**“Object Oriented Software Design”**

**Course**

**a.a. 2016-2017**

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Team Members** | | |  |
| **Name & Surname** | **Matriculation Number** | **E-mail address** | **Anno** |
| Stefano Corsetti | *227288* | *s.corsetti@hotmail.it* | *3°anno* |
| Luca D'Orazio | *227635* | *lucaadorazio@gmail.com* | *3°anno* |
| Tommaso Di Salle | *236202* | [Tommasodisalle@gmail.com](mailto:Tommasodisalle@gmail.com) | *3°anno* |
| Eugenio Mancini | *230024* | *Emancini1992@libero.it* | *3°anno* |

Indice

A.Documento dei requisiti

B. System Design

C. Software/Object Design

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 da moderatore 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

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

10)Gestione utenti - Gestita da Moderatore e Amministratore

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

10.2) Retrocessione - l'utente viene declassificato

11)Gestione livelli - Gestita da Amministratore

11.1) Inserimento - livello di gioco inserito

***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

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

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

-promozione

-retrocessione

11)gestione **livelli** (scenario 11)

-inserimento

***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)***

l 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 ha le sole funzionalità di:

- gestione delle recensioni, ovvero una recensione prima di essere pubblicata deve essere approvata da questo moderatore.

-Promuovere un utente a moderatore.

L'ultimo attore identificato è **l'Amministratore** con tutte le funzioni del sistema con accesso al BackOffice (pannello di gestione del sistema)

***A.8 Descrizione - Use Case Ad Alta Priorità***

|  |  |
| --- | --- |
|  |  |
| Nome Use-Case | **Login/Registrazione** |
| Attori partecipanti | Utente - Moderatore - Amministratore |
| Descrizione | Permette ad un utente/Moderatore/Amministratore registrato di accedere al sistema o ad un utente non registrato di registrarsi. |
| Evento scatenante | Click su relativo pulsante |
| Usi | Login/Registrazione effettuati con successo o errore |

|  |  |
| --- | --- |
|  |  |
| Nome Use-Case | **Visualizzazione Lista Giochi** |
| Attori partecipanti | Utente - Moderatore - Amministratore |
| Descrizione | Permette la visualizzazione della lista dei giochi anche a utenti non registrati |
| Nome Use-Case | **Gestione recensioni** |
| Attori partecipanti | Moderatore - Amministratore |
| Descrizione | Permette di gestire le recensioni, quindi approvarle oppure eliminarle |
| Evento scatenante | Click su relativo pulsante nel BackOffice\* |
| Usi | Recensione approvata o Eliminata |

|  |  |
| --- | --- |
|  |  |
| Nome Use-Case | **Sessione gioco** |
| Attori partecipanti | Utente |
| Descrizione | Permette a utenti registrati nel sistema (quindi "loggati") di giocare scegliendo un gioco dalla lista |
| Evento scatenante | Click su pulsante "Gioca" |

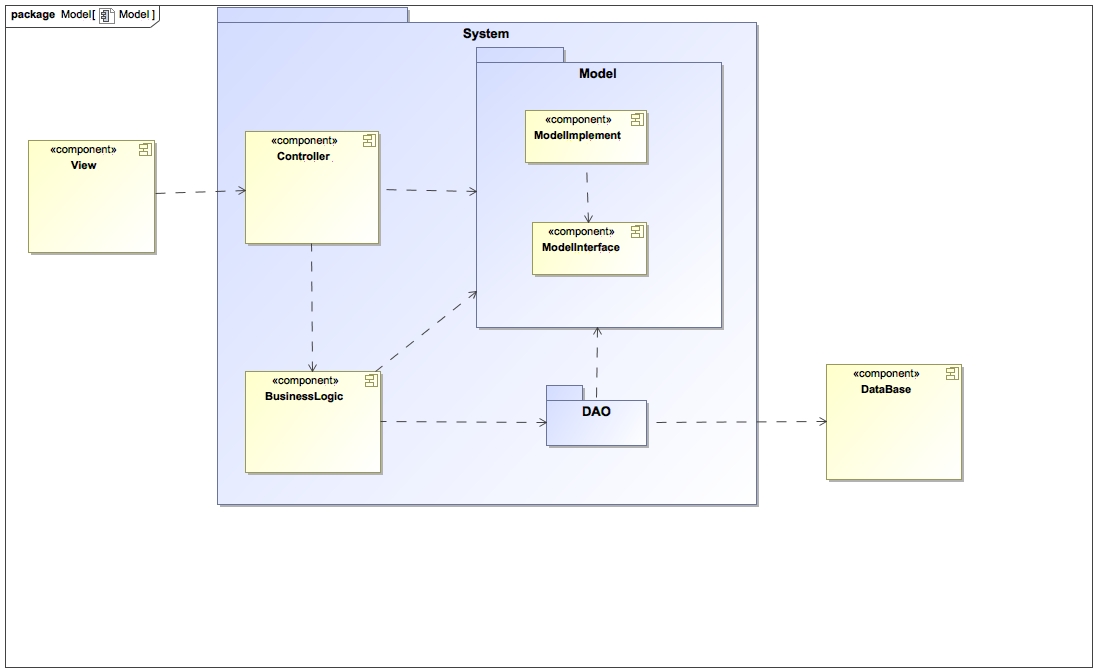
|  |  |
| --- | --- |
|  |  |
| Nome Use-Case | **Votazione/Recensione** |
| Attori partecipanti | Utente |
| Descrizione | Permette a utenti registrati nel sistema di valutare/votare il gioco da 0 a 5 punti nonché di poter scrivere delle recensioni sulla propria esperienza (aspettando l'approvazione) |
| Evento scatenante | Click su uno dei 5 bottoni per la votazione e "Scrivi recensione" per la recensione |

