Project Proposal

Team Page 404

Prepared for: Romina Eramo

Prepared by: Team “Page 404”

21/04/2017

Membri del team

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome | Cognome | e-mail | Year | Matricola |
| Andrea | Perelli | andrea.perelli@student.univaq.it | 3° | 236059 |
| Davide | De Marco | davide.demarco@student.univaq.it | 3° | 228737 |
| Francesco | Fuschi | francesco.fuschi@student.univaq.it | 3° | 239266 |
| Mattia | Caputo | mattia.caputo1@student.univaq.it | 2° | 243467 |

Esami effettuati

Nessuno ha effettuato l’esame del professore Della Penna.

Solo Perelli Andrea ha sostenuto l’esame di Tecnologie del Web.

# INDICE

[A. REQUIREMENTS 3](#_Toc488751632)

[A.1 Functional Requirements 3](#_Toc488751633)

[A.1.1 GUI Requirements 3](#_Toc488751634)

[A.1.2 DB Requirements 3](#_Toc488751635)

[A.1.3 Business Logic Requirements 3](#_Toc488751636)

[A.2 Non-Functional Requirements 4](#_Toc488751637)

[A.2.1 Efficiency NFR 4](#_Toc488751638)

[A.2.2 Security 4](#_Toc488751639)

[A.2.3 Usability 4](#_Toc488751640)

[A.2.4 Technology Requirements 4](#_Toc488751641)

[A.3 Use Cases 5](#_Toc488751642)

[A.4 Scenari 18](#_Toc488751643)

[B. System Design 19](#_Toc488751644)

[B.1 Component Diagram 19](#_Toc488751645)

[B.2 Design Decision 21](#_Toc488751646)

[B.2.1 Architectural design 21](#_Toc488751647)

[B.2.2 Software Design 22](#_Toc488751648)

[B.3 Design Pattern 24](#_Toc488751649)

[B.4 Pattern Architetturale 25](#_Toc488751650)

[B.5 E-R 27](#_Toc488751651)

[C. SOFTWARE/ OBJECT Design 29](#_Toc488751652)

[C.1 Class Diagram 29](#_Toc488751653)

[C.2 Sitemap 39](#_Toc488751654)

# A. REQUIREMENTS

## A.1 Functional Requirements

### A.1.1 GUI Requirements

1. Sarà la GUI che permetterà l’interazione tra l’utente e le funzionalità del sistema (registrazione, autenticazione, visualizzazione del profilo, effettuare una sessione di gioco, recensirlo…).
2. Attraverso la GUI l’utente potrà visualizzare e modificare le proprie informazioni anagrafiche quali: nome, cognome, età, email, password e immagine del profilo.
3. La GUI mostrerà i progressi (livello, trofei e punti esperienza) dell’utente nei vari giochi. Sarà inoltre possibile visualizzare la timeline del livello generale conquistato nel gioco.
4. Attraverso la GUI il moderatore e l’amministratore potranno accedere al rispettivo back office.

### A.1.2 DB Requirements

1. Il DB dovrà permettere di accedere ai propri dati personali.
2. Il DB dovrà essere strutturato in maniera tale da permettere il corretto salvataggio dei progressi di gioco.
3. Il DB dovrà permettere il corretto funzionamento del back office.

### A.1.3 Business Logic Requirements

1. Il Sistema potrà gestire una molteplicità di giochi.
2. Il Sistema dovrà gestire diversi tipi di utenza:

Amministratore

Moderatore

Utente base

1. Il Sistema prevedrà l’assegnazione di un trofeo, ad un utente, per ogni livello acquisito.
2. Il Sistema dovrà prevedere un range prefissato per l’assegnazione del voto relativo ad un gioco.
3. Il Sistema dovrà assicurare che il moderatore possa verificare le recensioni prima della loro pubblicazione.
4. Il Sistema prevedrà l’assegnazione di punti esperienza relativi al completamento di una sessione di gioco. Tale attributo servirà all’utente per conseguire i nuovi livelli e arricchire il proprio profilo.
5. Il Sistema prevedrà due tipi di livelli: un livello relativo al progresso generale dell’utente (dato dalla somma dei punti esperienza dei vari giochi) e un livello relativo al singolo gioco.
6. Il Sistema prevedrà una gestione del back office il quale permetterà di gestire i vari utenti e le loro recensioni. L’amministratore sarà in grado di gestire il sistema, aggiungendo, eliminando e/o modificando i vari servizi del sistema (e.g. aggiungere un nuovo gioco, un nuovo trofeo). L’amministratore inoltre sarà in grado di promuovere, retrocedere e “bannare” un utente e un moderatore.

## A.2 Non-Functional Requirements

### A.2.1 Efficiency NFR

1. Il committente non ha posto alcun vincolo riguardo lo spazio che dovrà occupare il nostro sistema e alle sue performance.

### A.2.2 Security

1. Il committente non ha posto alcun vincolo riguardo la sicurezza del sistema, pertanto il nostro team non prevedrà l’implementazione di particolari controlli di sicurezza.

### A.2.3 Usability

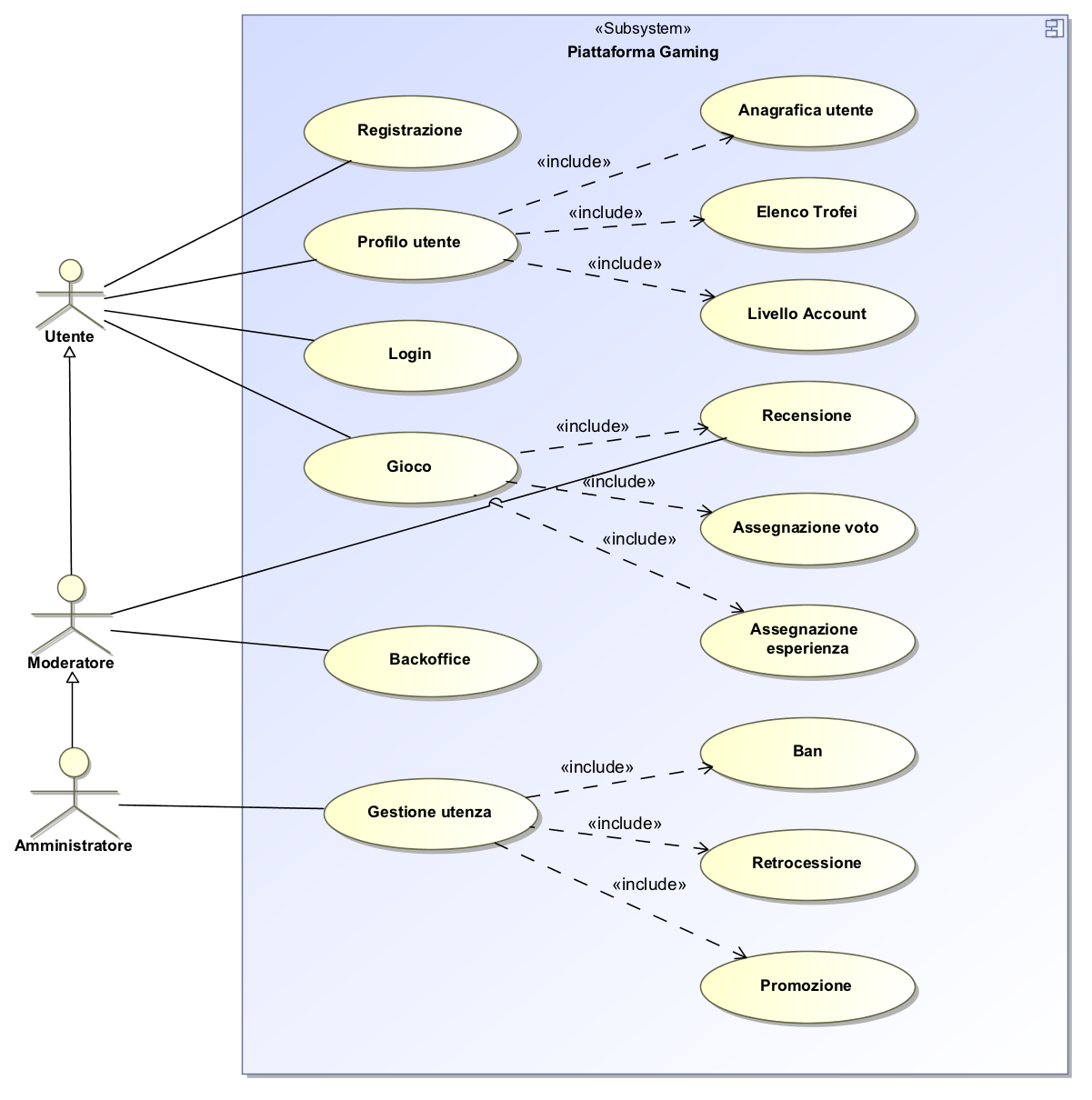
1. La GUI dovrà essere user-friendly al fine di facilitarne e invogliarne l’utilizzo da parte dell’utente.

