UNIVERSITÀ DEGLI STUDI DI SALERNO

Ingegneria del Software

Partecipanti:

Gerardo Peluso mat [0512102534]

Vincenzo Russo mat[0512102436]

OBJECT DESIGN DOCUMENT

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 la signature dei sottosistemi definiti nel System Design.

Comprensibilità vs Tempo:
Il codice deve essere più comprensibile possibile per facilitare la fase di testing ed eventuali future modifiche. Il codice sarà quindi accompagnato da commenti . Ovviamente questa caratteristica aggiungerà un incremento di tempo allo sviluppo del nostro progetto.

Prestazioni vs Costi:
Essendo il progetto sprovvisto di budget, al fine di mantenere prestazioni elevate, per alcune funzionalità verranno utilizzati dei template open source esterni.

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 da parte dell'utente finale.

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.

1.2 Linee Guida per la Documentazione delle Interfacce

Gli sviluppatori dovranno seguire alcune linee guida per la scrittura del codice:

Naming Convention

- E' buona norma utilizzare nomi:
  - Descrittivi
  - Pronunciabili
  - Di uso comune
  - 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 per facilitare la leggibilità.

- E' inoltre possibile, in alcuni casi, utilizzare il carattere **underscore** "_", ad esempio quando utilizziamo delle variabili costanti oppure come prefisso di variabili d'istanza.

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(). Le variabili dei metodi devono essere dichiarate appena prima del loro utilizzo e devono servire per un solo scopo, per facilitare la leggibilità.

- 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:

- I nomi delle classi devono cominciare con una lettera maiuscola, e anche le parole seguenti all'interno del nome devono cominciare con una lettera maiuscola. I nomi delle Classi devono fornire informazioni sul loro scopo.

1.3 Definizioni, acronimi e abbreviazioni

Acronimi:

- RAD: Requirements Analysis Document
- SDD: System Design Document
- ODD: Object Design Document

Abbreviazioni:

- DB: DataBase

1.4 Riferimenti

- Documento SDD del progetto CinemaItalia
- Documento RAD del progetto CinemaItalia

2. Packages

La gestione del nostro sistema è suddivisa in due livelli (thin-client):

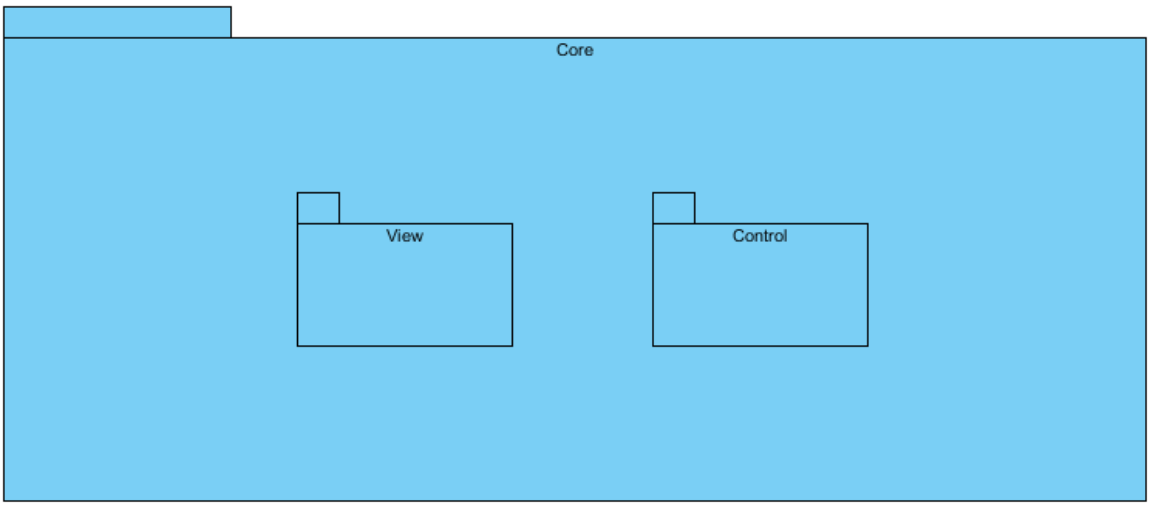
- Server
- Client

Il package CinemaItalia contiene sottopackage che a loro volta inglobano classi atte alla gestione delle richieste utente. Le classi contenute nel package svolgono il ruolo di gestore logico del sistema.

