



Università degli Studi di Salerno

Corso di Ingegneria del Software

**Analytics Films
Object Design Document
Versione 1.0**



Data: 28/12/2018

Coordinatore del progetto:

Nome	Matricola
Sammartino Vittorio	0512104780

Partecipanti:

Nome	Matricola
Aprea Pasqua	0512104990
Nappi Luca	0512104648
Armenio Vincenzo	0512104958
Sammartino Vittorio	0512104780

Scritto da:	Sammartino Vittorio
--------------------	---------------------

SOMMARIO

1	Introduzione.....	5
1.1	Object Design Trade-offs	5
1.2	Linee Guida per la Documentazione delle Interfacce	6
1.3	Definizioni, acronimi e abbreviazioni.....	7
1.4	Riferimenti.....	7
2	Packages.....	7
2.1	Package control	8
2.1.1	control.auth	8
2.1.2	control.film	8
2.1.3	control.user	9
2.1.4	control.search	9
2.1.5	control.review.....	10
2.1.6	control.statistics	10
2.2	Package bean.....	10
2.3	Package connectionPool.....	11
2.4	Package model.....	11
2.5	Amministratore	12
2.6	Utente	13
3	Class Interface	15
3.1	Interface Bean	15
3.2	Interface Manager	16
3.2.1	Authentication Manager.....	16
3.2.2	Film Manager.....	17
3.2.3	Review Manager	18
3.2.4	Search Manager	19
3.2.5	Statistics Manager	20
3.2.6	User Manager	21

4	Design Pattern.....	22
4.1	Object Pool Pattern	22

1 INTRODUZIONE

1.1 OBJECT DESIGN TRADE-OFFS

Dopo la realizzazione dei documenti RAD e SDD abbiamo descritto in linea di massima quello che sarà il nostro sistema e quindi i nostri obiettivi, tralasciando gli aspetti implementativi. Il seguente documento ha lo scopo di produrre un modello capace di integrare in modo coerente e preciso tutte le funzionalità individuate nelle fasi precedenti. In particolare, definisce le interfacce delle classi, le operazioni, i tipi, gli argomenti e le signature dei sottosistemi definiti nel System Design. Inoltre, sono specificati i trade-off e le linee guida.

Comprensibilità vs Tempo:

Il codice deve essere quanto più comprensibile possibile per facilitare la fase di testing ed eventuali future modifiche. Il codice sarà quindi accompagnato da commenti che ne semplifichino la comprensione. Ovviamente questa caratteristica aggiungerà un incremento di tempo allo sviluppo del nostro progetto.

Interfaccia vs Usabilità:

L'interfaccia grafica è stata realizzata in modo da essere molto semplice, chiara e concisa, fa uso di form e pulsanti disposti in maniera da rendere semplice l'utilizzo del sistema a quanti più utenti possibili.

Sicurezza vs Efficienza:

La sicurezza, come descritto nei requisiti non funzionali del RAD, rappresenta uno degli aspetti importanti del sistema. Tuttavia, dati i tempi di sviluppo molto limitati, ci limiteremo ad implementare sistemi di sicurezza basati su username e password degli utenti.

Response Time vs Hardware:

Il sistema garantisce una certa reattività alle richieste, e quindi essere in grado di poter comunque offrire una contemporaneità di servizi agli utenti. Ovviamente questa caratteristica sarà limitata dall'hardware del sistema.

Prestazioni vs Costi

Il sistema prevede l'utilizzo di template open source esterni per mantenere prestazioni elevate, essendo il progetto sprovvisto di budget.

1.2 LINEE GUIDA PER LA DOCUMENTAZIONE DELLE INTERFACCE

Gli sviluppatori dovranno seguire le seguenti convenzioni per la scrittura del codice:

Naming Convention

- È buona norma utilizzare nomi:
- Descrittivi
- Di uso comune
- Lunghezza medio-corta

Variabili

- I nomi delle variabili devono cominciare con una lettera minuscola, se il nome della variabile è costituito da più parole, solo l'iniziale delle altre parole sarà maiuscola (es: titoloFilm).