### A.2.4 Technology Requirements

1. Il committente non ha posto alcun tipo di vincolo tranne che per il linguaggio di programmazione, che dovrà essere “Java”, e l’utilizzo di almeno un design pattern.

## A.3 Use Cases

Di seguito viene riportato il diagramma:



|  |  |
| --- | --- |
| USE CASES# | FUNCTIONAL REQUIREMENTS |
| Registrazione | A.1.1a, A.1.2a |
| Profilo Utente | A.1.1a, A.1.1b, A1.2a |
| Login | A.1.1a, A.1.2a |
| Gioco | A.1.1a, A.1.2b |
| BackOffice | A.1.1d, A.1.2a, A.1.2c, A.1.3h |
| Gestione utenza | A.1.1d, A.1.2a, A.1.3.a, A.1.3h |
| Anagrafica Utente | A.1.1a, A.1.2a |
| Elenco trofei | A.1.1c, A.1.2b, A.1.3c, A.1.3h |
| Livello Account | A.1.1c, A.1.2b, A.1.3f, A.1.3g |
| Recensione | A.1.1a, A.1.3e, A.1.3h |
| Assegnazione voto | A.1.1a, A.1.3c, A.1.3d |
| Assegnazione Esperienza | A.1.1c, A.1.2b, A.1.3d |
| Ban | A1.3h |
| Retrocessione | A1.3h |
| Promozione | A1.3h |

|  |  |  |
| --- | --- | --- |
| USE CASE # | Registrazione | |
| Goal in Context | Permettere all’utente di registrarsi | |
| Scope & Level | Sistema Piattaforma Gaming.  Level: Primary Task | |
| Preconditions | L’utente si trova in una pagina del sito e l’utente non è autenticato. | |
| Success End Condition | I dati dell’utente vengono memorizzati nel database e l’utente viene autenticato e può usufruire delle funzionalità del Sistema. | |
| Failed End Condition | L’utente non viene registrato, riceve un messaggio di errore e viene invitato a correggere i campi scorretti | |
| Primary,  Secondary Actors | Utente  Moderatore, Amministratore | |
| Trigger | L’utente, in una pagina del sito, clicca su “registrazione” | |
| DESCRIPTION | Step | Action |
|  | 1 | L’utente, in una pagina del sito, clicca su “registrazione” |
|  | 2 | Compila la form di registrazione |
|  | 3 | Clicca su “Registrati” |

|  |  |
| --- | --- |
| RELATED INFORMATION | Registrazione |
| Priority: | Alta |
| Performance | Non espressamente dichiarate |
| Frequency | Non espressamente dichiarata |
| Channels to actors | Database, GUI |
| OPEN ISSUES | L’utente non può usufruire degli altri servizi del sistema |
| Subordinates | Nessuno |

|  |  |  |
| --- | --- | --- |
| USE CASE # | Profilo utente | |
| Goal in Context | Permettere ad un utente registrato di visualizzare il suo profilo | |
| Scope & Level | Sistema Piattaforma Gaming.  Livello: Primary task | |
| Preconditions | L’utente risulta essere autenticato | |
| Success End Condition | L’utente può visualizzare il suo profilo | |
| Failed End Condition | L’utente visualizza un messaggio di errore | |
| Primary,  Secondary Actors | Utente base  Moderatore, Amministratore | |
| Trigger | L’utente, in una pagina del sito, clicca su “Gestione profilo” | |
| DESCRIPTION | Step | Action |
|  | 1 | L’utente, in una pagina del sito, clicca su “Profilo” |
|  | 2 | Visualizza il suo profilo |
|  | 3 | Ha la possibilità di modificare alcuni campi |
| EXTENSIONS | Step | Branching Action |
|  | 2a | Informazioni sull’anagrafe dell’utente  Use case “Anagrafica utente” |
|  | 2b | Visualizzazione dell’elenco dei trofei finora ottenuti.  Use Case “Elenco trofei” |
|  | 2c | Visualizzazione del livello corrente del profilo, dei diversi giochi e dei punti esperienza.  Use Case “Livello account” |

|  |  |
| --- | --- |
| RELATED INFORMATION | Profilo utente |
| Priority: | Alta |
| Performance | Non espressamente dichiarata |
| Frequency | Non espressamente dichiarata |
| Channels to actors | Database, GUI |
| OPEN ISSUES | L’utente non può visualizzare il suo profilo |
| Subordinates | Anagrafica Utente, Elenco Trofei, Livello Account |

|  |  |  |
| --- | --- | --- |
| USE CASE # | Login | |
| Goal in Context | Permettere ad un utente già registrato di autenticarsi | |
| Scope & Level | Sistema di autenticazione  Level: Primary task | |
| Preconditions | L’utente si trova in una pagina del sito, non si è ancora autenticato ma è già registrato | |
| Success End Condition | L’utente viene autenticato e può accedere a nuove funzionalità del sistema | |
| Failed End Condition | L’utente visualizza un messaggio di errore dovuto o alla non correttezza dei dati o all’assenza dei dati inseriti nel database | |
| Primary,  Secondary Actors | Utente  Moderatore, Amministratore | |
| Trigger | L’utente, in una pagina del sito, clicca su “Login” | |
| DESCRIPTION | Step | Action |
|  | 1 | L’utente, in una pagina del sito, clicca su “Login” |
|  | 2 | Compila il form di autenticazione |
|  | 3 | Conferma i propri dati cliccando su “Login” |

|  |  |
| --- | --- |
| RELATED INFORMATION | Login |
| Priority: | Alta |
| Performance | Non espressamente dichiarate |
| Frequency | Non espressamente dichiarata |
| Channels to actors | Database, GUI |
| OPEN ISSUES | L’utente non può usufruire degli altri servizi del sistema |
| Subordinates | Nessuno |

|  |  |  |
| --- | --- | --- |
| USE CASE # | Gioco | |
| Goal in Context | Permette all’utente di effettuare una sessione di gioco tra quelli proposti | |
| Scope & Level | Sistema Piattaforma Gaming.  Level: Primary task | |
| Preconditions | L’utente si trova in una pagina del sito e si è autenticato | |
| Success End Condition | L’utente può avviare la sua sessione di gioco | |
| Failed End Condition | L’utente visualizza un messaggio di errore | |
| Primary,  Secondary Actors | Utente  Moderatore, Amministratore | |
| Trigger | Click su uno dei giochi proposti | |
| DESCRIPTION | Step | Action |
|  | 1 | L’utente clicca sul gioco desiderato |
|  | 2 | L’utente effettua la sua sessione di gioco |
| EXTENSIONS | Step | Branching Action |
|  | 0.a | Possibilità di recensire il gioco.  Use case “Recensione” |
|  | 0.b | Possibilità di assegnare un voto al gioco con range prefissato.  Use case “Assegnazione voto” |
|  | 2a | Viene assegnato un certo valore di punti esperienza all’utente in base alla sessione di gioco.  Use case “Assegnazione esperienza” |

|  |  |
| --- | --- |
| RELATED INFORMATION | Gioco |
| Priority: | Alta |
| Performance | Non espressamente dichiarate |
| Frequency | Non espressamente dichiarata |
| Channels to actors | GUI, Sistema esterno |
| OPEN ISSUES | L’utente non può avviare la sessione di gioco |
| Subordinates | Recensione, Assegnazione voto, Assegnazione esperienza |

|  |  |  |
| --- | --- | --- |
| USE CASE # | Gestione utenza | |
| Goal in Context | Permettere la gestione degli utenti e moderatori dando o togliendo loro privilegi. | |
| Scope & Level | Sistema di back office  Level: Primary Task | |
| Preconditions | L’amministratore del Sistema si è autenticato | |
| Success End Condition | L’amministratore può gestire i permessi | |
| Failed End Condition | L’amministratore visualizza un messaggio di errore | |
| Primary | Amministratore | |
| Trigger | L’amministratore effettua il login con i propri dati di accesso | |
| DESCRIPTION | Step | Action |
|  | 1 | L’amministratore effettua il login |
|  | 2 | L’amministratore effettua la gestione dei permessi |
| EXTENSIONS | Step | Branching Action |
|  | 2a | Possibilità di bannare determinati moderatori e/o utenti.  Use case “Ban” |
|  | 2b | Possibilità di revocare permessi a determinati moderatori.  Use case “Retrocessione” |
|  | 2c | Possibilità di promuovere un utente e/o un moderatore a moderatore e/o amministratore.  Use case “Promozione” |

