Università degli Studi dell'Aquila

Corso di Laurea in Informatica

Object- Oriented Software Design

Piattaforma Gaming

Relatore

Prof.ssa Romina Eramo

Studenti

Andrea Cantagallo

Lorenzo Andreoli

A.A. 2017/2018

INDICE

INTRODUZIONE

Il seguente progetto si propone di realizzare una piattaforma di gaming che coinvolga gli utenti a giocare al fine di guadagnare punti esperienza e nuovi livelli di gioco.
Una piattaforma online di gaming è quindi costituita da utenti, giochi e servizi. Lo scopo di questo progetto è quello di realizzare tale piattaforma in modo da garantire un'esperienza di gioco completa relativa ad un singolo gioco da parte di un utente. Tale esperienza è comprensiva di vari ruoli e di vari servizi relativi al giocatore.

A. REQUISITI FUNZIONALI

1. Accedere al portale attraverso un'autenticazione
2. Avviare una sessione di gioco
3. Visualizzare il profilo utente
4. Votare un gioco
5. Effettuare una sessione di gioco
6. Collezionare punti e trofei
7. Promuovere di livello un utente
8. Retrocedere di livello un utente
9. Nominare un utente come moderatore
10. Rimuovere nomina di moderatore
11. Gestire giochi disponibili

B. REQUISITI NON FUNZIONALI

Sviluppo attraverso il linguaggio di programmazione Java

C. USE CASE

L'attore utente rappresenta una componente basilare del sistema che può:

- Accedere al portale attraverso un'autenticazione
- Avviare una sessione di gioco
- Visualizzare il profilo
- Votare un gioco
- Effettuare una sessione di gioco
- Collezionare punti e trofei

L'attore moderatore potrà effettuare le medesime attività dell'utente base con la peculiarità di:

- Promuovere di livello un utente
- Retrocedere di livello un utente

L'attore Amministratore svolge le attività del moderatore con l'aggiunta di:

- Rimuovere nomina di moderatore
- Nominare un utente come moderatore
- Gestire inserimento giochi

Figura 1 – Use Case

Use Case	Registrazione
Descrizione	Registrazione dell'utente base alla piattaforma
Attori	Utente
Pre-condizioni	-
Flusso	L'utente inserisce: Username, Password, Email. La piattaforma registra le credenziali dell'utente
Post-Condizioni	Username ed Email devono essere univoche
Flusso alternativo	Notifica errata registrazione

Use Case	Login
Descrizione	Accesso alla piattaforma
Attori	Utente, Moderatore, Amministratore
Pre-condizioni	Utente registrato
Flusso	L'utente inserisce le proprie credenziali La piattaforma verifica i dati e consente l'accesso alla home page
Post-Condizioni	Username ed Password corrette
Flusso alternativo	Notifica errato inserimento

Use Case	Gameplay
Descrizione	Inizio sessione di gioco
Attori	Utente, Moderatore, Amministratore
Pre-condizioni	Login
Flusso	L'utente visualizza una lista di giochi. Selezionato il gioco avvia la sessione conquistando punti e trofei
Post-Condizioni	-
Flusso alternativo	Notifica errato caricamento

Use Case	Recensione
Descrizione	Accesso alla piattaforma
Attori	Utente, Moderatore, Amministratore
Pre-condizioni	Login
Flusso	L'utente scrive la recensione in un'apposita area di testo e valuta il gioco attraverso una scala di gradimento da 1 a 5
Post-Condizioni	-
Flusso alternativo	-

Use Case	Anagrafica
Descrizione	Anagrafica utente
Attori	Utente, Moderatore, Amministratore
Pre-condizioni	Login
Flusso	L'utente visualizza la propria anagrafica: Nome e Cognome. Opzionalmente può modificare i propri dati
Post-Condizioni	-
Flusso alternativo	Notifica errato salvataggio

Use Case	Timeline
Descrizione	Exp acquisita dall'utente nel tempo
Attori	Utente, Moderatore, Amministratore
Pre-condizioni	Login
Flusso	L'utente visualizza la data e i corrispondenti punti acquisiti
Post-Condizioni	-
Flusso alternativo	-