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 di un verbo che identifica una azione, seguito dal nome di un oggetto.
- I nomi dei metodi per l'accesso e la modifica delle variabili dovranno essere del tipo `getNomeVariabile()` e `setNomeVariabile()`.

Classi e pagine

- I nomi delle classi e delle pagine devono iniziare con una lettera maiuscola, le parole contenute al suo interno devono cominciare con lettera maiuscola. Il nome deve fornire informazioni utili relative al loro scopo.
- Ogni file sorgente contiene una singola classe e deve essere strutturato in un determinato modo:
- L'istruzione `package` che permette di inserire la classe in un determinato package
 - L'istruzione `import` che importa le librerie necessarie alla class
 - Una piccola descrizione della classe

1.3 DEFINIZIONI, ACRONIMI E ABBREVIAZIONI

RAD = Requirement Analysis Document

SDD = System Design Document

ODD = Object Design Document

1.4 RIFERIMENTI







Riferimenti a Rad_AnalyticsFilms e a Sdd_AnalyticsFilms

2 PACKAGES

Abbiamo diviso il sistema in :

- Control
- Bean
- ConnectionPool
- Model
- View
- Test

2.1 PACKAGE CONTROL







- >  control.auth
- >  control.film
- >  control.review
- >  control.search
- >  control.statistics
- >  control.user

2.1.1 control.auth

- ▼  control.auth
 - >  Login.java
 - >  Logout.java
 - >  RecovePass.java
 - >  Registration.java






Classe	Descrizione
Login.java	Servlet che gestisce il login
Logout.java	Servlet che gestisce il logout
RecovePass.java	Servlet che gestisce il recupero password
Registration.java	Servlet che gestisce la registrazione

2.1.2 control.film

- ▼  control.film
 - >  AddFilm.java
 - >  DeleteFilm.java
 - >  ModifyFilm.java
 - >  ShowFilm.java
 - >  ShowFilmList.java




Classe	Descrizione
AddFilm.java	Servlet che gestisce l'aggiunta di un film
DeleteFilm.java	Servlet che gestisce la rimozione di un film
ModifyFilm.java	Servlet che gestisce la modifica di un film
ShowFilm.java	Servlet che gestisce la visualizzazione di un film
ShowFilmList.java	Servlet che gestisce la visualizzazione di una lista di film

2.1.3 control.user

- ▼  control.user
 - >  DeleteAccount.java
 - >  ModifyPersonalInfo.java
 - >  ShowAccountList.java
 - >  ShowPersonalInfo.java




Classe	Descrizione
DeleteAccount.java	Servlet che gestisce l'eliminazione di un account
ModifyPersonalInfo.java	Servlet che gestisce la modifica delle info personali
ShowAccountList.java	Servlet che gestisce la visualizzazione di una lista di account
ShowPersonalInfo.java	Servlet che gestisce la visualizzazione delle info personali

2.1.4 control.search

- ▼  control.search
 - >  SearchCinema.java
 - >  SearchFilm.java



Classe	Descrizione
SearchCinema.java	Servlet che gestisce la ricerca di un cinema
SearchFilm.java	Servlet che gestisce la ricerca di un film

2.1.5 control.review

- ▼  control.review
 - >  InsertReview.java
 - >  ShowReviews.java

Classe	Descrizione
InsertReview.java	Servlet che gestisce l'inserimento di una recensione
ShowReviews.java	Servlet che gestisce la visualizzazione di più recensioni

2.1.6 control.statistics

- ▼  control.statistics
 - >  ShowStatistics.java



Classe	Descrizione
ShowStatistics.java	Servlet che gestisce la visualizzazione delle statistiche del sito

2.2 PACKAGE BEAN

- ▼  bean
 - >  Cinema.java
 - >  Film.java
 - >  Recensione.java
 - >  Utente.java








Class	Descrizione
Cinema	Descrive un cinema presente nell'elenco
Film	Descrive un film presente nell'elenco
Recensione	Descrive una recensione inserita da un utente
Utente	Descrive l'utente

