**Warrior’s Adventure – Object Design**

**1. Introduzione**

**1.1 Object Design trade-off:**

* **[Prestazioni vs Costi]:** Visto le scarse risorse finanziarie, le prestazioni punteranno ad offrire un esperienza di gioco accettabile, accettando qualche compromesso lato automatismi aggiornamenti senza riavviare la pagine di gioco.
* **[Interfaccia vs Easy-use]:** l’interfaccia verrà progettata in modo da risultare il più semplice e leggibile possibile, i contrasti saranno elevati in modo da semplificarne la lettura e le pagine saranno in numero ridotto in modo da non creare un senso di disorientamento nel giocatore. Per quanto riguarda la responsività dell’interfaccia si è scelto di adattarla unicamente a sistemi desktop per non andare contro ai requisiti precedentemente espressi.
* **[Interfaccia vs Tempo di risposta]:** L’interfaccia manterrà tempi di risposta più bassi possibili, in modo da permettere ai giocatori un esperienza d’uso ottimale compatibilmente alle risorse hardware della macchina che fungerà da server.
* **[Costi vs Mantenimento]:** Grazie all’utilizzo di Java EE il progetto sarà diviso in una componente client e una componente server questo permetterà un ottima scalabilità permettendo al progetto di girare su un'unica macchina server. Per quanto riguarda modifiche e aggiornamenti al codice, insieme al documento verrà rilasciato anche un java doc.

**1.2 Linee guida per l’interfaccia:**

Il Sistema è multi-utente (possono accedervi sia utenti comuni che amministratori, moderatori, gestori di gilda o gestori di torneo), per tale motivo il sistema nasconderà o mostrare funzionalità in base all’utente attualmente connesso.

**1.3 Componenti off-the-shell**

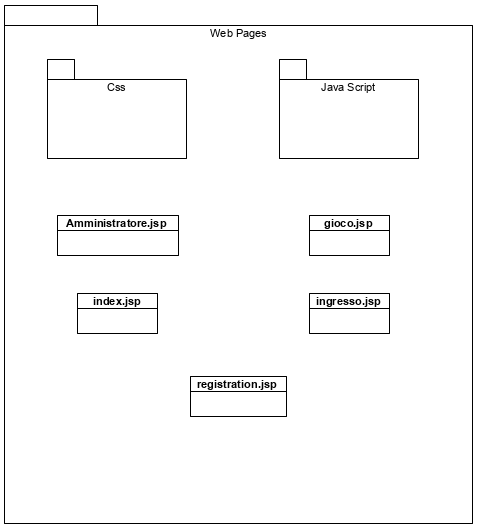
Componenti che andremmo a utilizzare sono:

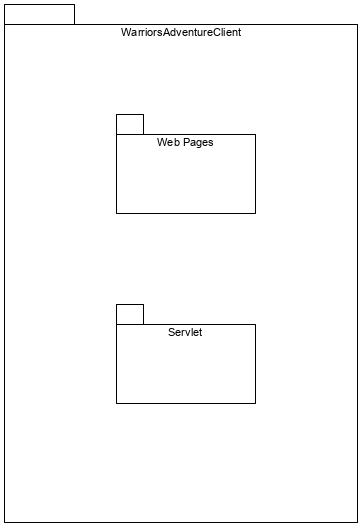
* Libreria open-source GSON distribuita da Google per linguaggio Java.
* Libreria jQuery per linguaggio Java Script.
* Libreria Open- Source AJAX per garantire aggiornamenti asincroni tra client e server.

**1.4 Definizioni, acronimi e abbreviazioni**

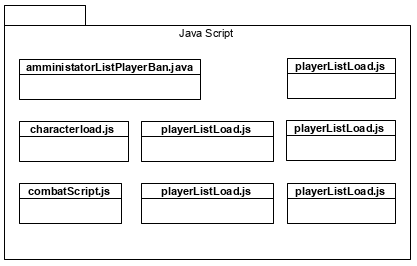
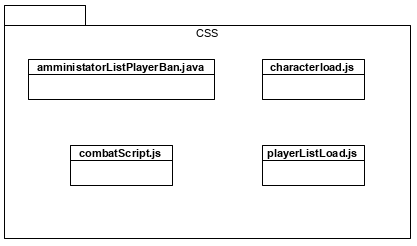
* DB = Database;
* DERBY = Apache Derby;
* RAD = Requirement Analysis Document;
* SDD = System Design Document;
* ODD = Object Design Document;
* PvP = Player vs Player;
* PvE = Player vs Environment;
* Player/s = Giocatore/I;
* JDBC = Java Data Base Connectivity;
* HTML = Linguaggio di Mark-Up per pagine web;
* CSS = Cascate Style Sheets;
* Amministratore: utente che si occupa di bannare e rimuovere e promuovere i giocatori all’interno del sito;

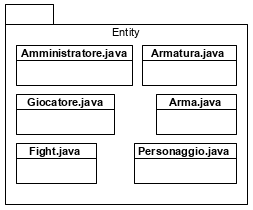
**2. Package**

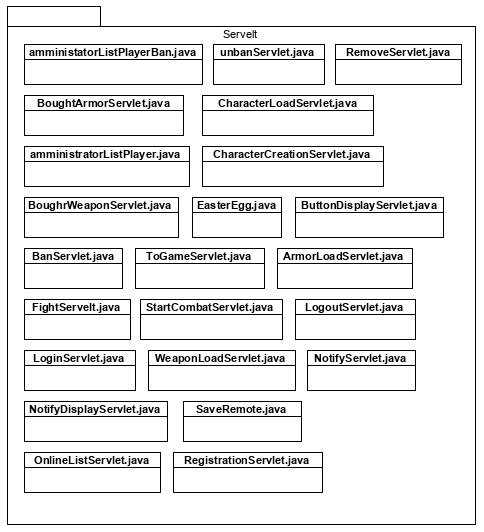
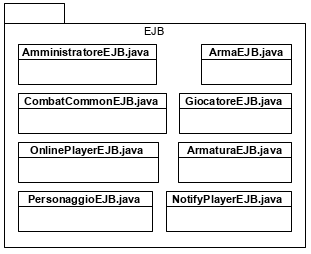


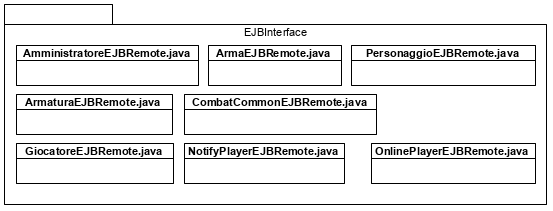
**2.1 Package Generale:**

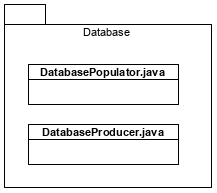
**2.2** **Package CSS e Java Script:**



**2.3 Package Servlet: 2.5 Package EJB:**

****

**2.5 Package Entity, Database e EJBInterface:**

****

**3. Interfaccia delle classi**

**3.1 Entity:**

**3.1.1 Amministratore**

|  |  |
| --- | --- |
| **Nome entità:** | **Amministratore** |
| **Attributi:** | * Long **id;** * String **username;** * String **password;** |
| **Metodi:** | * getUsername(); * setUsername(String username); * getPassword(); * setPassword(String password); * toString(); |

