|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

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
  2. 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\* |  |

LoginDAO

+Login doRetrieveByID(string id)

+Void doSave(Login login)

+void doUpdate(Login login)

+void doDelete(String id)

UtenteDAO

+Utente doRetrieeByid/email(id/email)

+Utente doRetrieveByEmailPassword(email password)

+void doSave(utente)

+void doUpdate(utente) //conntrollare solo password?

+void doDelete(id/email)

ContenutoDAO

+List<Contenuto> doRetrieveAll(offset, limit)

+List<Contenuto> doRetrieveLast(offset, limit)

+Contenuto doRetrieveById(id)

+List<Contenuto> doRetriveByGenere(genere, offset, limit)

+List<Contenuto> doRetriveByNome(against, offset, limit)

+List<Contenuto> doRetriveByNomeorDescrizione(against, offset, limit)

+List<Contenuto> doRetriveByNomeorDescrizioneandGenere(against, genere, offset, limit)

//doretrieve con regista o per durata (against, durata, offset, limit)

+void doSave(Contenuto)

+void doUpdate(Contenuto)

+void doDelete(id)

RecensioneDAO

+List<Recensione> doRetrieveByContenuto(idcontenuto, offset, limit)

+List<Recensione> doRetrieveByUtente(idutente, offset, limit)

+Recensione doRetrieveById(idutente, idcontenuto)

+void doSave(Recensione)

+void doUpdate(Recensione)

+void doDelete(idutente, idcontenuto) //da rivedere

GenereDAO

+List<Genere> doRetrieAll()

+Genere doRetrieveByName/ID (string nome/id)

ListaDAO

+List<Lista> doRetrieveByUtenet(idutente, offset, limit)

+Lista DoRetrievebyId (idutente, nomelista, offset, limit) // idlista = idutente+nomelista

+void DoSave(Lista)

+void doDelete(idutente,nomelista)

+Lista RemoveContenuto(contenuto)

+Lista AddContenuto(contenuto)

1. Class Interfaces
2. Design Patterns