**“Object Oriented Software Design”**

**Course**

**a.a. 2016-2017**

|  |  |
| --- | --- |
| **Team (Name)** | INDIFFERENTE 2.0 |

|  |  |  |
| --- | --- | --- |
| **Team Members** | | |
| **Name & Surname** | **Matriculation Number** | **E-mail address** |
| Stefano Corsetti | *227288* | *s.corsetti@hotmail.it* |
| Luca D'Orazio | *227635* | *lucaadorazio@gmail.com* |
| Tommaso Di Salle | *236202* | [Tommasodisalle@gmail.com](mailto:Tommasodisalle@gmail.com) |
| Eugenio Mancini | *230024* | *Emancini1992@libero.it* |

A. Documento dei requisiti

***A.0 Scenari***

0)L'utente può registrarsi

1)Login/Logout - l'utente, se registrato, può loggarsi nel sito inserendo le credenziali in una form e disconnettersi

2)Visualizzazione Profilo - L'utente può visualizzare le sue informazioni personali che comprendono anche la collezione punti esperienza e trofei relativi a giochi

3)Un utente può visualizzare la lista di giochi

4) Votazione/Recensione

3.1) L'utente può votare un gioco e/o recensirlo (con un punteggio compreso in un range)

3.2) L'utente può cancellare la propria recensione/votazione

5)Sessione gioco - L'utente può giocare scegliendo un gioco dalla lista

6)Un utente può collezionare trofei relativi ai giochi (al completamento di un livello l’utente riceverà un trofeo che sarà associato ad ogni livello [1,1])

7)Gestione Recensioni - Gestita dai due moderatori e Amministratore

7.1) Approvazione - Accetta la recensione dell'utente

7.2) Eliminazione - Rifiuta la recensione dell'utente

8)Gestione Gioco - Gestita da Amministratore

8.1) Aggiungere - l'amministratore può aggiungere un gioco

8.2) Rimozione - l'amministratore può rimuovere un gioco

9) Modifica Credenziali - Un utente può modificare le sue credenziali

10)Gestione utenti - Gestita da Moderatore1 e Amministratore

10.1) Promozione - utente viene promosso (in termini di permessi)

10.2) Retrocessione - l'utente viene declassificato

10.3) Rimozione - rimuove utente

***A.1 Requisiti Funzionali***

1)**Registrazione** (vedi scenario 0-1)

2)**Visualizzare profilo** (scenario 2)

3) **Visualizzare giochi** (scenario 3)

4)**Votare gioco** (scenario 4)

5)**Recensione gioco** (scenario 4)

6)**Sessione di gioco** (scenario 5)

7)**Collezionare trofei** (scenario 6)

8)**gestione recensioni** (scenario 7)

-approvazione

-eliminazione

9)**gestione gioco** (scenario 8)

-inserimento

-rimozione

11)**Modifica credenziali** (scenario 9)

10)**gestione utente** (scenario 10)

-promozione

-retrocessione

-rimozione

***A.2 Requisiti Non Funzionali***

1)**Sicurezza**

2)**Usability**: Il sistema, in particolare la UI, deve essere facile da utilizzare

3)**Performance**: Il sistema deve permettere all'utente di svolgere le funzioni in modo efficiente e veloce

4)**Maintainability**: Il sistema deve essere facilmente mantenibile, aperto a migliorie e aggiunte di nuove funzionalità

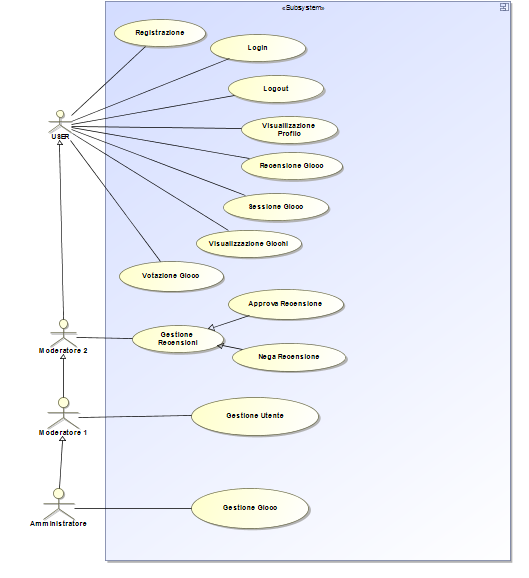
5)**Aviability**: La piattaforma dovrà essere sempre disponibile e garantire in ogni momento tutte le funzioni desiderate dall'utente