|  |  |
| --- | --- |
| Nome Use-Case | Esperienza Gioco |
| Attori partecipanti | Utente |
| Descrizione | Permette a utenti registrati nel sistema la funzione di visualizzare le proprie esperienze e trofei vinti |
| Evento scatenante | Bottone "Profilo" |

\*BackOffice: Pannello gestione di sistema per Amministratore e Moderatore

B. System Design

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

******

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

Il pattern architetturale di riferimento è MVC

Più in dettaglio:

**MVC:**

* **Model**: fornisce i metodi per accedere ai dati utili all'applicazione
* **View**: Si occuperà solo della gestione delle interfacce utente. Essa visualizza i dati contenuti nel Model e si occupa dell'interazione con gli utenti.
* **Controller**: Riceve i comandi dell'utente e li attua modificando gli stati delle altre due componenti.

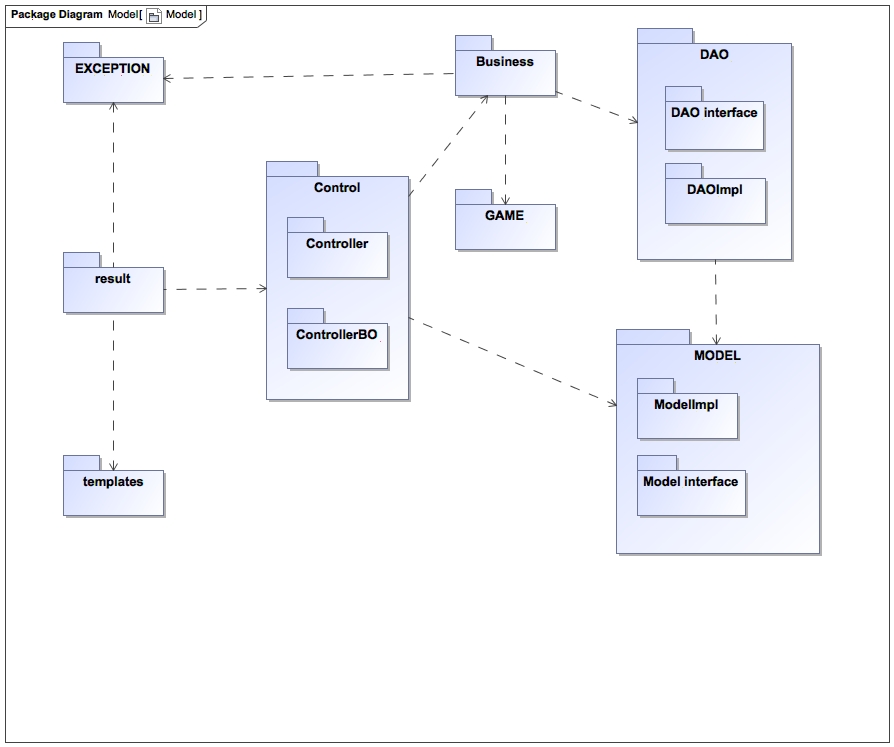
Il team ha deciso di utilizzare quest'ultimo in quanto, grazie ad esso abbiamo la cosiddetta separazione delle logiche che, nel nostro caso, è utile in quanto nel progetto non lavora una persona singola ma un team, e la suddivisione delle logiche risulta utile per la divisione del lavoro.

**DAO - Data Access Object:**

Altro pattern utilizzato è il DAO. Questo viene utilizzato per estrarre e incapsulare l'accesso ai dati. Questo ci permette di non avere impatto diretto con i moduli Controller e View.

Oltre a questo per enfatizzare ulteriormente questa separazione il team ha deciso di incorporare un nuovo modulo che chiameremo BusinessLogic che si occuperà di comunicare con il DAO.

