Project Proposal

Team Page 404

Prepared for: Romina Eramo

Prepared by: Team “Page 404”

21/04/2017

Membri del team

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

# INDICE

[A. REQUIREMENTS 3](#_Toc480471336)

[A.1 Functional Requirements 3](#_Toc480471337)

[A.1.1 GUI Requirements 3](#_Toc480471338)

[A.1.2 DB Requirements 3](#_Toc480471339)

[A.1.3 Business Logic Requirements 3](#_Toc480471340)

[A.2 Non Functional Requirements 4](#_Toc480471341)

[A.2.1 Efficiency NFR 4](#_Toc480471342)

[A.2.2 Security 4](#_Toc480471343)

[A.2.3 Usability 4](#_Toc480471344)

[A.2.4 Technology Requirements 4](#_Toc480471345)

[A.3 Use Cases 5](#_Toc480471346)

[A.4 Scenari 15](#_Toc480471347)

[B. System Design 16](#_Toc480471348)

[B.1 Component Diagram 16](#_Toc480471349)

[B.2 Design Decision 17](#_Toc480471350)

[B.2.1 Web Based 17](#_Toc480471351)

[B.2.2 Giochi 17](#_Toc480471352)

[B.2.3 Livello 18](#_Toc480471353)

[B.2.4 Trofei 18](#_Toc480471354)

[B.2.5 Timeline 18](#_Toc480471355)

[B.2.6 Technology 18](#_Toc480471356)

[B.3 Design Pattern 20](#_Toc480471357)

[B.4 E-R 21](#_Toc480471358)

[C. SOFTWARE/ OBJECT Design 23](#_Toc480471359)

[C.1 Class Diagram 23](#_Toc480471360)

[C.2 Sitemap 24](#_Toc480471361)

A. REQUIREMENTS

A.1 Functional Requirements

### A.1.1 GUI Requirements

* Sarà la GUI a permetterà l’interazione tra l’utente e le funzionalità del sistema (registrazione, autenticazione, visualizzazione del profilo, effettuare una sessione di gioco, recensirlo e votarlo).
* Attraverso la GUI l’utente potrà visualizzare e modificare le proprie informazioni anagrafiche.
* 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.
* Attraverso la GUI il moderatore e l’amministratore potranno accedere al rispettivo back office.

### A.1.2 DB Requirements

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

### A.1.3 Business Logic Requirements

* Il Sistema potrà gestire una molteplicità di giochi.
* Il Sistema dovrà gestire diversi tipi di utenza:
* Amministratore
* Moderatore
* Utente base
* Il Sistema prevedrà l’assegnazione di un trofeo, ad un utente, per ogni livello acquisito.
* Il Sistema dovrà prevedere un range prefissato per l’assegnazione del voto relativo ad un gioco.
* Il Sistema dovrà assicurare che il moderatore possa verificare le recensioni prima della loro pubblicazione.
* 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.
* 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.
* 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

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

### A.2.2 Security

* Il committente non ha posto alcun vincolo riguardo la sicurezza del sistema, ma noi implementeremo ugualmente una classe che si occuperà di gestire eccezioni di input nei form al fine di garantire un livello minimo di sicurezza per il committente e gli utilizzatori del sistema.