2.3 PACKAGE CONNECTIONPOOL

▼  connectionPool
 >  ConnectionPool.java

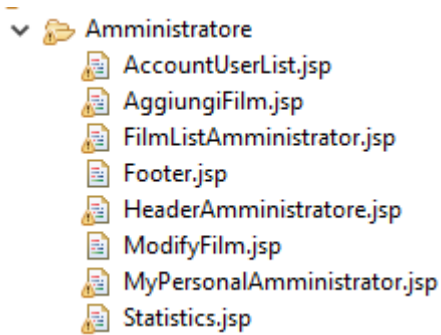
ConnectionPool.java	Classe che implementa l'object Pool Pattern, responsabile di fornire connessione con database.
---------------------	--

2.4 PACKAGE MODEL

▼  model
 >  AuthenticationManager.java
 >  FilmManager.java
 >  ReviewManager.java
 >  SearchManager.java
 >  StatisticsManager.java
 >  UserManager.java

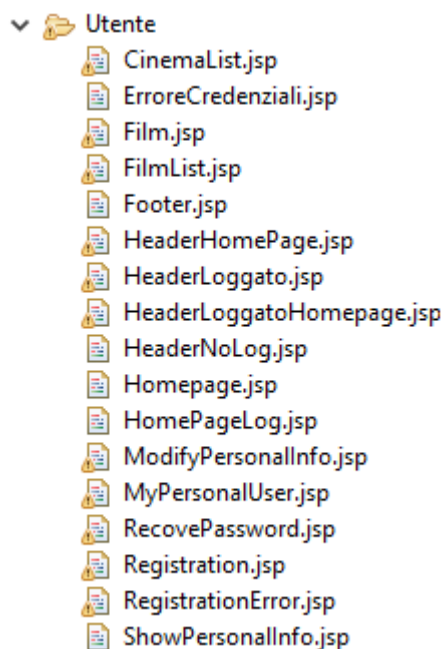
AuthenticationManager.java	Descrive l'interazione con il database per le operazioni di autenticazione
FilmManager.java	Descrive l'interazione con il database per le operazioni con film.
ReviewManager.java	Descrive l'interazione con il database per le operazioni di inserimento recensione
SearchManager.java	Descrive l'interazione con il database per le operazioni di ricerca
StatisticsManager.java	Descrive l'interazione con il database per le operazioni di visualizzazione statistiche
UserManager.java	Descrive l'interazione con il database per le operazioni sull'account

2.5 AMMINISTRATORE



AccountUserList.jsp	Sezione che mostra all'amministratore la lista di tutti gli utenti iscritti al sito che può eliminare
AggiungiFilm.jsp	Sezione che mostra all'amministratore un form per l'aggiunta di un film
FilmListAmministratore.jsp	Pagina che mostra all'amministratore la lista di tutti i film del sito che può modificare o eliminare
ModifyFilm.jsp	Sezione che mostra un form per la modifica di un film
MyPersonalAmministratore.jsp	Pagina principale dell'amministratore che mostra i bottoni per accedere alla lista dei film, alla lista degli utenti, per aggiungere un film e per visualizzare le statistiche
Statistics.jsp	Pagina che mostra all'amministratore le statistiche del sito

