Coordinatore del progetto:

Nome	Matricola

Partecipanti:

Nome	Matricola
Contente Antonio	0512103870
Mecca Vincenzo	0512103528

Scritto da:	Mecca Vincenzo

Revision History

Data	Versione	Descrizione	Autore
27/03/2020	1.0	Prima stesura	Mecca Vincenzo
30/03/2020	1.1	Modifiche	Mecca Vincenzo
01/04/2020	1.2	Modifiche e revisione	Mecca Vincenzo

Indice

1. INTRODUZIONE

Dopo aver realizzato il Requirements Analysis Document e il System Design Document, fornendo una descrizione sommaria di ciò che sarà il nostro sistema, ma tralasciando gli aspetti implementativi, andiamo ora a stilare il documento di Object Design. L'obiettivo di questo documento è produrre un modello che sia in grado di integrare in modo coerente e preciso tutte le funzionalità individuate nelle fasi precedenti.
In particolar modo, vengono definite le interfacce delle classi, le operazioni, i tipi, gli argomenti e le signature dei sottosistemi definiti nel System Design.

Comprensibilità vs Tempo:
Il source code deve essere quanto più comprensibile possibile, in modo da facilitare testing e possibili modifiche da apportare in futuro, e ovviamente accompagnato da commenti volti a semplificare la comprensione. Questo comporterà un lieve aumento di tempo di sviluppo del nostro progetto.

Interfaccia vs Usabilità:
L'interfaccia grafica è stata realizzata in maniera molto semplice, chiara e concisa. Form e pulsanti vengono utilizzati allo scopo di rendere semplice l'utilizzo del sistema da parte dell'utente finale.

Sicurezza vs Efficienza:
La sicurezza, come descritto nei requisiti non funzionali del Requirements Analysis Document, rappresenta uno degli aspetti importanti del sistema. Tuttavia, considerando i tempi di sviluppo molto limitati, saranno implementati dei semplici sistemi di sicurezza basati su username e password degli utenti.

1.1. *Linee guida per la documentazione delle interfacce*

Per la realizzazione del codice, gli sviluppatori dovranno seguire le seguenti linee guida:

Naming convention
È buona norma utilizzare nomi descrittivi, pronunciabili e utilizzare caratteri consentiti.

Variabili
I nomi delle variabili devono iniziare con la lettera minuscola, le parole successive con quella maiuscola (camel case). La dichiarazione delle variabili deve essere effettuata ad inizio blocco; ogni riga presenterà una sola variabile per migliorare la comprensione.

Metodi
I nomi dei metodi devono iniziare con la lettera minuscola e le parole successive con la lettera maiuscola (camel case). Di solito il nome del metodo è costituito da un verbo che identifica un'azione, seguito dal nome di un oggetto. I nomi dei metodi per l'accesso e la modifica delle variabili dovranno essere del tipo getNome(), setNome().

Classi e pagine
I nomi delle classi e delle pagine devono iniziare con la lettera maiuscola e anche le parole

successive all'interno del nome devono iniziare con lettera maiuscola (camel case). I nomi delle classi e delle pagine devono essere evocativi, in modo da fornire informazioni sullo scopo di quest'ultime.
La dichiarazione di una classe è caratterizzata da:

- Dichiarazione di costanti
- Dichiarazione di variabili di classe
- Dichiarazione di variabili d'istanza
- Costruttore
- Dichiarazione metodi

Per migliorare la comprensione del codice, è richiesta una corretta indentazione.

1.2. Definizioni, acronimi e abbreviazioni

RAD: Requirements Analysis Document
SDD: System Design Document
ODD: Object Design Document
RDBMS: Relational Database Management System, sistema per la gestione di basi di dati relazionali