### A.2.3 Usability

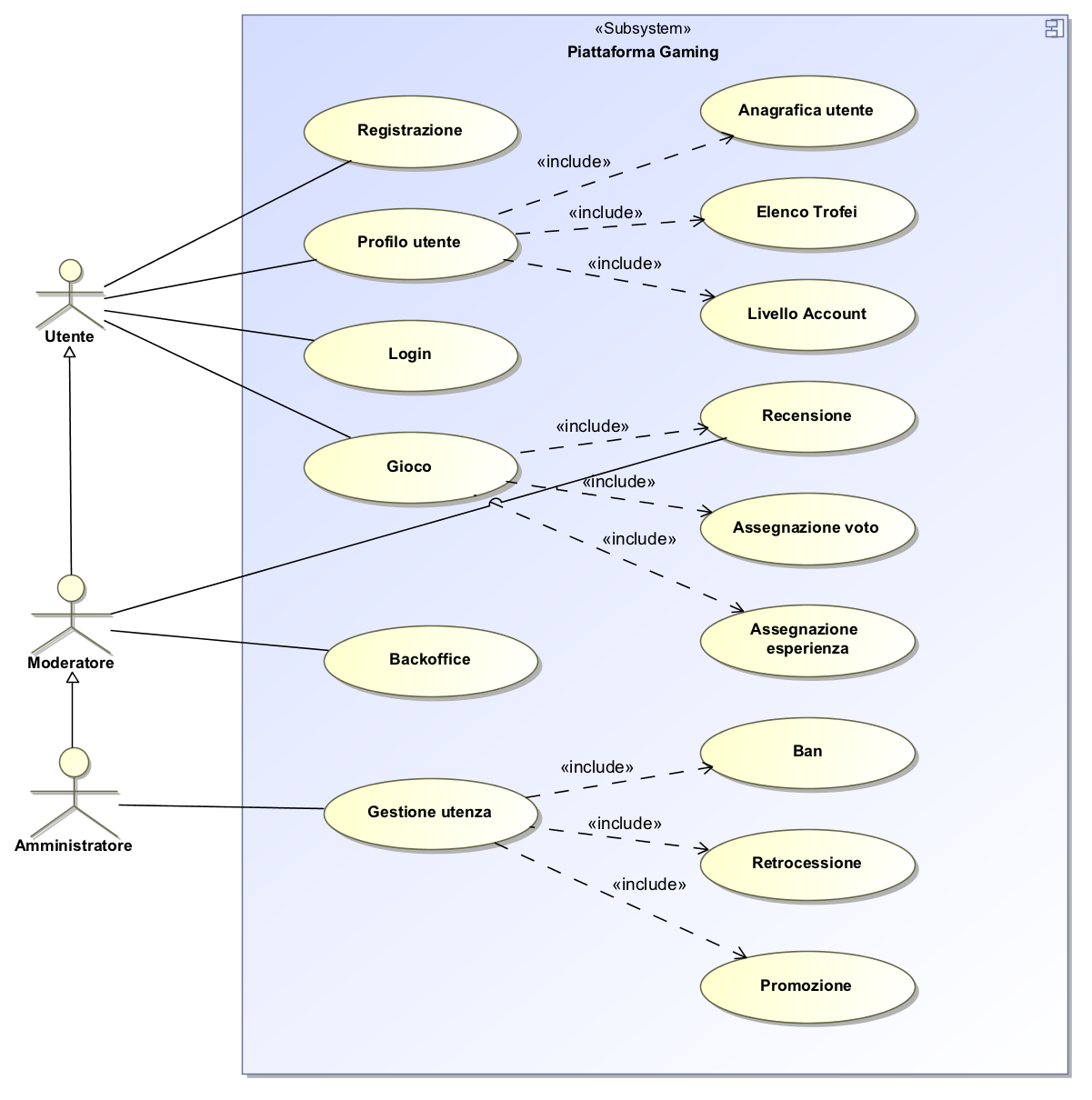
* La GUI dovrà essere user-friendly al fine di facilitarne e invogliarne l’utilizzo da parte dell’utente.

### A.2.4 Technology Requirements

* Il committente non ha posto alcun tipo di vincolo tranne che per il linguaggio di programmazione: dovrà essere “Java”.