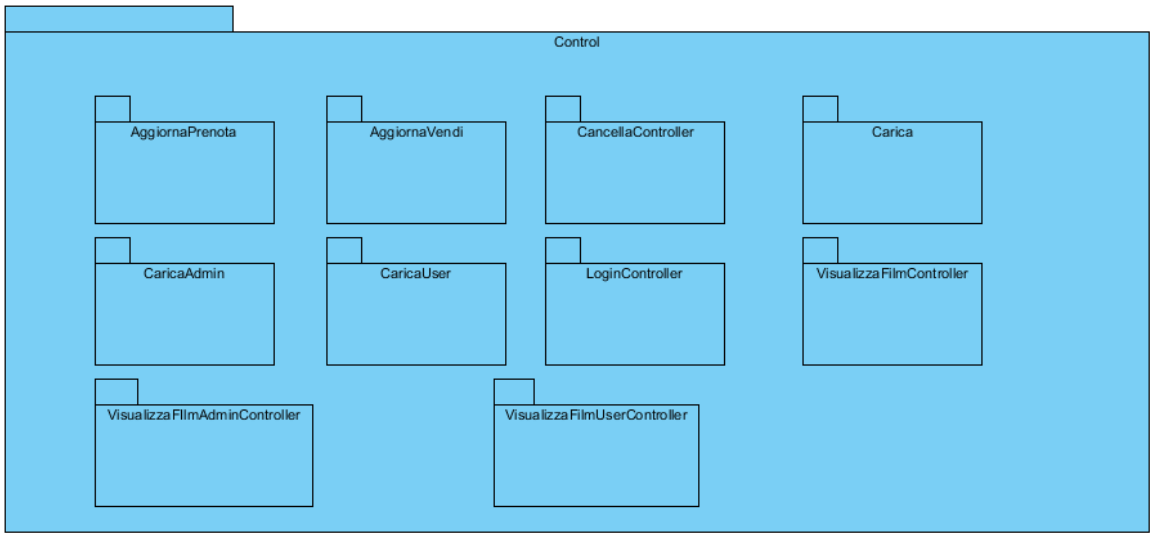
Client	Rappresenta l'interfaccia del sistema ed offre la possibilità all'utente di interagire con quest'ultimo, offrendo sia la possibilità di inviare, in input, che di visualizzare, in output, dati.

Server	Ha il compito di elaborare i dati da inviare al client e spesso grazie a delle richieste fatte al database accede ai dati persistenti. Ha il compito di memorizzare i dati sensibili del sistema, utilizzando un DBMS, inoltre riceve le varie richieste client inoltrandole al DBMS e restituendo i dati richiesti.

	Si occupa di varie gestioni quali: • **Gestione Registrazione** • **Gestione Autenticazione** • **Gestione Cliente** • **Gestione Film** • **Gestione Sala** • **Gestione Prenotazione**



