Object-Oriented Programming

Progetto Piattaforma di Gaming

Università degli studi dell'Aquila

Membri del team (2° Anno):

- Cerone Luigi, matricola: 242895
- Rosati Danilo, matricola: 243748
- Taglieri Alessandro, matricola: 243488

Sommario

Sommario

Requirements

1.1 Documento dei requisiti

I requisiti del sistema sono i seguenti; a ciascuno di essi è associata una priorità su scala da 1 a 5 (1: importanza minima; 5: massima importanza).

- **Registrazione\Login** (5): Funzionalità che permette all'utente di potersi registrare una tantum al portale gaming e successivamente accedere al proprio profilo iniziando la sua esperienza di gioco.

- **Visualizzazione User-Profile**:

  - Sezione **Gaming** (4): Funzionalità che mostra all'utente tutte le informazioni relative al suo profilo gaming (timeline e livelli conquistati).

  - Sezione **anagrafica** (1): Funzionalità che mostra all'utente tutte le informazioni relative al suo profilo anagrafico.

- **Visualizzazione lista giochi** (5): Funzionalità che permette di entrare nella sezione dedicata alla lista dei giochi presenti nel sistema.

- **Attribuzione punteggio** (5): Funzionalità che ci indica i punti XP che si possono guadagnare per ogni gioco presente nella lista.

- **Guadagno XP** (5): Funzionalità che ad ogni bet assegna un totale Y di punti XP, relativi ad un determinato gioco, all'utente.

- **Crescita livello** (5): Funzionalità che ad ogni range di XP assegna un livello all'utente.

- **Collezione trofei** (4): Funzionalità che permette all'utente di ricevere particolari trofei rispettivamente al livello raggiunto.

- **Sessione di gioco** (5): Funzionalità che permette all'utente di selezionare il gioco e cominciare la sessione.

- **Valutazione del gioco** (2): Funzionalità che permette all'utente di attribuire al gioco una valutazione personale espressa attraverso l'assegnamento di un voto.

- **Recensione del gioco** (2): Funzionalità che permette all'utente di attribuire al gioco una valutazione personale espressa attraverso un commento, il quale sarà sottoposto ad una revisione prima della sua pubblicazione.

- **Validazione della recensione** (4): Funzionalità possibile solo al moderatore che gli permette di accettare o rifiutare una recensione inserita da un utente.

- **Promozione\Retrocessione** (2): Funzionalità possibile solo al moderatore che gli permette di aumentare o diminuire il livello di un giocatore a suo piacimento.

Requisiti non funzionali

- **Usability**: Il sistema dovrà essere pulito e di facile utilizzo.

- **Reliability**: Il portale dovrà garantire all'utente le funzioni messe a disposizione (vedi Use Case Diagram) senza errori.

- **Availability**: Il portale dovrà essere sempre disponibile e deve poter garantire in qualsiasi momento tutte le funzioni desiderate.

Attori

Il primo attore identificato nel nostro sistema è **l'Amministratore,** il cui compito principale è la gestione della piattaforma gaming. In particolare i suoi compiti possono essere quelli di controllo sull'efficienza del sistema, il quale deve essere pulito e di facile utilizzo.

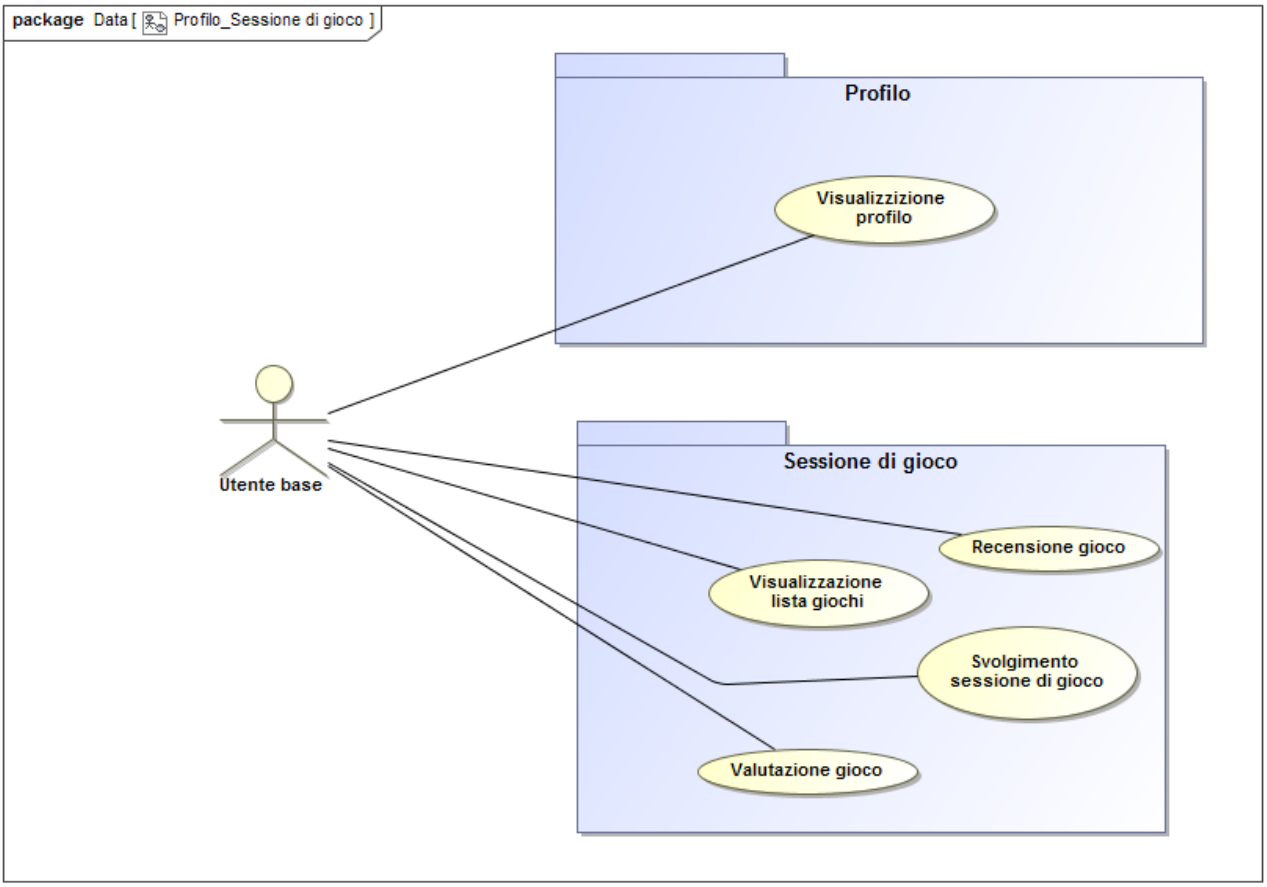
Il secondo attore che incontriamo è il **Moderatore** che si occupa della gestione dell'utenza incluse attività di controllo. Infatti questa figura ha la possibilità di modificare il livello di un giocatore del sistema (aumentandolo/diminuendolo). Inoltre si occupa dell'approvazione delle recensioni relative ai giochi, pubblicate dagli utenti, che non verranno pubblicate fino alla sua decisione. Egli può eventualmente decidere, con dovute motivazioni, di non rendere visibile la recensione andando ad eliminarla (ad es. per violazioni al regolamento del sistema).