***A.5 Priorità***

*Requisiti listati secondo ordine di priorità.*

|  |  |
| --- | --- |
| Sessione Gioco | **HIGH** |
| Login | **HIGH** |
| Recensione | **MEDIUM** |
| Votazione | **MEDIUM** |
| Visualizzazione profilo | **LOW** |

***A.6 Use Case***

******

***A.7 Descrizione - Use Case***

Identificazione attori: Il primo attore individuato nel sistema è l'utente che ha diritto di accesso alle funzionalità di login/registrazione, visualizzazione giochi, visualizzazione profilo, sessione gioco, votazione, recensione.

Il secondo attore identificato è il Moderatore che il team ha deciso di suddividere in due ruoli: Moderatore1 e Moderatore2.

-Il moderatore1 ha accesso alle funzionalità che riguardano la gestione degli utenti.

-Il moderatore2 ha accesso alle funzionalità di gestione delle recensioni, ovvero una recensione prima di essere pubblicata deve essere approvata da questo moderatore.

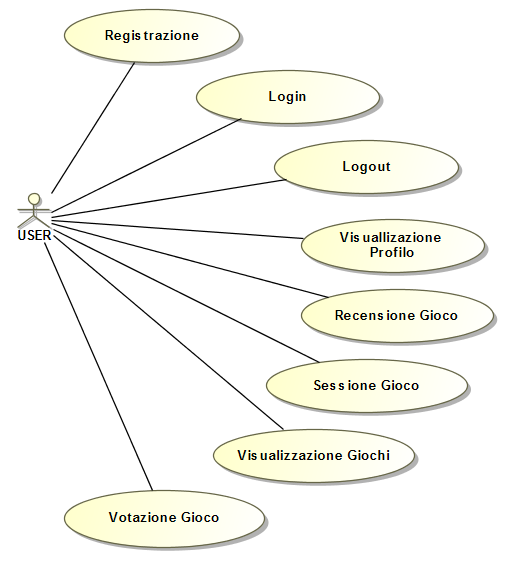
L'ultimo attore identificato è l'Amministratore che ha le funzioni di gestione giochi, ovvero inserimento di un nuovo gioco o eliminare giochi già presenti nella piattaforma

**Naturalmente ci sono le generalizzazioni quindi l'amministratore può usufruire di tutte le funzionalità degli altri attori, poi c'è il moderatore1 che può usufruire delle funzionalità di Moderatore2 e Utente e il Moderatore2 può usufruire delle funzionalità di utente.**

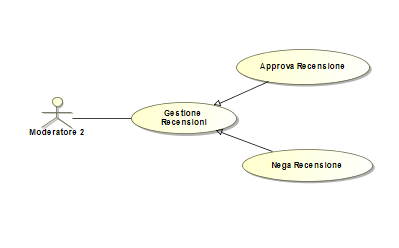
NOTA:

Per problemi di spazio, il team ha deciso di dividere (quindi espandere) lo use-case di ogni attore.

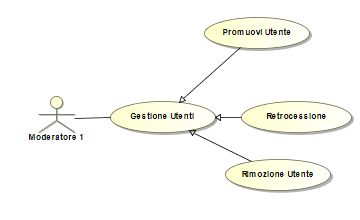
-USER (come in figura precedente):

******

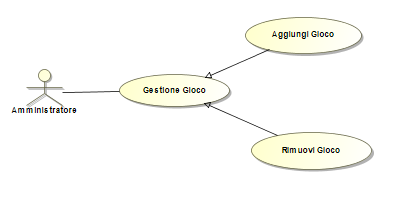
-MOD2(come in figura precedente):

******

-MOD1 (esteso):



-AMMINISTRATORE (esteso):