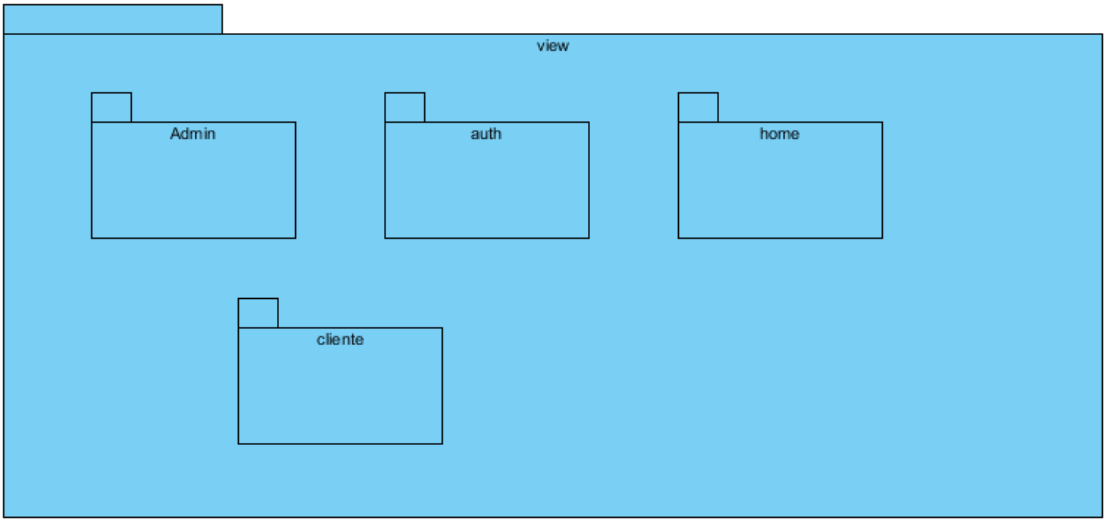
2.1.1 Package control



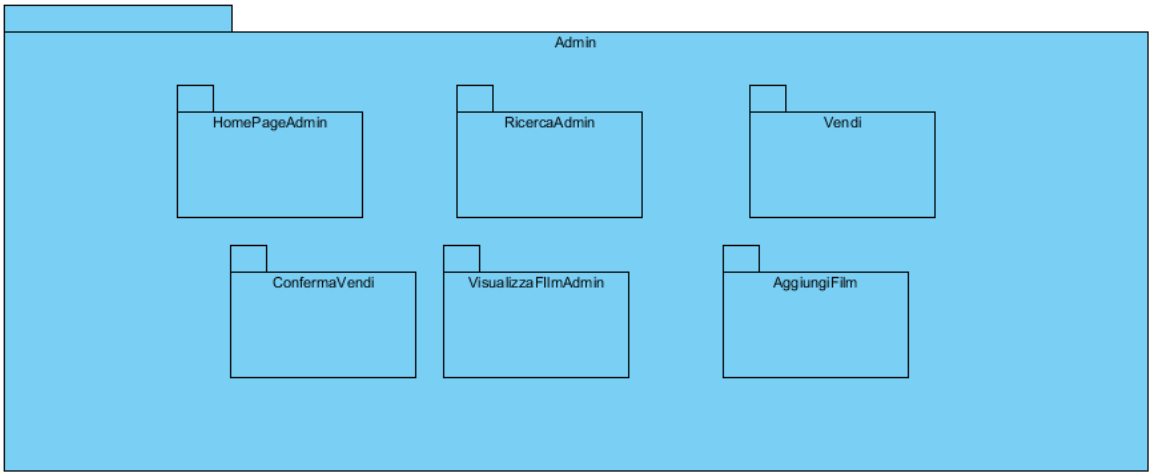
Classe:	**Descrizione:**
AggiornaPrenota.php	Il controller che gestisce le prenotazioni che effettua un cliente.
AggiornaVendi.php	Il controller che gestisce la conferma delle vendite.
LoginController.php	Il controller che gestisce gli utenti iscritti al sistema
CancellaController.php	Il controller che gestisce la cancellazione dei film
Carica.php	Il controller che gestisce la visualizzazione dei film per chi non è registrato.
CaricaAdmin.php	Il controller che gestisce la visualizzazione dei film per l'Admin.
CaricaUser.php	Il controller che gestisce la visualizzazione dei film per l'User.
VisualizzaFilmController.php	Il controller che gestisce la visualizzazione delle informazioni dei film per chi non è registrato.
VisualizzaFilmAdminController.php	Il controller che gestisce la visualizzazione delle informazioni dei film per l'Admin

VisualizzaFilmUserController.php	Il controller che gestisce la visualizzazione delle informazioni dei film per l'User.