**3.1.2 Arma**

|  |  |
| --- | --- |
| **Nome entità:** | **Arma** |
| **Attributi:** | * Long **id;** * String **name;** * Int **forza;** * int **livello;** * int **prezzo;** * String **FIND\_WEAPON="Arma.Find\_Weapon";** * String **DELETE\_WEAPON = "Arma.Delete\_Weapon";** * String **BOUGHT\_WEAPON = "Arma.Bought\_Weapon";** |
| **Metodi:** | * getId():long * setId(Long id):void * getName():String * setName(String name) :void * getForza():int * setForza(int forza) :void * getLivello():int * setLivello(int livello) :void * getPrezzo():int * setPrezzo(int prezzo) :void * toString():String |

**3.1.3 Armatura**

|  |  |
| --- | --- |
| **Nome entità:** | **Armatura** |
| **Attributi:** | * Long **id;** * String **name;** * Int **forza;** * int **vitalità;** * int **difesa;** * int **livello;** * int **prezzo;** * String **FIND\_ARMOR="Armatura.FIND\_Armor"** * String **BOUGHT\_ARMOR = "Armatuta.Bought\_Armor";** * String **DELETE\_ARMOR = "Armatura.Delete\_Armor";** |
| **Metodi:** | * getId():long * setId(Long id) :void * getName():String * setName(String name) :void * getForza():int * setForza(int forza) :void * getVitalità():int * setVitalità(int vitalità) :void * getDifesa():int * setDifesa(int difesa) :void * getLivello():int * setLivello(int livello) :void * getPrezzo():int * setPrezzo(int prezzo) :void * toString():String |

**3.1.4 Giocatore**

|  |  |
| --- | --- |
| **Nome entità:** | **Giocatore** |
| **Attributi:** | * Long **id;** * String **username;** * String **email;** * String **password;** * boolean **ban;** * String **FIND\_ALL = "Giocatore.FindAllGiocatore";** * String **FIND\_BYUSERNAME = "Giocatore.findbyUsername";** * String **FIND\_BYUSERPW = "Giocatore.findbyUserpw";** * String **FIND\_BYEMAIL = "Giocatore.findBbyEmail";** * String **FIND\_BYUSEM = "Giocatore.findbyUSEM";** * String **FIND\_ALL\_BAN = "Giocatore.findbyBan";** * String **REMOVE\_PLAYER = "Giocatore.RemovePlayer";** |
| **Metodi:** | * getUsername():String * getPassword():String * getEmail():String * setUsername(String username) :void * setPassword(String password) :void * setEmail(String email) :void * isBan():boolean * setBan(boolean ban) :void * toString():String |

**3.1.5 Personaggio**

|  |  |
| --- | --- |
| **Nome entità:** | **Personaggio** |
| **Attributi:** | * long **id**; * String **name**; * String **classe**; * int **livello**; * int **vitality**; * int **defense**; * int **strenght**; * int **gold**; * String **guild**; * int **exp**; * int **remainexp**; * String **username**; * String **arma**; * String **armatura**; * int **base\_vit**; * int **base\_def**; * int **base\_str**; * int **armorStr**; * int **armorVit**; * int **armorDef**; * int **weaponStr**; * int **armorLv**; * int **weaponLv**; * long **combat\_key**; * String **FIND\_BYUS = "Personaggio.findByUsername";** * String **FIND\_BYID = "Personaggio.fidnByID";** * String **FIND\_BYUSNA = "Personaggio.findByUsernameName";** * String **SAVE\_BYID = "Personaggio.Save\_by\_id";** |
| **Metodi:** | * getRemainexp():int * getName():String * getClasse():String * getLivello():int * getVitality():int * getDefense():int * getStrenght():int * getGold():int * getGuild():String * getExp():int * setRemainexp(int remainexp):void * setName(String name):void * setClasse(String classe):void * setLivello(int level):void * setVitality(int vitality):void * setDefense(int defense):void * setStrenght(int strenght):void * setGold(int gold):void * setGuild(String guild):void * setExp(int exp):void * getArma():String * setArma(String arma):void * getArmatura():String * setArmatura(String armatura):void * getBase\_vit():int * setBase\_vit(int base\_vit):void * getBase\_def():int * setBase\_def(int base\_def):void * getBase\_str():int * setBase\_str(int base\_str):void * getArmorStr():int * setArmorStr(int armorStr):void * getWeaponStr():int * setWeaponStr(int weaponStr):void * getArmorLv():int * setArmorLv(int armorLv):void * getWeaponLv():int * setWeaponLv(int weaponLv):void * getArmorVit():int * setArmorVit(int armorVit):void * getArmorDef():int * setArmorDef(int armorDef):void * getId():long * setId(long id):void * getCombat\_key():long * setCombat\_key(long combat\_key):void |

**3.1.6 Fight**

|  |  |
| --- | --- |
| **Nome entità:** | **Fight** |
| **Attributi:** | * Personaggio **a;** * Personaggio **b;** * Random **ram;** * boolean **dif\_a**; * boolean **dif\_b**: |
| **Metodi:** | attacca(Personaggio attack):String  difesa(Personaggio defense):String  cura(Personaggio cura):String  fuga(Personaggio fuga):String  verifyWinner():Long |

**3.1.6.1 Metodo attacca**

|  |  |
| --- | --- |
| **Nome metodo:** | **Attacca** |
| **Descrizione:** | Esegue le operazioni di attacco durante un combattimento. |
| **Precondizione:** | **Context:** Fight::attacca (Personaggio attack); |
| **Postcondizione:** | **Context:** Fight::attacca (Personaggio attack);  **post:** se ha attaccato A/B le statistiche di B/A vengono aggiornate. |

**3.1.6.2 Metodo difesa**

|  |  |
| --- | --- |
| **Nome metodo:** | **Difesa** |
| **Descrizione:** | Esegue le operazioni di difesa durante un combattimento. |
| **Precondizione:** | **Context:** Fight::difesa(Personaggio defense);  **Pre:** attack!= null |
| **Postcondizione:** | **Context:** Fight::difesa(Personaggio defense);  **post:** se A/B si è difeso le statistiche di B/A vengono aggiornate. |

**3.1.6.3 Metodo cura**

|  |  |
| --- | --- |
| **Nome metodo:** | **Cura** |
| **Descrizione:** | Esegue le operazioni di cura durante un combattimento. |
| **Precondizione:** | **Context:** Fight::cura(Personaggio cura);  **Pre:** attack!= null |
| **Postcondizione:** | **Context:** Fight::cura(Personaggio cura);  **post:** se A/B si è curato le statistiche di B/A vengono aggiornate. |

**3.1.6.4 Metodo fuga**

|  |  |
| --- | --- |
| **Nome metodo:** | **Fuga** |
| **Descrizione:** | Esegue le operazioni di fuga durante un combattimento. |
| **Precondizione:** | **Context:** Fight::fuga(Personaggio fuga);  **Pre:** attack!= null |
| **Postcondizione:** | **Context:** Fight::fuga(Personaggio fuga);  **post:** se A/B è fuggito, B/A viene proclamato giocatore. |