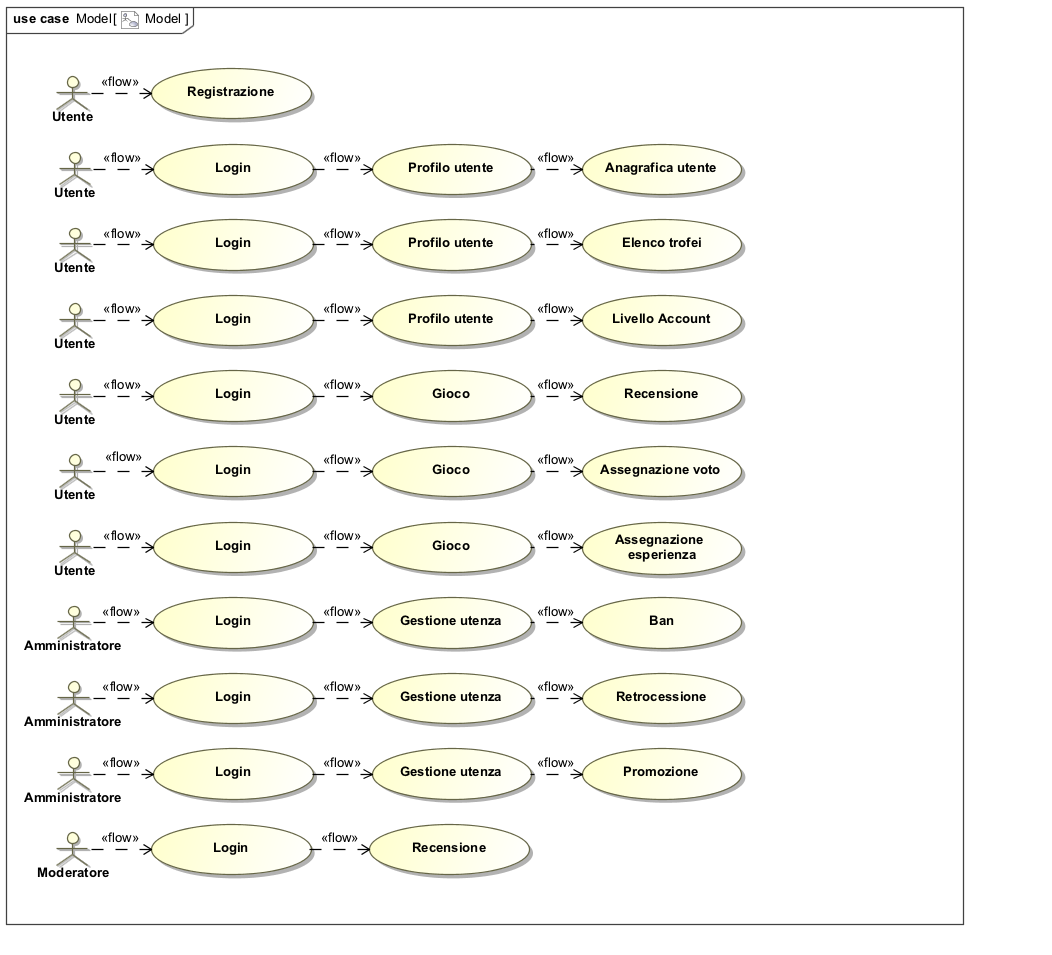
|  |  |
| --- | --- |
| RELATED INFORMATION | Gestione utenza |
| Priority: | Alta |
| Performance | Non espressamente dichiarate |
| Frequency | Non espressamente dichiarata |
| Channels to actors | GUI, back office |
| OPEN ISSUES | L’utente non può gestire i permessi |
| Subordinates | Ban, Retrocessione, Promozione |

|  |  |  |
| --- | --- | --- |
| USE CASE # | Backoffice | |
| Goal in Context | Permettere al moderatore e/o all’amministratore di gestire il sito in base ai loro permessi | |
| Scope & Level | Sistema piattaforma gaming  Level: Primary Task | |
| Preconditions | Il moderatore e/o l’amministratore sono autenticati | |
| Success End Condition | Il moderatore e/o l’amministratore possono gestire il sito | |
| Failed End Condition | Non viene permesso al moderatore e/o all’amministratore di accedere al backoffice e viene restituito un messaggio di errore | |
| Primary | Moderatore, Amministratore | |
| Trigger | Il moderatore e/o l’amministratore in una pagina del sito, cliccano su “Login” | |
| DESCRIPTION | Step | Action |
|  | 1 | Il moderatore e/o l’amministratore effettuano il login |
|  | 2 | Possono effettuare le operazioni a seconda dei permessi |
| EXTENSIONS | Step | Branching Action |
|  | 2a | Possibilità di gestire i permessi degli utenti e/o moderatori  Use case “Gestione utenza” |

|  |  |
| --- | --- |
| RELATED INFORMATION | Backoffice |
| Priority: | Alta |
| Performance | Non espressamente dichiarate |
| Frequency | Non espressamente dichiarata |
| Channels to actors | Database, GUI |
| OPEN ISSUES | Il moderatore e/o l’amministratore non possono utilizzare il backoffice |
| Subordinates | Gestione utenza |

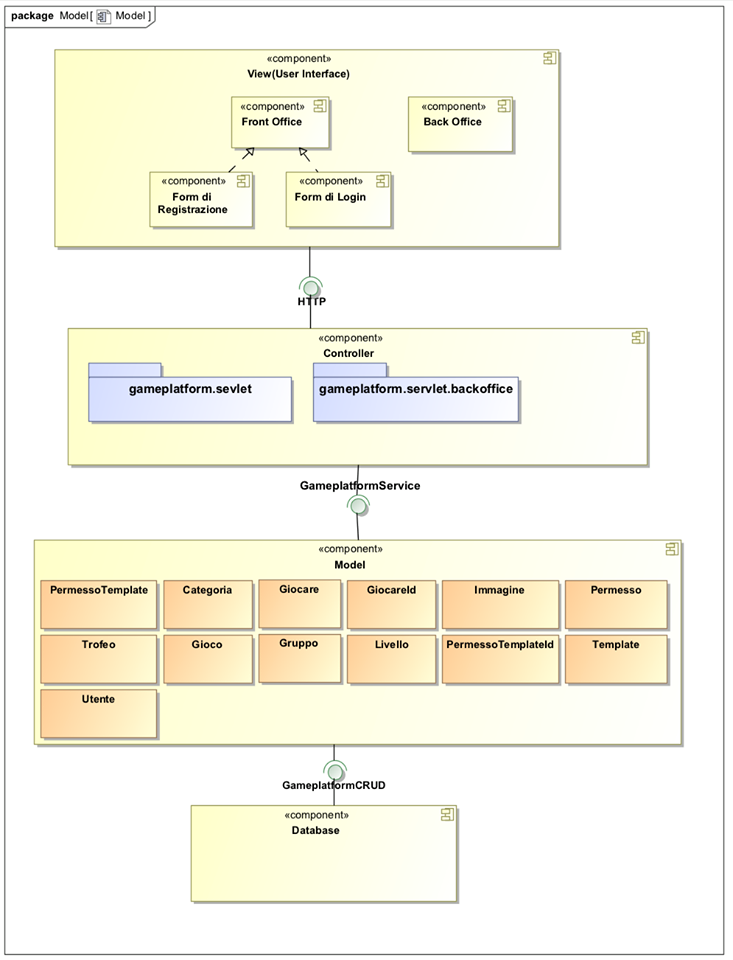
## A.4 Scenari

Nei seguenti scenari vengono mostrati tutti i possibili utilizzi delle funzionalità del sistema che possono utilizzare gli stakeholder.

­­­­­

# B. System Design

## B.1 Component Diagram



Il Sistema è stato sviluppato in quattro macro-componenti:

1. View: rappresenta l’interazione tra il sistema e l’utente nella quale potrà immettere dati in input e ricevere risposte in output grazie alla logica computazionale dell’engine. È divisa in due macro-componenti: Front Office e Back Office.

La prima si occupa della gestione della vista con cui si interfaccerà l’utente. In questa macro-componente abbiamo esplicitato due sotto-componenti, Form di Registrazione e Form di Login, al fine di sottolinearne l’importanza all’interno del sistema: nessun utente può accedere alle funzionalità del sistema senza essere registrato e aver effettuato il log in. La seconda componente si occupa della vista relativa all’amministratore e al moderatore.

1. Controller: implementa tutte le funzionalità del sistema e manipola i dati ricevuti dalla View interfacciandosi a sua volta con il Model. È diviso in due package: gamplatform.servlet e gamplatform.servlet.backoffice. Il primo conterrà le servlet che si occuperanno di prendere una richiesta HTTP e la attua modificando lo stato degl’altri due componenti: Model e View. Il secondo conterrà le servlet relative al back office che avranno lo stesso comportamento, ma relativo alle pagine di back office.
2. Model: si occupa di gestire l’accesso e l’immagazzinamento dei dati presenti nel Db. Mapperemo le tabelle del Database in classi java (sopra descritte): in questo modo abbiamo implementato una persistenza dei dati.
3. Database: rappresenta il contenitore di tutti i dati (vedi B.4). Utilizzeremo il database relazionale MySql, il quale comunicherà con il Model attraverso l’interfaccia CRUD da noi implementata: essa implementa i metodi CRUD (create, read, update, delete) necessari alla comunicazione con il Database.