A.3 Use Cases

Di seguito viene riportato il diagramma:



|  |  |  |
| --- | --- | --- |
| **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 “gestione 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” |

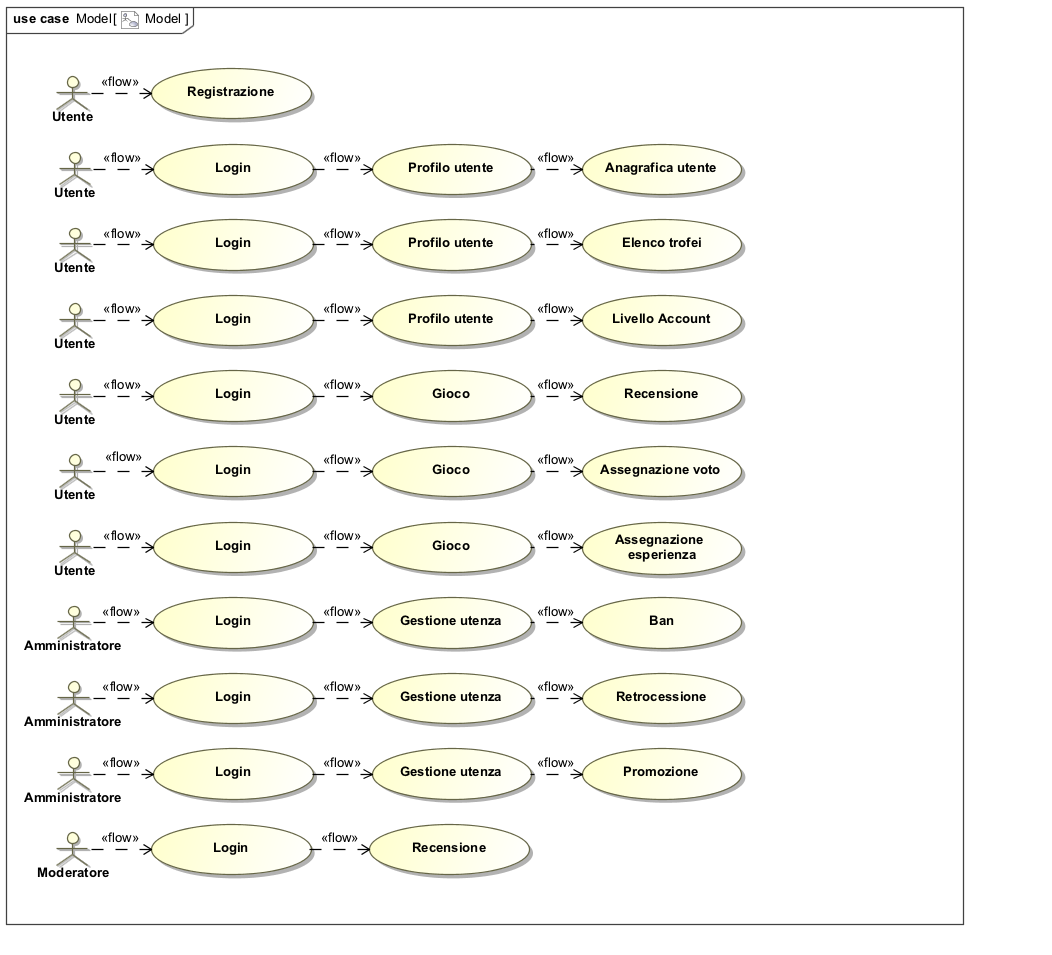
|  |  |
| --- | --- |
| **RELATED INFORMATION** | Gioco |
| **Priority:** | Alta |
| **Performance** | Non espressamente dichiarate |
| **Frequency** | Non espressamente dichiarata |
| **Channels to actors** | GUI, Sistema esterno |
| **OPEN ISSUES** | L’utente non può avviare la sessione di gioco |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