**3.1.6.5 Metodo verifyWinner**

|  |  |
| --- | --- |
| **Nome metodo:** | **verifyWinner** |
| **Descrizione:** | Verifica se c’è stato un vincitore all’interno del torneo. |
| **Precondizione:** | **Context:** Fight:: verifyWinner();  **Pre:** attack!= null |
| **Postcondizione:** | **Context:** Fight:: verifyWinner();  **post:** se la vitalità di A è minore di zero, return a.getId();  se la vitalità di B è minore di zero, return b.getId(); |

**3.2 Enterprise Java Beans:**

**3.2.1 AmministratoreEJB**

|  |  |
| --- | --- |
| **Nome entità:** | **AmministratoreEJB** |
| **Descrizione** | EJB che si occupa della gestione di tutte le operazioni che riguardano l’amministrazione del sito |
| **Metodi:** | * logginAmministratore(String username, String password):Amministratore * insertAmministratore(Amministratore am):Amministratore * updateAmministratore(Amministratore am):Amministratore * removeAmministratore(Amministratore am):void * allPlayers():List<Giocatore> * BannedPlayers():List<Giocatore> * banPlayer(String username)String * unbanPlayer(String username)String |

**3.2.1.1 Metodo logginAmministratore**

|  |  |
| --- | --- |
| **Nome metodo:** | **logginAmministratore** |
| **Descrizione:** | Presi username e password verifica se l’utente è un amministratore all’interno del DB. |
| **Precondizione:** | **Context:**AmministratoreEJB::logginAmministratore(String username, String password) |
| **Postcondizione:** | **Context:** AmministratoreEJB::logginAmministratore(String username, String password)  **post:** se la coppia username, password all’interno del DB è annotata come amministratore, viene restituita l’entità amministratore, altrimenti viene restituito null. |

**3.2.1.2 Metodo insertAmministratore**

|  |  |
| --- | --- |
| **Nome metodo:** | **insertAmministratore** |
| **Descrizione:** | Dato un oggetto amministratore lo rende persistenze e lo ritorna. |
| **Precondizione:** | **Context:** AmministratoreEJB::insertAmministratore(Amministratore am); |
| **Postcondizione:** | **Context:** AmministratoreEJB::insertAmministratore(Amministratore am);  **post:** am viene reso persistente e ritornato al chiamante. |

**3.2.1.3 Metodo updateAmministratore**

|  |  |
| --- | --- |
| **Nome metodo:** | **updateAmministratore** |
| **Descrizione:** | Dato un oggetto amministratore lo rende persistenze e lo ritorna. |
| **Precondizione:** | **Context:** AmministratoreEJB::insertAmministratore(Amministratore am); |
| **Postcondizione:** | **Context:** AmministratoreEJB::insertAmministratore(Amministratore am);  **post:** am viene reso persistente e ritornato al chiamante. |

**3.2.1.4 Metodo removeAmministratore**

|  |  |
| --- | --- |
| **Nome metodo:** | **removeAmministratore** |
| **Descrizione:** | Dato un oggetto amministratore lo rende persistenze e lo ritorna. |
| **Precondizione:** | **Context:** AmministratoreEJB::insertAmministratore(Amministratore am); |
| **Postcondizione:** | **Context:** AmministratoreEJB::insertAmministratore(Amministratore am);  **post:** am viene reso persistente e ritornato al chiamante. |

**3.2.1.5 Metodo BannedPlayers**

|  |  |
| --- | --- |
| **Nome metodo:** | **BannedPlayers** |
| **Descrizione:** | Dato un oggetto amministratore lo rende persistenze e lo ritorna. |
| **Precondizione:** | **Context:** AmministratoreEJB::insertAmministratore(Amministratore am); |
| **Postcondizione:** | **Context:** AmministratoreEJB::insertAmministratore(Amministratore am);  **post:** am viene reso persistente e ritornato al chiamante. |

**3.2.1.6 Metodo allPlayers**

|  |  |
| --- | --- |
| **Nome metodo:** | **allPlayers** |
| **Descrizione:** | Dato un oggetto amministratore lo rende persistenze e lo ritorna. |
| **Precondizione:** | **Context:** AmministratoreEJB::insertAmministratore(Amministratore am); |
| **Postcondizione:** | **Context:** AmministratoreEJB::insertAmministratore(Amministratore am);  **post:** am viene reso persistente e ritornato al chiamante. |

**3.2.1.7 Metodo banPlayer**

|  |  |
| --- | --- |
| **Nome metodo:** | **banPlayer** |
| **Descrizione:** | Dato un oggetto amministratore lo rende persistenze e lo ritorna. |
| **Precondizione:** | **Context:** AmministratoreEJB::insertAmministratore(Amministratore am); |
| **Postcondizione:** | **Context:** AmministratoreEJB::insertAmministratore(Amministratore am);  **post:** am viene reso persistente e ritornato al chiamante. |

**3.2.1.8 Metodo unbanPlayer**

|  |  |
| --- | --- |
| **Nome metodo:** | **unbanPlayer** |
| **Descrizione:** | Dato un oggetto amministratore lo rende persistenze e lo ritorna. |
| **Precondizione:** | **Context:** AmministratoreEJB::insertAmministratore(Amministratore am); |
| **Postcondizione:** | **Context:** AmministratoreEJB::insertAmministratore(Amministratore am);  **post:** am viene reso persistente e ritornato al chiamante. |

**3.2.2 ArmaEJB**

|  |  |
| --- | --- |
| **Nome entità:** | **ArmaEJB** |
| **Descrizione** | EJB che si occupa di tutta la gestione delle armi. |
| **Metodi:** | * findWeapons(Personaggio p): ist<Arma> * findWeapons\_bylv(long id); :Arma * insertWeapon(Arma weapon):Arma * updateWeapon(Arma weapon):Arma * deleteWeapon(Arma weapon):void |

**3.2.2.1 Metodo findWeapons**

|  |  |
| --- | --- |
| **Nome metodo:** | **findWeapons** |
| **Descrizione:** | Restituisce una lista di tutte le armi che un dato personaggio è in grado di acquistare. |
| **Precondizione:** | **Context:** ArmaEJB::findWeapons(Personaggio p);  **Pre:** P!=null; |
| **Postcondizione:** | **Context:** ArmaEJB::findWeapons(Personaggio p);  **post:** Restituisce una lista con le armi che il personaggio è in grado di acquistare. |

**3.2.2.2 Metodo findWeapons\_bylv**

|  |  |
| --- | --- |
| **Nome metodo:** | **findWeapons\_bylv** |
| **Descrizione:** | Dato l’id del arma ritorna l’oggetto arma associato. |
| **Precondizione:** | **Context:** ArmaEJB:: findWeapons\_bylv(long id);  **Pre:** id!=null; |
| **Postcondizione:** | **Context:** ArmaEJB:: findWeapons\_bylv(long id);  **Post:** Ritorna l’oggetto arma; |