## B.2 Design Decision

### B.2.1 Architectural design

B.2.1.a Web Based

Il team, dopo varie discussioni, ha deciso di sviluppare l’applicazione orientata al Web (HTML 5, CSS, Javascript) in quanto si possiedono conoscenze maggiori rispetto allo sviluppo lato Desktop.

B.2.1.b Technology

Il team, dopo alcune discussioni, ha deciso di usare le Java Servlet, server Tomcat, JDBC, il framework Hibernate, MySql e Jsp.

Abbiamo scelto di utilizzare Tomcat, un motore in grado di eseguire lato Server applicazione Web basate su tecnologie J2EE e costituite da componenti Servlet e pagine Jsp. Non è da sottovalutare il fatto che Tomcat è disponibile per tutti i Sistemi Operativi, permettendo la portabilità dell’applicazione.

Si è reso necessario l’utilizzo delle Java Servlet in quanto il committente ci ha espressamente richiesto di sviluppare il sistema in ambiente Java.

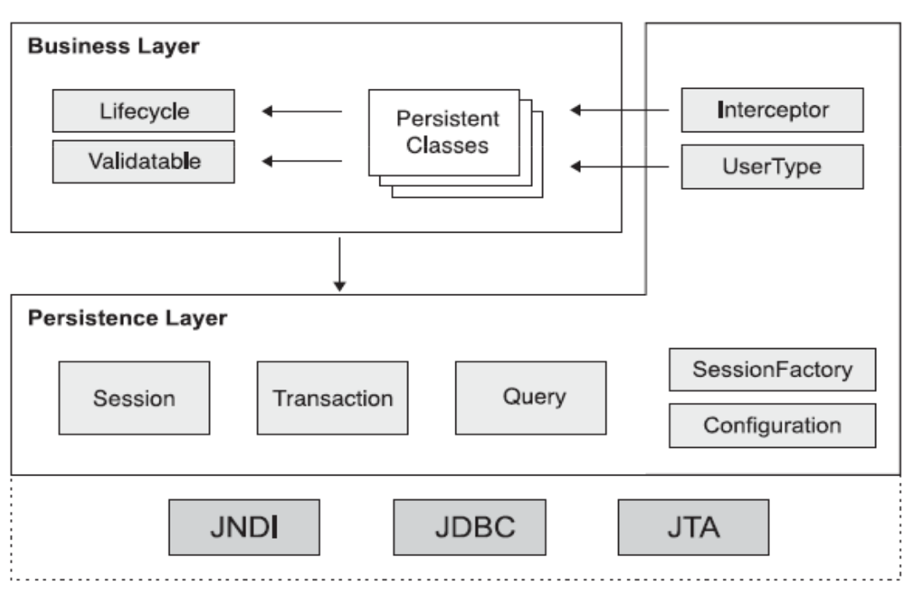
Utilizzeremo il database MySql, in quanto il team ha sufficienti conoscenze pregresse a riguardo. Per tal motivo verrà utilizzata l’interfaccia JDBC, la quale conterrà MySql Connector, che permetterà la connessione con il nostro database.

Il team, dopo un approfondito studio, ha scelto di utilizzare il framework Hibernate, che fornisce un servizio di ORM (Object Relational Mapping).

Un ORMFramework mira ad offrire una soluzione completa alla gestione della persistenza dei dati.

Hibernate è uno dei framework di maggior successo che consente allo sviluppatore di automatizzare le procedure per le operazioni cosiddette CRUD (Create, Read, Update, Delete) dei database. Inoltre non richiede nessun particolare ambiente per poter funzionare: basta un collegamento ad un datasource e le sue librerie.

L’architettura di Hibernate:



Il team, dopo un attento studio, ha deciso di implementare le Jsp (Java Server Pages), ovvero una tecnologia di programmazione Web in Java la quale ci permetterà di sviluppare contenuti dinamici in HTML5, visto che si interfacciano in maniera ottimale con Tomcat e Java. Sarà inoltre possibile, grazie a degli speciali tag all’interno delle pagine HTML, invocare funzioni in Javascript.

### B.2.2 Software Design

B.2.2.a Giochi

Il sistema prevedrà l’esistenza di più giochi. Ogni utente potrà giocare ad ogni gioco del sistema in modo gratuito. Il team ha scelto di implementare differenti giochi con lo scopo di ampliare la scelta del cliente.

Ogni gioco sarà assegnato ad una categoria al fine di implementare una ricerca più specifica.

E’ stata implementata una gallery, all’interno della pagina gioco, dove vengono inserite varie immagini di schermate riguardanti il gameplay per una visione più completa dello stesso.

B.2.2.b Livello

Il team ha deciso di tenere traccia del livello sia del singolo gioco sia dell’account, poiché avendo più giochi ci è sembrato necessario poter classificare un giocatore anche rispetto ad un singolo game. Il livello generale viene calcolato sommando tutti i punti esperienza.

La formula utilizzata per calcolare il livello corrente del gioco è: “*exp/100*”, dove *exp* è l’esperienza attuale (approssimando per difetto).

Utilizzeremo la seguente formula per calcolare il livello totale: ((somma esperienza di ogni singolo gioco)/100)\*(numero di giochi).

B.2.2.c Trofei

Il sistema oltre a prevedere i trofei per livello (del singolo gioco), come richiesto dal committente, prevedrà altre tipologie di trofei:

* Totale partite giocate
* Ore di gioco
* Numero di accessi alla piattaforma

Questa decisione è stata presa al fine di incentivare il giocatore a giocare il più possibile.

B.2.2.d Timeline

Il sistema prevedrà due Timeline: una che mostrerà la cronologia dei livelli acquisiti dell’utente, mentre l’altra mostrerà i trofei acquisiti. Tale scelta è stata dettata dal fatto che riteniamo opportuno mantenere traccia solo dei progressi generali e non di ogni singolo gioco.

## B.3 Design Pattern

**­­**Il design pattern adottato è il:

Singleton.



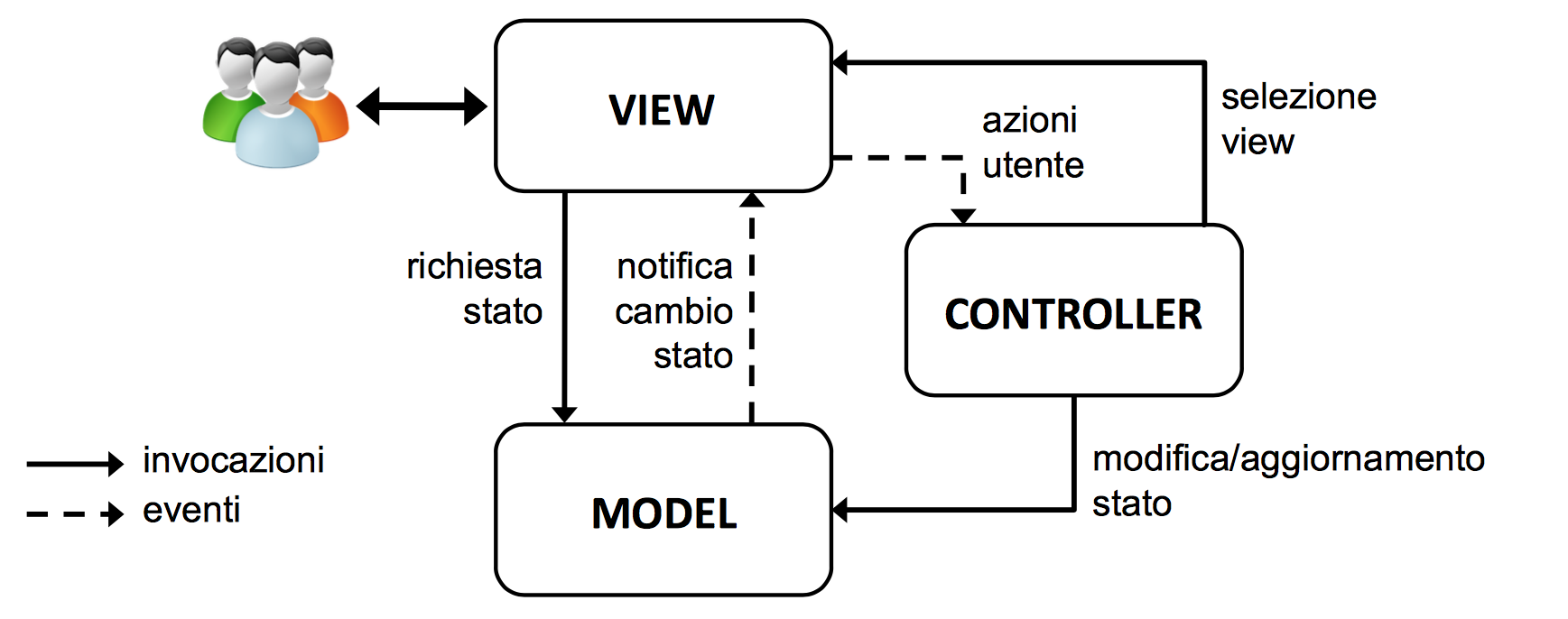
Al fine di rendere la nostra piattaforma portabile, il team ha deciso di implementare il design pattern Singleton, il quale ha lo scopo di garantire che venga creata una e una sola istanza di una classe, fornendo un punto di accesso globale a tale istanza.