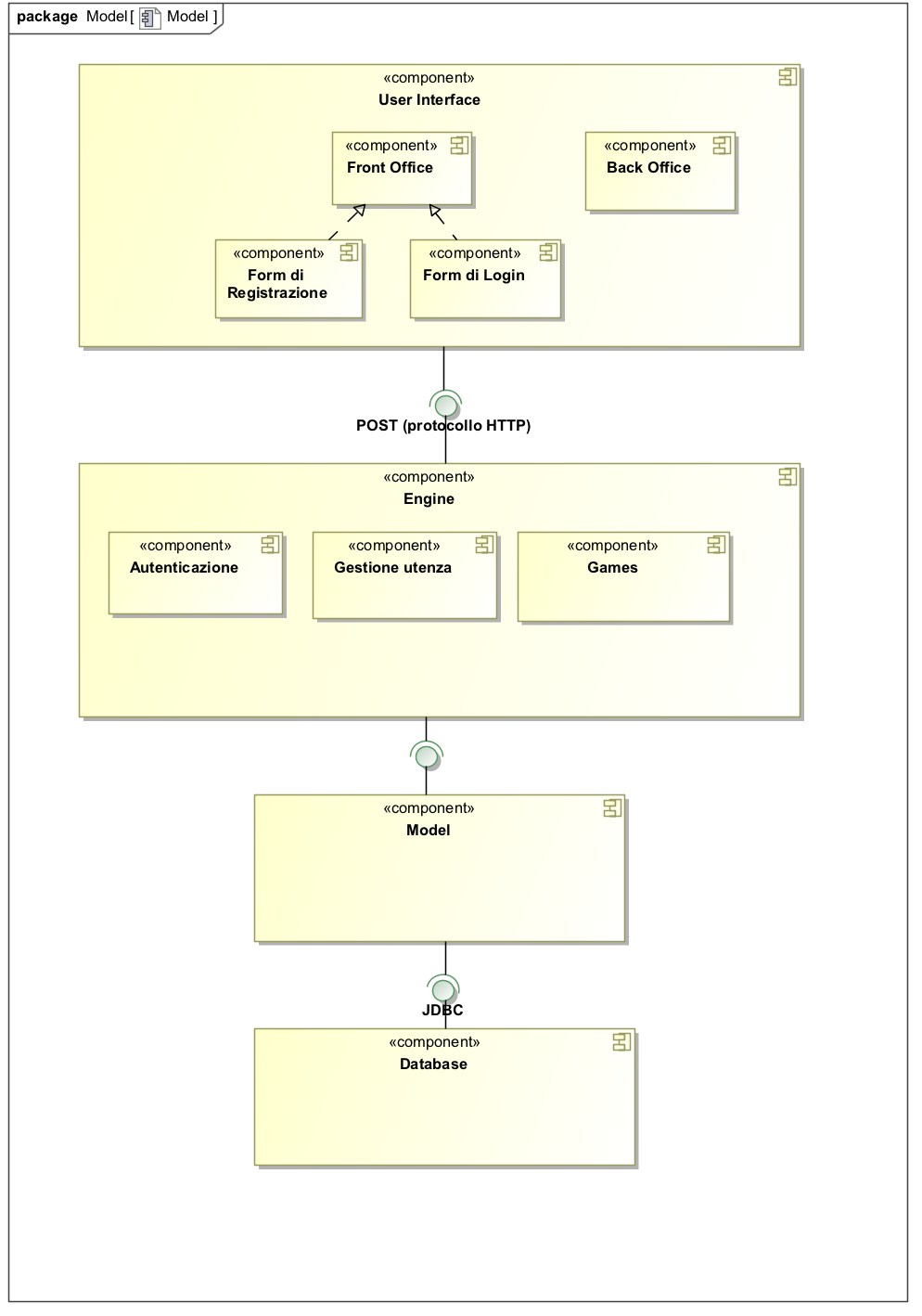
A.4 Scenari

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:

* User Interface: 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.

* Engine: implementa tutte le funzionalità del sistema e manipola i dati ricevuti dall’User Interface interfacciandosi a sua volta con il Model. È divisa in tre macro-componenti: Autenticazione, Gestione utenza e Games.

La prima si occupa della gestione dell’utenza attraverso il controllo della veridicità dell’identità di un utente. La seconda gestisce i permessi degli utenti (e i relativi servizi disponibili) e le varie operazioni di gestione utenza. La terza rappresenta l’esperienza di gioco completa. (vedi A.1.3)

* Model: si occupa di gestire l’accesso e l’immagazzinamento dei dati presenti nel Db. Mapperemo le tabelle del Database in classi java: in questo modo abbiamo implementato una persistenza dei dati.
* 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 standard JDBC (Java Database Connectivity).