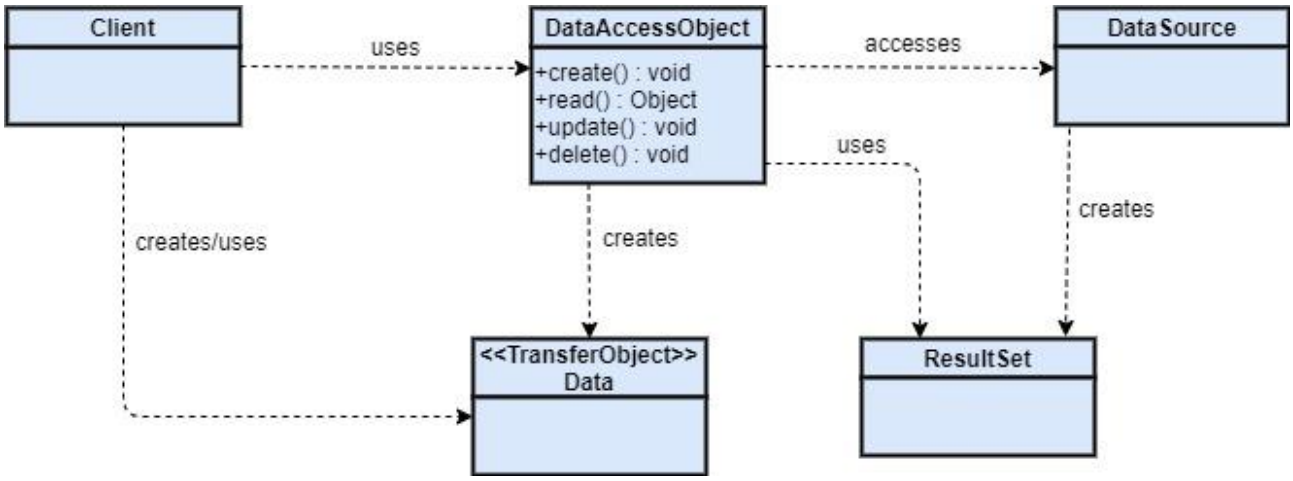
1.3. Riferimenti

Il materiale di riferimento utilizzato per la stesura di questo stesso documento comprende:

- Libro di testo: B.Bruegge, A.H. Dutoit, Object Oriented Software Engineering –Using UML, Patterns and Java, Prentice Hall, 3rd edition, 2009
- RAD_PlanetVideo
- GestioneDatiPersistenti_PlanetVideo

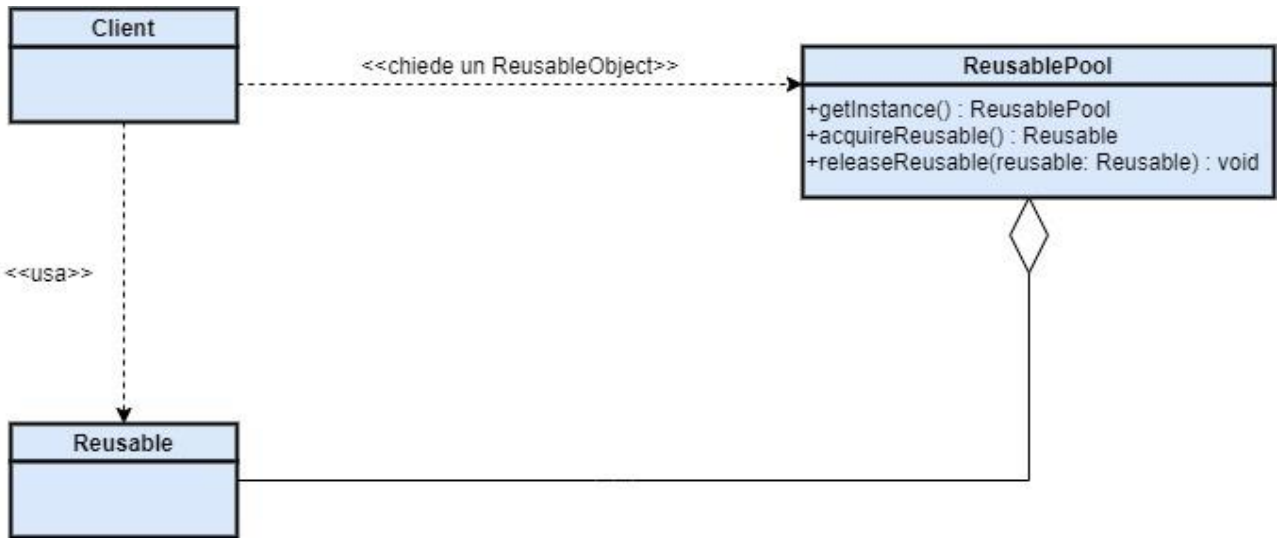
2. DESIGN PATTERN

2.1. Design Access Object Pattern (DAO Pattern)



PlanetVideo fa uso del DAO Pattern. Si tratta di un pattern architetturale per la gestione della persistenza: fondamentalmente, è una classe con relativi metodi che rappresenta un'entità tabellare di un RDBMS, usata principalmente in applicazioni web sia di tipo Java EE sia di tipo EJB, per stratificare e isolare l'accesso ad una tabella tramite query (poste all'interno dei metodi della classe) ovvero al data layer da parte della business logic creando un maggiore livello di astrazione ed una più facile manutenibilità. I metodi del DAO con le rispettive query verranno così richiamati dalle classi della business logic.
Il seguente design pattern è stato utilizzato nell'implementazione dei vari Model, dunque nelle seguenti classi: *AbbonModel, CatalogModel, FattModel, ReviewModel, UserModel*, dove sono presenti dunque i vari metodi con le query per interrogare e riportare informazioni presenti nel database.