Use Case	Trofei
Descrizione	Visualizza trofei conquistati
Attori	Utente, Moderatore, Amministratore
Pre-condizioni	Login
Flusso	L'utente visualizza i trofei acquisiti
Post-Condizioni	-
Flusso alternativo	Notifica errato inserimento

Use Case	Gestione Utenti
Descrizione	Promozione livello
Attori	Moderatore, Amministratore
Pre-condizioni	Login
Flusso	L'attore visualizza una lista di utenti e decide se promuovere o retrocedere di un livello un utente
Post-Condizioni	-
Flusso alternativo	-

Use Case	Gestione Recensioni
Descrizione	Approvazione recensioni
Attori	Moderatore, Amministratore
Pre-condizioni	Login
Flusso	L'attore visualizza la lista delle recensioni in attesa di approvazione e decide se pubblicare o rimuovere la recensione
Post-Condizioni	-
Flusso alternativo	-

Use Case	Gestione Moderatori
Descrizione	Nomina Moderatore
Attori	Amministratore
Pre-condizioni	Login
Flusso	L'amministratore visualizza una lista di utenti e moderatori e decide se promuovere o retrocedere di grado un utente
Post-Condizioni	-
Flusso alternativo	-

Use Case	Gestione Giochi
Descrizione	Inserimento Giochi
Attori	Amministratore
Pre-condizioni	Login
Flusso	L'amministratore può inserire un nuovo gioco oppure rimuoverlo
Post-Condizioni	-
Flusso alternativo	-

D. ASSUNZIONI

Non sono definiti specifici vincoli riguardo usabilità, sicurezza e performance

F. ARCHITETTURA DEL SOFTWARE

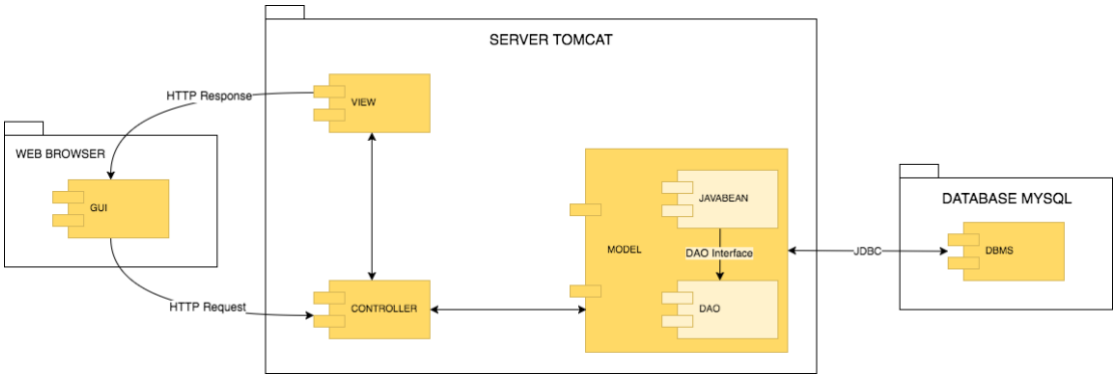

Fig. 2 Component Diagram

Model View Controller

Il pattern architetturale di riferimento è il pattern MVC (Model View Controller), è un pattern architetturale particolarmente indicato per la realizzazione di sistemi che hanno una interfaccia utente.

Permette di suddividere il sistema in 3 macro componenti interconnesse che permettono di rappresentare le funzionalità al loro interno in modo indipendente così da favorire riuso del codice e parallelizzazione dello sviluppo.

1. *Gui*

Componete caratterizzata dal web browser che invia richieste http al Controller e riceve la risposta dalla View.

2. *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), viene effettuato dalla View in maniera semplice e user-friendly. Poiché, la creazione delle interfacce utente ricavata attraverso le servlet è particolarmente onerosa e prona ad errori, la gestione della View viene totalmente delegata alle pagine HTML e/o JSP.

3. *Controller*

Si occupa di gestire il flusso dell'applicazione, sincronizzando il contenuto della View con quello del Model. Tale componente è composta da servlet, la quale interagendo direttamente con il Model è facilmente in grado di aggiornare la View corrispondente.