B.2 Design Decision

### B.2.1 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.2.2 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.

### B.2.3 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 sequenza di Fibonacci per calcolare il livello totale: *Fibonacci di n* corrisponderà all’esperienza totale necessaria per il raggiungimento del livello n.

### B.2.4 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.2.5 Timeline

* Il sistema prevedrà una Timeline che mostrerà la cronologia dei livelli acquisiti dall’account dell’utente. Tale scelta è stata dettata dal fatto che riteniamo opportuno mantenere traccia solo dei progressi generali e non di ogni singolo gioco.

### B.2.6 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.

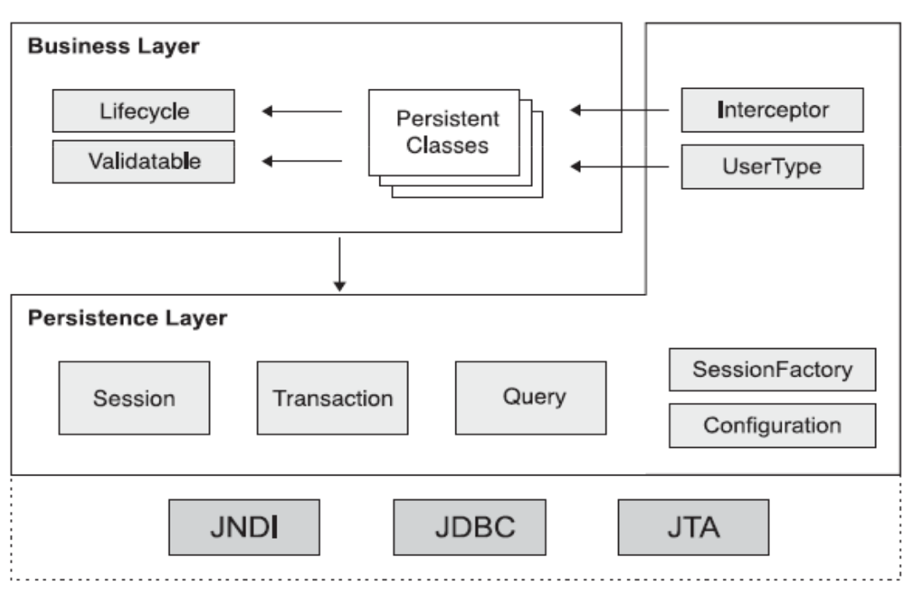
Utilizzeremo il database MySql, in quanto il team ha sufficienti conoscenze pregresse a riguardo. Per tal motivo verrà utilizzata l’interfaccia JDBC, la quale conterrà MySql Connector, che permetterà la connessione con il nostro database.

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:

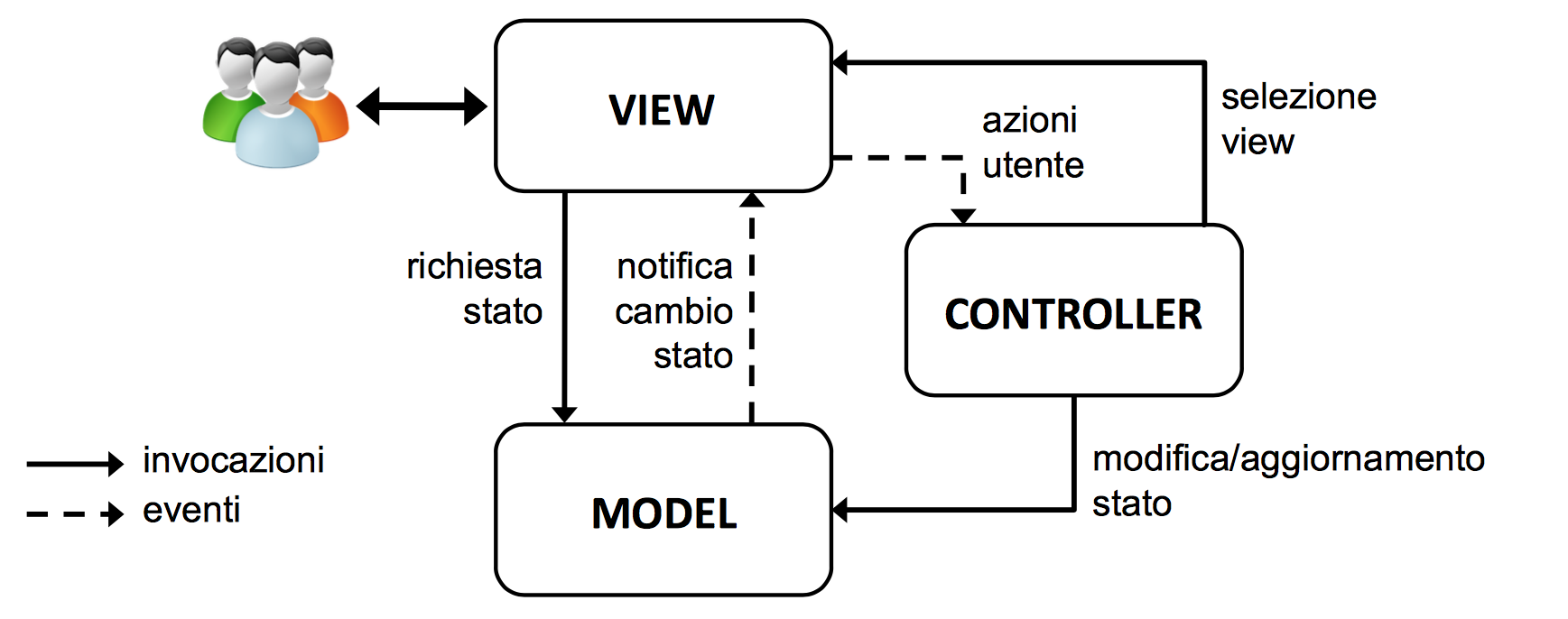


Il team, dopo un attento studio, ha deciso di implementare le Jsp (Java Server Pages), ovvero una tecnologia di programmazione Web in Java la quale ci permetterà di sviluppare contenuti dinamici in HTML5, visto che si interfacciano in maniera ottimale con Tomcat e Java. Sarà inoltre possibile, grazie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

B.3 Design Pattern

Il design pattern 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.4 E-R

# ../../../Downloads/18035616_10213359008036100_672026958_n.jpg

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 e “Exp Totale” per calcolare il livello totale dell’account, ottenuto sommando l’esperienza dei vari giochi cui l’utente ha giocato, applicando la sequenza di Fibonacci (vedi B.2.3).

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

Il secondo, 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’ “Obbiettivo”, 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.

“Recensione” è l’entità 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”.