**3.2.2.3 Metodo insertWeapon**

|  |  |
| --- | --- |
| **Nome metodo:** | **insertWeapon** |
| **Descrizione:** | Dato un arma la rende persistente. |
| **Precondizione:** | **Context:** ArmaEJB:: insertWeapon(Arma weapon);  **Pre:** weapon!=null; |
| **Postcondizione:** | **Context:** ArmaEJB:: insertWeapon(Arma weapon);  **Post:** Rende l’arma persistente; |

**3.2.2.4 Metodo updateWeapon**

|  |  |
| --- | --- |
| **Nome metodo:** | **updateWeapon** |
| **Descrizione:** | Dato un arma l’aggiorna all’interno del DB; |
| **Precondizione:** | **Context:** ArmaEJB::updateWeapon(Arma weapon);  **Pre:** weapon!=null; |
| **Postcondizione:** | **Context:** ArmaEJB:: updateWeapon(Arma weapon);  **Post:** aggiorna l’arma all’interno del database. |

**3.2.2.5 Metodo deleteWeapon**

|  |  |
| --- | --- |
| **Nome metodo:** | **deleteWeapon** |
| **Descrizione:** | Elimina l’arma dal database; |
| **Precondizione:** | **Context:** ArmaEJB:: delete(Arma weapon);  **Pre:** weapon!=null; |
| **Postcondizione:** | **Context:** ArmaEJB:: delete(Arma weapon);  **post:** cancella l’entità arma dal database |

**3.2.3 ArmaturaEJB**

|  |  |
| --- | --- |
| **Nome entità:** | **ArmaturaEJB** |
| **Descrizione** | EJB che si occupa di tutta la gestione delle armature. |
| **Metodi:** | * findArmor(Personaggio p): ist<Armatura> * findArmor\_bylv(long id); : Armatura * insertArmor(Armatura armor): Armatura * updateArmor(Armatura armor): Armatura * deleteArmor(Armatura armor):void |

**3.2.3.1 Metodo findArmor**

|  |  |
| --- | --- |
| **Nome metodo:** | **findArmor** |
| **Descrizione:** | Restituisce una lista di tutte le armature che un dato personaggio è in grado di acquistare. |
| **Precondizione:** | **Context:** ArmaturaEJB::findArmor(Personaggio p);  **Pre:** P!=null; |
| **Postcondizione:** | **Context:** ArmaturaEJB::findArmor(Personaggio p);  **post:** Restituisce una lista con le armature che il personaggio è in grado di acquistare. |

**3.2.3.2 Metodo findArmor\_bylv**

|  |  |
| --- | --- |
| **Nome metodo:** | **findArmor\_bylv** |
| **Descrizione:** | Dato l’id dell’armatura ritorna l’oggetto armatura associato. |
| **Precondizione:** | **Context:** ArmaturaEJB:: findArmor\_bylv(long id);  **Pre:** id!=null; |
| **Postcondizione:** | **Context:** ArmaturaEJB:: findArmor\_bylv(long id);  **Post:** Ritorna l’oggetto armatura; |

**3.2.3.3 Metodo insertArmor**

|  |  |
| --- | --- |
| **Nome metodo:** | **insertArmor** |
| **Descrizione:** | Dato un’armatura la rende persistente. |
| **Precondizione:** | **Context:** ArmaturaEJB:: insertArmor(Armatura armor);  **Pre:** armor!=null; |
| **Postcondizione:** | **Context:** ArmaturaEJB:: insertArmor(Armatura armor);  **Post:** Rende l’armatura persistente; |

**3.2.3.4 Metodo updateWeapon**

|  |  |
| --- | --- |
| **Nome metodo:** | **updateWeapon** |
| **Descrizione:** | Dato un’armatura l’aggiorna all’interno del DB; |
| **Precondizione:** | **Context:** ArmaturaEJB::updateWeapon(Armatura armor);  **Pre:** armor!=null; |
| **Postcondizione:** | **Context:** ArmaturaEJB:: updateWeapon(Armatura armor);  **Post:** aggiorna l’armatura all’interno del database. |

**3.2.3.5 Metodo deleteWeapon**

|  |  |
| --- | --- |
| **Nome metodo:** | **deleteWeapon** |
| **Descrizione:** | Data un’armatura la elimina dal database; |
| **Precondizione:** | **Context:** ArmaturaEJB:: delete(Armatura armor);  **Pre:** armor!=null; |
| **Postcondizione:** | **Context:** ArmaturaEJB:: delete(Armatura armor);  **post:** cancella l’entità armatura dal database |

**3.2.4 CombatCommonEJB**

|  |  |
| --- | --- |
| **Nome entità:** | **CombatCommonEJB** |
| **Descrizione** | Gestisce tutta quella che è la logica di combattimento tra due personaggi |
| **Metodi:** | * inizializationCombat(Personaggio a, Personaggio b):void * removeCombat(Personaggio a, Personaggio b):void * returnFight(Personaggio a):Fight * attack(Personaggio a):List<String> * defense(Personaggio a) :List<String> * heals(Personaggio a) :List<String> * leave(Personaggio a) :List<String> * winner(Long combatKey):Long |

**3.2.4.1 Metodo inizializationCombat**

|  |  |
| --- | --- |
| **Nome metodo:** | **inizializationCombat** |
| **Descrizione:** | Inizializza il combattimento fra due personaggi |
| **Precondizione:** | **Context:** CombatCommonEJB:: inizializationCombat(Personaggio a, Personaggio b);  **Pre:** a!=null && b!=b; |
| **Postcondizione:** | **Context:** CombatCommonEJB:: inizializationCombat(Personaggio a, Personaggio b);  **post:** il combattimento viene inizializzato e viene aggiunta la stringa di inizio combattimento all’interno del log. |

**3.2.4.2 Metodo removeCombat**

|  |  |
| --- | --- |
| **Nome metodo:** | **removeCombat** |
| **Descrizione:** | Elimina il combattimento tra due giocatori; |
| **Precondizione:** | **Context:** CombatCommonEJB:: removeCombat(Personaggio a, Personaggio b);  **Pre:** a!=null && b!=b; |
| **Postcondizione:** | **Context:** CombatCommonEJB:: removeCombat(Personaggio a, Personaggio b);  **post:** il combattimento viene eliminato |

**3.2.4.3 Metodo returnFight**

|  |  |
| --- | --- |
| **Nome metodo:** | **returnFight** |
| **Descrizione:** | Dato un giocatore ritorna la chiave del combattimento a cui sta partecipando; |
| **Precondizione:** | **Context:** CombatCommonEJB::returnFight(Personaggio a);  **Pre:** a!=null; |
| **Postcondizione:** | **Context:** CombatCommonEJB::returnFight(Personaggio a);  **post:** ritorna l’id del combattimento; |

**3.2.4.4 Metodo attack**

|  |  |
| --- | --- |
| **Nome metodo:** | **Attack** |
| **Descrizione:** | Dato un personaggio attaccante attacca. |
| **Precondizione:** | **Context:** CombatCommonEJB:: attack(Personaggio a);  **Pre:** a!=null; |
| **Postcondizione:** | **Context:** CombatCommonEJB:: attack(Personaggio a);  **post:** Personaggio A attacca Personaggio B; |