4. Model

Contiene lo stato dell'applicazione, definisce la logica implementativa dei dati e delle operazioni che possono essere svolte su di essi; in Java, seguendo la politica della programmazione ad oggetti, incapsula le classi che definiscono le parti principali della piattaforma. Manipola i dati attraverso la componente JavaBean e dialoga con il databaase attraveso il DAO.

5. DBMS

Implementato tramite Database relazionale MySQL, rappresenta lo stato di storage dell'applicazione in cui vengono memorizzati i dati.

Design Pattern

Il DAO è un design pattern che fornisce un'interfaccia astratta, mappando le chiamate della business logic sul modello di persistenza dati.
Il vantaggio di usare DAO consiste nel costruire una separazione semplice e rigorosa tra parti del sistema che ci si aspetta evolveranno nel tempo e che possono, ma non dovrebbero "conoscersi" (tutti i dettagli dell'immagazzinamento dati sono nascosti al resto del sistema), è quindi particolarmente adatto a sistemi basati su MVC e in particolare aderenti al paradigma OO.
In poche parole la business logic farà affidamento sempre sui metodi esposti dall'interfaccia DAO, mentre la sua implementazione può cambiare al cambiare del metodo di persistenza adottato.

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 Concrete**: 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).

G. CLASS DIAGRAM

Package Controller

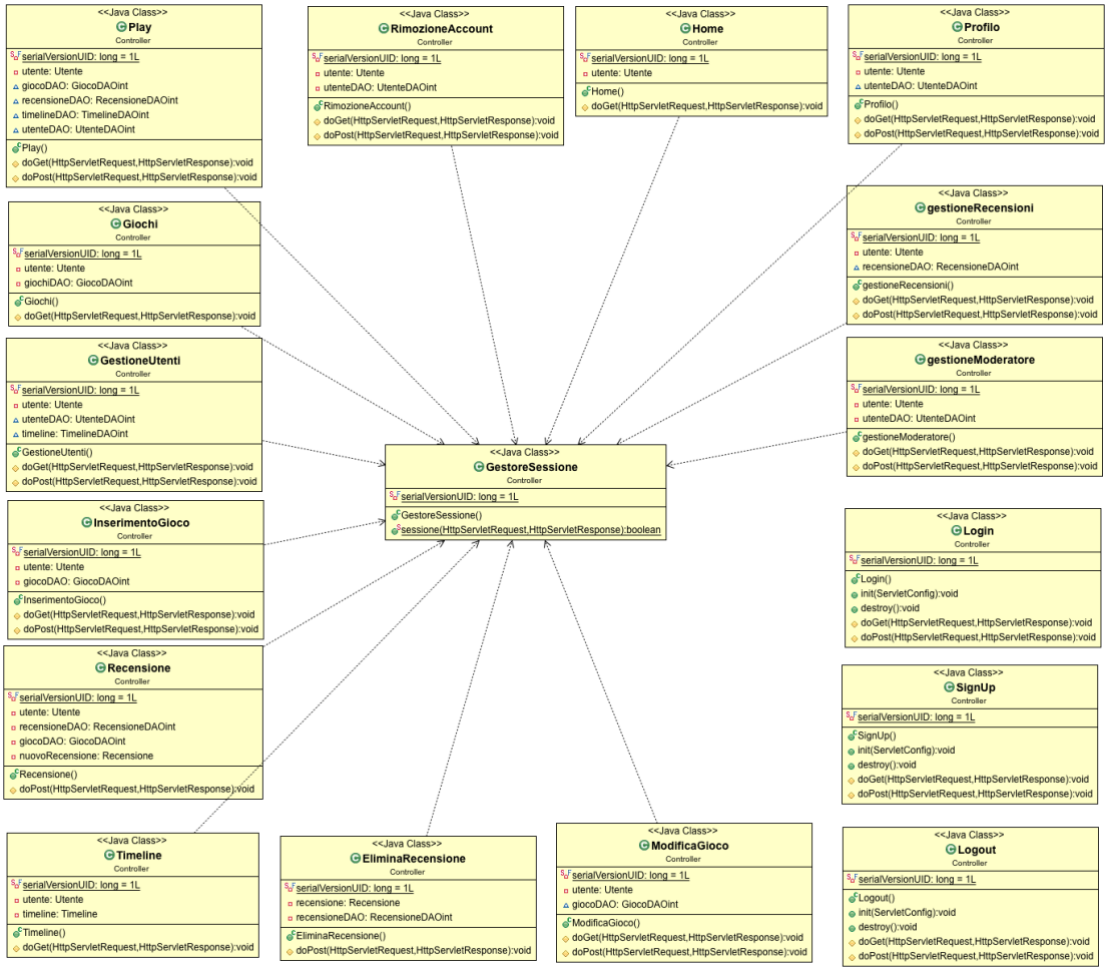

Fig.4 Package Controller

Package Model

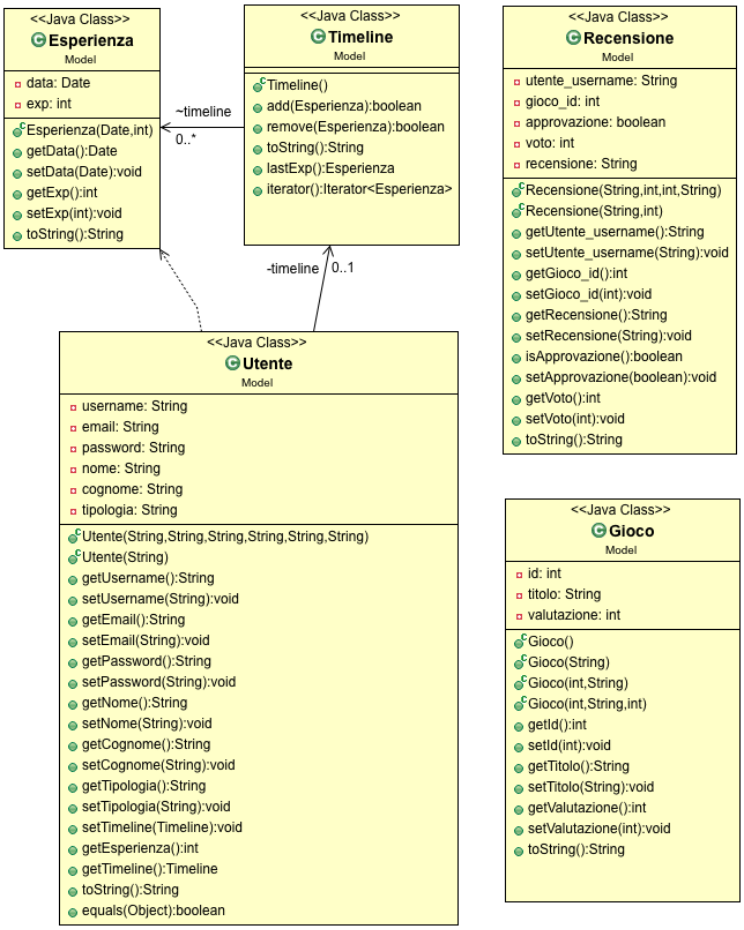

Fig.6 Package Model

Package Model. Concrete

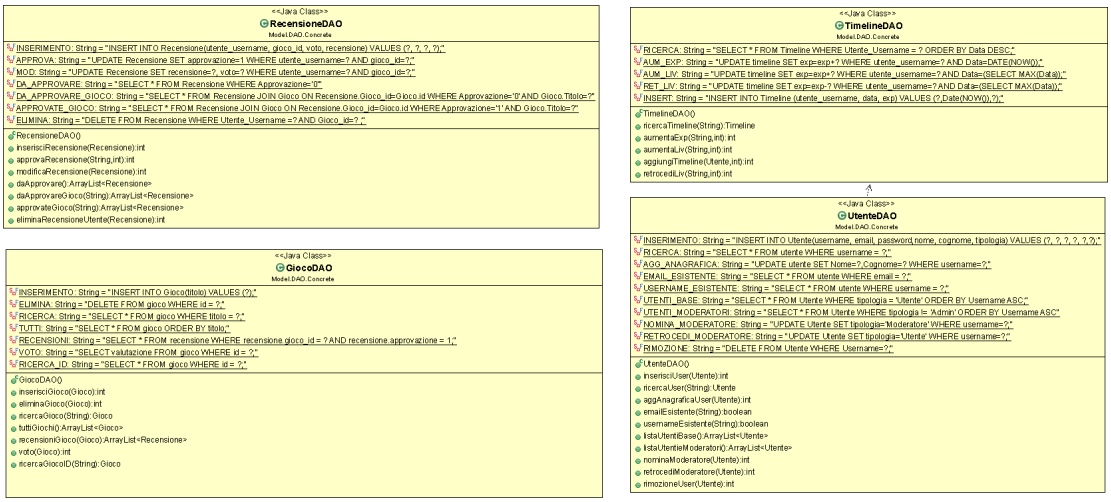

Fig.7 Package Model.Concrete

Package Model.Interface

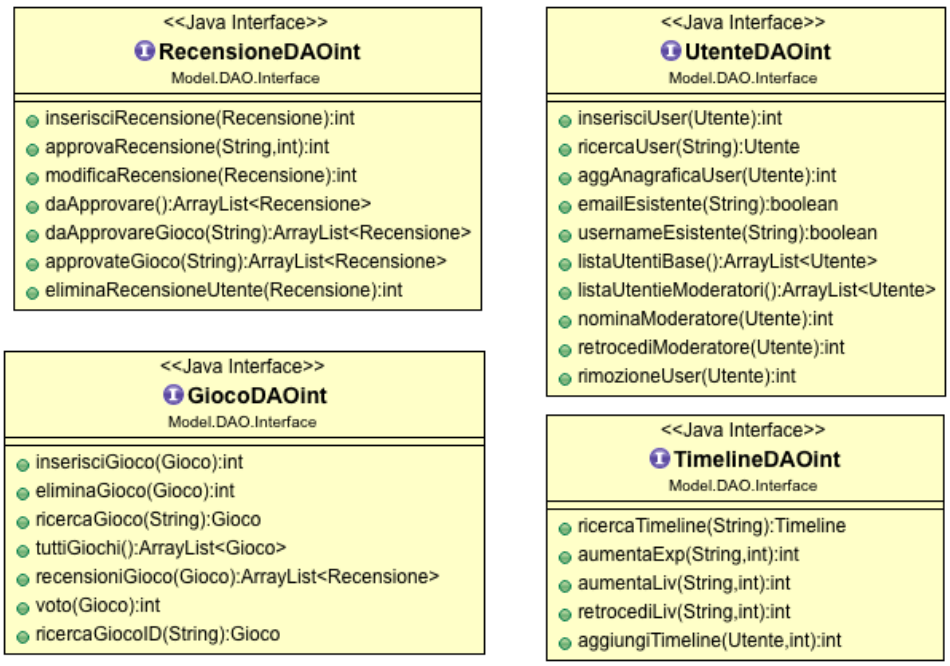

Fig. 8 Model.Interface

Package Model.Database

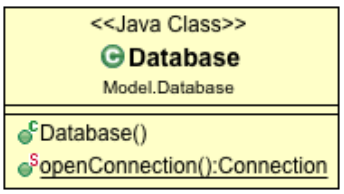

Fig.9 Model.Database

H. ER