2.1.2 Package view

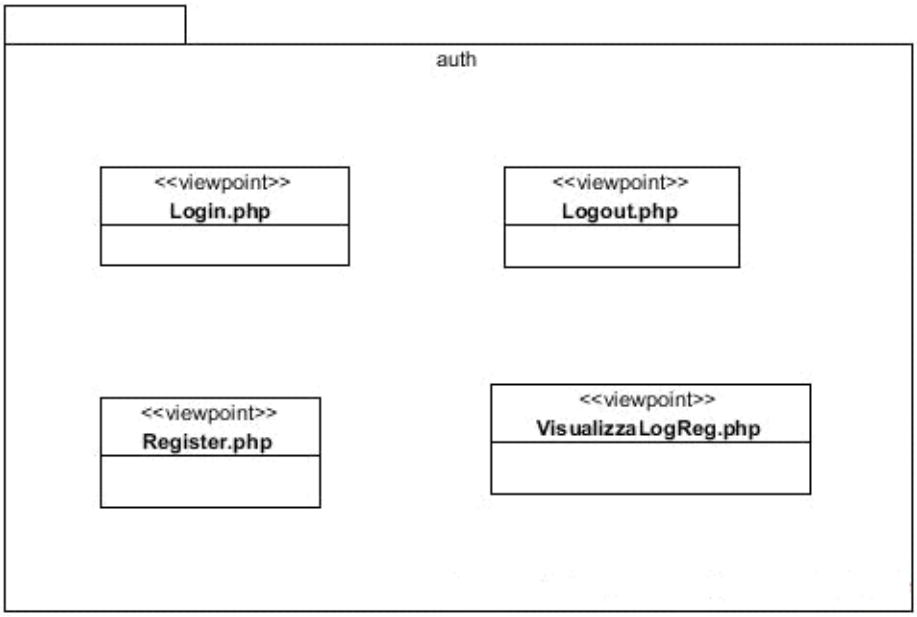


2.1.2.1 Package admin



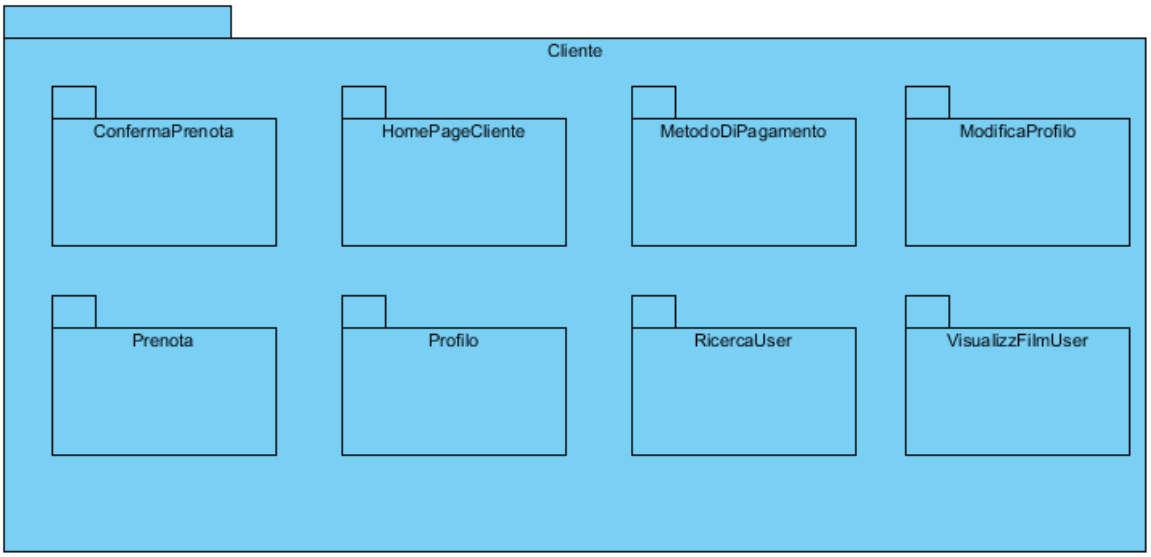
AggiungiFIlm.php	La view che consente all’admin di aggiungere un film.
VisualizzaFilmAdmin.php	La view che consente all’admin visualizzare le informazioni dei film.
Vendi.php	La view che consente all’admin di vendere un biglietto.
ConfermaVendi.php	La view che consente all’admin di stampare un biglietto
RicercaAdmin.php	La view che consente all’admin di ricercare un film.
HomePage.php	La view che rappresenta homepage dell'admin