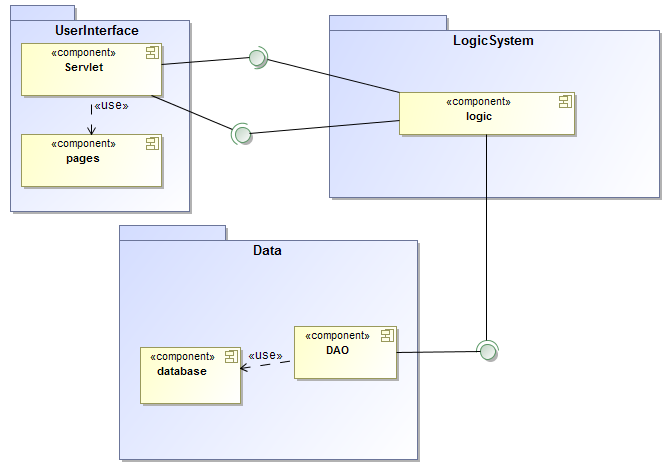


B. System Design

Definire gli obiettivi di design, decomporre il sistema, selezionare le strategie per la costruzione del sistema, definire l’architettura software del sistema, scegliere design patterns.

*Documenti richiesti: (2.1) modello dell’architettura software (o più modelli con diverso grado di dettaglio), (2.2) descrizione dell’architettura, (2.3) descrizione delle scelte e strategie adottate (compresi design patterns)*

***B.1 Component Diagram- Modello architettura software***

******

***B.2 Descrizione Architettura - Component Diagram***

Dopo aver letto la specifica del progetto, il team si è interrogato su quale fosse il pattern da utilizzare in questo sistema.

Si è arrivati alla decisone di utilizzare due design pattern: MVC design pattern e DAO design pattern che andremo a descrivere successivamente.

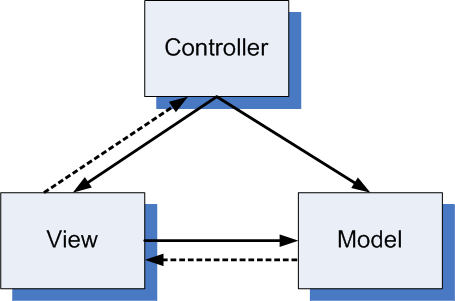
Il team ha deciso di rappresentare le tre MACRO, ovvero UserInterface, Login e Data.

La **UserInterface** rappresenta la **VIEW** del pattern MVC ed è composta da **Servlet,** che rappresenta il mezzo logico con cui l'utente interagirà con il sistema e permetterà il passaggio di dati in input dalla parte front-end alla parte Logic.

**Logic**, invece, rappresenta la parte **CONTROLLER** che riceve come detto i comandi dell'utente attraverso la VIEW ed elabora dati, richieste ed accessi

La parte **MODEL**, infine, identificata nel package **Data** cattura il comportamento del CONTROLLER ed è la parte di memoria del sistema. Sono state identificate due sotto-componenti: **DAO** e **Database.**

DAO fornisce semplicemente i metodi per accedere ai dati utili dell'applicazione, ovvero i metodi per eseguire le operazioni sul database ed incapsulare i dati in oggetti.

******

***B.3 Descrizione scelte e strategie adottate (design pattern)***

MVC

Il team ha deciso di utilizzare il design pattern MVC in quanto questo modello facilita il "ri-uso" del codice. Grazie ad esso abbiamo la cosiddetta separazione delle logiche che, nel nostro caso, è utile in quanto nel progetto non lavora una persona singola ma un team, e la suddivisione delle logiche risulta utile per la divisione del lavoro. Appunto per quanto detto risulterà più facile in un secondo momento modificare (se necessario) una delle logiche, ad esempio, in futuro se volessimo aggiungere nuovi attori non andremo a modificare tutte le logiche.