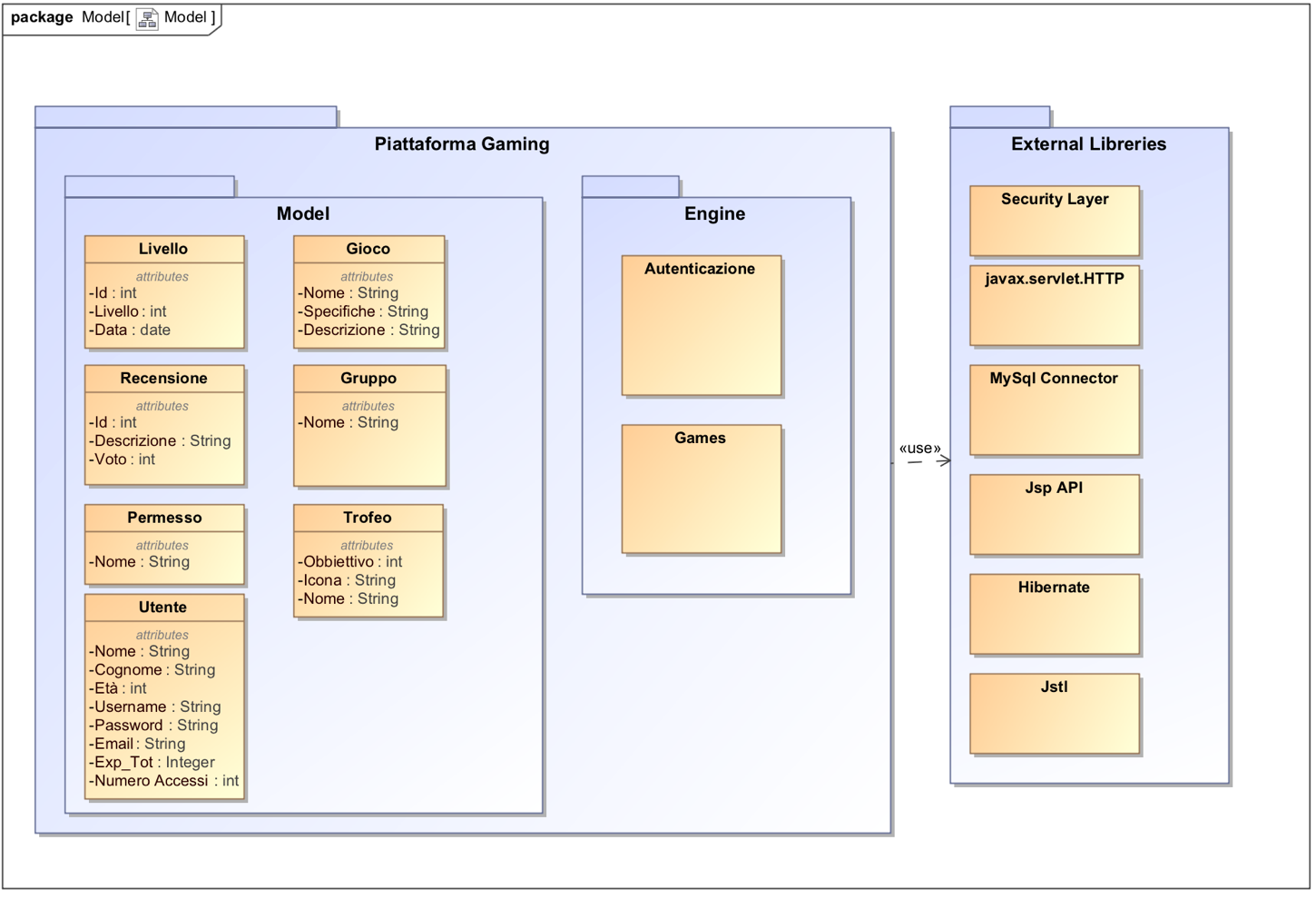
L’unica relazione rilevante del nostro sistema è “Giocare”, la quale serve a tenere traccia dei dati delle sessioni di gioco dell’utente tramite tre attributi: “Numero accessi”, “Ore” ed “Exp”.

I primi due serviranno per l’assegnazione di determinati trofei elencati (vedi B.2.4), mentre l’ultimo servirà per calcolare il livello relativo ad un gioco e all’utente che ci gioca.

Infine “Permesso” è l’entità in cui vengono salvati i file del sistema nell’apposito attributo “Nome”.

C. SOFTWARE/ OBJECT Design

C.1 Class Diagram



Abbiamo suddiviso il nostro sistema di “Piattaforma Gaming” in due sotto-package “Model” e “Engine”.

All’interno del primo abbiamo fondamentalmente i dati del database espressi come classi (per una documentazione più dettagliata delle classi intese come entità vedi B.4), mentre nel secondo è presente la classe che gestirà l’ “Autenticazione” e la classe che gestirà l’esperienza di gioco (classe “Games”).

Il nostro sistema farà uso di librerie esterne quali: un “Security Layer” per garantire un livello di sicurezza minimo (vedi B.2.2), “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).

C.2 Sitemap

Ogni “vista” rappresenterà una pagina Jsp:

