Object Oriented Software Design (titolo)

Progetto 2016/2017

Collevecchio Lorenzo 242384 lorenzo.collevecchio@student.univaq.it

Ficorilli Matteo 242078 matteo.ficorilli@student.univaq.it

Giostra Francesco 236997 [francesco.giostra@student.univaq.it](mailto:francesco.giostra@student.univaq.it)

Repository GitHub: Clicca qui

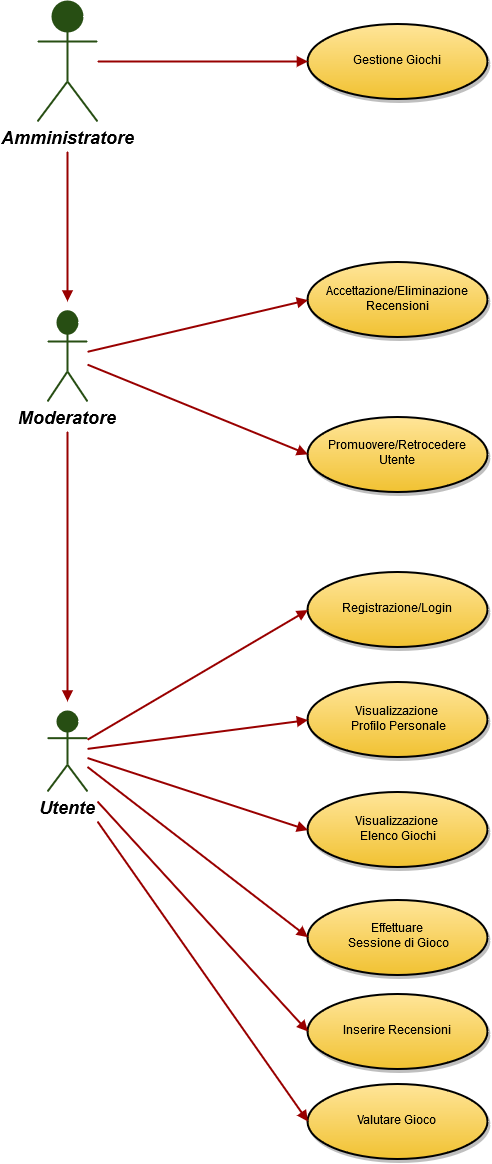
Indice

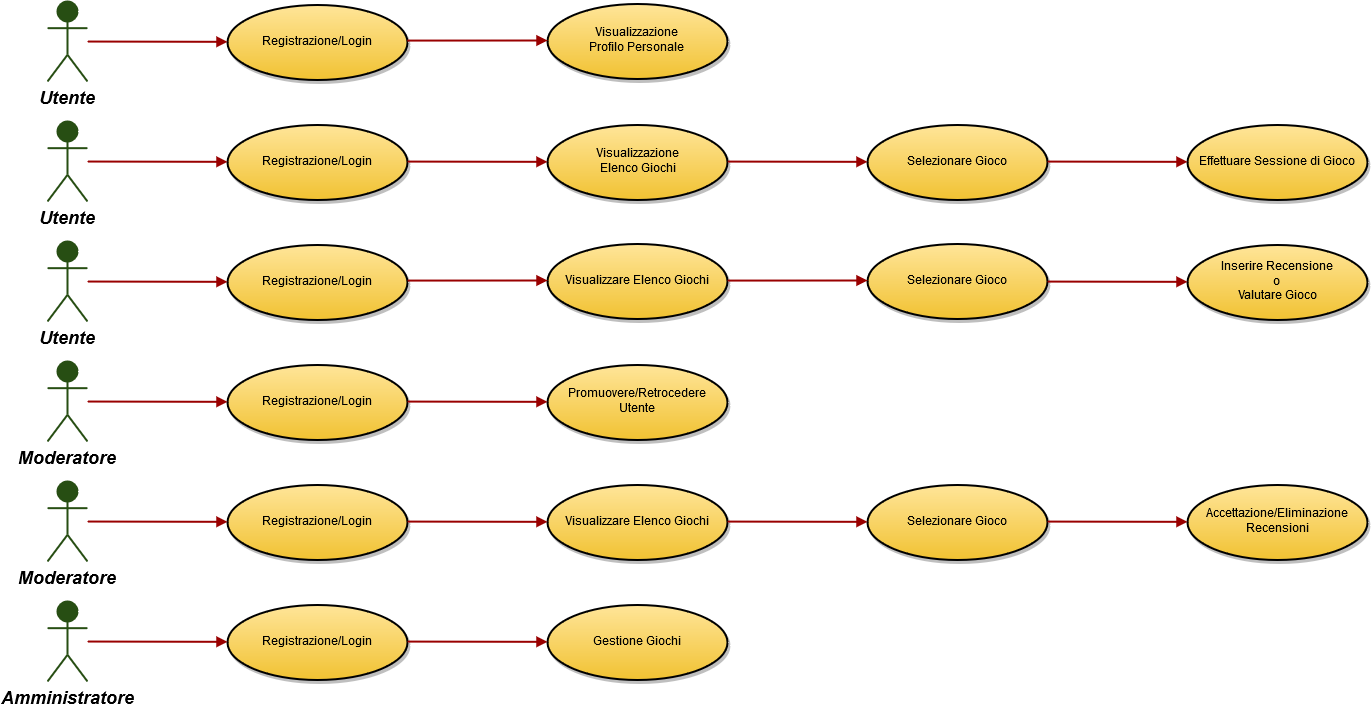
1. Requirments
   1. Requisiti Funzionali
   2. Requisiti non Funzionali
   3. Attori del Sistema e Use Cases
2. System Design
   1. Modello dell’architettura software
   2. Descrizione dell’architettura
   3. Descrizione delle scelte
      1. Strategie adottate
      2. Design Patterns
3. Software Object Design
   1. Modello Object Design
   2. Descrizione dettagli di design
4. Requisiti
   1. Requisiti Funzionali