2.2. Object Pool Pattern

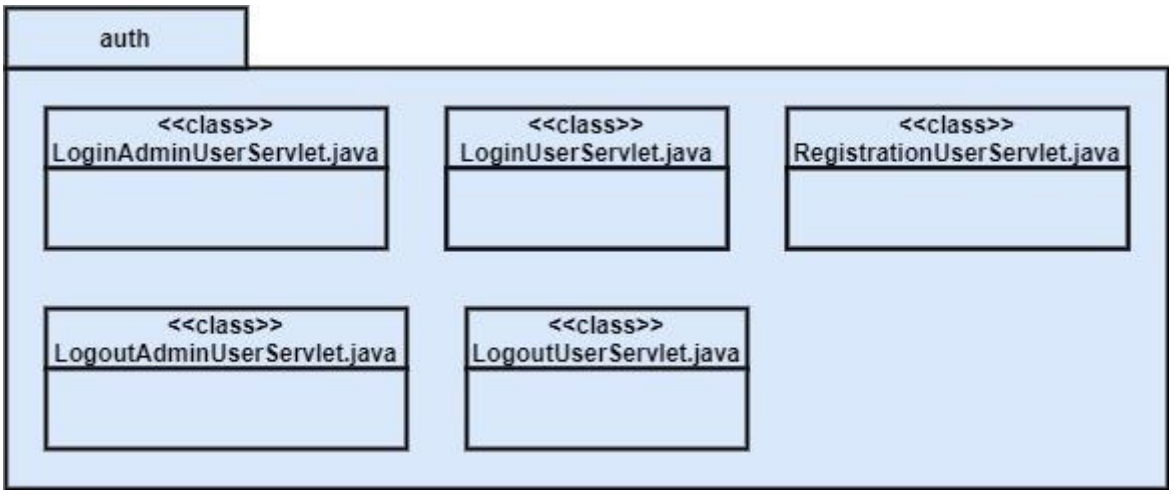


L'Object pool pattern è un design pattern creazionale che usa un insieme di oggetti inizializzati pronti per l'uso - una "pool" – piuttosto che allocarli e de-allocherli ripetutamente su richiesta. Il client della pool richiedeà un oggetto dalla pool ed eseguirà operazioni sull'oggetto restituito. Quando il client ha terminato, ritorna l'oggetto alla pool invece che distruggerlo; questo può avvenire in modo manuale o automatico.
Il seguente design pattern è stato utilizzato nell'implementazione della connessione al database, dunque nella classe *DriverManagerConnectionPool.*

3. PACKAGE COMPONENTS

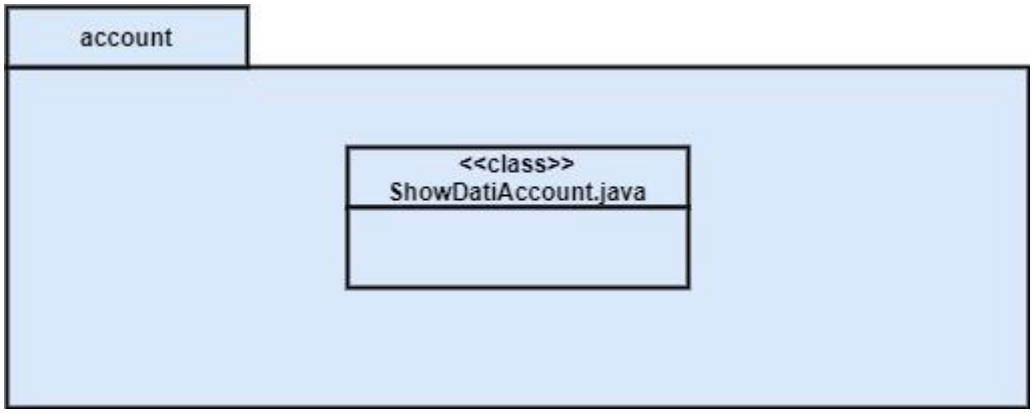
3.1. Package control

3.1.1. Control.auth



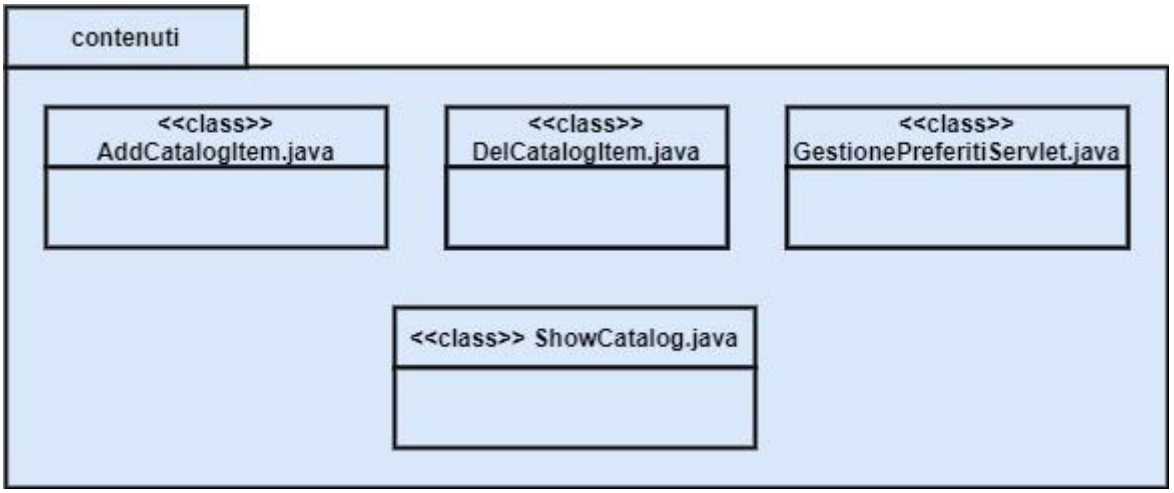
Classe	Descrizione
LoginUserServlet.java	Servlet che gestisce il login per un utente registrato
LogoutUserServlet.java	Servlet che gestisce il logout per un utente registrato
RegistrationUserServlet.java	Servlet che gestisce la registrazione di un utente
LoginAdminUserServlet.java	Servlet che gestisce il login per un utente amministratore
LogoutAdminUserServlet.java	Servlet che gestisce il logout per un utente amministratore