2.6 UTENTE

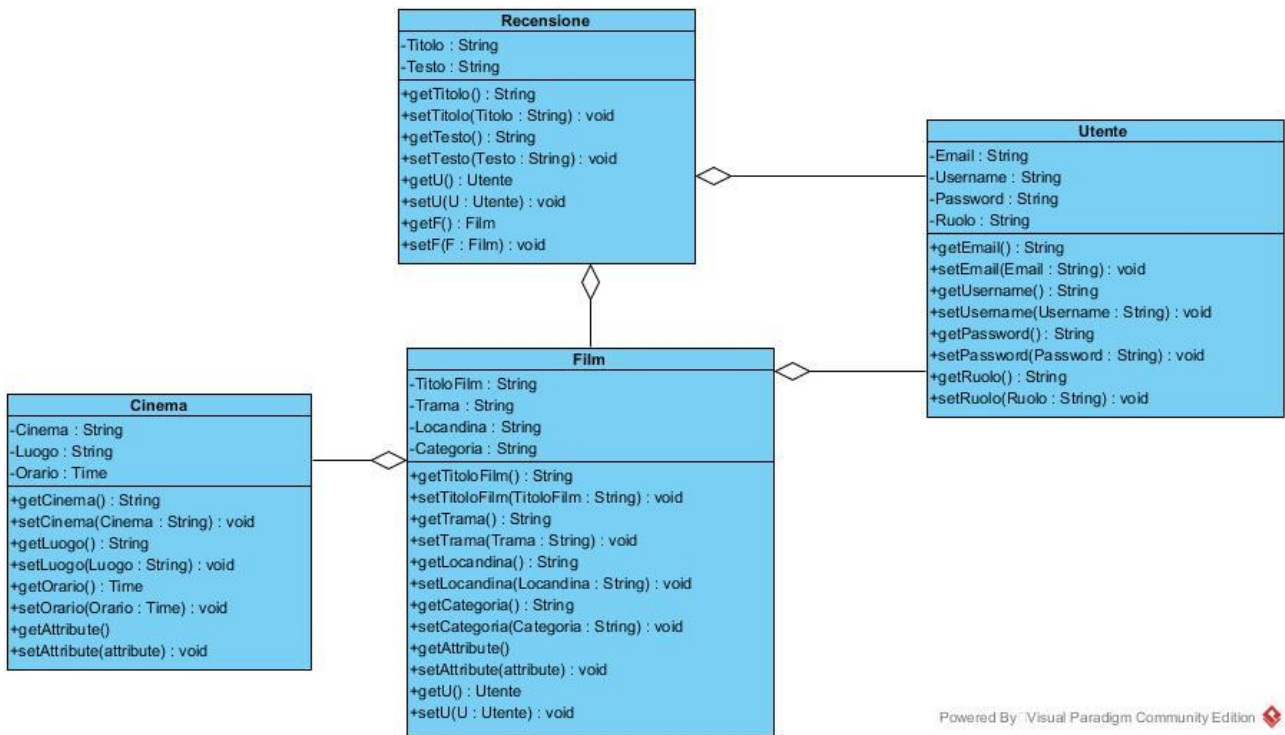


CinemaList.jsp	Pagina che mostra all'utente la lista di tutti i cinema per un determinato film
ErroreCredenziali.jsp	Pagina che mostra un messaggio di errore
Film.jsp	Pagina che mostra all'utente un determinato film dove è possibile vedere le recensioni degli altri utenti e di inserire una recensione
FilmList.jsp	Pagina che mostra la lista di tutti i film ricercati
Homepage.jsp	Pagina principale del sito visibile a tutti
HomePageLog.jsp	Pagina principale del sito visibile solo agli utenti registrati
ModifyPersonallInfo.jsp	Pagina che mostra un form per la modifica dell'username utente
MyPersonalUser.jsp	Pagina personale dell'utente
RecovePassword.jsp	Pagina per il recupero della password mediante email
Registration.jsp	Pagina che mostra un form per la registrazione

RegistrationError.jsp	Pagina che mostra un messaggio di errore per una registrazione errata
ShowPersonallInfo.jsp	Pagina che mostra informazioni personali dell'utente