Infine troviamo l'attore più importante della nostra piattaforma, cioè **l'Utente base.** Tra le varie funzionalità, egli può scegliere un gioco tra quelli presenti nel sistema, effettuare

sessioni di gaming guadagnando punti esperienza (XP), lasciare votazioni, proporre recensioni, consultare il proprio profilo di gioco con la possibilità di visualizzare balance del punteggio e awards del livello.

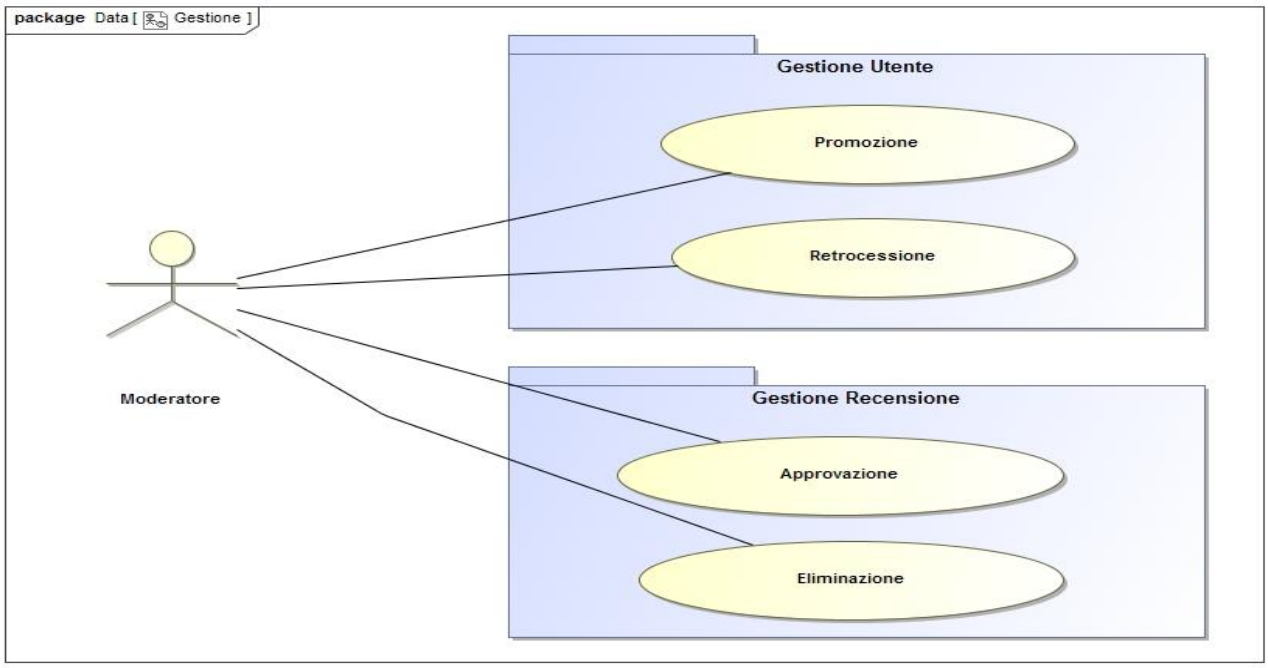
Use Case

USE CASE UTENTE-BASE



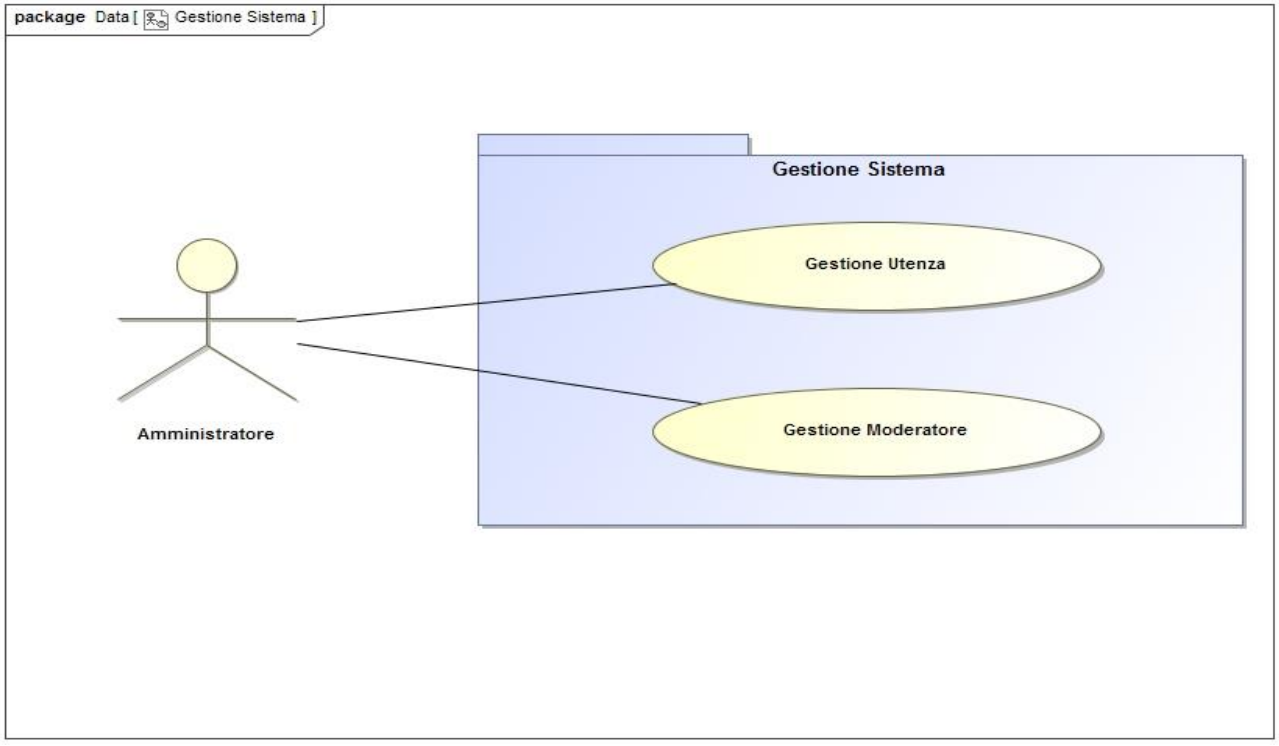
Use Case 1 - Utente base

USE CASE MODERATORE



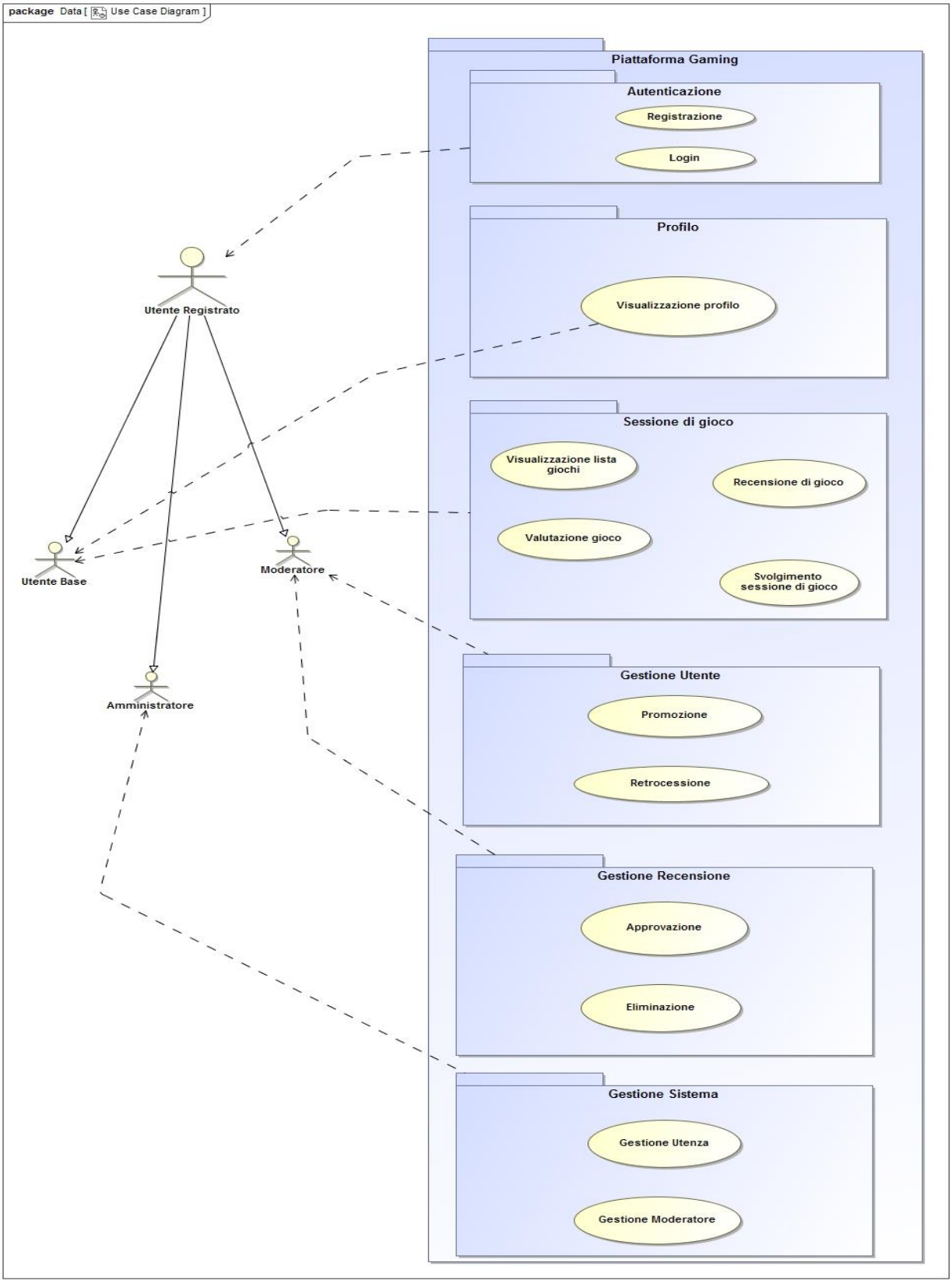
Use Case 2 - Moderatore

USE CASE AMMINISTRATORE



Use Case 3 - Amministratore

USE CASE GENERALE DEL SISTEMA



Use Case 4 - Generale

Scenari:

Autenticazione: L'utente base "Giovanni" e il moderatore "Gigi", effettuano la registrazione una tantum, inserendo i propri dati anagrafici e quelli relativi alla piattaforma gaming (username e password). Questi ultimi dati, permettono al giocatore e moderatore di accedere al proprio profilo e svolgere ciò che è a loro consentito.

Profilo: L'utente base "Giovanni" visualizza il suo profilo all'interno della piattaforma gaming, il quale è suddiviso in due parti: sezione anagrafica e sezione gaming. La prima permette a Giovanni di vedere le proprie informazioni anagrafiche mentre la seconda consente all'utente di interagire con i giochi messi a disposizione dalla piattaforma.

Sessione di gioco: L'utente base "Giovanni" visualizza la lista dei giochi, dove per ciascuno di essi sarà indicato il nome con un eventuale media relativa ai voti che gli utenti hanno precedentemente dato. Per ogni gioco, Giovanni può intraprendere una sessione di gioco, con lo scopo di accumulare punti esperienza. Tale sessione consiste in un bet, rappresentato da un click sulla relativa icona, il quale permetterà all'utente base di accrescere il proprio livello di gioco con l'assegnamento di punti XP. L'utente base può scrivere una recensione relativa alla sue esperienza di gaming (la quale verrà sottoposta all'approvazione o meno da parte del moderatore), e/o esprimere una propria valutazione personale attraverso l'assegnamento di un voto.