**3.2.4.5 Metodo defense**

|  |  |
| --- | --- |
| **Nome metodo:** | **defense** |
| **Descrizione:** | Gestisce l’operazione di difesa di un dato personaggio. |
| **Precondizione:** | **Context:** CombatCommonEJB:: defense(Personaggio a);  **Pre:** a!=null; |
| **Postcondizione:** | **Context:** CombatCommonEJB:: defense(Personaggio a);  **post:** Personaggio A si difende; |

**3.2.4.3 Metodo heals**

|  |  |
| --- | --- |
| **Nome metodo:** | **heals** |
| **Descrizione:** | Gestisce l’operazione di cura di un dato personaggio. |
| **Precondizione:** | **Context:** CombatCommonEJB:: heals(Personaggio a);  **Pre:** a!=null; |
| **Postcondizione:** | **Context:** CombatCommonEJB:: heals (Personaggio a);  **post:** Personaggio A si cura; |

**3.2.4.4 Metodo leave**

|  |  |
| --- | --- |
| **Nome metodo:** | **leave** |
| **Descrizione:** | Gestisce l’operazione di fuga di un dato personaggio. |
| **Precondizione:** | **Context:** CombatCommonEJB:: leave(Personaggio a);  **Pre:** a!=null; |
| **Postcondizione:** | **Context:** CombatCommonEJB:: leave(Personaggio a);  **post:** Personaggio A fugge; |

**3.2.4.5 Metodo winner**

|  |  |
| --- | --- |
| **Nome metodo:** | **winner** |
| **Descrizione:** | Dato l’ID di un combattimento verifica se c’è un vincitore; |
| **Precondizione:** | **Context:** CombatCommonEJB::winner(Long combatKey);  **Pre:** combatKey!=null; |
| **Postcondizione:** | **Context:** CombatCommonEJB::winner(Long combatKey);  **post:** Se c’è un vincitore viene ritornato; |

**3.2.5 GiocatoreEJBRemote**

|  |  |
| --- | --- |
| **Nome entità:** | **GiocatoreEJBRemote** |
| **Descrizione** | EJB della gestione del giocatore e dei relativi dati |
| **Metodi:** | * findPlayer(String username):Giocatore; * insertPlayer(Giocatore newPlayer):Giocatore; * updatePlayer(Giocatore player):Giocatore; * removePlayer(Giocatore player):void; * login(String username, String password):Giocatore; * registration(String username, String password, String repassword, String email):String; * findUsername(String username):boolean; * findEmail(String email):boolean; |

**3.2.5.1 Metodo findPlayer**

|  |  |
| --- | --- |
| **Nome metodo:** | **findPlayer** |
| **Descrizione:** | Data un username cerca e ritorna il giocatore corrispondente all’interno del database. |
| **Precondizione:** | **Context:** GiocatoreEJBRemote:: findPlayer(String username);  **Pre:** username!=null; |
| **Postcondizione:** | **Context:** GiocatoreEJBRemote:: findPlayer(String username);  **post:** se il giocatore è stato trovato lo ritorna, altrimenti ritorna null; |

**3.2.5.2 Metodo insertPlayer**

|  |  |
| --- | --- |
| **Nome metodo:** | **insertAmministratore** |
| **Descrizione:** | Rende persistente il Giocatore passato |
| **Precondizione:** | **Context:** GiocatoreEJB:: insertPlayer(Giocatore newPlayer);  **Pre**: newPlayer != null; |
| **Postcondizione:** | **Context:** GiocatoreEJB:: insertPlayer(Giocatore newPlayer);  **Post:** Inserisce nel database il nuovo account. |

**3.2.5.3 Metodo updatePlayer**

|  |  |
| --- | --- |
| **Nome metodo:** | **updatePlayer** |
| **Descrizione:** | Aggiorna nel database lo stato del Giocatore |
| **Precondizione:** | **Context:** GiocatoreEJB:: updatePlayer(Giocatore player);  **Pre**: player != null |
| **Postcondizione:** | **Context:** GiocatoreEJB:: updatePlayer(Giocatore player);  **Post:** Se presente aggiorna il Giocatore. |

**3.2.5.4 Metodo removePlayer**

|  |  |
| --- | --- |
| **Nome metodo:** | **removePlayer** |
| **Descrizione:** | Rimuove dal database un giocatore |
| **Precondizione:** | **Context:** GiocatoreEJB:: removePlayer(Giocatore player);  Pre: player != null; |
| **Postcondizione:** | **Context:** GiocatoreEJB:: removePlayer(Giocatore player);  **Post:** se presente il giocatore viene rimosso |

**3.2.5.5 Metodo login**

|  |  |
| --- | --- |
| **Nome metodo:** | **login** |
| **Descrizione:** | Dato un username e una password effettua il login |
| **Precondizione:** | **Context:** GiocatoreEJB:: login(String username, String password);  Pre: username != null && password != null |
| **Postcondizione:** | **Context:** GiocatoreEJB:: login(String username, String password);  **Post:** Se presente all’interno del database restituisce il giocatore. |

**3.2.5.6 Metodo registration**

|  |  |
| --- | --- |
| **Nome metodo:** | **registration** |
| **Descrizione:** | crea un nuovo giocatore rendendolo persistente |
| **Precondizione:** | **Context:** GiocatoreEJB::registration(String username, String password, String repassword, String email);  **Pre**: username != null && password && null && repassword && null && email != null |
| **Postcondizione:** | **Context:** GiocatoreEJB::registration(String username, String password, String repassword, String email);  **Post:** Inserisce un nuovo account nel database |

**3.2.5.7 Metodo findUsername**

|  |  |
| --- | --- |
| **Nome metodo:** | **findUsername** |
| **Descrizione:** | Data un username controlla se è già presente nel database |
| **Precondizione:** | **Context:** GiocatoreEJB:: findEmail(String username);  **Pre**: username != null |
| **Postcondizione:** | **Context:** GiocatoreEJB:: findEmail(String username);  **Post:** true se è già presente, false altrimenti. |

**3.2.5.8 Metodo findEmail**

|  |  |
| --- | --- |
| **Nome metodo:** | **findEmail** |
| **Descrizione:** | Data un’e-mail controlla se è già presente nel database |
| **Precondizione:** | **Context:** GiocatoreEJB:: findEmail(String email);  **Pre**: email != null |
| **Postcondizione:** | **Context:** GiocatoreEJB:: findEmail(String email);  **Post:** true se è già presente, false altrimenti. |

**3.2.6 NotifyPlayerEJB**

|  |  |
| --- | --- |
| **Nome entità:** | **AmministratoreEJB** |
| **Descrizione** | EJB che si occupa della gestione degli inviti |
| **Metodi:** | * inizialization(Long id); * addNotify(Long id, Personaggio enemy); * removeNotify(Long id); * returnMap(Long id): Collection<Personaggio> |