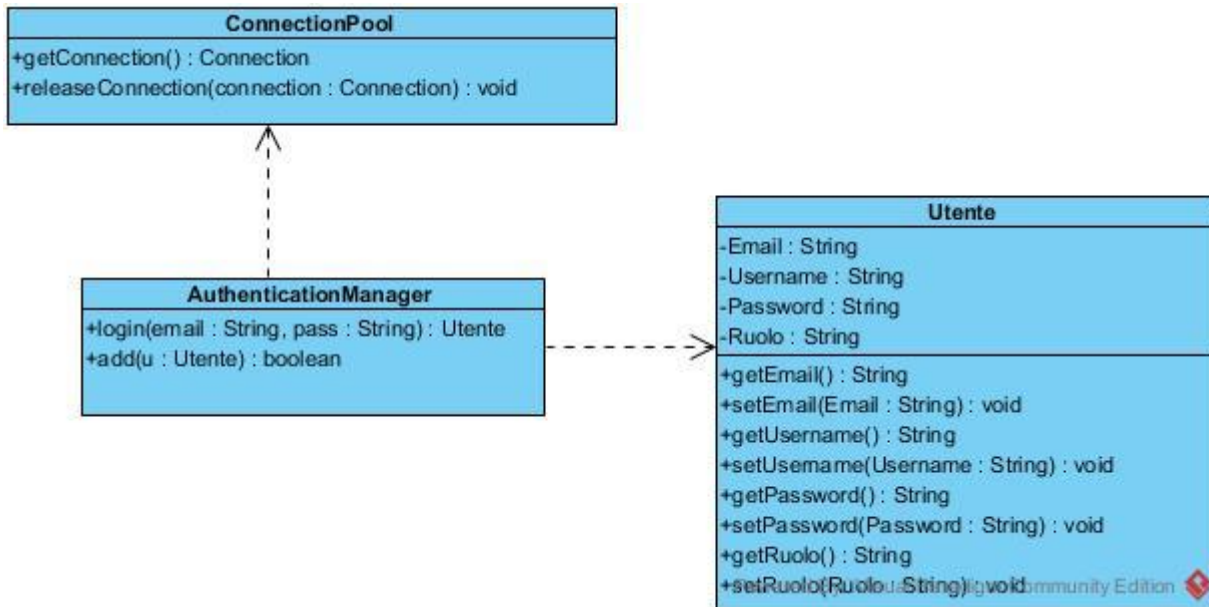
3 CLASS INTERFACE

3.1 INTERFACE BEAN



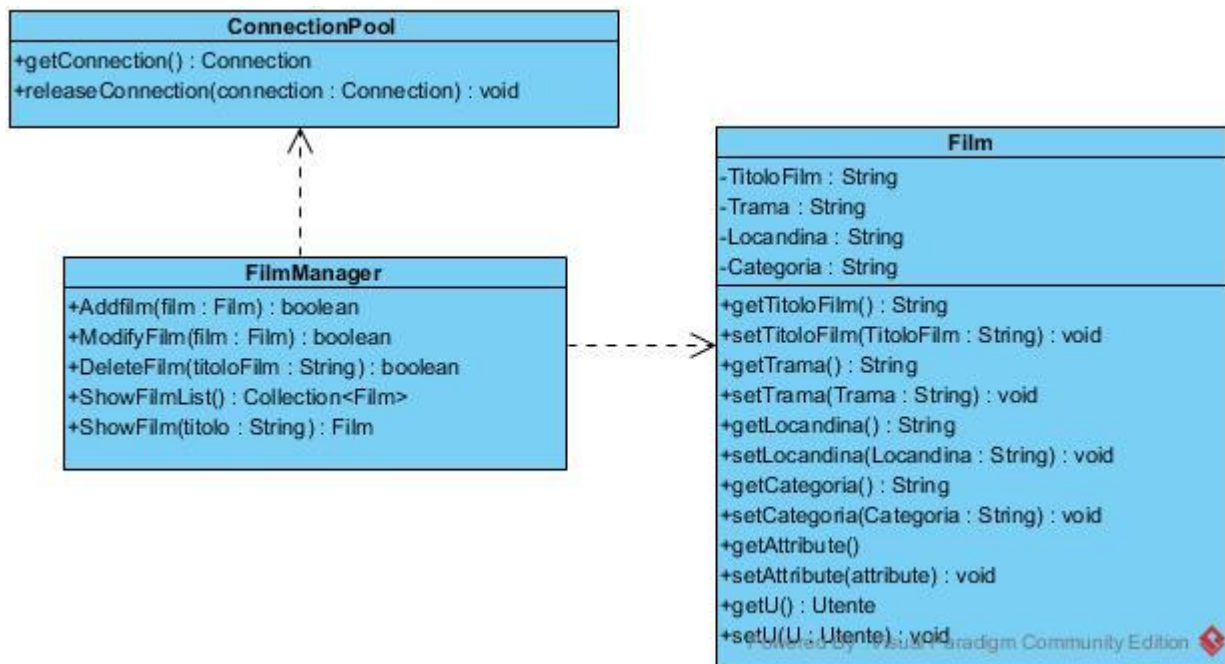
3.2 INTERFACE MANAGER

3.2.1 Authentication Manager



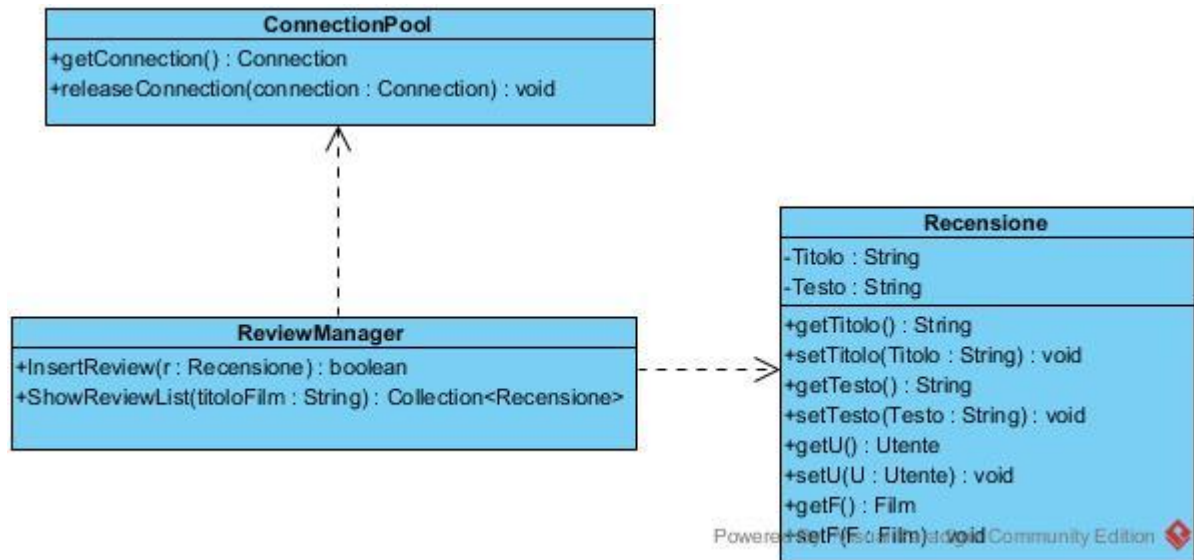
Nome: Authentication Manager	
Pre-condition	context AuthenticationManager::login(email:String,pass:String): Utente pre: email!=null && password !=null context AuthenticationManager::add(u:Utente): boolean pre: u!=null