Il design pattern è stato usato nelle due classi “GameplatformCRUD” e “GameplatformService”. Nella prima sono dichiarate le operazioni di “save”, “update” e “delete” necessarie per lo sviluppo del back office. Nella seconda sono dichiarate alcune operazioni comuni nel lato front-end: “login”, “logout”, “registrazione”, “controllo-Permessi”, “lista-Giochi”, “gioco-Singolo”, “lista-Immagini-Gioco” e “recensioni”.

## B.4 Pattern Architetturale

Il pattern architetturale adottato principalmente è il:

Model View Controller (MVC).



Il pattern è basato sulla separazione dei compiti fra le componenti del sistema che interpretano tre ruoli principali:

Model: rappresenta il modello dei dati di interesse per l’applicazione

* Incapsula lo stato dell’applicazione.
* Gestisce l’accesso ai dati.
* Fornisce le funzionalità per l’aggiornamento dello stato e l’accesso ai dati.

View: fornisce una rappresentazione grafica del model

* Consente l’interazione con l’utente.
* Definisce le modalità di presentazione dei dati e dello stato dell’applicazione

Controller: definisce la logica di controllo e le funzionalità applicative

* Opera sul model in base agli eventi ed ai comandi ricevuti
* Può aggiornare il View in base al risultato dell’elaborazione o alle scelte dell’utente
* Gestisce gli eventi e i comandi generati dall’utente

Si è scelto di utilizzare tale pattern in modo da tenere i tre strati (Model, View e Controller) quanto più separati e renderli il più indipendenti possibile, inoltre tale pattern apporta altri vantaggi come:

* facilitare il riutilizzo del codice: più una parte del lavoro è indipendente dal resto, più possibilità ci sono che ci possa venire utile per altre applicazioni.
* utile per suddividere il lavoro nel caso ci debbano lavorare più persone o team.
* utilizzando un modello rigido e regole standard facilita un eventuale lavoro di manutenzione e agevola la comprensione anche da parte di altri programmatori.
* nel caso si cambi tipo di database sarà possibile adattare l’applicazione senza dover mettere mano a tutto il codice ma solo al modello, quindi avremo maggiore flessibilità.

Concludendo possiamo affermare aver fatto questa scelta al fine di facilitare la scalabilità e la manutenzione dell’applicazione.

## B.5 E-R

# 

Il nostro diagramma E-R ha come entità principale “Utente”, la quale possiede come primary key l’attributo “Username” che è unico in tutto il sistema. Oltre ai banali attributi quali “Nome”, “Cognome”, “Età”, “Password” e “email” abbiamo inserito “Numero accessi” (che corrisponde al numero di accessi giornalieri effettuati dall’utente) per poter ricompensare i giocatori con trofei in base al numero di accessi giornalieri, “Exp Totale” per calcolare il livello totale dell’account, ottenuto sommando l’esperienza dei vari giochi cui l’utente ha giocato, dividendola per 100 e moltiplicandola per il numero di giochi giocati (vedi B.2.3) e “Ban” per poter gestire il ban degli utenti del sistema.

Le altre entità correlate ad “Utente” sono: “Gioco”, “Livello”, “Trofeo” e “Gruppo”.

Il primo possiede come attributi, oltre alla banale primary key “Nome” e a “Descrizione”, anche “Specifiche” che indica i requisiti minimi per eseguire il gioco, “Media” che indica la media dei voti espressi dagli utenti sul gioco e “Data” che rappresenta la data di inserimento nella piattaforma.

L’entità “Immagine”, a sua volta, è collegata con l’entità “Gioco”. Abbiamo definito questa entità al fine di poter definire le immagini da poter inserire nel sito. Ha come attributi: la primary key “Id” e il “Path” nel quale si trova l’immagine.

L’entità “Livello”, che rappresenta il livello totale degli account degli utenti, oltre ad “id” e al “livello” possiede l’attributo “Data” per tenere traccia di quando un utente ha ottenuto un determinato livello e poter quindi visualizzarlo nella timeline.

L’entità “Trofeo”, oltre al “Nome” dello stesso e all’ “Obiettivo”, che indica come ottenere un determinato trofeo, possiede anche l’attributo “icona” che indica il path relativo alla posizione del file .ico relativo all’icona scelta per il trofeo.

“Giocare” è la relazione all’interno della quale vengono salvate tutte le recensioni degli utenti relative ai giochi; essa è composta da un “id” e dalla sua eventuale “Descrizione”, nel caso un utente volesse lasciare solo un commento scritto relativo al gioco, o “Voto” nel caso volesse lasciare un feedback numerico, o da entrambi.

L’ultima entità collegata ad “Utente” è “Gruppo”, che differenzia i tre tipi di attori del nostro sistema: Utente, Amministratore e Moderatore, salvati nel campo “Nome”.

L’unica relazione rilevante del nostro sistema è “Giocare”, la quale serve a tenere traccia sia dei dati delle sessioni di gioco dell’utente tramite tre attributi: “Numero accessi”, “Minuti” ed “Exp”, e sia delle recensioni (massimo una per utente) del gioco tramite gli attributi “Data”, “Voto” e “Approvazione”: quest’ultimo attributo si è reso necessario per la visualizzazione della recensione sul sito previa moderazione da parte dei moderatori preposti.

Gli attributi “Numero accessi” e “Minuti” serviranno per l’assegnazione di determinati trofei elencati (vedi B.2.4), mentre “Exp” servirà per calcolare il livello relativo ad un gioco e all’utente che ci gioca.

“Permesso” è l’entità in cui vengono salvati i file del sistema nell’apposito attributo “Nome”, e il relativo “Indirizzo”.

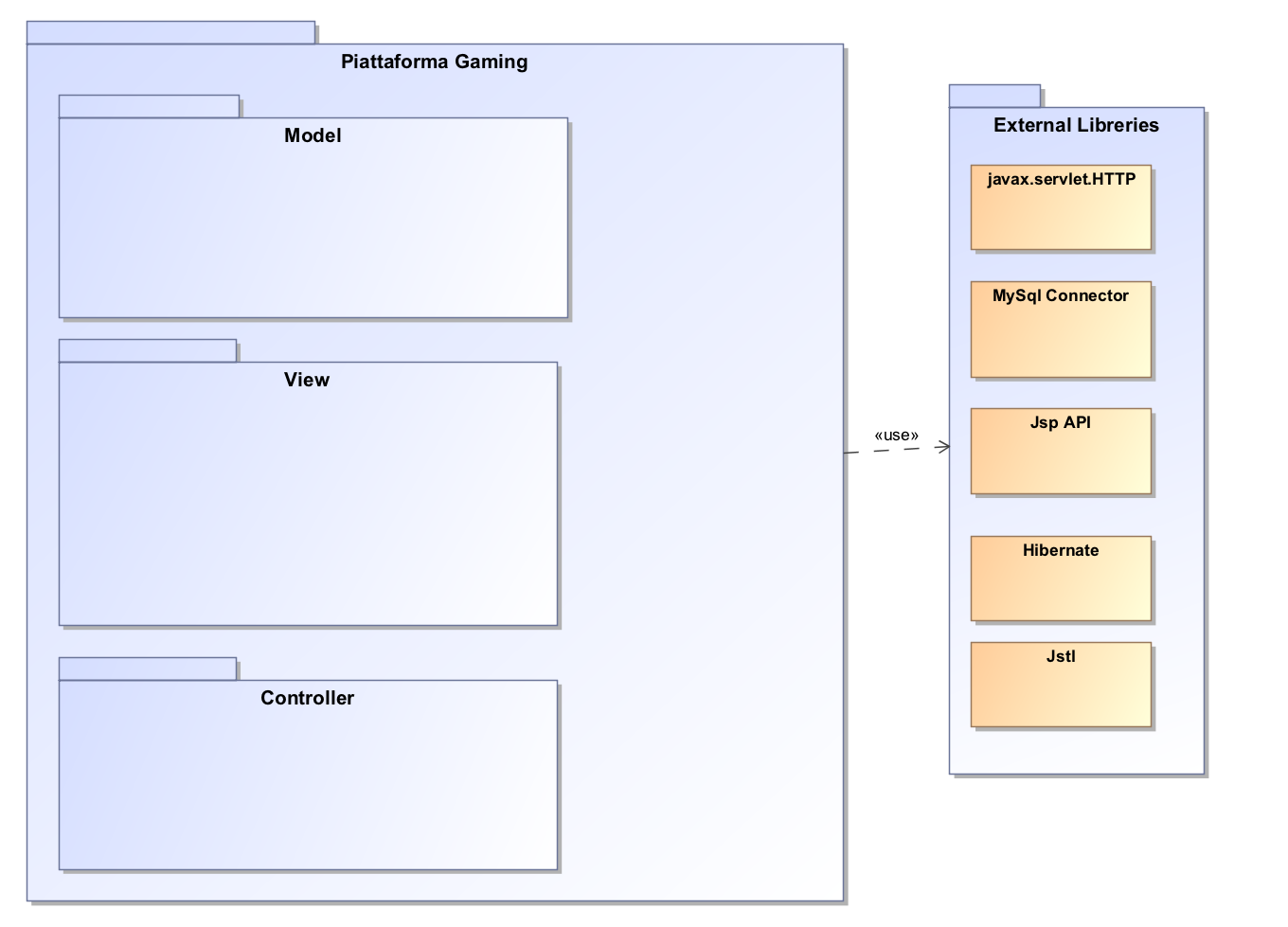
Infine l’entità “Template” è correlata con “Permesso” ed è costituita dall’unico campo “Nome” utilizzato per identicare il template. Nella relazione “Possedere” è specificata la posizione che il template deve ricoprire all’interno della pagina partendo dall’alto.