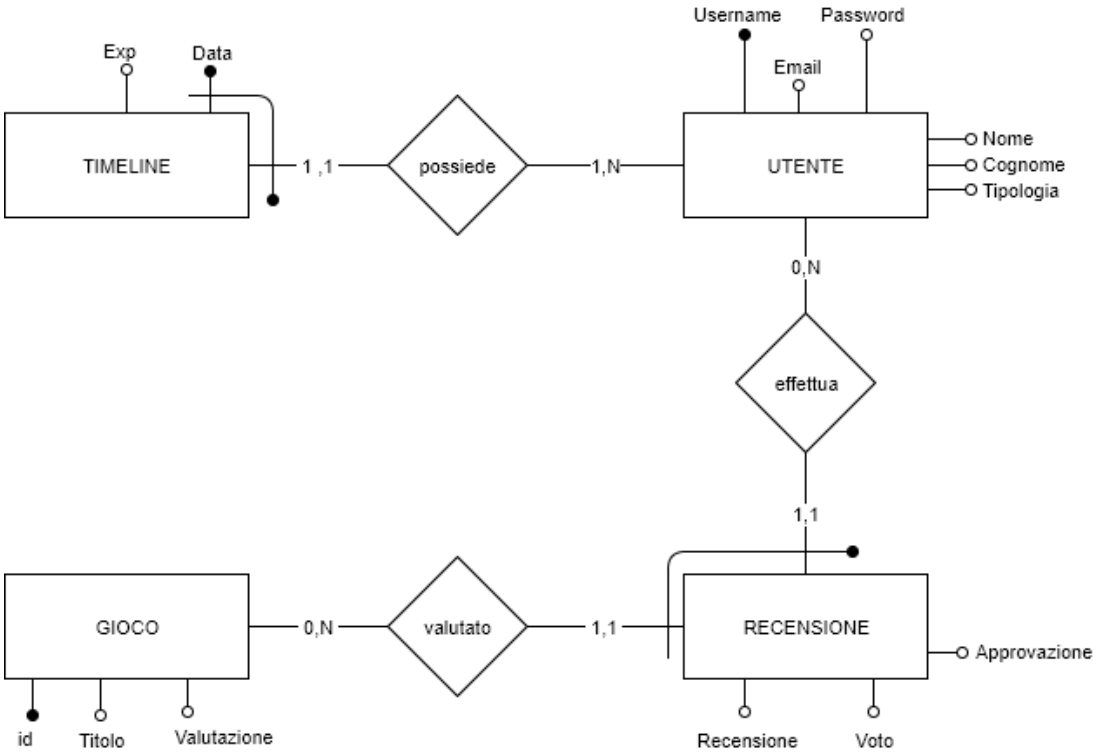

Fig. 10 ER

Entità

Gioco

Id: Id Univoco E Chiave Del Gioco Auto-Incrementato Ad Ogni Inserimento

Titolo: Titolo Univoco Del Gioco

Valutazione: Valutazione Media Del Gioco(Range 0-5)

Recensione

Gioco_id: id del gioco

Utente_username: username dell'utente

Approvazione: indica se un gioco è stato approvato o meno(Range 0-1)

Voto: votazione data al gioco(Range 1-5)

Recensione: testo della recensione

Utente

Username: nominativo univoco e chiave dell'utente

Email: email dell'utente

Password: password dell'utente

Nome: nome dell'utente

Cognome: cognome dell'utente

Tipologia: indica il livello di autorizzazione nel sistema(Range "Utente"-"Moderatore"-"Admin")

Timeline

Utente_username: username dell'utente

Data: data univoca di creazione Timeline

Exp: esperienza accumulata in una Timeline(Range 0-100)

Relazioni:

Valutato