Post-condition	context AuthenticationManager::login(email:String,pass:String): Utente post: utente!=null
Invarianti	

3.2.2 Film Manager



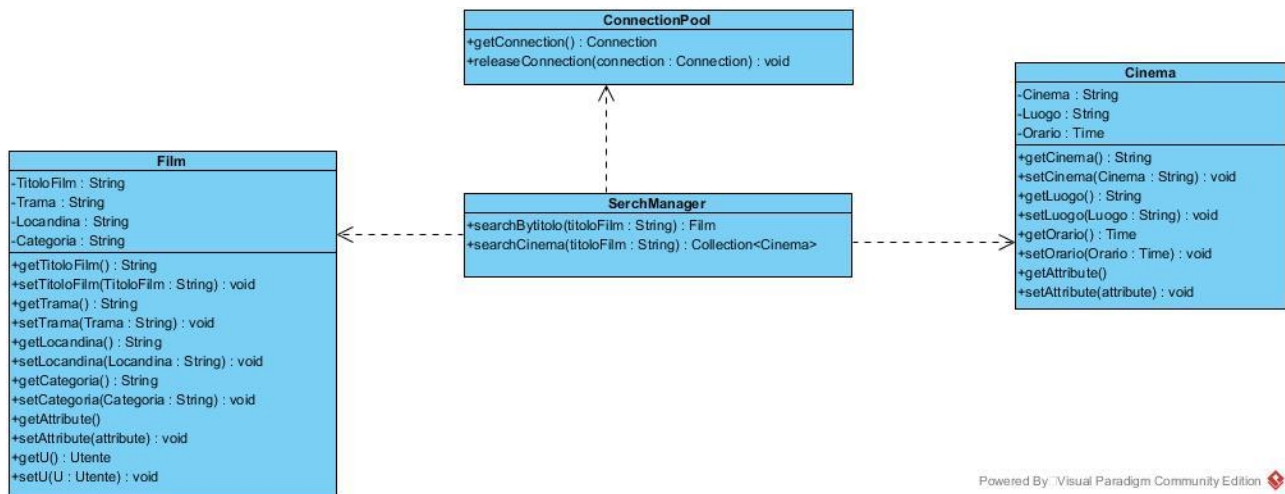
Nome: Film Manager	
Pre-condition	context FilmManager::AddFilm(film:Film): boolean pre: film!=null context FilmManager::ModifyFilm(film:Film): boolean pre: film!=null context FilmManager::DeleteFilm(titoloFilm:String): boolean pre: titoloFilm!=null context FilmManager::ShowFilm(titolo:String): Film pre: titolo!=null
Post-condition	
Invarianti	