2.1.2.2 Package auth



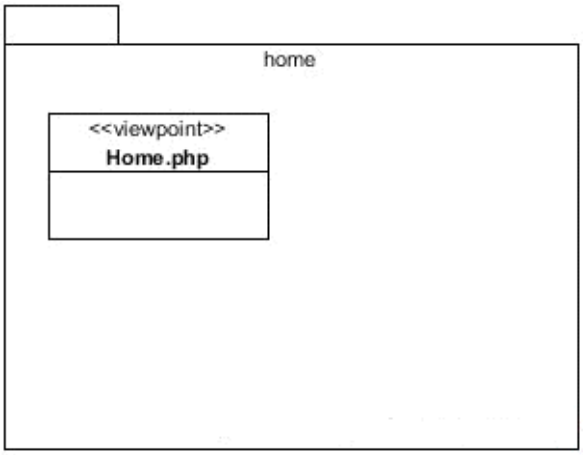
Login.php	Gestisce il login degli utenti.
Logout.php	Gestisce il logout degli utenti.
Register.php	Gestisce la registrazione degli utenti
VisualizzaLogReg.php	Gestisce tutti i log della registrazione

2.1.2.4 Package cliente



VisualizzaFilmUser.php	La view che consente all'user visualizzare le informazioni dei film.
Prenota.php	La view che consente al cliente di prenotare un biglietto.
MetodoPagamento.php	La view che consente al cliente di scegliere un metodo di pagamento.
ModificaProfilo.php	La view che consente al cliente di modificare il proprio profilo.
HomePageCliente.php	La view che rappresenta homepage dell'user
Profilo.php	La view che consente al cliente di visualizzare il proprio profilo.
RicercaUser.php	La view che consente al cliente di ricercare un film.
ConfermaPrenota.php	La view che consente di stampare il biglietto.