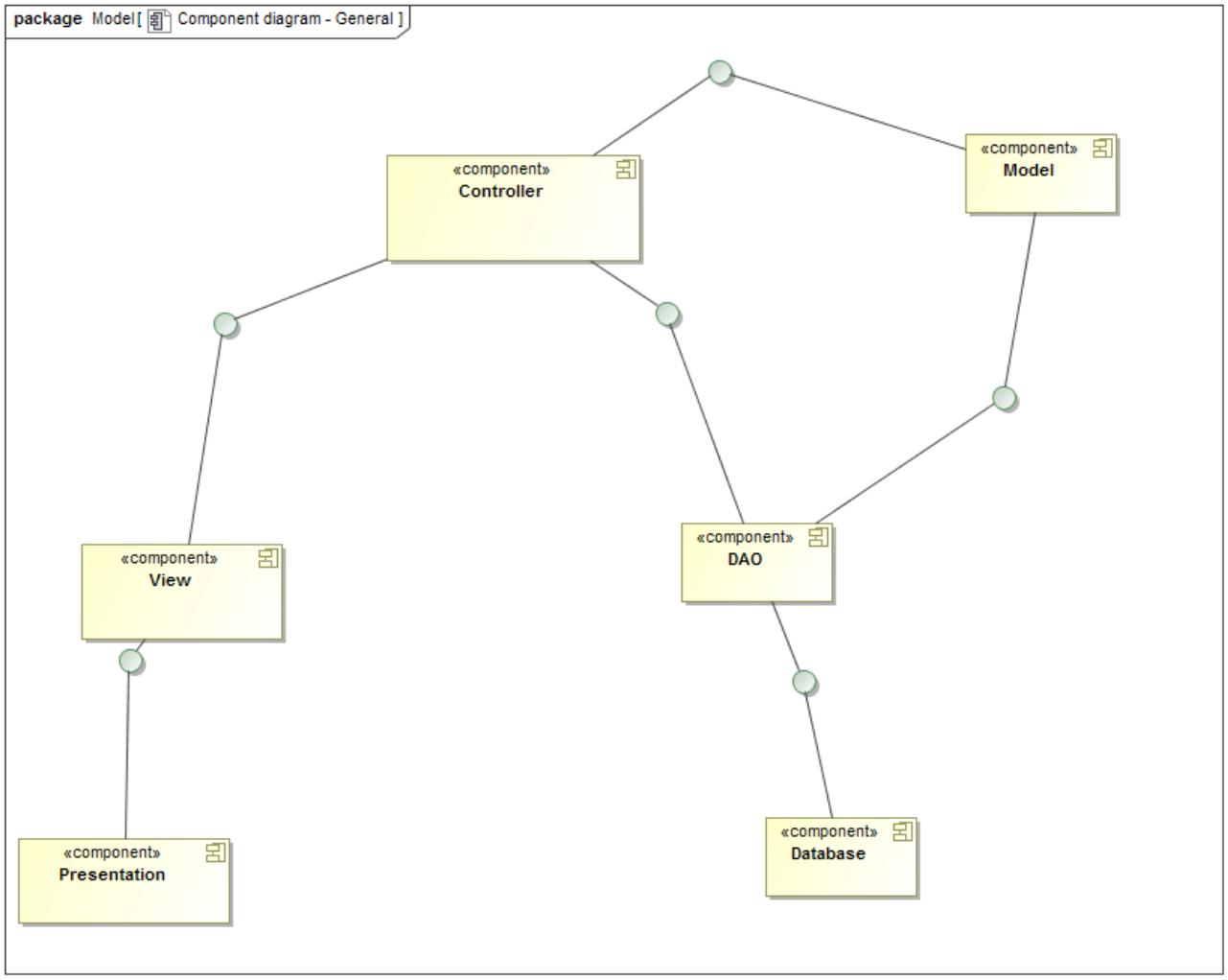
Gestione Utente: Il moderatore "Gigi" può modificare a suo piacimento il livello acquisito dall'utente base Giovanni. In altre parole, può far accrescere l'esperienza gaming del giocatore, oppure far retrocedere e quindi diminuire il relativo livello dell'utente base.

Gestione Recensione: Il moderatore "Gigi" ha il compito di approvare un eventuale recensione scritta dal giocatore Giovanni in merito ad una sua esperienza gaming relativa ad un determinato gioco. In altre parole, una recensione scritta dall'utente base non può essere pubblicata prima dell'approvazione da parte di Gigi.

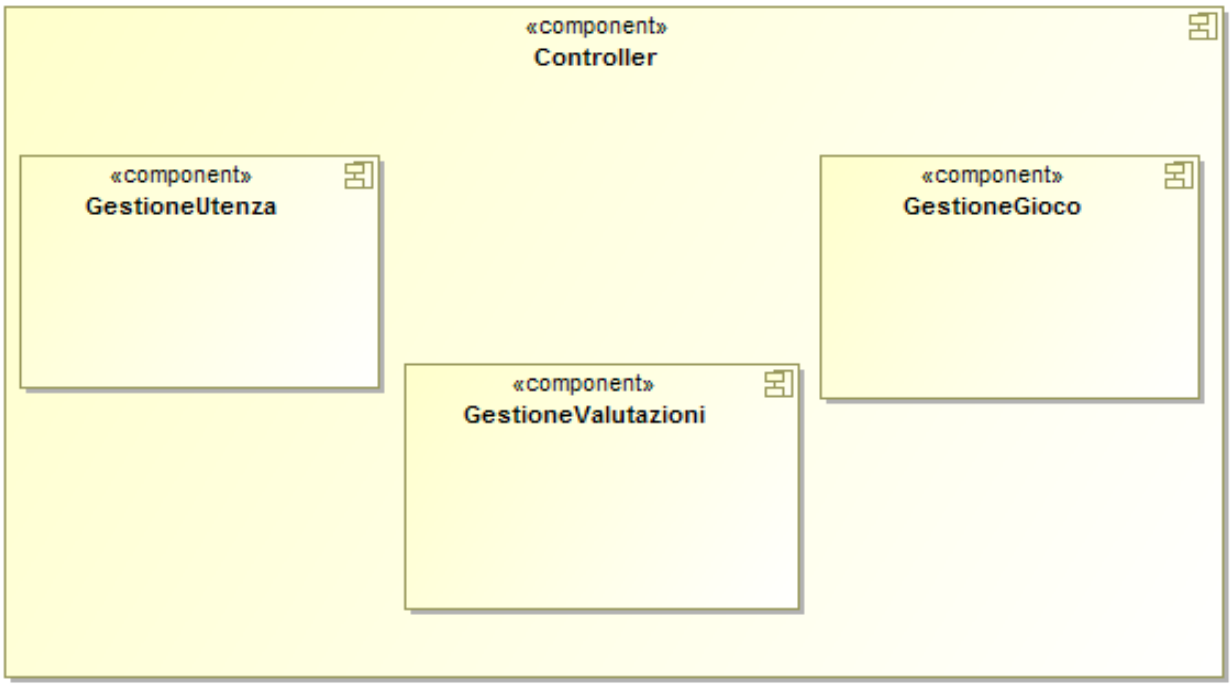
Gestione Sistema: L'amministratore "Dino" ha il compito di garantire il regolare funzionamento della piattaforma Gaming. Quest'ultima deve essere sempre efficiente e di facile utilizzo sia per quanto riguarda le sessioni di gioco da parte di Giovanni, ma anche per le relative azioni relative all'attività del moderator. Un qualsiasi problema, riscontrato dall'utente base o da Gigi, deve essere gestito e risolto da Dino.

System Design

2.1 Modello dell'architettura del sistema



Component diagram 1 - Generale



Component diagram 2 - Controller

2.2 Descrizione dell'architettura

Il component diagram presente nella sezione 2.1 ha lo scopo di elencare i componenti principali del nostro sistema software e tutte le relazioni che intercorrono tra essi.

Partendo dal livello più vicino, fino ad arrrivare a quello dell'utente finale, individuiamo la componente **Presentation** dedicata alla gestione dell'aspetto grafico, svolgendo un compito di intermediazione tra user e sistema.

Un elemento connesso a quest'ultimo componente è quello della **View** il quale costituisce una sorta di "ponte" tra Presentation e la logica vera e propria del sistema, identificata nel **Controller.** Essenzialmente, quest'ultimo, si occupa della gestione di tutte quelle che sono le funzionalità essenziali. Data la sua complessità è stato suddiviso in sotto componenti quali **GestioneUtenza** , **GestioneGioco** e **GestioneValutazioni**. Il primo legato all'elaborazione e al mantenimento delle informazioni relative all'utente finale del sistema;

il secondo tratta tutte le funzionalità relative al gioco ed infine l'ultimo sotto componente del Controller è delegato all'aggiornamento, al mantenimento e alla gestione di tutte le informazioni relative alle valutazioni, sia quelle espresse sotto forma di voto sia quelle sotto forma di commento (recensione).