3.2.3 Review Manager



Nome: Review Manager	
Pre-condition	context ReviewManager::InsertReview(r:Recensione): boolean pre: r!=null context ReviewManager::ShowReview(titoloFilm:String): Collection <Recensione> pre titoloFilm!=null
Post-condition	
Invarianti	

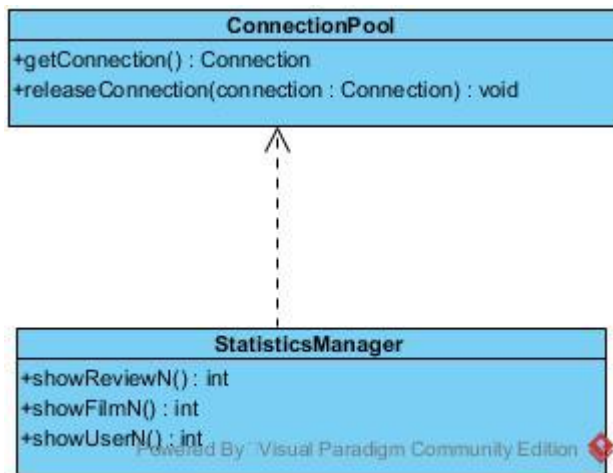
3.2.4 Search Manager



Powered By Visual Paradigm Community Edition

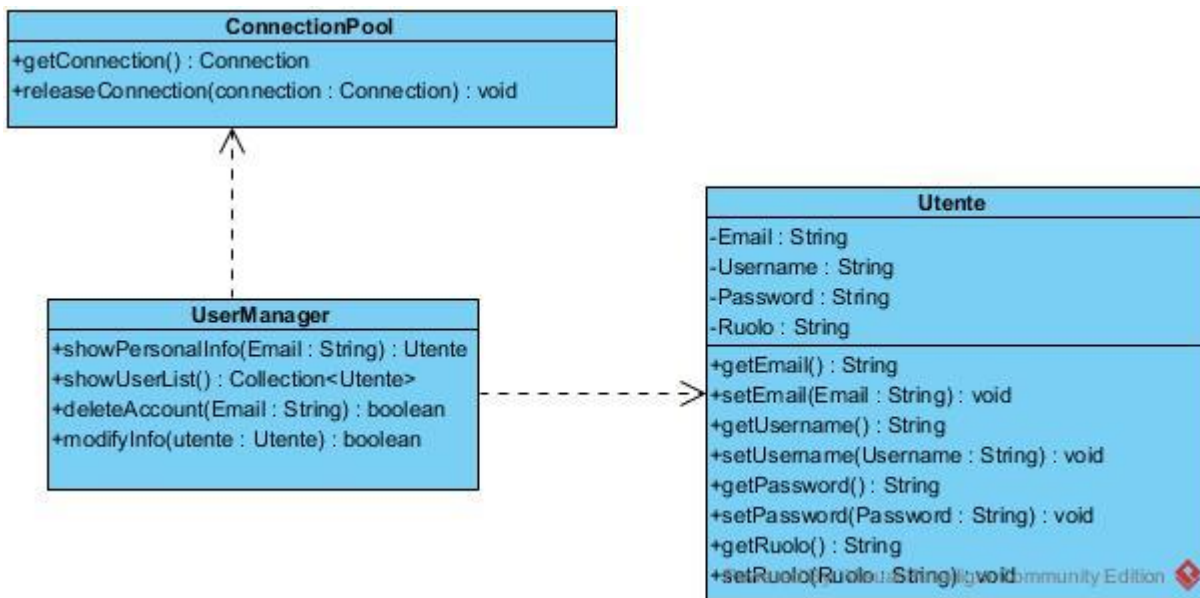
Nome: Search Manager	
Pre-condition	context SearchManager::searchBytitolo(titoloFilm:String): Film pre titoloFilm!=null context SearchManager::searchCinema(titoloFilm:String): Collection <Cinema> pre titoloFilm!=null
Post-condition	
Invarianti	