3.1.2. Control.account



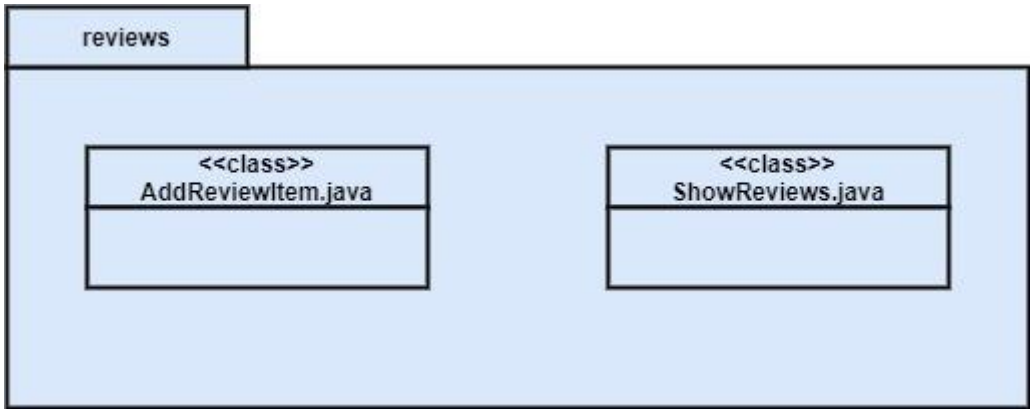
Classe	Descrizione
ShowDatiAccount.java	Servlet che gestisce la visualizzazione dei dati di un account utente

3.1.3. Control.contenuti



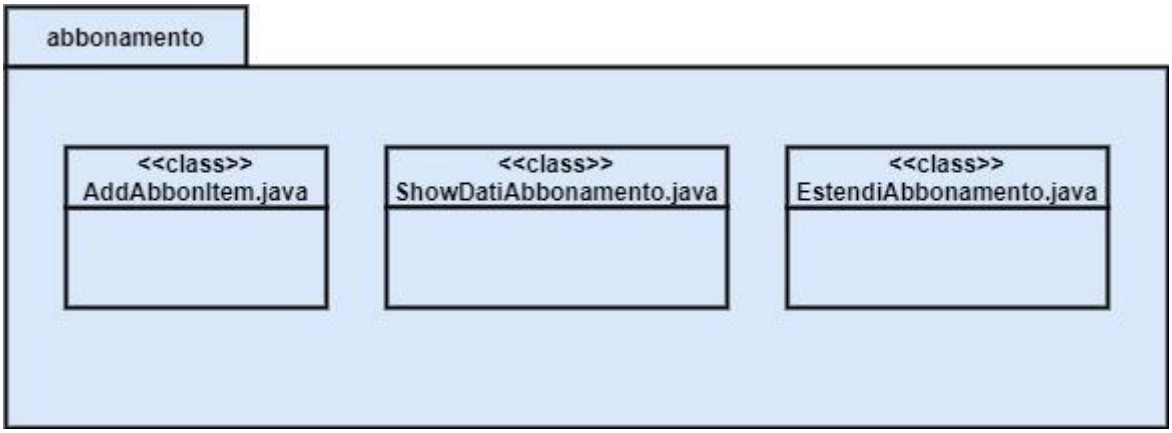
Classe	Descrizione
AddCatalogItem.java	Servlet che gestisce l'aggiunta di un contenuto nel catalogo
DelCatalogItem.java	Servlet che gestisce la rimozione di un contenuto nel catalogo
GestionePreferitiServlet.java	Servlet che gestisce i contenuti preferiti
ShowCatalog.java	Servlet che gestisce la visualizzazione dei contenuti del catalogo

