MovieTips

1. Introduzione

Nonostante il successo dei social network è ancora difficile trovare persone con gli stessi interessi e molto spesso si incappa in community non gestite propriamente o con persone non realmente interessate all'argomento trattato. Per questo motivo abbiamo deciso di creare **MovieTips**, un sito web che dà la possibilità agli appassionati di cinema di trovare una community accogliente e piena d'entusiasmo, con un sistema di recensioni appagante, in cui ogni utente può lasciare mi piace alle recensioni degli altri e visitare il loro profilo.

1.1 Object design trade-offs

Functionality vs. **Usability**
Considerando il sistema che stiamo realizzando abbiamo preferito concentrarci sull'usabilità del sito anziché implementare troppe funzionalità. In particolare, abbiamo posto grande attenzione alle interazioni che l'utente ha col sito.

Rapid development vs. Functionality
Considerando i tempi ridotti a nostra disposizione abbiamo preferito adottare uno sviluppo rapido anziché implementare funzionalità secondarie non necessarie nei primi mesi di vita del sito.

Cost vs. Reusability
Considerando il budget ristretto a nostra disposizione abbiamo preferito minimizzare i costi di sviluppo a discapito della riusabilità.

1.2 Linee guida per l'interfaccia

- Le classi vengono denominate con nomi al singolare, i metodi con verbi, le variabili con nomi e le costanti con parole scritte con lettere maiuscole separate dal trattino basso.
- i nomi vengono scelti per esprimere il significato di ciò che rappresentano in modo da facilitare la comprensione del codice al programmatore;
- le classi e le interfacce vengono denominate utilizzando la notazione "upper camel case".
- gli errori vengono restituiti tramite un'eccezione e non per mezzo di un valore.

2. Packages

2.1 View

error: pagina di errore.
exception: pagina di errore.
film: la pagina consente di vedere le informazioni relative ad uno specifico film.
homePage: pagina principale, dove viene mostrato l'header e alcuni film.
login: pagina utilizzata dagli utenti per effettuare il login.
profilo: pagina che permette la visualizzazione dei dati utente.
ricerca: pagina dove vengono mostrati tutti i risultati all'input inserito per la ricerca.
segnalazioni: pagina esclusiva per il moderatore, dove vengono visualizzate tutte le recensioni segnalate.

2.2 Control

Film: gestisce la corretta visualizzazione della pagina di un film.
Home: gestisce la corretta visualizzazione della homepage.
Ricerca Film: gestisce la corretta visualizzazione della pagina dei risultati della ricerca.

AggiungiRecensione: permette a un filmino di aggiungere la recensione relativa a un film all'interno del database.
EliminaRecensione: permette a un un filmino di eliminare una sua recensione.
GestioneSegnalazioni: gestisce la corretta visualizzazione della pagina riservata al moderatore inserendo tutte le recensioni segnalate.
IgnoraSegnalazione: permette a un moderatore di ignorare una recensione segnalata.
ModeraRecensione: permette a un moderatore di eliminare una recensione segnalata.
SegnalaRecensione: permette a un utente di di segnalare la recensione di un altro utente.

Accesso: gestisce il login inserendo l'utente nella sessione.
BannaUtente: permette a un moderatore di bannare un utente.
Disconnessione: gestisce il logout rimuovendo l'utente dalla sessione.
Profilo: gestisce la corretta visualizzazione della pagina profilo di un utente.

2.3 Model

ConPool: consente la connessione al database.
Film: classe che rappresenta un oggetto Film.
FilmDAO: gestisce le interazioni con la tabella del database Film.
Recensione: classe che rappresenta un oggetto Recensione.
RecensioneDAO: gestisce le interazioni con la tabella del database Recensione.
Utente: classe che rappresenta un oggetto Utente.
UtenteDAO: gestisce le interazioni con la tabella del database Utente.

3. Class interface

FilmDAO
public Film getFilm(int id) **pre**: **post**: Il metodo restituisce il film con il campo id_film = id se esiste, altrimenti restituisce null **public List<Film> doRetrieveLastTen()** **pre**: **post**: Il metodo restituisce 10 film ordinati in modo decrescente rispetto all'id del film **public List<Film> searchFilms(String ricerca)** **pre**: 1 ≤ ricerca.length() ≤ 255 **post**: Il metodo restituisce una lista di film che contengono la stringa "ricerca" all'interno del titolo o, se non c'è nessuna corrispondenza, restituisce una lista vuota

RecensioneDAO

public List<Recensione> getReviewsByFilm(int idFilm)
pre:
post: Il metodo restituisce la lista di recensioni, caricate nel database, associate ad un film (il campo id_film della recensione deve essere uguale a idFilm), altrimenti restituisce una lista vuota

public int addReview(int valutazione, String testo, int idFilm, Utente utente)
pre: utente ≠ null e utente.isFilmino() = true e 1 ≤ valutazione ≤ 5 e 1 ≤ testo.size() ≤ 255
post: La recensione viene salvata nel database

public int deleteReview(int idRecensione, Utente utente)
pre: utente ≠ null
post: La recensione con il campo id_recensione = idRecensione viene eliminata dal database

public int getReportedReviews(Utente utente, List<Recensione> recensioni)
pre: utente ≠ null e utente.isModeratore = true
post: Il metodo restituisce una lista di recensioni che hanno il campo segnalazione = true, altrimenti restituisce una lista vuota

public int ignoreReporting(int idRecensione, Utente utente)
pre: utente ≠ null e utente.isModeratore() = true**post**: Il campo segnalazione della recensione con il campo id_recensione = idRecensione viene settato a false

public int moderateReview(int idRecensione, Utente utente)
pre: utente ≠ null e utente.isModeratore() = true
post: La recensione con il campo id_recensione = idRecensione viene eliminata dal database

public int reportReview(int idRecensione, Utente utente)
pre: utente ≠ null e utente.isActive() = true
post: Il campo segnalazione della recensione con il campo id_recensione = idRecensione viene settato a true

public List<Recensione> getReviewsByUser(String username)
pre:
post: Il metodo restituisce la lista di recensioni, caricate nel database, associate ad un username (il campo username_utente della recensione deve essere uguale a username), altrimenti restituisce una lista vuota

UtenteDAO
public int signIn(String mail, String password, Utente utente) **pre**: **post**: Il metodo restituisce un utente con il campo mail (database) = mail e password (database) = SHA1(password) se il campo utente.ruolo ≠ “100000” (utente bannato), altrimenti restituisce null **public int banUser(String username, Utente utente)** **pre**: utente ≠ null e utente.isModeratore() = true e utente.username ≠ username **post**: Il campo ruolo dell’utente con il campo username (database) = username viene settato a “100000” (utente bannato) **public Utente getUser(String username)** **pre**: **post**: Il metodo restituisce l’utente che ha il campo username (database) = username

4. Glossario

Filmino: utente normale registrato.