Troviamo poi il **Model**, che contiene dati strutturati in classi con i relativi metodi di accesso, necessari per la manipolazione da parte del Controller. Le stesse verranno organizzate in un **Database** che ne permette la memorizzazione fisica e la persistenza. Questa funzione viene delegata ad un altro componente del sistema, il **DAO, D**ata **A**ccess **O**bject.

2.3.1 Descrizione delle scelte

Il component diagram realizzato dal team è stato sviluppato secondo i principi classici del paradigma OOP, separando logicamente i vari moduli che costituiscono ciascun componente.

L'interfaccia, con l'utente, rappresentata dalla componente Presentation, è composta da due parti: quella di front-office dedicata a tutti gli utenti base che hanno lo scopo di giocare e di visualizzare il loro profilo sulla piattaforma , e una parte di back-office, utilizzata unicamente dall'amministratore per la gestione del sistema e del relativo database.

2.3.2 Design Patterns

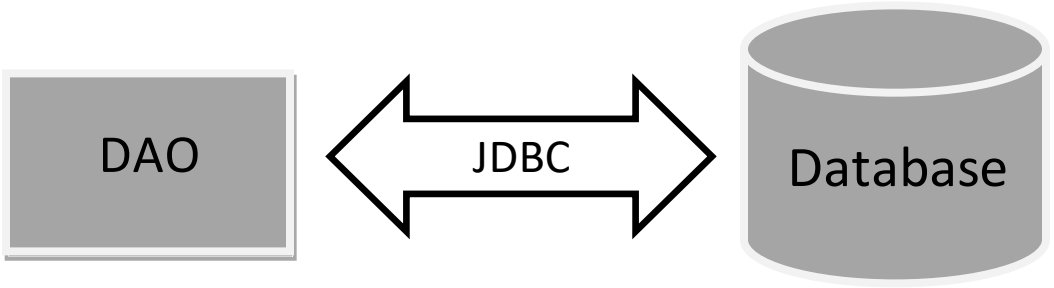
DATA **A**CCESS **O**BJECT

Abbiamo scelto di utilizzare il DAO come design pattern per gestire la persistenza dei dati del nostro sistema su Database. Anzichè far effettuare le interrogazioni dal componente

Model, utilizzando questo pattern, deleghiamo il compito di aggiornamento, di aggiunta e di modifica dei dati del database al DAO.

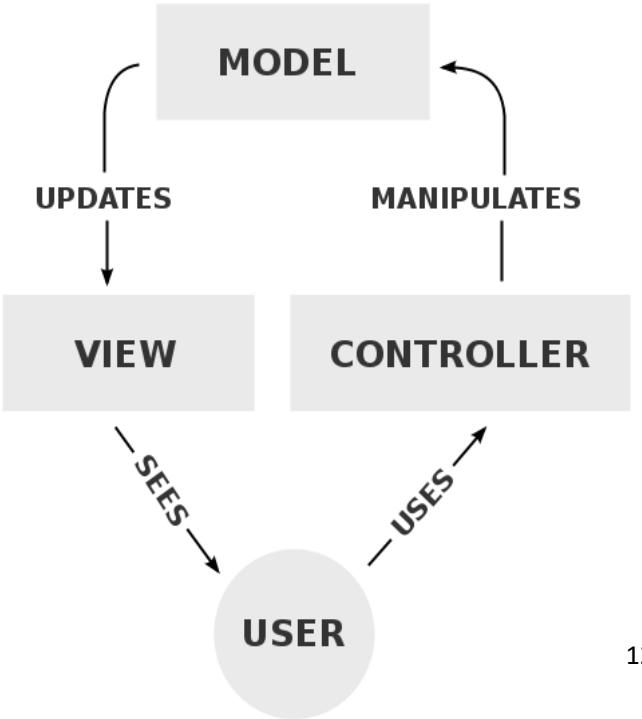
Il pattern si basa su:

- **Data Access Object Interface:** Un'interfaccia che definisce le operazioni standard da eseguire sui modelli dei nostri oggetti.
- **Data Access Object classe concreta:** Una classe che implementa l'interfaccia definita al punto precedente, i cui compiti sono quelli di prendere i dati dalla sorgente (nel nostro caso un database).
- **Model Object o Value Object:** Un semplice oggetto contenente metodi *getters* e *setters* per leggere/memorizzare dati ottenuti attraverso le classi DAO.



MODEL-**V**IEW-**C**ONTROLLER

Il pattern è basato sulla separazione dei compiti tra tre componenti del sistema:



- **Model**: Uno dei componenti più importanti dell'applicazione è il model, che, incapsulando lo stato dell'applicazione, definisce i dati e le operazioni che possono essere eseguite su questi. Nella nostra implementazione in Java, il model rappresenta una classe che contiene informazioni utili al nostro

sistema. Può contenere dei metodi utilizzati per notificare al controller la modifica di alcuni dati.

- **View**: Si occupa di gestire la presentazione dei dati. In altre parole, tutto quello che è correlato alla creazione/gestione della GUI (Graphical User Interface), ovvero lo strumento che gli utenti utilizzano per interagire con il sistema; ciò viene effettuato dalla View in maniera semplice e user-friendly.
- **Controller**: Questo componente ha la responsabilità di trasformare le iterazioni tra utente e View, in azioni eseguite dal Model. Ma il Controller non rappresenta un semplice "ponte" tra View e Model. Realizzando la mappatura tra input dell'utente e processi eseguiti dal Model e selezionando la schermate della View richieste, il Controller implementa la logica di controllo dell'applicazione. Questa logica può infatti risultare molto complessa tanto da poter essere decomposta in ulteriori sotto componenti proprio come nel nostro caso.

Utilizzando questo pattern, e quindi la separazione in tre strati, è possibile mantenere separati i concetti avendo un duplice vantaggio ovvero codice di facile manutenzione e modulare.