3.2.5 Statistics Manager



Nome: Statistics Manager	
Pre-condition	context StatisticsManager::showReiewN(): int pre utente!=null context StatisticsManager::showFilmN(): int pre utente!=null context StatisticsManager::showUserN(): int pre utente!=null
Post-condition	
Invarianti	

3.2.6 User Manager

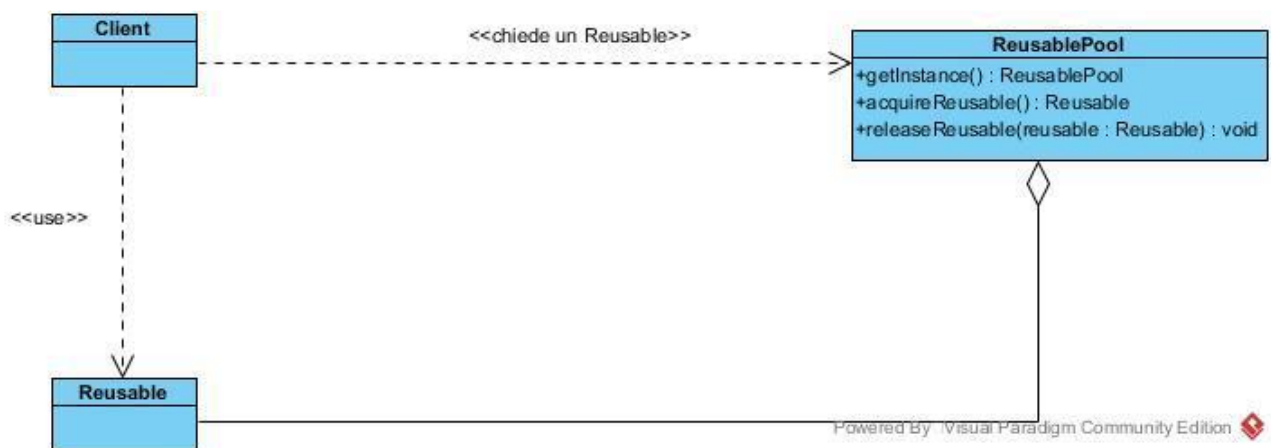


Nome: User Manager	
Pre-condition	context UserManager::showPersonalInfo(Email:String): Utente pre Email!=null context UserManager::deleteAccount(Email:String): boolean pre Email!=null context UserManager::modifyInfo(utente:Utente): boolean pre utente!=null
Post-condition	
Invarianti	

4 DESIGN PATTERN

4.1 OBJECT POOL PATTERN

L'Object pool pattern è un design pattern creazionale che usa un insieme di oggetti inizializzati pronti per l'uso mantenuti in una "pool", che si occupa di allocarli e de-allocherà su richiesta. Il client della pool invierà richiesta a un oggetto nella pool ed eseguirà operazioni sull'oggetto ritornato. Quando il client ha finito, ritorna l'oggetto alla pool che lo de-allocherà.



Utilizzo: L'Object Pool Pattern sarà utilizzato per gestire le connessioni con il database. Più precisamente, un oggetto Model richiederà connessioni al DriverManagerPool che ritornerà un oggetto Connection, l'oggetto Model effettuerà operazioni con l'oggetto connection e successivamente richiederà al DriverManagerPool la de-allocazione.