# C. SOFTWARE/ OBJECT Design

## C.1 Class Diagram

Abbiamo suddiviso il nostro sistema di “Piattaforma Gaming” in tre sotto-package dettati dal tipo di architettura da noi utilizzato: “Model”, “View” e “Controller”.

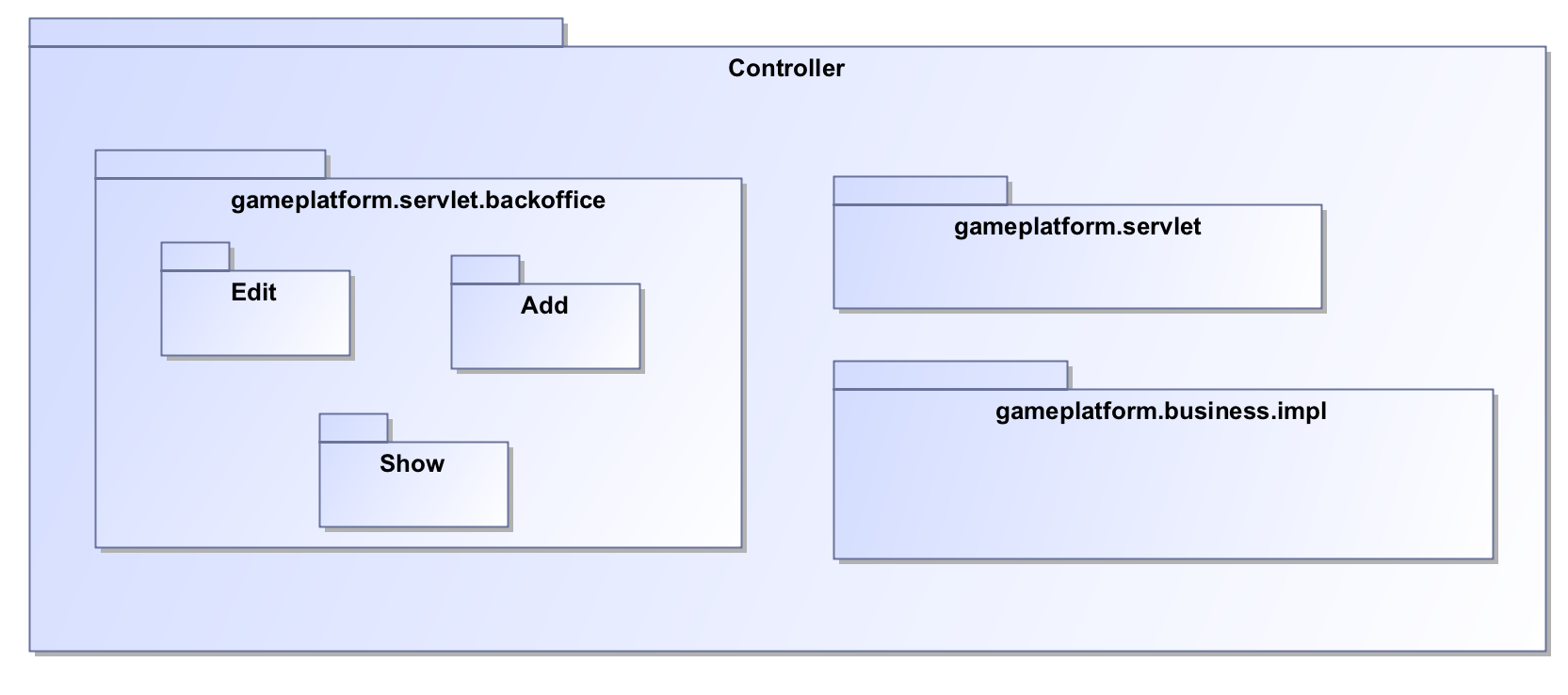
Il nostro sistema farà uso di librerie esterne quali: “My Sql Connector” per la connessione con il database, le “java servlet”, “Hibernate”, le API “Jsp” (per dettagli sulle ultime tre vedi B.2.6).



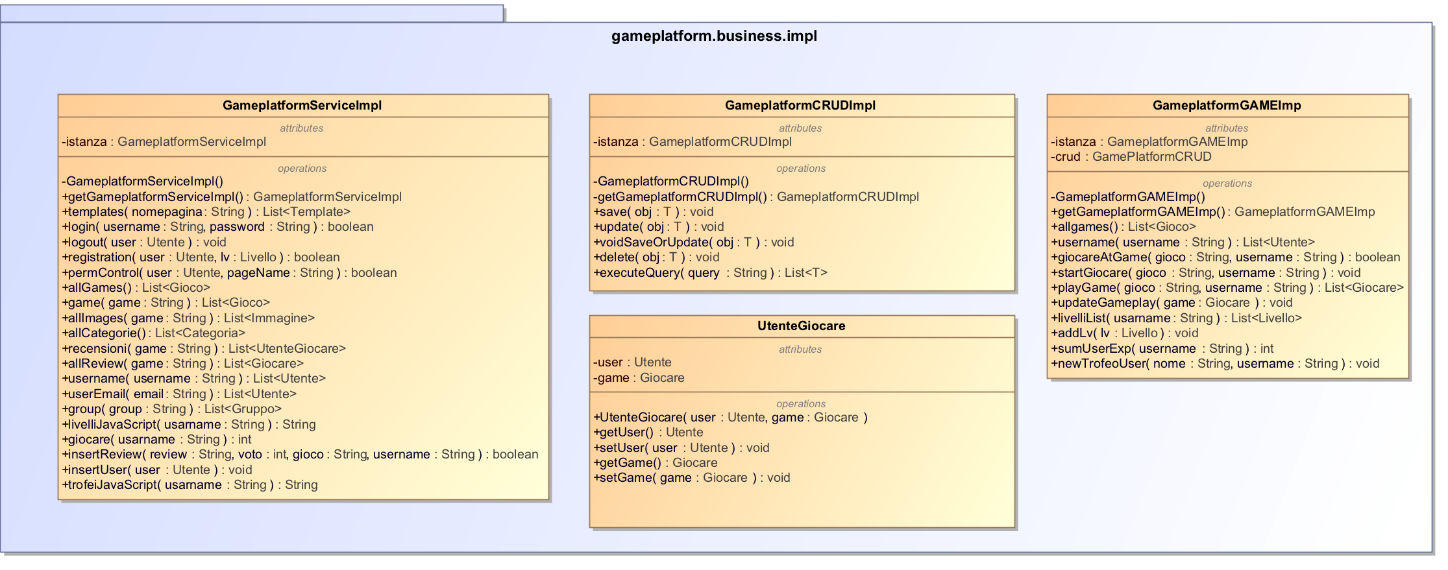
Model

## C:\Users\davx9\AppData\Local\Microsoft\Windows\INetCache\Content.Word\20179585_467181280316970_653867262_n.png

All’interno del Model abbiamo fondamentalmente le tabelle del database espresse come classi (per una documentazione più dettagliata delle classi intese come entità vedi B.5). Ognuna di esse ha al suo interno gli attributi presenti nel database con le relative operazioni di get e set.



Per mettere in risalto l’architettura MVC utilizzata, abbiamo racchiuso i package “gameplatform.servlet.backoffice”, “gameplatform.servlet” e “gameplatform.business.impl” all’interno di “Controller” per far capire il ruolo di questi all’interno della piattaforma.



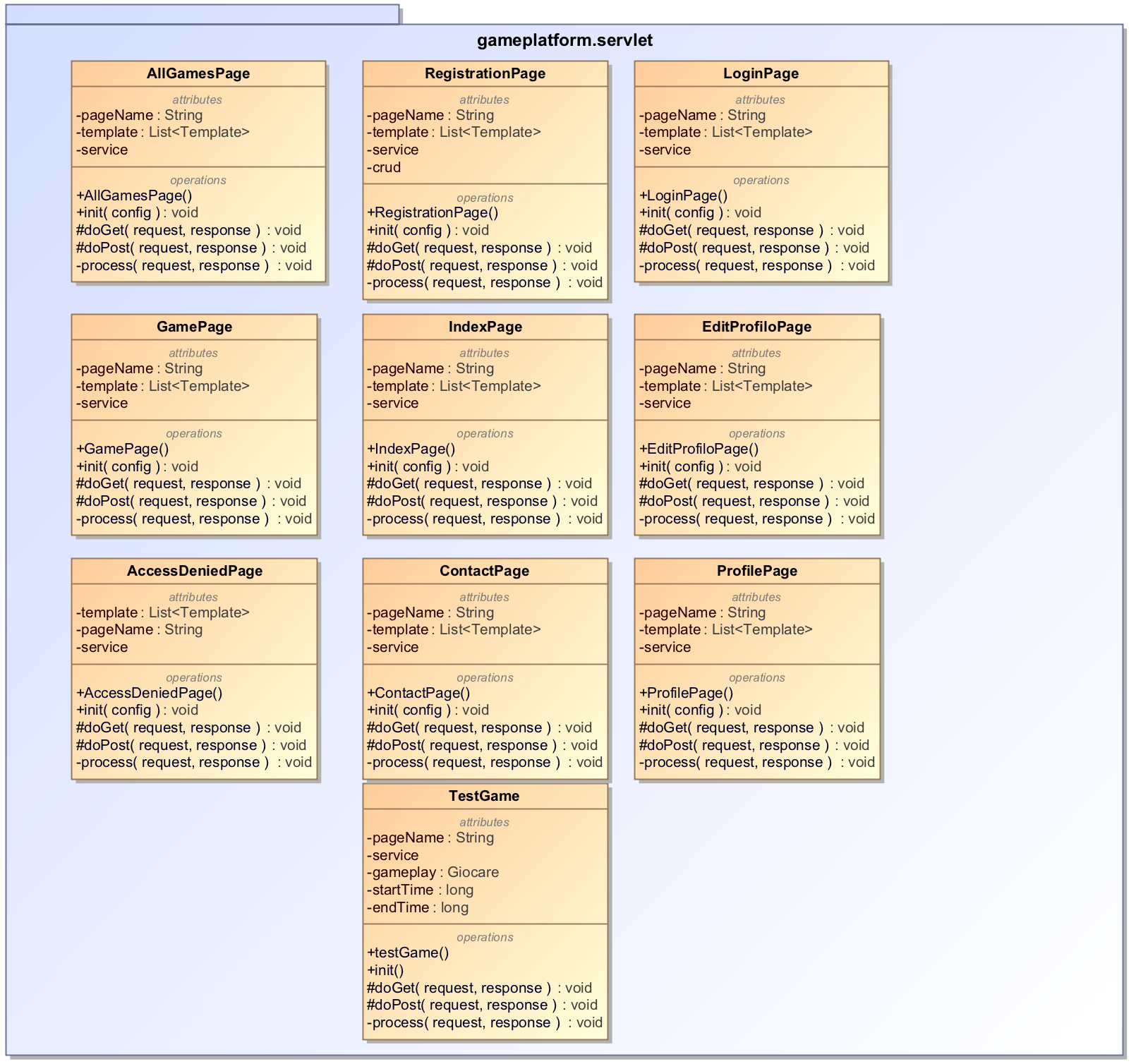
All’interno del package “gameplatform.business.impl” sono presenti alcune classi volte ad assicurare la portabilità della piattaforma: “GameplatformServiceImpl”, “GameplatformCRUDImpl”, e “GameplatformGAMEImp”.

Nella prima abbiamo inserito i metodi che andremo ad utilizzare più frequentemente nelle varie servlet. Tale classe, che implementa i metodi dell’interfaccia “GameplatformService” viene utilizzata soprattutto per interfacciarsi con le classi del Model al fine di evitare di scrivere lo stesso codice in più classi, favorendone la leggibilità del codice.

Nella seconda sono stati inseriti i metodi CRUD (create, read, update, delete). Tale classe implementa l’interfaccia “GameplatformCRUD”. Nel nostro caso all’interno dei metodi richiamiamo le operazioni CRUD di hibernate (save, update, saveorupdate e delete) che permettono di gestire le istanze delle nostre classi mappate. Tale classe ci permette di poter cambiare tipo di database senza avere un impatto negativo nella manutenzione del codice: dovremo solo cambiare l’implementazione dell’interfaccia.

“GameplatformGAMEImp” è una classe dove vengono implementati tutti i metodi richiamati dai giochi. Tale classe implementa l’interfaccia “GameplatformGAME”. La sua creazione è stata motivata al fine di poter accettare giochi di produttori esterni. Per poter far ciò il team dovrà solo mettere tale interfaccia al produttore esterno, il quale la userà nella creazione del proprio gioco.

La classe “UtenteGiocare” è stata creata al fine di assegnare un utente ad un gioco: si è resa necessaria tale classe solo per il passaggio di valori dalla servlet al jsp nella creazione della servlet “gamepage”.



Il package “gameplatform.servlet” sono inserite tutte le servlet riguardanti il front office. Queste comunicano i jsp associati (View) e con l’interfaccia “GameplatformService”. Tutte le classi presentano il metodo “init” (per la gestione della creazione della servlet), il metodo doGet ( per la gestione delle chiamate in Get), il metodo doPost (per la gestione delle chiamate in Post) e il metodo process (per la gestione del codice condiviso sia da doPost che da doGet).

“AllGamesPage”

Tale servlet presenta tutti i giochi inseriti nel sistema a disposizione dell’utente.

“RegistrationPage”

Tale servlet permette all’utente la registrazione presso il nostro sistema: nessuno può accedere al sistema senza essere registrato.

“LoginPage”

Tale servlet permette all’utente (moderatore, amministratore e utente) di accedere al sistema. Nessuno può accedere al sistema senza aver effettuato il login.

“GamePage”

Tale servlet permette all’utente di aver accesso ai dettagli di un gioco: da tale servlet potranno essere effettuate le recensioni. Le recensioni possono essere effettuate solo se l’utente ha già effettuato almeno una sessione di gioco. Ogni recensione dovrà essere approvata da un moderatore, dopo non potrà più essere modificata.

“IndexPage”

Tale servlet rappresenta la pagina principale del sistema nella quale è contenuto uno slider, con gli ultimi tre giochi inseriti nel sistema, e la lista dei giochi.

“EditProfiloPage”

Tale servlet permette all’utente di modificare i propri dati, la propria immagine del profilo e la password.

“AccessDeniedPage”

Si viene reindirizzati su questa servlet quando non si hanno i permessi per l’accesso a una data pagina o non si è effettuato il login. Da tale servlet si verrà reindirizzati sulla “IndexPage” tramite un link.