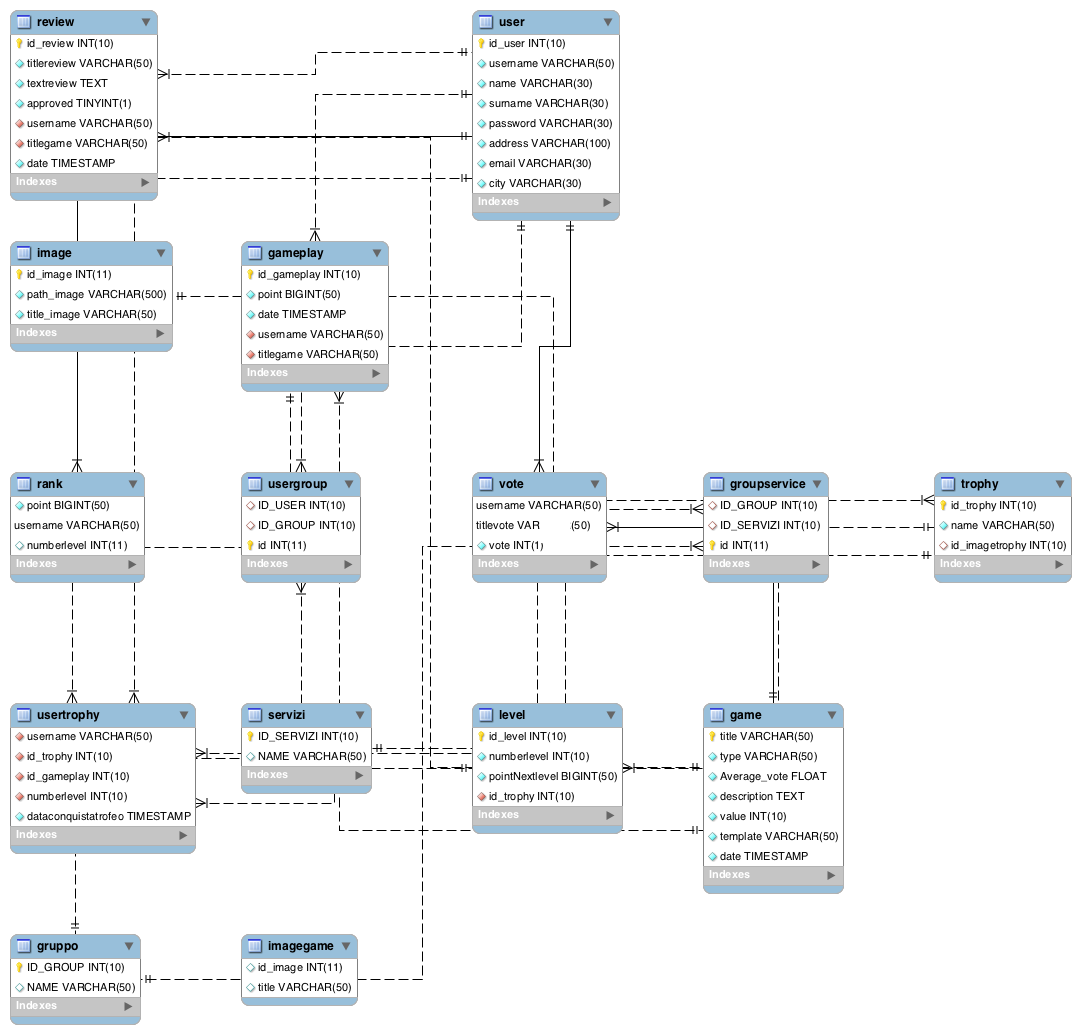
***B.3 Package Diagram***

******

Il team cercando di massimizzare il riuso, l'anticipazione di nuovi requisiti, le modifiche ai requisiti esistenti e la progettazione del sistema in modo che possa evolvere conseguentemente, ha identificato i seguenti package:

* **Controller**: contiene tutte le Servlet relative alle singole pagine del sistema. E' stato suddiviso ulteriormente in: Controller e ControllerBO. Esso utilizza Model, Business, Result e Exception.
* **DAO:** contiene tutte le classi e le interfacce per la realizzazione del DAO, suddiviso in:
* *nomeClasse:*interfacce che definiscono le operazioni effettuabili per entità
* *nomeClasse*Impl: classi che implementano le interfacce realizzanod in modo concreto le operazioni, usando il Model.
* **Exception**: package adibito alla gestione degli errori.
* **Model**: tutte le classi rappresentanti le entità gestite dal sistema
* **Result**: esso carica e compila il template
* **Game**: package per la gestione dell'assegnazione dell'esperienza
* **Business:** package di separazione tra Controller e DAO