3.1.4. Control.reviews



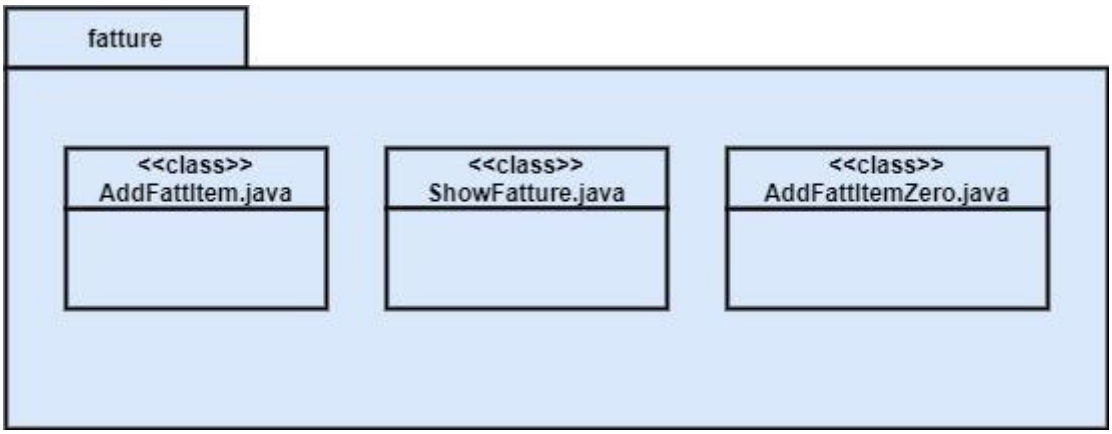
Classe	Descrizione
AddReviewItem.java	Servlet che gestisce l'aggiunta di una recensione
ShowReviews.java	Servlet che gestisce la visualizzazione delle recensioni

3.1.5. Control.abbonamento



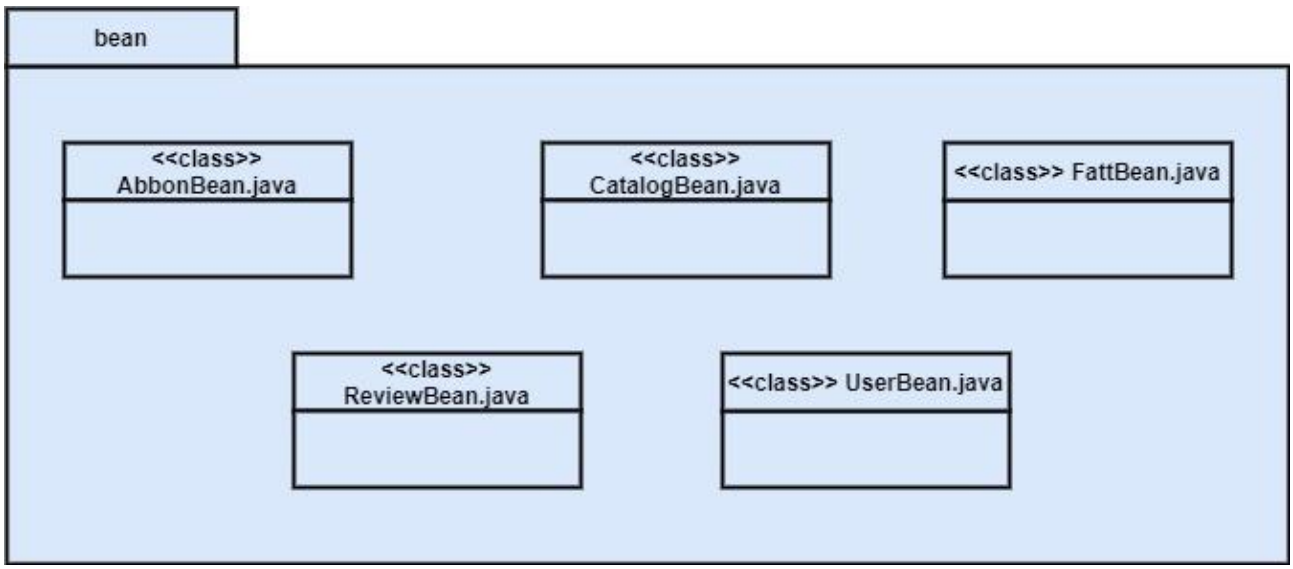
Classe	Descrizione
AddAbbonItem.java	Servlet che gestisce l'aggiunta di un abbonamento per un utente
ShowDatiAbbonamento.java	Servlet che gestisce la visualizzazione dei dati dell'abbonamento di un utente
EstendiAbbonamento.java	Servlet che gestisce il rinnovo di un abbonamento per un utente