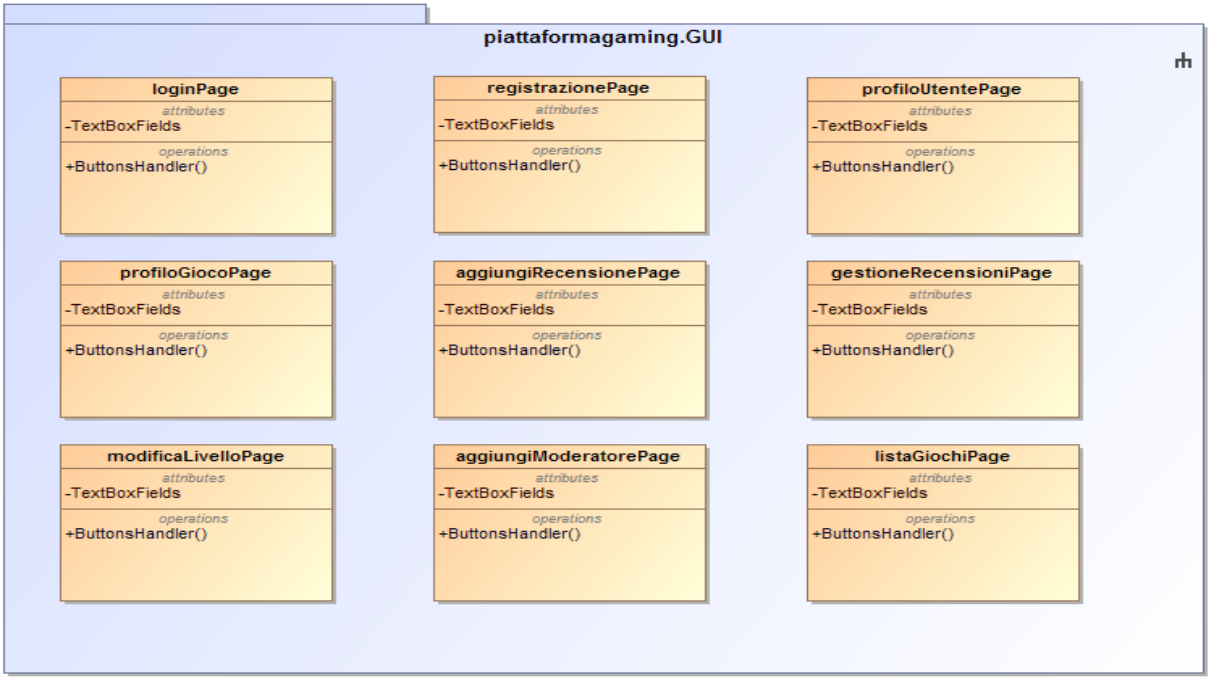
Software Design

La piattaforma gaming è stata separata in diversi package, con lo scopo di mantenere una specifica organizzazione del sistema, ma anche per garantire una separazione logica delle varie classi. Sono stati individuati i seguenti package:

- GUI,
- View
- Controller
- Model
- DAO
- Database

Di seguito sono elencati i class diagram per ciascun package sopra citato, che hanno il compito di descrivere tipi di entità con relative caratteristiche, operazioni ed eventuali correlazioni logiche tra gli stessi.

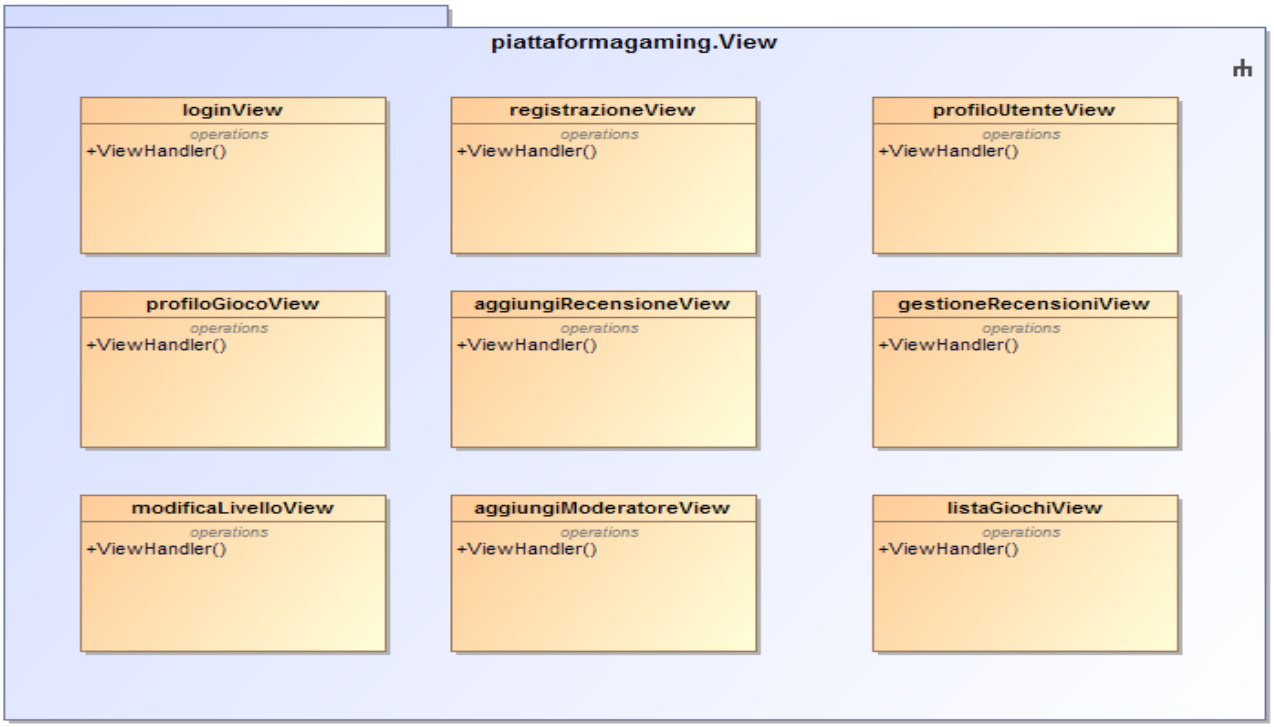
Class diagram – GUI



Class diagram 1 – GUI

Questo package è costituito da tutte le classi utilizzate per la gestione dell'aspetto grafico e appunto della **GUI** (Graphical User Interface). Ognuna di queste classi comprenderà i metodi del framework **Swing** dedicati all'implementazione di un'interfaccia che sia allo stesso tempo funzionale e user-friendly. Ogni classe del pacchetto costituisce una pagina del sistema.