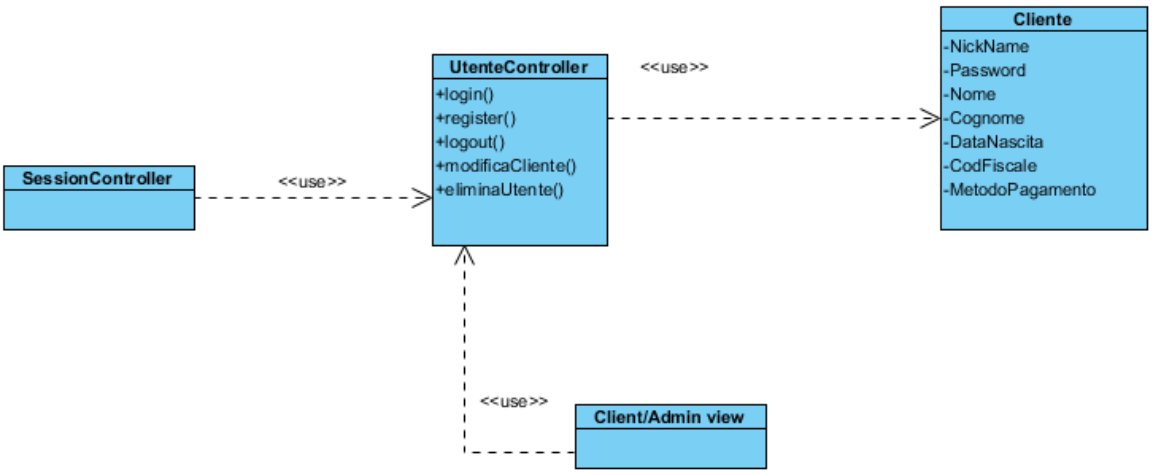
2.1.2.5 Package home



Home.php	La home di CinemaItalia

3. Class interfaces

3.0 Gestione Cliente



Classe: UtenteController			
Descrizione	La classe *UtenteController* si occupa di far interagire *Utente* e *Film* con le view *VisualizzaFilm, Profilo, ModificaProfilo e Prenota*		
Dipendenze	*UtenteController.php, VisualizzaFilm.php*		
Metodi			
Nome	**Accesso**	**Descrizione**	**Eccezioni**
login	pubblico	Metodo effettua il login, dato l'id e la password. Inizializza la sessione	
logout	pubblico	Effettua l'eliminazione delle sessioni	
register	pubblico	Metodo prende tutti i parametri necessari per una registrazione di un nuovo utente, effettua se i campi sono tutti inseriti e validi e crea l'utente	
modificaCliente	pubblico	Effettua la modifica delle informazioni relative all'utente	

Classe: *Utente*		
Descrizione	*Utente* rappresenta l'oggetto "Utente" nel database	
Dipendenze	*UtenteController.php*	
Attributi		
Nome	**Accesso**	**Descrizione**
ID	Privato	Rappresenta l'ID nonchè

		l'identificativo di un singolo Utente
nickname	Privato	Rappresenta il nickname di un utente
password	Privato	Rappresenta la password dell'utente
Tipologia	Privato	Rappresenta la tipologia dell'utente, Admin o cliente
nome	Privato	Rappresenta il nome dell'utente
cognome	Privato	Rappresenta il cognome dell'utente
CodFiscale	Privato	Rappresenta il codice fiscale dell'utente.
DataNascita	Privato	Rappresenta la data di nascita dell'utente
MetodoPagamento	Privato	Rappresenta il metodo di pagamento con cui il cliente acquista i biglietti.

Gestione Film

Classe: *Film*		
Descrizione	*film* rappresenta l'oggetto "film " nel database	
Dipendenze	VisualizzaFilmController.php,	
Attributi		
Nome	**Accesso**	**Descrizione**
ID	Privato	Rappresenta l'ID nonchè l'identificativo di un singolo film
Titolo	Privato	Rappresenta il titolo del film
Genere	Privato	Rappresenta la tipologia a cui appartiene il FIlm
Cast	Privato	Rappresenta il cast del film.
Durata	Privato	Rappresenta la durata del film.
Sala	Privato	Rappresenta la sala in cui viene proiettato il film
Prezzo	Privato	Rappresenta il costo del biglietto.
Data	Privato	Rappresenta la data della

		proiezione del film
Ora	Privato	Rappresenta l'ora in cui viene proiettato il film.
Regia	Privato	Rappresenta la regia del film

Classe: VisualizzaFilmController			
Descrizione	La classe Film*Controller* si occupa di far interagire *Film, Utenti* e *Sala*		
Dipendenze	Film		
Metodi			
Nome	**Accesso**	**Descrizione**	**Eccezioni**
AggiungiFilm	Pubblico	Metodo che prende in input tutti gli attributi e inserisce nel Database un film.	
visualizzaFIlm	Pubblico	Metodo che prende in input Id film e lo fa visualizzre	
cancellaFilm	Pubblico	Metodo che permette di eliminare i film	

Gestione Sala

Classe: Sala			
Descrizione	*Sala rappresenta l'oggetto Sala nel Database*		
Dipendenze			
Attributi			
Nome	**Accesso**	**Descrizione**	
Metodi			
Nome	**Accesso**	**Descrizione**	**Eccezioni**
ID	Pubblico	ID compreso tra 1-4 per identifaicare la sala	
NumeroColonne	Pubblico	Attributo che ci indica il nunmero di posti per fila	
NumeroRIghe	Pubblico	Attributo che ci indica il nunmero di file	

Classe: SalaController			
Descrizione	La classe SalaController si occupa di far interagire Utente e FIlm.		
Dipendenze			
Metodi			
Nome	**Accesso**	**Descrizione**	**Eccezioni**
prenota	Pubblico	Metodo che permette di aggionare la disponibilita dei posti in sala.	
createSala	Pubblico	Metodo che prende in input una Sala e lo inserisce nel DB attraverso l'apposita query.	
readSala	Pubblico	Metodo che prende in input l'ID della sala e restituisce la mappa della sala..	