**3.2.6.1 Metodo inizialization**

|  |  |
| --- | --- |
| **Nome metodo:** | **inizialization** |
| **Descrizione:** | inserisce id del Personaggio all’interno della Collezione. |
| **Precondizione:** | **Context:** NotifyPlayerEJB:: inizialization(Long id);  **Pre**: id > 0 && id != null; |
| **Postcondizione:** | **Context:** NotifyPlayerEJB:: inizialization(Long id);  **Post: il** personaggio è all’interno della collezione, pronto a ricevere inviti |

**3.2.6.2 Metodo addNotify**

|  |  |
| --- | --- |
| **Nome metodo:** | **addNotify** |
| **Descrizione:** | Dato id dello sfidante e l’entità Personaggio che si intende sfidare, aggiunge lanotifica nella Collezione |
| **Precondizione:** | **Context:** NotifyPlayerEJB:: addNotify(Long id, Personaggio enemy);  **Pre**: id != null && id > 0 && enemy != null |
| **Postcondizione:** | **Context:** NotifyPlayerEJB:: addNotify(Long id, Personaggio enemy);  **Post:** Aggiunge una notifica all’interno della collezione |

**3.2.6.3 Metodo removeNotify**

|  |  |
| --- | --- |
| **Nome metodo:** | **removeNotify** |
| **Descrizione:** | Rimuove la notifica che corrisponde alla key passata |
| **Precondizione:** | **Context:** NotifyPlayerEJB:: removeNotify(Long id);  **Pre**: id > 0 && id != null; |
| **Postcondizione:** | **Context:** NotifyPlayerEJB:: removeNotify(Long id);  **Post:** L’invito è rimosso |

**3.2.6.4 Metodo returnMap**

|  |  |
| --- | --- |
| **Nome metodo:** | **returnMap** |
| **Descrizione:** | Dato id del personaggio restituisce gli inviti ricevuti. |
| **Precondizione:** | **Context:** NotifyPlayerEJB:: returnMap(Long id): Collection<Personaggio>  Pre: id > 0 && id != null; |
| **Postcondizione:** | **Context:** NotifyPlayerEJB:: returnMap(Long id): Collection<Personaggio>  **Post:** Restituisce tutte le notifiche corrispondenti al Personaggio passato |

**3.2.7 OnlinePlayerEJB**

|  |  |
| --- | --- |
| **Nome entità:** | **OnlinePlayerEJB** |
| **Descrizione** | EJB che si occupa della gestione della lista dei giocatori online |
| **Metodi:** | * addPlayer(Giocatore key, Personaggio value); * removePlayer(Giocatore key) ; * returnHashMap() : Collection<Personaggio> |

**3.2.7.1 Metodo addPlayer**

|  |  |
| --- | --- |
| **Nome metodo:** | **addPlayer** |
| **Descrizione:** | Presi l’id del giocatore e il personaggio attualmente selezionato viene aggiunto alla Collezione dei giocatori online |
| **Precondizione:** | **Context:** OnlinePlayerEJB:: addPlayer(Giocatore key, Personaggio value);  **Pre**: key != null && value != null; |
| **Postcondizione:** | **Context:** OnlinePlayerEJB:: addPlayer(Giocatore key, Personaggio value);  **Post:** Aggiunge il Personaggio alla Mappa. |

**3.2.7.2 Metodo removePlayer**

|  |  |
| --- | --- |
| **Nome metodo:** | **removePlayer** |
| **Descrizione:** | Rimuove il Giocatore e il Personaggio dalla Mappa. |
| **Precondizione:** | **Context:** OnlinePlayerEJB::removePlayer(Giocatore key);  **Pre**: key != null |
| **Postcondizione:** | **Context:** OnlinePlayerEJB::removePlayer(Giocatore key);  **Post:** Rimuove il giocatore dalla Mappa perché non più online. |

**3.2.7.3 Metodo returnHashMap**

|  |  |
| --- | --- |
| **Nome metodo:** | **returnHashMap** |
| **Descrizione:** | Dato un oggetto amministratore lo rende persistenze e lo ritorna. |
| **Precondizione:** | **Context:** OnlinePlayerEJB::returnHashMap() : Collection<Personaggio> |
| **Postcondizione:** | **Context:** OnlinePlayerEJB::returnHashMap() : Collection<Personaggio>  **Post:** Restituisce la Collezione dei Personaggi online. |

**3.2.8 PersonaggioEJB**

|  |  |
| --- | --- |
| **Nome entità:** | **PersonaggioEJB** |
| **Descrizione** | EJB che si occupa della gestione di tutte le operazioni che riguardano il personaggio controllato dal giocatore |
| **Metodi:** | * addGold(Personaggio chr,int gold) * subGold(Personaggio chr,int gold) * addExp(Personaggio chr,int exp) * createCharacter(String name, String classe, Giocatore p): String * findCharacter(String username, String name): Personaggio * findByID(long id): Personaggio * insertPersonaggio(Personaggio p): Personaggio * updatePersonaggio(Personaggio p): Personaggio * removePersonaggio(Personaggio p): Personaggio * listCharacter(Giocatore p): List<Personaggio> * levelUp(Personaggio p): Personaggio * boughtWeapon(Personaggio p, Arma weapon): Personaggio * boughtArmor(Personaggio p, Armatura armor): Personaggio * fineDuello(Personaggio winner); |

**3.2.8.1 Metodo addGold**

|  |  |
| --- | --- |
| **Nome metodo:** | addGold |
| **Descrizione:** | Aggiunge al personaggio passato come paramentro l’ammontare di gold passato al metodo |
| **Precondizione:** | **Context:** PersonaggioEJB:: addGold(Personaggio chr,int gold); |
| **Postcondizione:** | **Context:** PersonaggioEJB:: addGold(Personaggio chr,int gold);  **Post:** Restituisce il Personaggio con l’attributo gold aggiornato. |

**3.2.8.2 Metodo subGold**

|  |  |
| --- | --- |
| **Nome metodo:** | subGold |
| **Descrizione:** | Sottrae al personaggio passato come parametro l’ammontare di gold passato al metodo |
| **Precondizione:** | **Context:** PersonaggioEJB:: subGold(Personaggio chr,int gold);  **Pre**: gold > 0; |
| **Postcondizione:** | **Context:** PersonaggioEJB:: subGold(Personaggio chr,int gold);  **Post:** Restituisce il Personaggio con l’attributo gold aggiornato; |

**3.2.8.3 Metodo addExp**

|  |  |
| --- | --- |
| **Nome metodo:** | **addExp** |
| **Descrizione:** | Aggiunge al personaggio passato come paramentro l’ammontare di esperienza passato al metodo |
| **Precondizione:** | **Context:** PersonaggioEJB:: addExp(Personaggio chr,int exp);  **Pre**: exp > 0; |
| **Postcondizione:** | **Context:** PersonaggioEJB:: addExp(Personaggio chr,int exp);  **Post:** Restituisce il Personaggio con l’attributo exp aggiornato; |

**3.2.8.4 Metodo createCharacter**

|  |  |
| --- | --- |
| **Nome metodo:** | createCharacter |
| **Descrizione:** | Passato nome, classe e il Giocatore, crea un nuovo Personaggio che viene associato al Giocatore |
| **Precondizione:** | **Context:** PersonaggioEJB::createCharacter(String name, String classe, Giocatore p) |
| **Postcondizione:** | **Context:** PersonaggioEJB::createCharacter(String name, String classe, Giocatore p)  **Post:** Restituisce un nuovo Personaggio associato al Giocatore. |