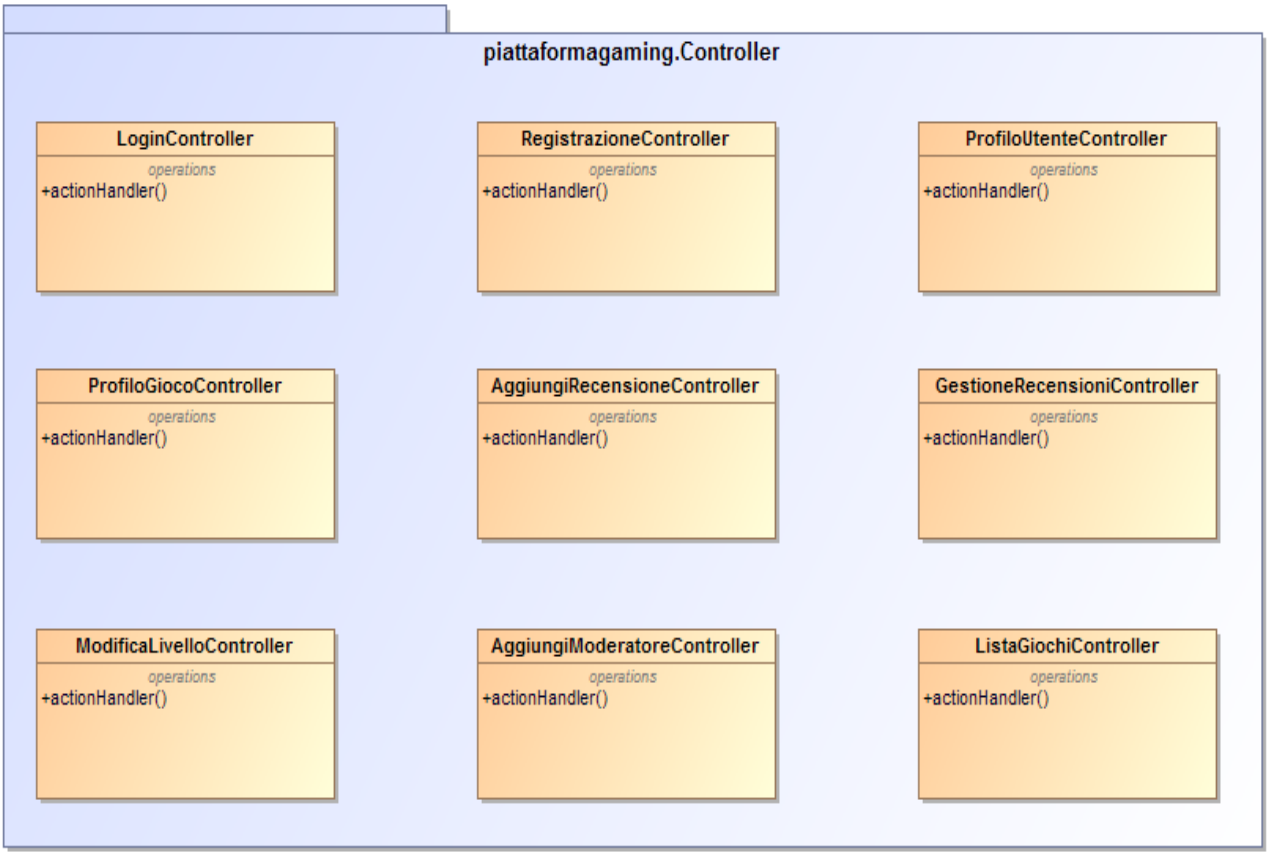
Class diagram – View



Class diagram 2 – View

Il package GUI, relativo alla creazione ed alla gestione dell'interfaccia grafica si avvarrà dell'utilizzo di metodi presenti in un altro package del sistema, quello chiamato **View**. Esso è composto da una serie di classi (una per ogni pagina della GUI) i cui metodi si occupano della gestione dell'input dell'utente. Infatti secondo il pattern MVC, la view ha il compito di notificare al Controller l'avvenuto input utente.

Class diagram - Controller

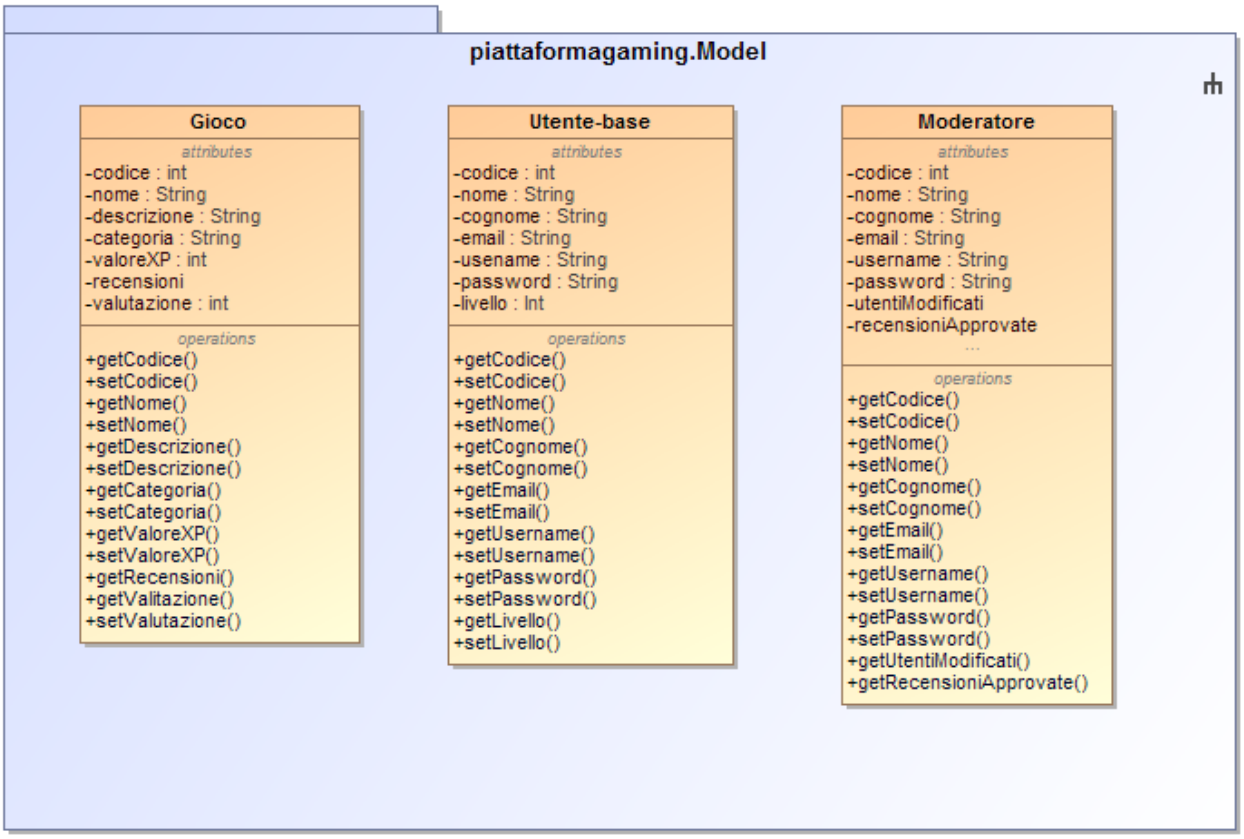


Class diagram 3 – Controller

Così come accennato nel precedente paragrafo, un altro package del sistema è il Controller. Questo costituisce la logica del sistema occupandosi di "rispondere" alle iterazioni dell'utente con la visualizzazione di risultati.

Per ogni classe del package View (e quindi per ognuna di quella della GUI) è presente una rispettiva classe nel controller.

