|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Object Design Document  Progetto  GuardaTV   |  |  | | --- | --- | | Riferimento |  | | Versione | 0.1 | | Data | 22/12/2021 | | Destinatario | Studenti di Ingegneria del Software 2021/22 | | Presentato da | Gruppo 16:  N. Cacace  S. Pastore  A. Prezioso  A. Ricchetti | | Approvato da |  | |

RevisionHistory

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

1. Introduction

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. Object design goal

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 fare le query sono stati utilizzati oggetti PreparedSatement per la protezione da attacchi Sql Injection.

* 1. Linee guida per la documentazione dell’interfaccia

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

Naming convention

● E’ buona norma utilizzare nomi:

1. Descrittivi

2. Pronunciabili

3. Di uso comune

4. Di lunghezza medio-corta

5. Non abbreviati

6. Evitando la notazione ungherese

7. 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:

1. Dichiarazione della classe pubblica

2. Dichiarazioni di costanti

3. Dichiarazioni di variabili di classe

4. Dichiarazione di variabili d’istanza

5. Costruttore

6. Commento e dichiarazione dei metodi

1.3 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. Object Trade-off
  2. Definizioni, acronimi e abbreviazioni
  3. Riferimenti

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

Statement of work

Requirements analysis document

System design document

1. Packages

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

• .mvn, contiene tutti i file di configurazione per Maven

• src, contiene tutti i file sorgente o main

▪ java, contiene le classi Java relative ai layer di application e storage

▪ resources, contiene i file relativi al layer di presentation

▪ test, contiene tutto il necessario per il testing

▪ java, contiene le classi Java per l’implementazione del testing

Package Guardatv

Nella presente sezione si mostra la struttura del package pricipale 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 eventuli classi di utlità per il sistema e usabili da più sottosistemi.

Gestione utente

|  |  |
| --- | --- |
| Classe: | Descrizione |
| RegistrazioneServlet |  |
| LoginServlet |  |
| LogoutServlet |  |
| ModificaPasswordServlet |  |
| ElencoRecensioniServlet |  |
| ElencoListeServlet |  |
| VisualizzaUtenteServlet |  |

Gestione contenuto

|  |  |
| --- | --- |
| ContenutoServlet |  |

Gestione ricerca

|  |  |
| --- | --- |
| RicercaServlet |  |

Gestione amministratore

|  |  |
| --- | --- |
| AggiuntaContenutoServlet |  |
| ModificaContenutoServlet |  |
| RimozioneContenutoServlet |  |

Gestione recensione

|  |  |
| --- | --- |
| AggiuntaRecensioneServlet |  |
| ModificaRecensioneServlet |  |
| RimozioneRecensioneServlet |  |

Gestione lista

|  |  |
| --- | --- |
| CreazioneListaServlet |  |
| EliminazioneListaServlet |  |
| AggiuntaContenutoListaServlet |  |
| RimozioneContenutoListaServlet |  |
| VisualizzaListaServlet |  |
| CondivisioneLista\* |  |

1. Class Interfaces
2. Design Patterns

•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 patter 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.