**3.2.8.5 Metodo findCharacter**

|  |  |
| --- | --- |
| **Nome metodo:** | **findCharacter** |
| **Descrizione:** | Dato l’username del Giocatore e Il nome del Personaggio da cercare, interroga il database che restituisce il Personaggio. |
| **Precondizione:** | **Context:** PersonaggioEJB**::**findCharacter(String username, String name); |
| **Postcondizione:** | **Context:** PersonaggioEJB**::**findCharacter(String username, String name);  **Post:** Restituisce il Personaggio cercato se esiste; |

**3.2.8.6 Metodo findByID**

|  |  |
| --- | --- |
| **Nome metodo:** | **findByID** |
| **Descrizione:** | Dato un id di un Personaggio restituisce il personaggio crcato. |
| **Precondizione:** | **Context:** PersonaggioEJB:: findByID(long id);  **pre**: id != null; |
| **Postcondizione:** | **Context:** PersonaggioEJB:: findByID(long id);  **Post:** Restituisce il personaggio cercato se esiste. |

**3.2.8.7 Metodo insertPersonaggio**

|  |  |
| --- | --- |
| **Nome metodo:** | **insertPersonaggio** |
| **Descrizione:** | Rende persistente il personaggio passato come parametro. |
| **Precondizione:** | **Context:** PersonaggioEJB:: insertPersonaggio(Personaggio p);  **Pre:** p != null; |
| **Postcondizione:** | **Context:** PersonaggioEJB:: insertPersonaggio(Personaggio p);  **Post:** Effettua una chiamata al database per rendere persistente il personaggio passato come parametro |

**3.2.8.8 Metodo updatePersonaggio**

|  |  |
| --- | --- |
| **Nome metodo:** | **updatePersonaggio** |
| **Descrizione:** | Dato un oggetto Personaggio effettua una chiamata per aggiornare l’entità nel database. |
| **Precondizione:** | **Context:** PersonaggioEJB:: updatePersonaggio(Personaggio p);  **Pre**: p != null; |
| **Postcondizione:** | **Context:** PersonaggioEJB:: updatePersonaggio(Personaggio p);  **Post:** Nel database si ritroverà la versione aggiornata del personaggio. |

**3.2.8.9 Metodo removePersonaggio**

|  |  |
| --- | --- |
| **Nome metodo:** | **removePersonaggio** |
| **Descrizione:** | Elimina un’istanza dell’entità Personaggio dal database. |
| **Precondizione:** | **Context:** PersonaggioEJB::removePersonaggio(Personaggio p);  **Pre**: p != null; |
| **Postcondizione:** | **Context:** PersonaggioEJB::removePersonaggio(Personaggio p);  **Post:** Il personaggio passato come parametro è eliminato dal database. |

**3.2.8.10 Metodo levelUp**

|  |  |
| --- | --- |
| **Nome metodo:** | **levelUp** |
| **Descrizione:** | Preso il personaggio aumenta il suo livello e le sue statistiche. |
| **Precondizione:** | **Context:** PersonaggioEJB:: levelUp(Personaggio p);  **Pre**: p != null; |
| **Postcondizione:** | **Context:** PersonaggioEJB:: levelUp(Personaggio p);  **Post:** restituisce il personaggio aggiornato. |

**3.2.8.11 Metodo boughtWeapon**

|  |  |
| --- | --- |
| **Nome metodo:** | **boughtWeapon** |
| **Descrizione:** | Aggiorna l’attributo dell’arma all’interno del Personaggio passato. |
| **Precondizione:** | **Context:** PersonaggioEJB::boughtWeapon(Personaggio p, Arma weapon);  **Pre**: p != null && weapon != null |
| **Postcondizione:** | **Context:** PersonaggioEJB::boughtWeapon(Personaggio p, Arma weapon);  **Post:** restituisce il personaggio aggiornato col l’acquisto appena effettuato. |

**3.2.8.12 Metodo boughtArmor**

|  |  |
| --- | --- |
| **Nome metodo:** | **boughtArmor** |
| **Descrizione:** | Aggiorna l’attributo dell’armatura all’interno del Personaggio passato. |
| **Precondizione:** | **Context:** PersonaggioEJB:: boughtArmor (Personaggio p, Armatura armor);  **Pre**: p != null && armor != null; |
| **Postcondizione:** | **Context:** PersonaggioEJB:: boughtArmor (Personaggio p, Armatura armor);  **post:** restituisce il personaggio aggiornato col l’acquisto appena effettuato. |

**3.2.8.13 Metodo fineDuello**

|  |  |
| --- | --- |
| **Nome metodo:** | **fineDuello** |
| **Descrizione:** | Aggiunge al personaggio che ha vinto il duello l’oro e l’esperienza. |
| **Precondizione:** | **Context:** PersonaggioEJB:: fineDuello(Personaggio winner);  **Pre**: winner != null; |
| **Postcondizione:** | **Context:** PersonaggioEJB:: fineDuello(Personaggio winner);  **post:** Restituisce il personaggio aggiornato. |

**3.3 Servlet:**

Le servlet implementano tutte il medesimo metodo con la seguente firma:

***protected void processRequest(HttpServletRequest request, HttpServletResponse response);***

In modo da garantire, il passaggio dei dati in modo sicuro e non visibile dalla barra degli indirizzi.

|  |  |
| --- | --- |
| **Nome Servlet:** | **AmministratorListPlayerServlet** |
| **Descrizione:** | Restituisce in formato JSON la lista dei giocatori iscritti al sito in modo che possa essere visualizzata all’interno dell’interfaccia dell’amministratore. |
| **Pre Condizioni:** | **Context:** AmministratorListPlayerServlet:: processRequest(request, response) |
| **Post Condizioni:** | **Context:** AmministratorListPlayerServlet:: processRequest(request, response)  **Post:** La lista contente i giocatori viene inviata tramite JSON; |

|  |  |
| --- | --- |
| **Nome Servlet:** | **AmministratorListPlayerServletBan** |
| **Descrizione:** | Restituisce in formato JSON la lista dei giocatori bannati dall’amministratore in modo che possa essere visualizzata all’interno dell’interfaccia dell’amministratore. |
| **Pre Condizioni:** | **Context:** AmministratorListPlayerServletBan:: processRequest(request, response) |
| **Post Condizioni:** | **Context:** AmministratorListPlayerServletBan:: processRequest(request, response)  **Post:** La lista contente i giocatori bannati viene inviata tramite JSON. |

|  |  |
| --- | --- |
| **Nome Servlet:** | **ArmorLoadServlet** |
| **Descrizione:** | Restituisce tramite JSON, la lista delle armature che il personaggio è in grado di acquistare. |
| **Pre Condizioni:** | **Context:** ArmorLoadServlet:: processRequest(request, response)  **Pre:** session.getAttribute("character")!=null |
| **Post Condizioni:** | **Context:** ArmorLoadServlet:: processRequest(request, response)  **Post:** La lista contente le armi viene restituita tramite JSON. |