* Registrazione/Login: Funzionalità che permette agli utenti e ai moderatori di accedere al sistema.
* Visualizzazione Elenco Giochi: Funzionalità che permette agli utenti di selezionare un gioco da recensire o giocare.
* Visualizzazione Profilo Personale: Funzionalità che permette agli utenti in generale di visualizzare i dettagli del proprio profilo personale.
* Inserire recensioni: Funzionalità che permette di inserire recensioni su un determinato gioco con la possibilità di esprimere un voto da 1 a 10 e di aggiungere un breve commento.
* Accettazione/Eliminazione recensioni: Funzionalità che permette ai soli moderatori di visualizzare le recensioni in attesa di approvazione e di accettarle o rifiutarle.
* Incremento EXP/Livello: Funzionalità che ad ogni sessione di gioco assegna 50 punti esperienza per ogni ora di gioco.
* Effettuare Sessione di Gioco: Funzionalità che permette agli utenti in generale di giocare ad un determinato gioco, simulata in questo progetto tramite un bottone.
* Ottenimento Premi: Funzionalità che permette agli utenti in generale di ottenere premi a seconda del livello raggiunto.
* Promuovere/Retrocedere Utente: Funzionalità che permette ai soli moderatori di visualizzare l’elenco degli utenti e di promuoverli da utente a moderatore e viceversa.
  1. Requisiti non Funzionali
* Usability: il sistema dovrà essere pulito e di facile utilizzo.
* Reliability: il sistema dovrò garantire all’utente le funzioni messe a disposizione senza errori.
* Availability: il sistema dovrà essere disponibile in qualsiasi momento
  1. Attori del Sistema
* Amministratore: L’amministratore non fa parte degli utenti presenti nel database perché non è un utilizzatore di questa piattaforma, l’amministratore è considerato come colui che ha accesso diretto al database e che ha la completa gestione di tutti i dati; può aggiungere giochi al sistema, rimuovere utenti, etc…
* Moderatore: Il Moderatore è una versione evoluta dell’utente che oltre alle funzioni base dell’utente ha altre funzionalità a disposizione; può promuovere un utente a moderatore o declassare un moderatore a utente.

Il Moderatore può accettare/rifiutare recensioni richieste dagli utenti prima di essere realmente pubblicate.

* Utente: L’utente è l’utilizzatore standard del sistema; può effettuare sessioni di gioco e richiedere ai moderatori recensioni con voto.

Use Case Diagram



1. System Design
   1. Modello dell’architettura software
   2. Descrizione dell’architettura software  
      Il sistema si appoggia su un Database MySQL che in fase di progettazione viene lanciato tramite l’applicazione XAMPP.  
      L’architettura è divisa nella View che sostanzialmente è l’interfaccia grafica con cui moderatori e utenti interagiscono con il sistema.  
      Essa è connessa al controller.

Il Controller si occupa di gestire tutte le funzionalità relative all’elaborazione e al mantenimento delle informazioni relative al sistema.  
Il Model contiene le classi che vengono successivamente utilizzate dal controller.

* 1. Descrizione delle scelte

2.3.1 Strategie adottate

Sono stati seguiti i principi del paradigma Object-Oriented, l’applicazione è stata divisa in due parti, una è quella che vede l’utenti e moderatori che è finalizzata a giocare e gestire le funzioni base del sistema; l’altra è quella che è destinata all’amministratore tramite la quale ha il controllo totale del sistema.

2.3.2 Design Patterns

Model View Controller

Model:Il Model è la struttura dei dati e delle funzioni che possono essere eseguite sui dati

View: La view è lo strumento utilizzato