3.1.6. Control.fatture



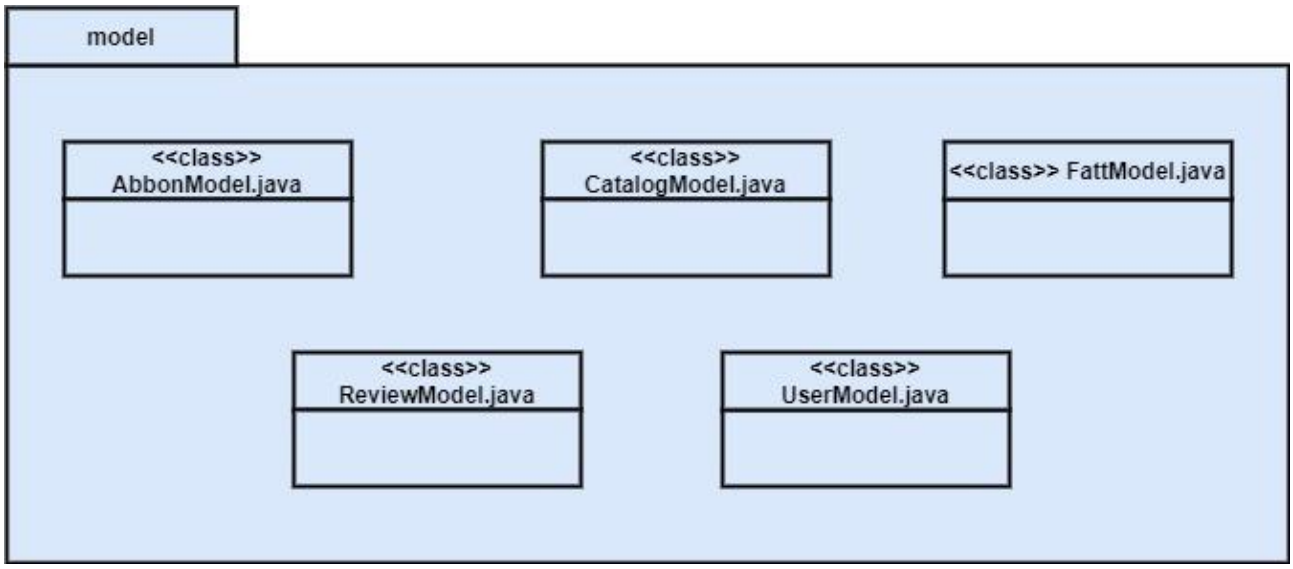
Classe	Descrizione
AddFattItem.java	Servlet che gestisce l'aggiunta di una fattura con importo di 11.99 euro per un utente
ShowFatture.java	Servlet che gestisce la visualizzazione delle fatture di un utente
AddFattItemZero.java	Servlet che gestisce l'aggiunta di una fattura con importo di 0 euro per un utente

3.2. Package bean



Classe	Descrizione
AbbonBean.java	Questa classe rappresenta un abbonamento
CatalogBean.java	Questa classe rappresenta un elemento del catalogo
FattBean.java	Questa classe rappresenta una fattura
ReviewBean.java	Questa classe rappresenta una recensione
UserBean.java	Questa classe rappresenta un utente registrato

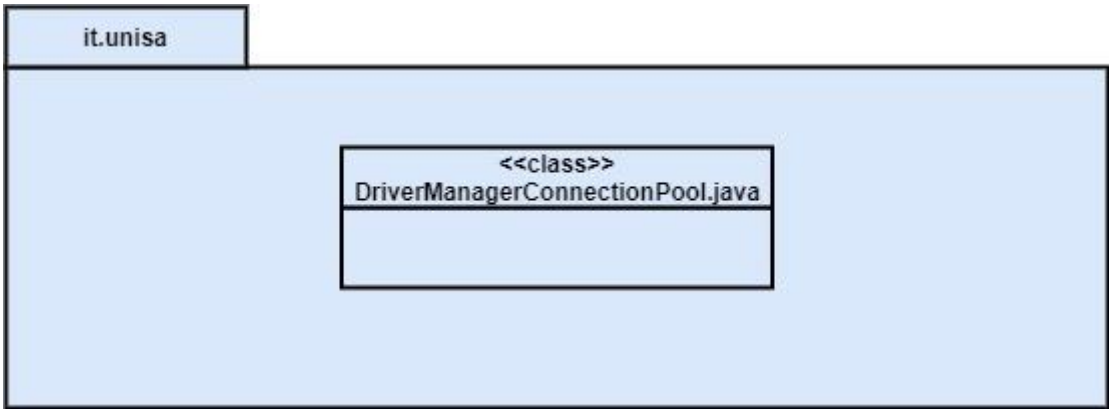
3.3. Package model



Classe	Descrizione
AbbonModel.java	Questa classe contiene metodi che permettono di effettuare query sul database che riguardano gli abbonamenti