Class diagram – Model



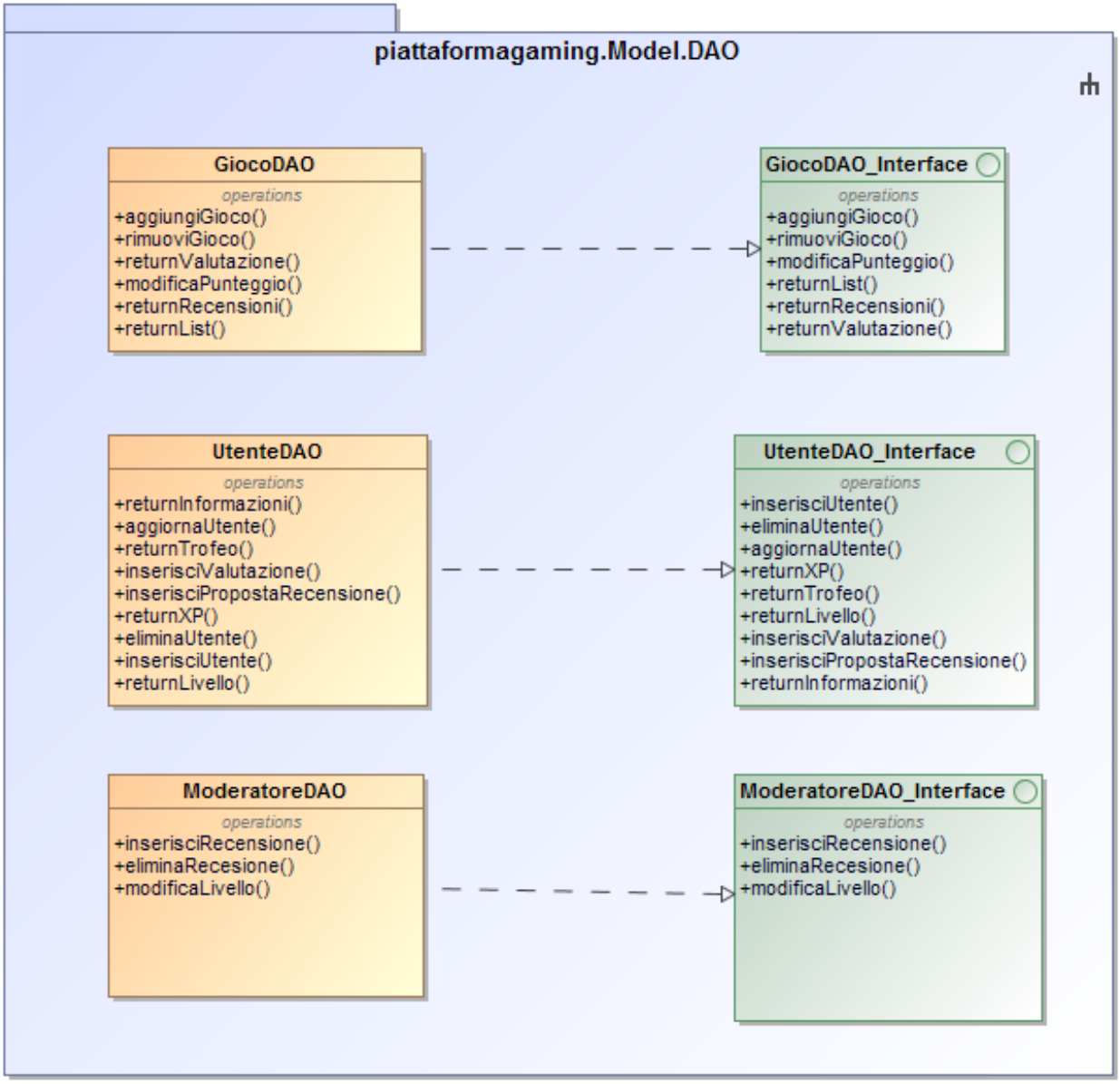
Class diagram 4 - Model

Nel package Model sono presenti le classi relative alle principali entità del sistema:

- **Gioco:** Rappresenta il gioco nel dettaglio, quindi tutte le informazioni che sono di interesse per un particolare gioco e che quindi il nostro sistema deve poter memorizzare. Per ognuno di essi appartenenti alla piattaforma sono stati individuate le caratteristiche più importanti che costituiscono gli attributi della classe Gioco. Per questa classe (come per le altre presenti in questo package) sono presenti metodi **getters** e metodi **setters** per ogni attributo.
- **Utente-Base:** Rappresenta l'utente del sistema, cioè il giocatore della piattaforma. Anche qui per ognuno di essi, sono memorizzate le informazioni di interesse espresse sotto forma di attribuiti.
- **Moderatore:** Rappresenta il moderatore del sistema, cioè l'utente registrato che viene selezionato dall'amministratore ed acquisisce un ruolo privilegiato. Esso infatti può approvare o rifiutare recensioni per un particolare gioco e può modificare il livello di un utente-base sia in positivo che in negativo. Dal class diagram si può

notare che molti attributi sono in comune con la classe Utente-Base e quindi si potrebbe pensare ad un refactoring delle due entità con la creazione di una superclasse. Il team però ha preferito mantenere le due entità separate per non appesantire troppo la gestione del sistema.

Class diagram - Dao



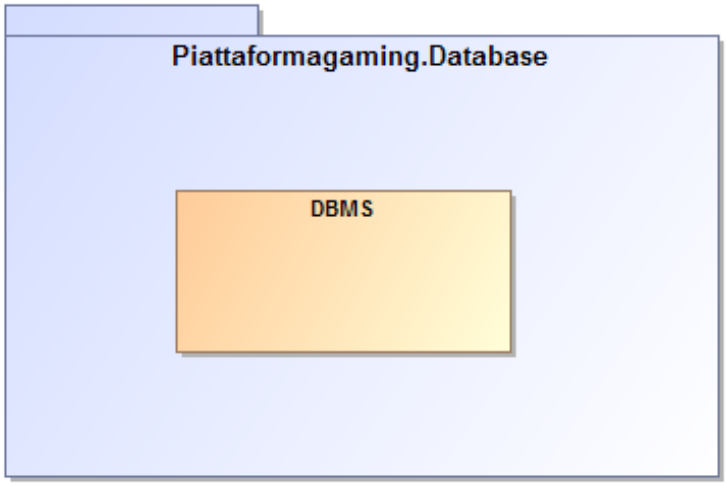
Class diagram 5 – DAO

Rispecchiando il pattern DAO, nel sistema troviamo un package chiamato DAO, dedicato all'iterazione con il Database e quindi alla gestione della persistenza dei dati. Per ogni

concetto principale del sistema è stata realizzata un'interfaccia contenente la dichiarazione dei metodi implementati dalle rispettive classi.

Ogni qualvolta sarà necessario interagire con il database (ad es. quando si dovrà inserire un nuovo utente appena registrato o quando si dovrà inserire una recensione per un particolare gioco) lo si farà tramite l'ausilio di una classe di questo package.

Class diagram – Database



Class diagram 6 - Database

Infine abbiamo il Package Database che contiene al suo interno una sola classe DBMS che si occupa della connessione al Database per accedere ai dati memorizzati fisicamente al suo interno.