Più in dettaglio:

**Model**: Si occuperà solo dei dati utili all'applicazione ed a fornire i metodi per accedere ad essi

**View**: Si occuperà solo della gestione delle interfacce utente. Non si occupa della logica applicativa o dell'elaborazione/validazione dell'input. Garantirà che l'interfaccia rifletta lo stato corrente del modello

**Controller**: Si occuperà soltanto di validare/elaborare l'input dell'utente (fornito dalla parte view) e tradurlo in aggiornamenti(messaggi) che verranno passati al modello

DAO

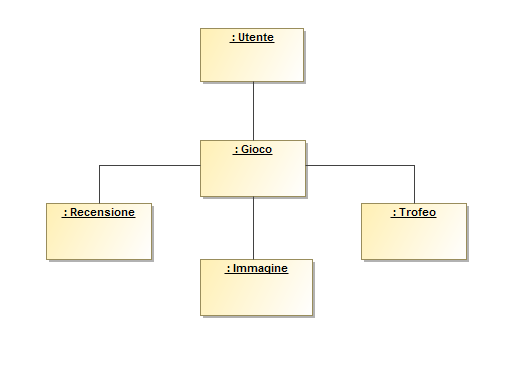
Si è scelto di utilizzare il DAO principalmente per: Separazione logiche e manutenzione.

La manutenzione in quanto se un giorno si volesse modificare il database bisognerà cambiare solamente la parte riguardante il nuovo database senza preoccuparsi delle altre logiche.

C. Software/Object Design

Definire gli oggetti del dominio, definire in modo dettagliato componenti, classi, interfacce e membri. Documenti richiesti: (3.1) modelli rappresentanti l’object design con classi, interfacce e membri, (3.2) descrizione dei dettagli di design scelti.

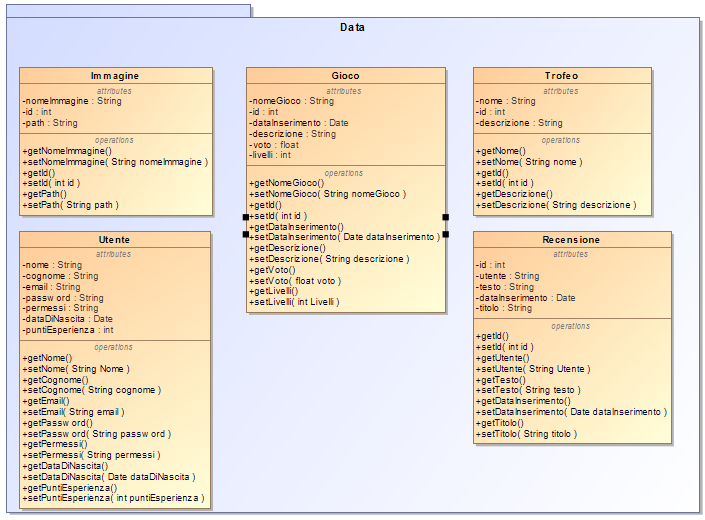
***C.1 Modello – Object Diagram***



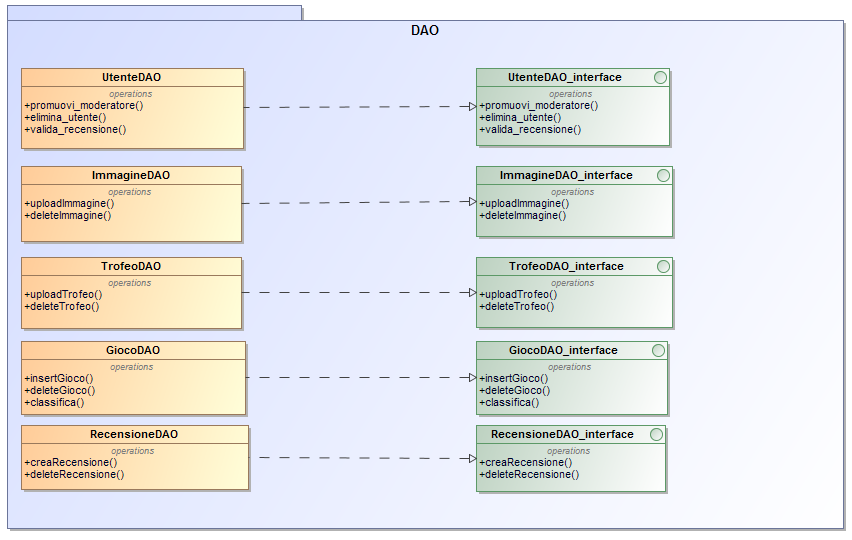
Gli oggetti identificati, che costituiscono la parte del model, sono:

* Immagine: rappresenta l’immagine fisica che potrà essere associata ad un trofeo, oppure una serie di immagini che potranno essere associate ad un gioco. I campi di Immagine sono nomeImmagine, ovvero il nome dell’immagine. Il team ha scelto di attribuire un nome all’immagine, perché nel caso si volesse eliminare un’immagine da un gioco, sarà piu facile eliminarla da parte dell’amministratore. Il campo id, rappresenta l’identificativo dell’immagine, mentre il path indica dove si trova fisicamente l’immagine.
* Gioco: reppresenta il gioco nel dettaglio, ed è caratterizzata dai suoi campi. Ogni gioco avrà un nome ed un id identificativi, la data in cui è stato inserito il gioco sulla piattaforma, la descrizione, dove saranno spiegate le regole, ed il funzionamento del gioco. Il voto, rappresenta la media dei voti dati al gioco dagli utenti. Il campo livelli invece indica i livelli di un gioco.
* Utente: è l’utente vero e proprio che interagirà con il nostro sistema. È caratterizzato dai suoi campi, che sono l’anagrafica ed i campi utili alle funzionalità del sistema.
* Trofeo: è l’entità, che raggiunto un numero di punti, verrà attribuita all’utente
* Recensione: è la recensione che verrà creata da un utente su un gioco. I suoi campi identificano le informazioni sulla recensione.

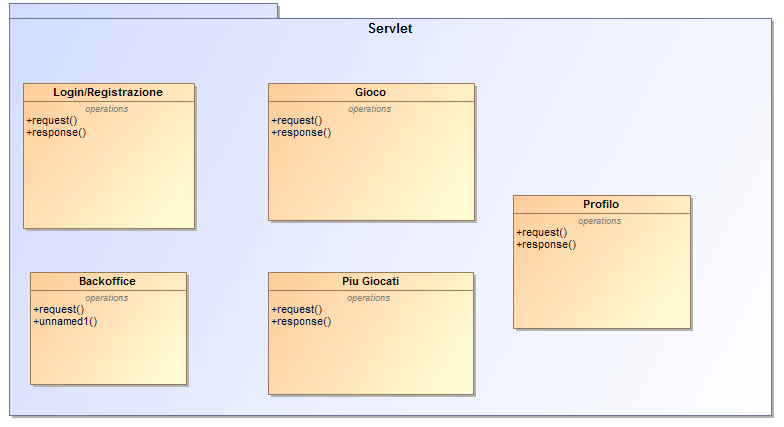
***C.1 Modello – Class Diagram***



Model: rappresenta il modello dei dati di interesse per la nostra applicazione. Rappresenta la prima parte che viene creata. Come già detto i nostri oggetti sono Utente, Immagine,Gioco,Trofeo,Recensione. Sono stati descritti precedentemente nell’object diagram.



Le classi DAO, sono utilizzate per l’iterazione con il database, e le relative interfacce le estenderanno. Nelle interfacce, i metodi vengo solo definiti ma non implementati.



Servlet.

Questo package, contiene la classe in cui sono incapsulate e gestite tutte le iterazioni che l’utente avrà con il nostro sistema. È opportuno per il passaggio di dati in input dall’utente, alla nostra LogicSystem, che poi elaborerà i dati, e dopo averli manipolati attraverso le operazioni DAO, restituirà l’output.

***C.3 Descrizione dettagli design scelti***

Come già detto in precedenza, il team ha deciso di utilizzare due patterns: MVC design pattern e DAO design pattern.

Per quanto riguarda le librerie, il team utilizzerà:

Apache Tomcat: ovvero l’application server su cui il nostro sistema girerà in locale.

Mysql Connector: sarà la libreria utilizzata per comunicare con il nostro database.

Altre librerie saranno utilizzate dal team. Per ora il team sta ancora discutendo se utilizzare il java tamplate engine “FreeMarker” oppure optare per l’utilizzo delle JSP. Anche a causa di queste decisioni, e di altre su cui il team sta discutendo, il class diagram in particolare, ma come gli altri diagram, risultano parzialmente incompleti.