***B.4 Modello ER***

******

C. Software/Object Design

***C.1 Class Diagram***

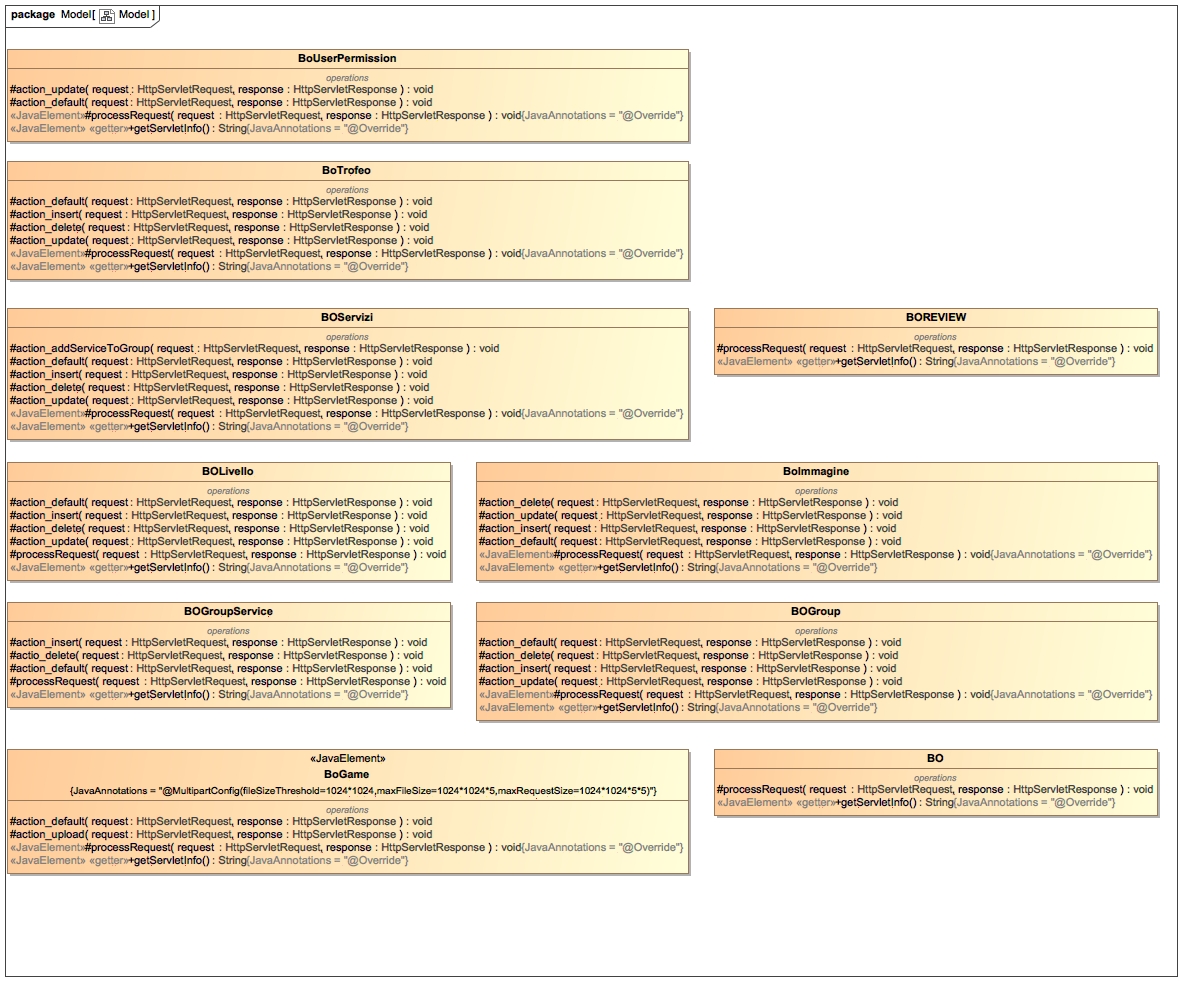
Il team, per rispettare le design decision descritte in sezione B3, ha deciso di descrivere non le classi singolarmente (facilmente deducibili dal class diagram di seguito allegato), ma analizzare a livello di package, dato che ele funzionalità sono le stesse.

1. **Control (Controller / ControllerBO)**

* Controller



* ControllerBO



Control comprende tutte le servlet, per una questione di praticità è stato suddiviso in due package:

**Controller** e **ControllerBO**.

-In Controller troviamo le seguenti classi:

GamePlay: Servlet che gestisce la sessione di gioco

Home: Servlet che gestisce la pagina iniziale dell'applicazione dove viene mostrata la lista dei giochi

Logout: Servlet che gestisce la terminazione di una sessione

MyProfile: Servlet che gestisce la pagina dei profili personali

PaginaGioco: Servlet che gestisce la pagina del gioco singolo,la possibilità di votare e recensire un gioco

Ranking: Servlet che gestisce la visualizzazione della classifica generale

Register: Servlet che gestisce la pagina di registrazione

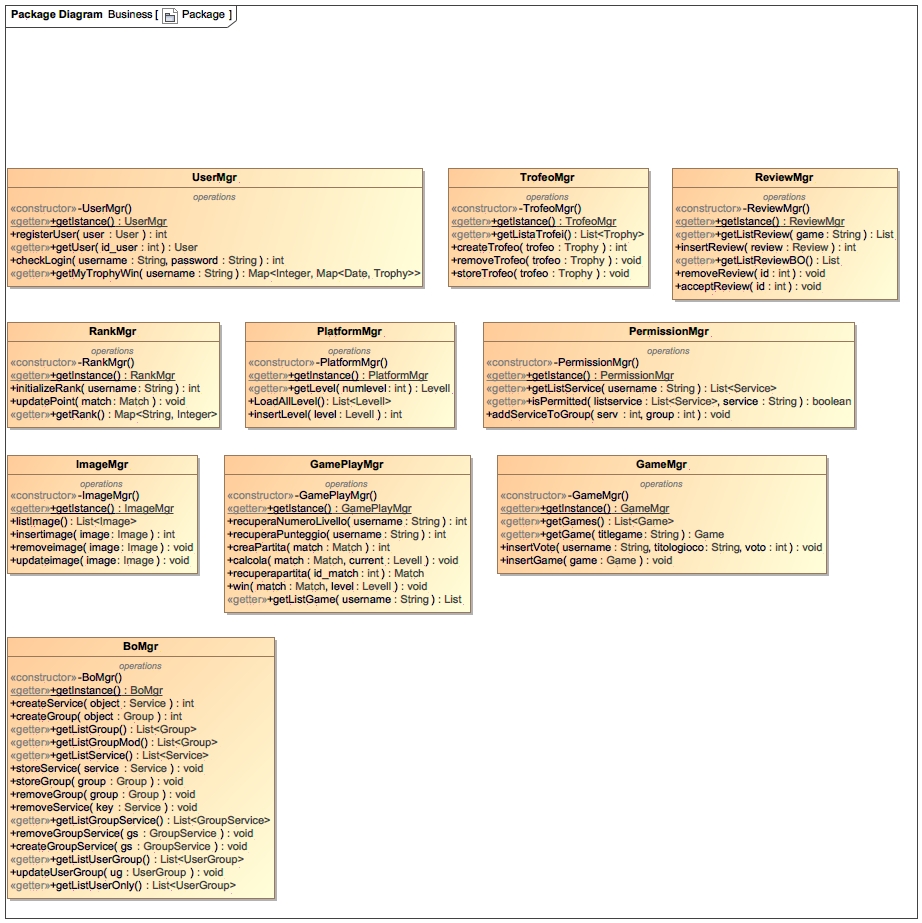
-In ControllerBO troviamo:

BO, BOgroup, BOgroupeservice, BOlivello, BOreview, BOservizi, BOgame, BOimmagine, BOtrofeo, BOuserpermission.

Qui ogni classe è adibita alle operazioni CRUD del relativo modello.

Tutte le classi dei due package estendono una classe astratta BaseServlet (che a sua volta estende HttpServlet), adibita alle funzioni di login.

1. **Business**



Questo package contiene le seguenti classi:

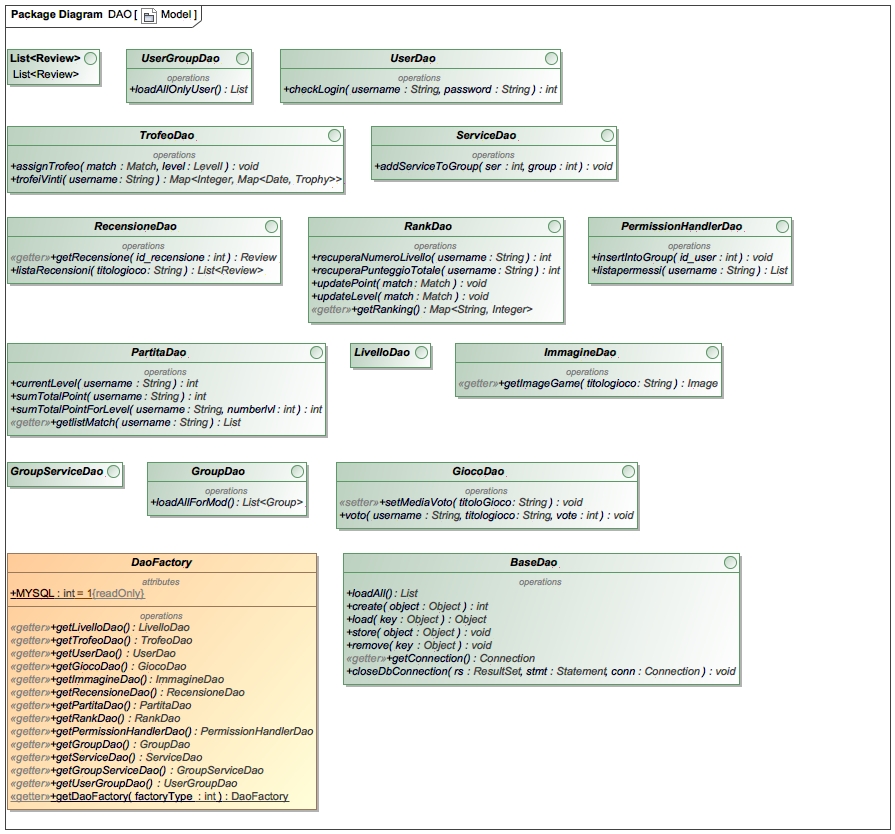
-BOMgr, GameMgr, GamePlayMgr, ImageMgr, PermissionMgr, PlatformMgr, RankMgr, ReviewMgr, TrofeoMgr, UserMgr. Ogni classe di questo package è cosi strutturata:

ha un solo campo privato *dbfactory* di tipo DAOfactory. Un costruttore che inizializza tale campo e un metodo statico *getIstance* che ritorna un istanza della classe.

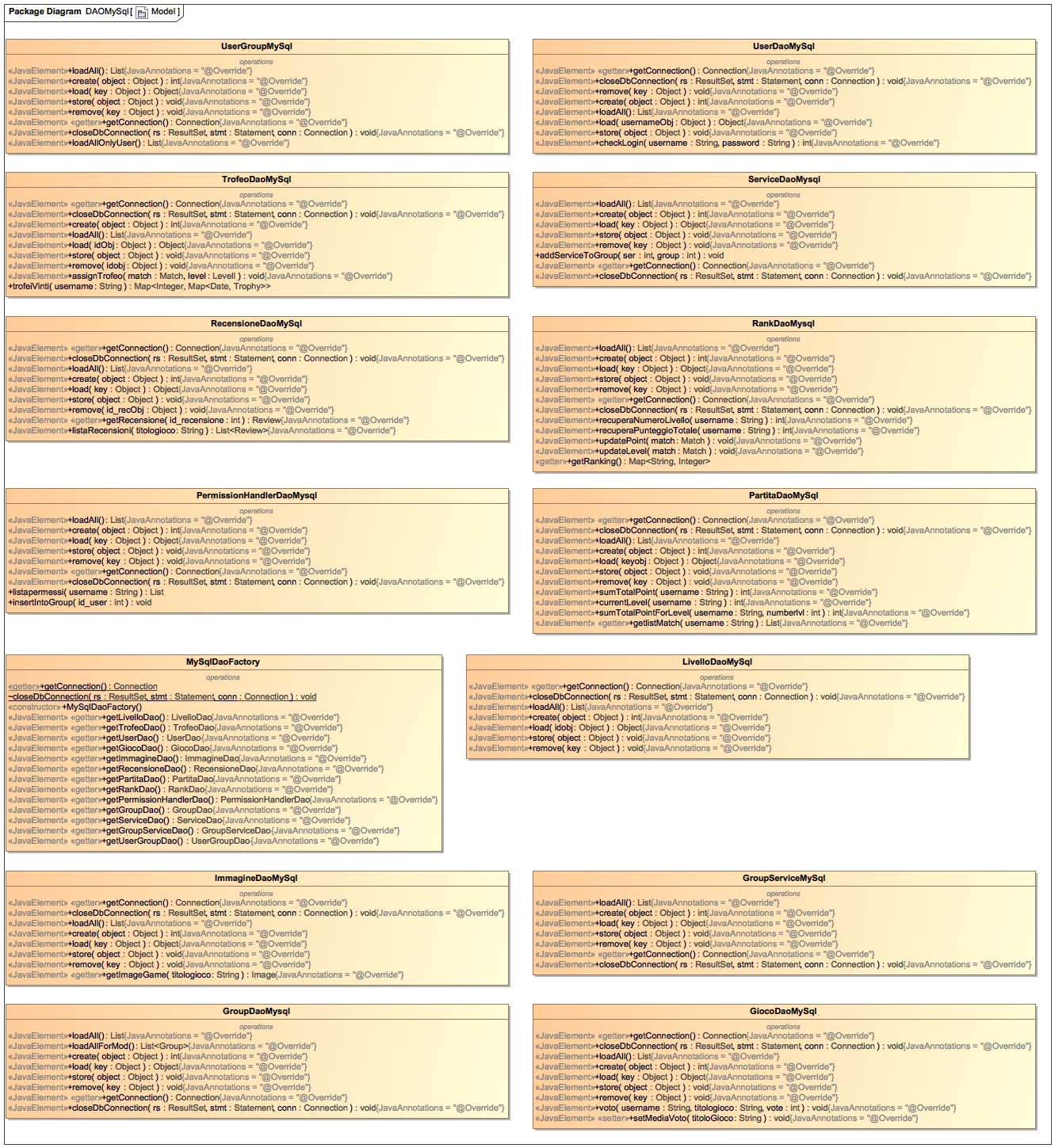
Ogni classe di questo package ha metodi relativi a metodi della relativa classe del package DAO.

1. **DAO**

* **DAO**

****

* **DAOMySql**

******

Questo package è stato suddiviso in due sottopackage: DAOInterface e DAOMySql.

DAOinterface è composta le interfacce del DAO, ognuna delle quali è implementata da una particolare classe del sottopackage DAOMySql e ne definisce in astratto la struttura ed il comportamento.

I metodi comuni ad ogni interfaccia hanno nomi autoesplicativi e sono i seguenti:

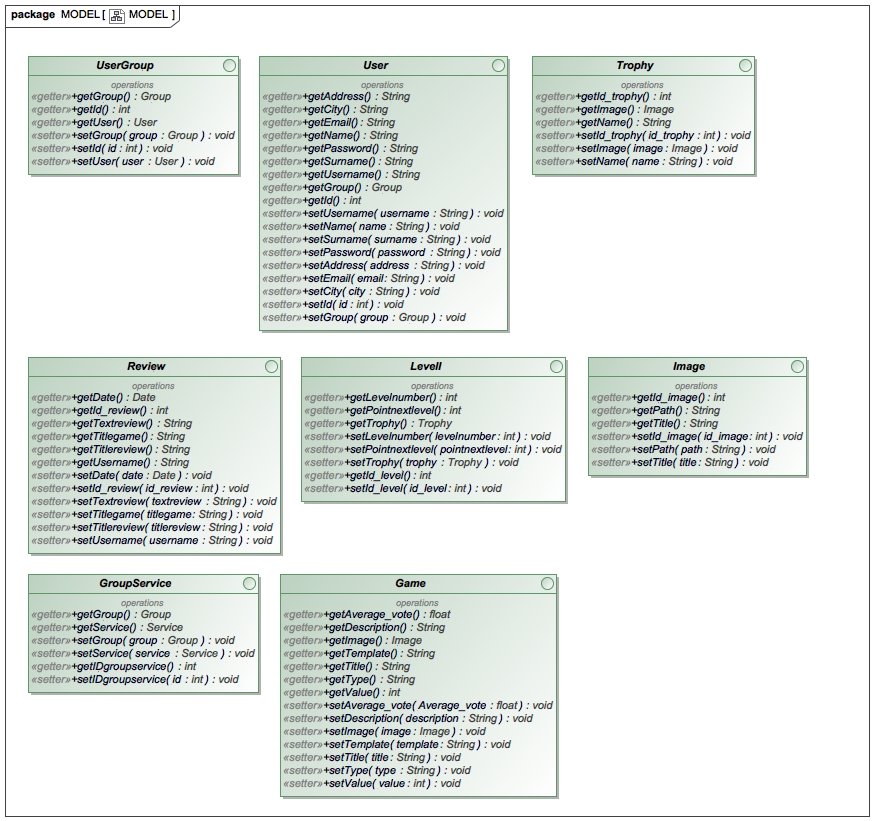
loadAll(), create(), load(), store(), remove(), getConnection(), closeDbConnection()

oltre ai metodi appena elencati, sono presenti all'interno di ogni interfaccia, metodi ad hoc per la gestione delle singole entità nel database. Infine la classe astratta DAOFactory contiene tutti i metodi astratti che ritornano un interfaccia per ogni entità del package Model.

Il sottopackage DAOMySql infine è un implementazione concreta della classe DAOfactory.

1. **Model**

* **Model**

****

* **ModelImpl**

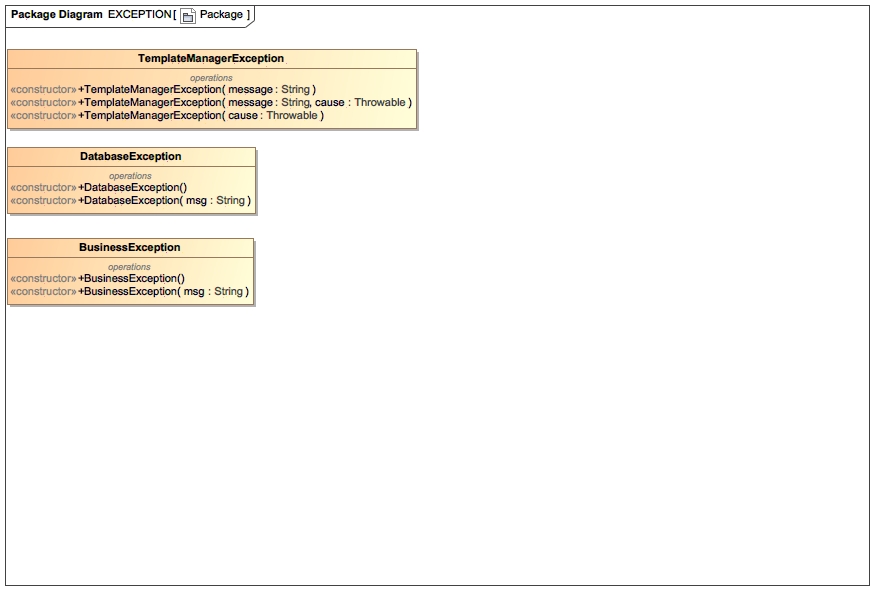
****

Questo package è stato suddiviso in: Model e ModelImpl.

Ogni classe del package Model contiene le interfacce con i metodi get() e set().

Mentre ModelImpl è il package che contiene le classi concrete. Ogni classe corrisponde ad una tabella nel database.

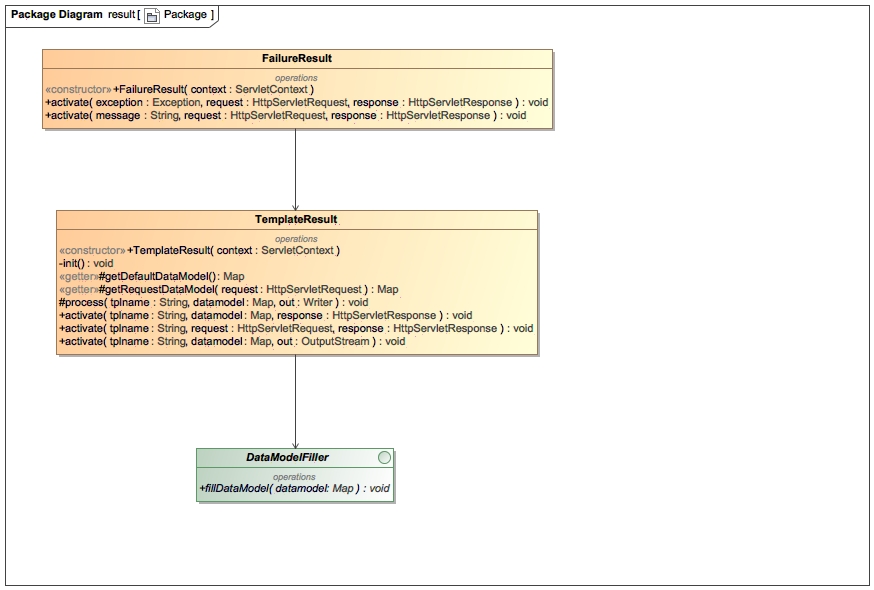
1. **Exception**



Contiene tre classi che estendono Exception, denominate: BusinessException, DataBaseException e TemplateManagerException. Esse hanno il compito di gestire le eccezioni del relativo package.

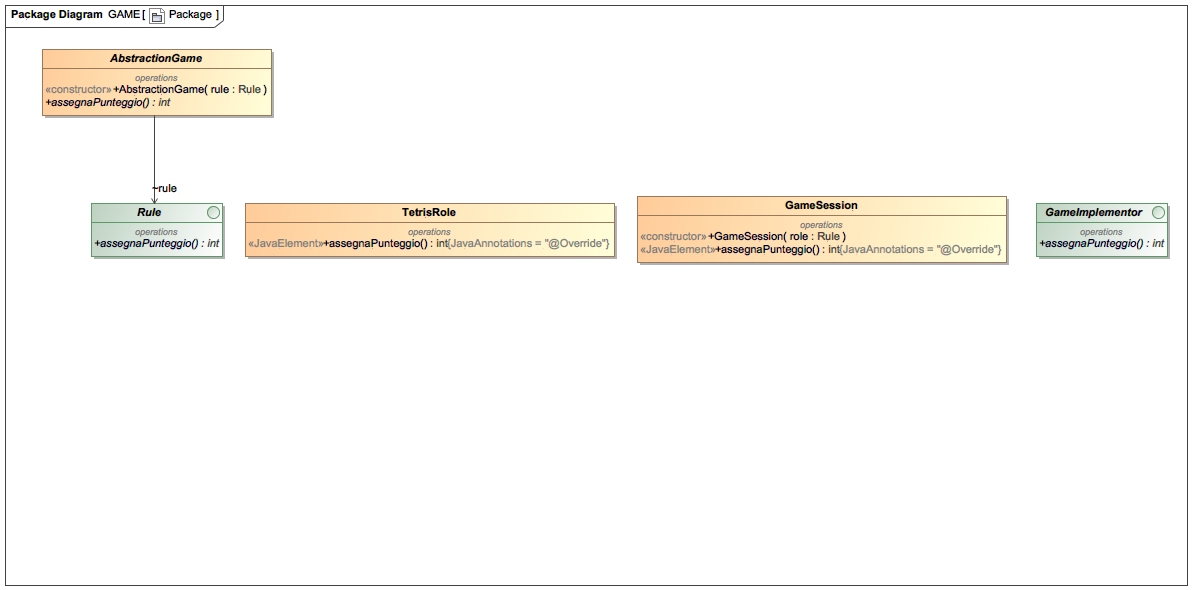
BusinessException per le eccezioni di Business, DataBaseException per le eccezioni del database e infine TemplateManagerException per le eccezioni del package result.

1. **Result**



Package contenente le classi per la visualizzazione del template in output.

1. **Game**



Questo package contiene le classi che specificano le regole di assegnazione dell'esperienza dei giochi.

***C.4 Descrizione dettagli design scelti***

**1) AbstractFactory**

Il pattern AbstractFactory può essere utlizzato quando:

* un sistema deve essere indipendente dalle modalità di creazione, composizione e rappresentazione dei suoi prodotti.
* un sistema deve poter essere configurato scegliendo una tra più famiglie di prodotti
* esistono famiglie di oggetti correlati, progettati per essere utilizzati inisieme, e occorre gestire che questo vincolo sia rispettato
* si vuole fornire una libreria di classi, rivelando soltanto le loro interfacce e non le implementazioni.

L'uso di questo pattern ha i seguenti vantaggi:

* Isola le classi concrete
* Consente di cambiare in modo semplice la famiglia di prodotti utilizzata
* promuove la coerenza nell'utilizzo dei prodotti
* L'aggiunta del supporto per nuove tipologie di prodotti è difficile

**2) Bridge**

L'uso di questo pattern ha le seguenti conseguenze:

* Disaccoppiamento tra interfaccia e implementazione
* Maggiore estendibilità
* Mascheramento dei dettagli dell'implementazione ai Client