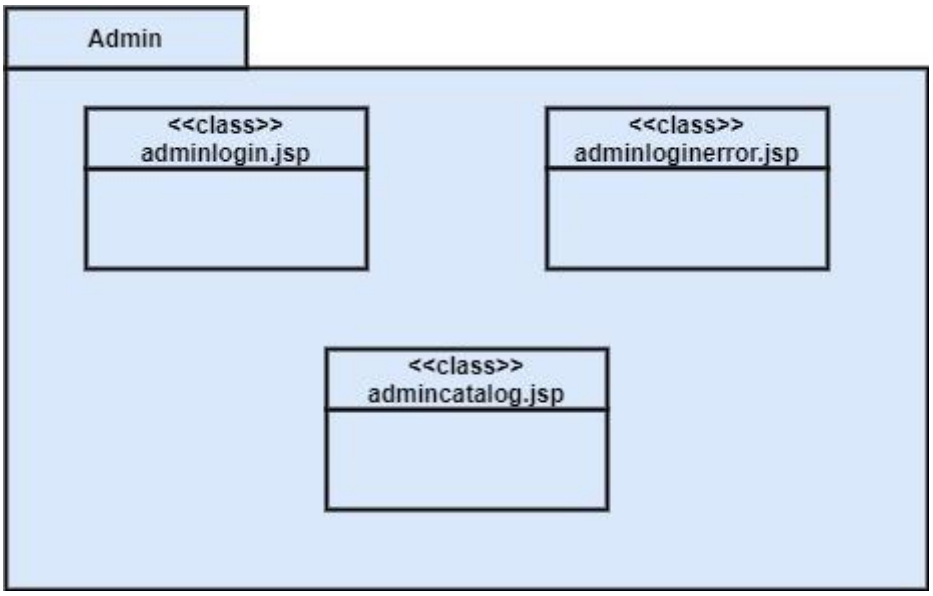
CatalogModel.java	Questa classe contiene metodi che permettono di effettuare query sul database che riguardano i contenuti del catalogo
FattModel.java	Questa classe contiene metodi che permettono di effettuare query sul database che riguardano le fatture
ReviewModel.java	Questa classe contiene metodi che permettono di effettuare query sul database che riguardano le recensioni
UserModel.java	Questa classe contiene metodi che permettono di effettuare query sul database che riguardano gli utenti registrati

3.4. Package it.unisa



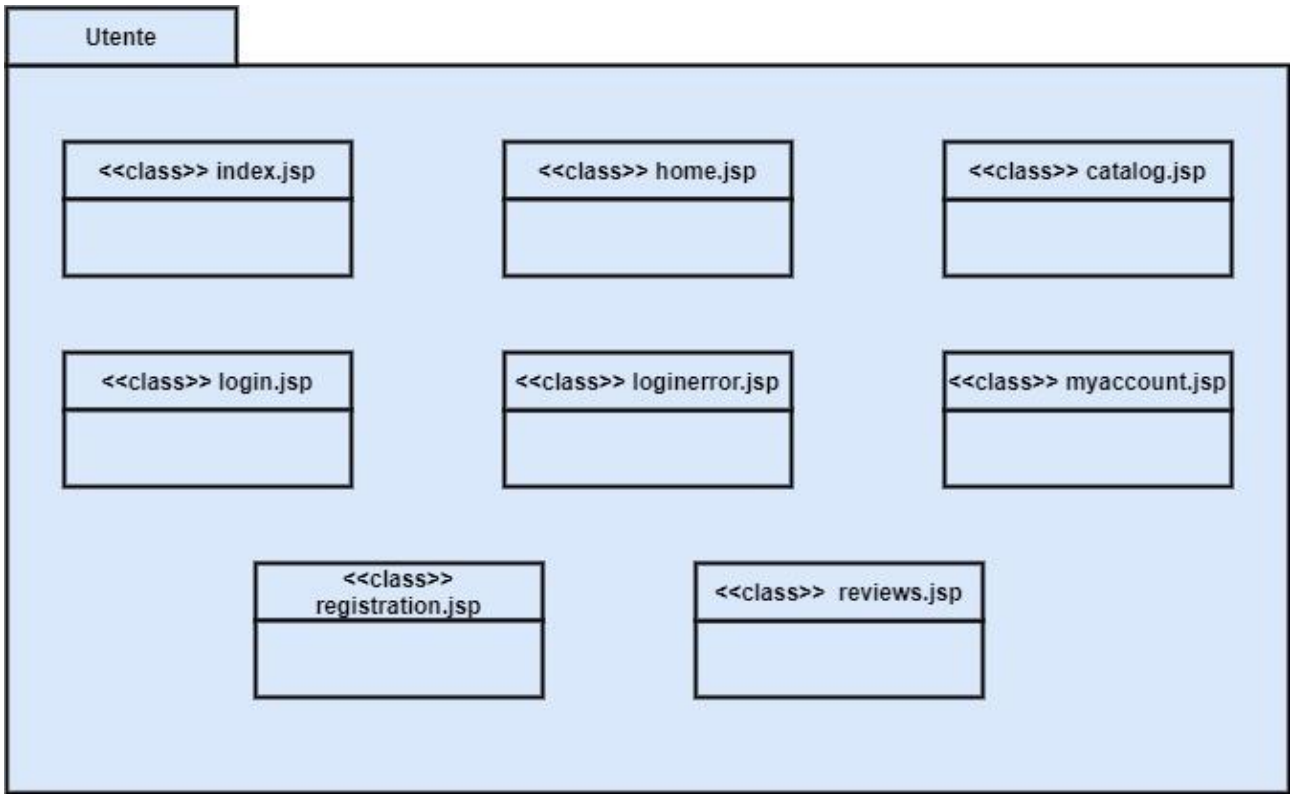
Classe	**Descrizione**
DriverManagerConnectionPool.java	Classe che si occupa di gestire una pool di connessioni al database