“ContactPage”

Tale servlet fornirà le informazioni per mettersi in contatto con gli amministratori del sistema. Il team ha implementato i metodi per l’invio della email, ma questi sono stati saltati per difficoltà nell’attivazione di un server smtp in tomcat

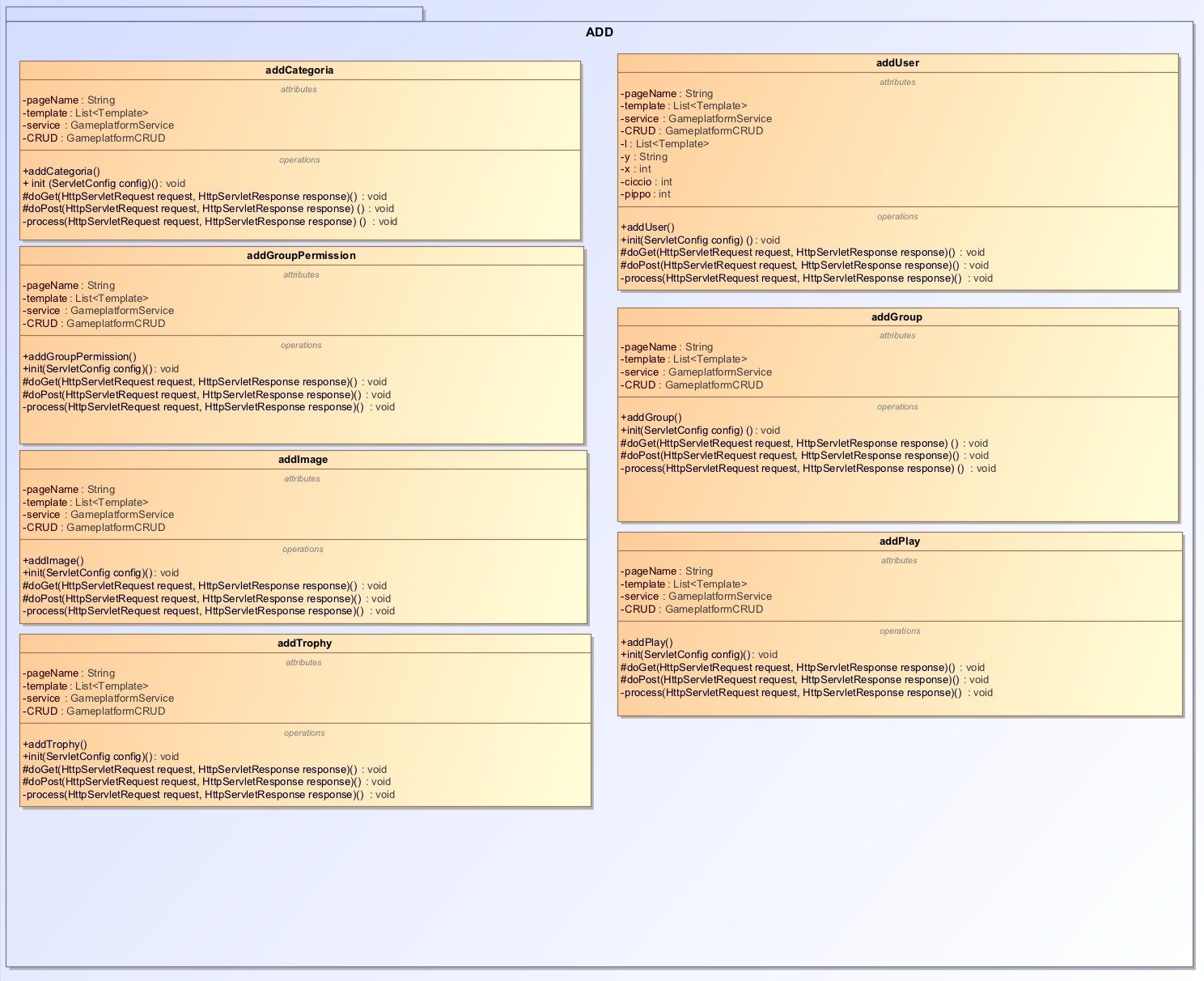
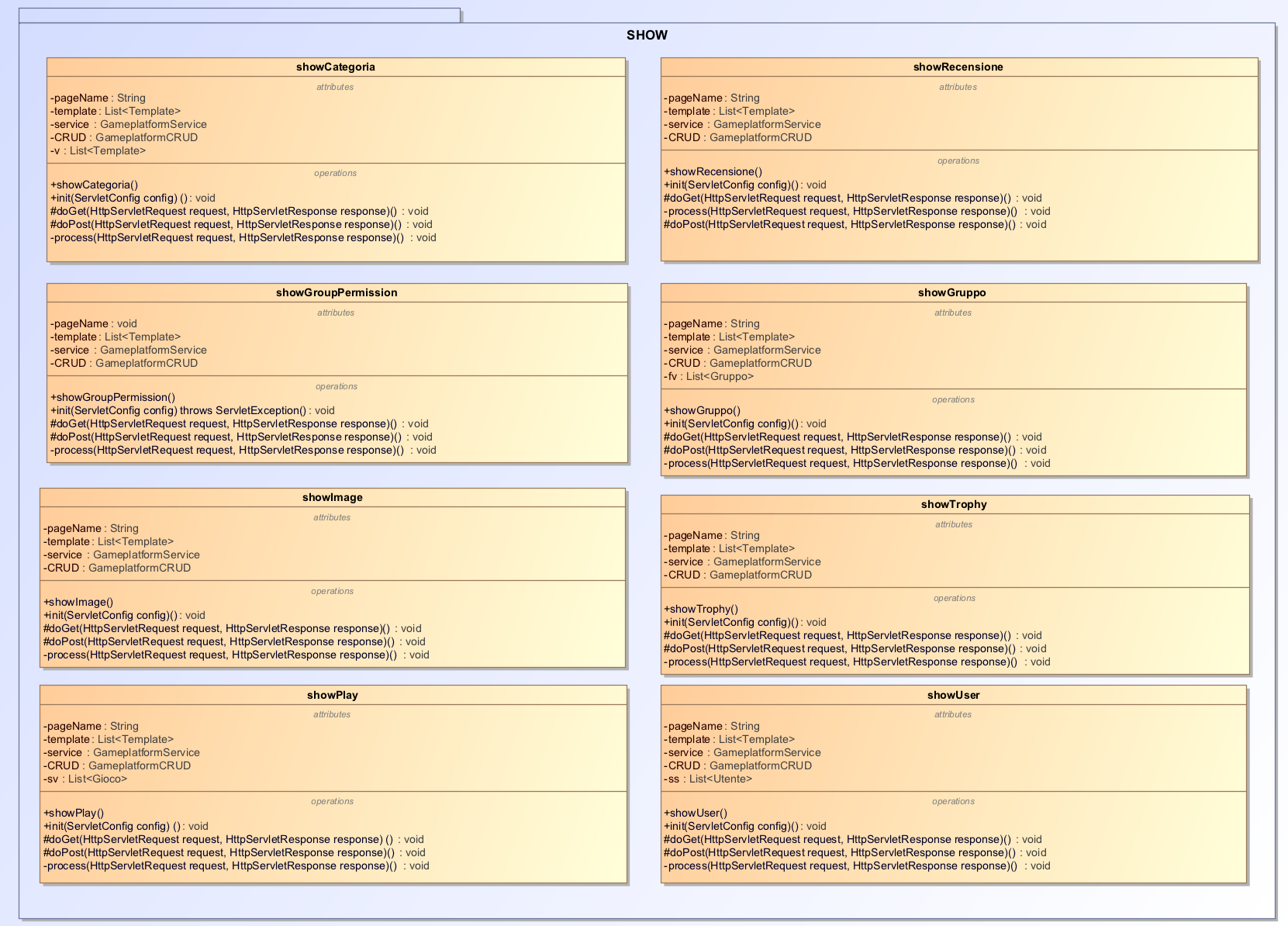
“ProfiloPage”

Tale servlet fornirà l’accesso ai propri dati anagrafici e alla timeline dei livelli dell’account e di tutti i trofei conquistati.

“TestGame”

Tale servlet permetterà all’utente di effettuare una sessione di gioco.

## C:\Users\davx9\AppData\Local\Microsoft\Windows\INetCache\Content.Word\EditBackofficeClass.png



All’interno del package gameplatform.servlet.backoffice sono presenti a loro volta altri sottopackage( EDIT, SHOW, ADD) dove sono presenti le classi volte a rappresentare il nostro Backoffice. I metodi raffigurati saranno sempre gli stessi come da definizione di “Servlet”. Tuttavia al loro interno ci sarà tutta la logica che farà “girare” il backoffice.

ADD: in questo sottopackage troviamo tutte le classi che ci permettono di inserire attraverso il backoffice tutti i record del database.

EDIT: in questo sottopackage troviamo tutte le classi che ci permettono di modificare attraverso backoffice un qualunque campo di un determinato record del database.

SHOW: in questo sottopackage troviamo tutte le classi che ci permettono di visionare attraverso il backoffice tutti i record del database.

View

All’interno del sotto-package “View” sono presenti tutti i file .jsp del nostro sistema che hanno il compito di gestire la costruzione dell’interfaccia grafica (GUI) che rappresenta il mezzo mediante il quale gli utenti interagiranno con il sistema.

## C.2 Sitemap

Trattandosi di una piattaforma Web il team ha deciso di aggiungere una sezione nel documento relativa alla spiegazione della piattaforma da un punto di vista web in modo da capirne bene il funzionamento. Ciò sarà effettuato tramite raffigurazione del sitemap nella quale ogni “card” corrisponderà ad una pagina HTML.