Relazione tra una recensione ed un gioco associato ad essa

Effettua

Relazione tra una recensione ed un utente associato ad essa

Possiede

Relazione tra una timeline ed un utente associato ad essa

Trigger

```
--
-- Trigger `utente`
--
DROP TRIGGER IF EXISTS `Utente_AFTER_INSERT`;
DELIMITER $$
CREATE TRIGGER `Utente_AFTER_INSERT` AFTER INSERT ON `utente` FOR
EACH ROW BEGIN
INSERT INTO Timeline(Utente_Username,Data,Exp)
VALUES(NEW.Username,CURRENT_DATE,'0');
END
$$
DELIMITER ;


--
-- Trigger `recensione`
--
DROP TRIGGER IF EXISTS `Recensione_AFTER_UPDATE`;
DELIMITER $$
CREATE TRIGGER `Recensione_AFTER_UPDATE` AFTER UPDATE ON
`recensione` FOR EACH ROW BEGIN
      IF NEW.Approvazione=1
      THEN
            UPDATE Gioco
        SET Valutazione=((SELECT SUM(Voto) From Recensione WHERE
Gioco_id=NEW.Gioco_id)/(SELECT COUNT(Voto) From Recensione WHERE
Gioco_id=NEW.Gioco_id)) WHERE id=NEW.Gioco_id;
      END IF;
END
$$
DELIMITER ;
```