3.5. Admin



Classe	Descrizione
adminlogin.jsp	Pagina che mostra all'utente amministratore il form di login per l'accesso
adminloginerror.jsp	Pagina che mostra all'utente amministratore un messaggio d'errore per credenziali errate e il form di login per tentare nuovamente l'accesso
admincatalog.jsp	Pagina che mostra due form all'utente amministratore, rispettivamente per l'aggiunta e la rimozione di un contenuto del catalogo

3.6. Utente



Classe	**Descrizione**
index.jsp	Pagina principale del sito visibile a tutti
home.jsp	Pagina principale del sito. Mostra all'utente i prossimi contenuti in arrivo sul sito
catalog.jsp	Pagina che mostra all'utente i contenuti presenti nel catalogo. Qui è possibile visualizzare i dettagli di un contenuto, aggiungere un contenuto alla lista preferiti e andare alla sezione recensioni per un contenuto
login.jsp	Pagina che mostra all'utente il form di login per l'accesso
loginerror.jsp	Pagina che mostra all'utente un messaggio d'errore per credenziali errate e il form di login per tentare nuovamente l'accesso
myaccount.jsp	Pagina che mostra all'utente i dati dell'account, i dati dell'abbonamento e le rispettive fatture
registration.jsp	Pagina che mostra all'utente il form di registrazione
reviews.jsp	Pagina che mostra all'utente le recensioni per un determinato contenuto del catalogo. Inoltre, mostra all'utente un form per l'aggiunta di una recensione

4. CLASS INTERFACES

4.1. User Model

doSave	*Context*: UserModel::doSave(user) *Pre*: username != null && password != null && nome != null && cognome != null && datanascita != null && numcarta != null && email != null *Post*: -
doRetrieveByKey	*Context*: UserModel::doRetrieveByKey(username) *Pre*: username != null *Post*: -
checkUser	*Context*: UserModel::checkUser(username, password) *Pre*: username != null && password != null *Post*: -
checkAdminUser	*Context*: UserModel::checkUser(username) *Pre*: username != null *Post*: -

4.2. Abbonamento Model

doSave	*Context*: AbbonamentoModel::doSave(abbonamento) *Pre*: datainizio != null && datafine != null && user.username != null *Post*: -
doUpdate	*Context*: AbbonamentoModel::doUpdate(abbonamento) *Pre*: datafine != null && user.username != null *Post*: -
doRetrieveByKey	*Context*: AbbonamentoModel:: doRetrieveByKey(username) *Pre*: user.username != null *Post*: -

4.3. Fattura Model

doRetrieveByKey	*Context*: FatturaModel:: doRetrieveByKey(username) *Pre*: user.username != null *Post*: -
doSave	*Context*: FatturaModel:: doSave(fattura) *Pre*: dataemissione != null && importo != null && user.username != null *Post*: -

4.4. Catalog Model

doRetrieveAll	*Context*: CatalogModel:: doRetrieveAll(order) *Pre*: - *Post*: -
doSave	*Context*: CatalogModel:: doSave(catalog) *Pre*: codice != null && titolo != null && regista != null && anno != null && genere != null && urlimg != null *Post*: -
doRetrieveByKey	*Context*: CatalogModel:: doRetrieveByKey(codice) *Pre*: codice != null *Post*: -
doDelete	*Context*: CatalogModel:: doDelete(codice) *Pre*: codice != null *Post*: -

4.5. Review Model

doRetrieveByKey	*Context*: ReviewModel:: doRetrieveByKey(codice) *Pre*: catalog.codice != null *Post*: -
doSave	*Context*: ReviewModel:: doSave(review) *Pre*: testo != null && voto != null && catalog.codice != null *Post*: -

4.6. Servlet authentication

LoginUser	*Pre*: username != null && password != null *Post*: -
LoginAdminUser	*Pre*: username != null && password != null *Post*: -
LogoutUser	*Pre*: - *Post*: -
LogoutAdminUser	*Pre*: - *Post*: -
RegistrationUser	*Pre*: username != null && password != null && nome != null && cognome != null && datanascita != null && numcarta != null && email != null *Post*: -

4.7. Servlet account

ShowDatiAccount	*Pre*: username != null *Post*: -

4.8. Servlet contenuti

ShowCatalog	*Pre*: - *Post*: -
AddCatalogItem	*Pre*: codice != null && titolo != null && regista != null && anno != null && genere != null && urlimg != null *Post*: -
DelCatalogItem	*Pre*: codice != null *Post*: -
GestionePreferiti	*Pre*: action != null && codice != null *Post*: -

4.9. Servlet reviews

ShowReviews	*Pre*: codice != null *Post*: -
AddReviewItem	*Pre*: testo != null && voto != null && codice != null *Post*: -

4.10. Servlet abbonamento

ShowDatiAbbonamento	*Pre*: username != null *Post*: -
AddAbbonItem	*Pre*: numeroabb != null && costo != null && datainizio != null && username != null *Post*: -
EstendiAbbonamento	*Pre*: username != null *Post*: -

4.11. Servlet fatture

ShowFatture	*Pre*: username != null *Post*: -
AddFattItem	*Pre*: username != null *Post*: -
AddFattItemZero	*Pre*: username != null *Post*: -