|  |
| --- |
| **Università degli Studi di Salerno Corso di Ingegneria del Software** |

**Analytics Films  
Object Design Document  
Versione 1.0**



Data: 11/01/2019

**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 |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autore |
| 12/11/2018 | 0.1 | Prima stesura |  |
| 23/11/2018 | 0.3 | Aggiunti diagrammi packages |  |

Indice

1. Introduzione ………………………………………………………………….

1.1 Object Design Trade-offs ……………………………………………..

1.2 Linee Guida per la Documentazione delle Interfacce……………..

1.3 Definizioni, acronimi e abbreviazioni…..…………………………..

1.4 Riferimenti ………………………………………..

2. Packages ……

3. Class interfaces …

4.Design Pattern

1. **Introduzione**
   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. **Linee Guida per la Documentazione delle Interfacce**

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

**Naming Convention**

* 1. • È buona norma utilizzare nomi:
  2. • Descrittivi
  3. • Di uso comune
  4. • 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. **Definizioni, acronimi e abbreviazioni**
  2. **Riferimenti**

1. **Packages**

**Control**

**Bean**

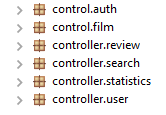
**ConnectionPool**

**Model**

**View**

**Test**

**Package control**

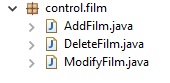
****

**controller.auth**

****

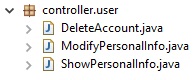
|  |  |
| --- | --- |
| **Classe** | **Descrizione** |
| Login.java | Servlet che gestisce il login |
| Logout.java | Servlet che gestisce il logout |
| RecovePass.java | Servlet che gestisce il recupero password |
| Registration.java | Servlet che gestisce la registrazione |

**controller.film**

****

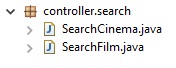
|  |  |
| --- | --- |
| **Classe** | **Descrizione** |
| AddFilm.java | Servlet che gestisce l’aggiunta di un film |
| DeleteFilm.java | Servlet che gestisce la rimozione di un film |
| ModifyFilm.java | Servlet che gestisce la modifica di un film |

**controller.user**

****

|  |  |
| --- | --- |
| **Classe** | **Descrizione** |
| DeleteAccount.java | Servlet che gestisce l’eliminazione di un account |
| ModifyPersonalInfo.java | Servlet che gestisce la modifica delle info personali |
| ShowPersonalInfo.java | Servlet che gestisce la visualizzazione delle info personali |

**controller.search**

****

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

**controller.review**

****

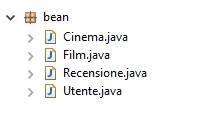
|  |  |
| --- | --- |
| **Classe** | **Descrizione** |
| InsertReview.java | Servlet che gestisce l’inserimento di una recensione |

**controller.statistics**

****

|  |  |
| --- | --- |
| **Classe** | **Descrizione** |
| ShowStatisticsFilms.java | Servlet che gestisce la visualizzazione delle statistiche dei films. |
| ShowStatisticsUsers.java | Servlet che gestisce la visualizzazione delle statistiche degli utenti. |

**Package bean**

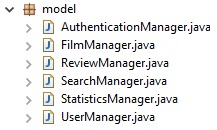
****

|  |  |
| --- | --- |
| **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 |

**Package connectionPool**

|  |  |
| --- | --- |
| **Classe** | **Descrizione** |
| **ConnectionPool.java** | **Classe che implementa l’object Pool Pattern, responsabile di fornire connessione con database.** |

**Package model**

****

**AuthenticationManager.java**

Descrive l’interazione con il database per le operazioni di login, recovePass e registration**.**

**FilmManager.java**

Descrive l’interazione con il database per le operazioni di add, modify e delete film.

**ReviewManager.java**

Descrive l’interazione con il database per le operazioni di insertReview.

**SearchManager.java**

Descrive l’interazione con il database per le operazioni di searchCinema e searchFilm.

**StatisticsManager.java**

Descrive l’interazione con il database per le operazioni di showStatisticsFilms e showStatisticsUsers.

**UserManager.java**

Descrive l’interazione con il database per le operazioni di showPersonalInfo, modifyPersonalInfo, deleteAccount.

**Package view**

**Homepage.jsp**

Package test

4.Design Pattern

Spiegazione cosa fanno e foto class diagram su internet)

Façade Pattern

Object Pool Pattern