|  |  |
| --- | --- |
| **Nome Servlet:** | **BanServlet** |
| **Descrizione:** | Preleva dalla form contente il bottone di ban, il nome del giocatore del giocatore che l’amministratore desidera bannare e lo banna. |
| **Pre Condizioni:** | **Context:** BanServlet:: processRequest(request, response)  **Pre:** request.getParameter("name\_ban")!=null; |
| **Post Condizioni:** | **Context:** BanServlet:: processRequest(request, response)    **Post:** banPlayer(username); |

|  |  |
| --- | --- |
| **Nome Servlet:** | **BoughtArmorServelt** |
| **Descrizione:** | Preleva dalla form contente il bottone di acquisto, l’ID dell’item che il giocatore desidera acquistare e lo acquista. |
| **Pre Condizioni:** | **Context:** BoughtArmorServelt:: processRequest(request, response)  **Pre:** request.getParameter("id\_value") !=null && session.getAttribute("character") !=null; |
| **Post Condizioni:** | **Context:** BoughtArmorServelt:: processRequest(request, response)  **Post:** boughtArmor(session\_player, findArmorByid(id)), L’armatura viene acquistata; |

|  |  |
| --- | --- |
| **Nome Servlet:** | **BoughtWeaponServelt** |
| **Descrizione:** | Preleva dalla form contente il bottone di acquisto, l’ID dell’item che il giocatore desidera acquistare e lo acquista. |
| **Pre Condizioni:** | **Context:** BoughtWeaponServelt:: processRequest(request, response)  **Pre:** request.getParameter("id\_value") !=null && session.getAttribute("character") !=null; |
| **Post Condizioni:** | **Context:** BoughtWeaponServelt:: processRequest(request, response)  **Post:** boughtWeapon(session\_player, findArmorByid(id)), l’arma viene acquistata; |

|  |  |
| --- | --- |
| **Nome Servlet:** | **ButtonDisplayServlet** |
| **Descrizione:** | Attiva l’interfaccia di combattimento ad entrambi gli sfidanti. |
| **Pre Condizioni:** | **Context:** ButtonDisplayServlet:: processRequest(request, response)  **Pre:** session.getAttribute("character")!=null; |
| **Post Condizioni:** | **Context:** ButtonDisplayServlet:: processRequest(request, response)  **Post:** l’interfaccia di combattimento viene attivata |

|  |  |
| --- | --- |
| **Nome Servlet:** | **CharacterCreationServlet** |
| **Descrizione:** | Preleva dalla form contenente il nome e la classe del personaggio che il giocatore desidera creare e lo crea. |
| **Pre Condizioni:** | **Context:** CharacterCreationServlet:: processRequest(request, response)  **Pre:** request.getParameter("name") != null && request.getParameter("classe") !== null && session.getAttribute("player")!= null |
| **Post Condizioni:** | **Context:** CharacterCreationServlet:: processRequest(request, response)  **Post:** createCharacter(name, classe, logged), il personaggio viene creato; |

|  |  |
| --- | --- |
| **Nome Servlet:** | **CharachterLoadServlet** |
| **Descrizione:** | Restituisce in formato JSON la lista di tutti i personaggi associati a un giocatore in modo che possano essere visualizzati da quest’ultimo all’interno della pagina di ingresso. |
| **Pre Condizioni:** | **Context:** CharachterLoadServlet:: processRequest(request, response)  **Pre:** session.getAttribute("player") !=null |
| **Post Condizioni:** | **Context:** CharachterLoadServlet:: processRequest(request, response)  **Post:** lista personaggi inviata tramite JSON; |

|  |  |
| --- | --- |
| **Nome Servlet:** | **FightServlet** |
| **Descrizione:** | Gestisce il combattimento tra due personaggi, gestendo il tipo di bottone che viene cliccato e inviando sotto forma JSON la stringa risultante dall’interazione per essere visualizzata. |
| **Pre Condizioni:** | **Context:** FightServlet:: processRequest(request, response)  **Pre:** session.getAttribute("character") != null; |
| **Post Condizioni:** | **Context:** FightServlet:: processRequest(request, response)  **Post:** attack(session\_ch) || defense(session\_ch) || heals(session\_ch) || leave(session\_ch),viene inviato tramite JSON la stringa risultante dall’operazione effettuata. |

|  |  |
| --- | --- |
| **Nome Servlet:** | **WeaponLoadServlet** |
| **Descrizione:** | Restituisce in formato JSON la lista delle armi corrispondenti al livello del Personaggio inserito nella sessione di gioco. |
| **Pre Condizioni:** | **Context:** WeaponLoadServelt :: processRequest(request, response)  **Pre:** session.getAttribute(“character”) != null; |
| **Post Condizioni:** | **Context:** WeaponLoadServelt :: processRequest(request, response)  **Post:** Vengono trasformate in formato JSON le armi trovate all’interno del database |

|  |  |
| --- | --- |
| **Nome Servlet:** | **UnbanServlet** |
| **Descrizione:** | Imposta a False l’attributo BAN di un giocatore in modo da permettergli di accedere al sito. |
| **Pre Condizioni:** | **Context:** UnbanServlet:: processRequest(request, response)  **Pre: r**equest.getParameter(“name\_unban”) != null; |
| **Post Condizioni:** | **Context:** UnbanServlet:: processRequest(request, response)  **Post:** Il Giocatore viene sbannato e gli è permesso di acceder al sito |

|  |  |
| --- | --- |
| **Nome Servlet:** | **ToGameServlet** |
| **Descrizione:** | Manda alla pagina del gioco utilizzando il nome del Giocatore e il nome del personaggio scelto per caricare le statistiche del personaggio correttamente, sincronizzandolo alla sessione. |
| **Pre Condizioni:** | **Context:** ToGameServlet::processRequest(request, response)  **Pre:** request.getParameter(“id\_value”) != null && session.getAttribute(“character”) != null; |
| **Post Condizioni:** | **Context:** ToGameServlet::processRequest(request, response)  **Post:** Il Giocatore viene portato alla pagina di gioco col Personaggio selezionato. |

|  |  |
| --- | --- |
| **Nome Servlet:** | **StartCombatServlet** |
| **Descrizione:** | Crea una nuova istanza di un combattimento tra due giocatori. |
| **Pre Condizioni:** | **Context:** StartCombatServlet:: processRequest(request, response)  **Pre:** session.getAttribute(“character”) != null &&request**.**getParameter(“id\_enemy”) != null; |
| **Post Condizioni:** | **Context:** StartCombatServlet:: processRequest(request, response)  **Post:** Viene inizializzato un nuovo combattimento |

|  |  |
| --- | --- |
| **Nome Servlet:** | **SaveServlet** |
| **Descrizione:** | Permette di salvare i progressi di gioco di un Personaggio. |
| **Pre Condizioni:** | **Context:** SaveServlet::processRequest(request, response)  **Pre:** session.getAttribute(“character”) != null; |
| **Post Condizioni:** | **Context:** SaveServlet::processRequest(request, response)  **Post:** I dati del personaggio vengono salvati |

|  |  |
| --- | --- |
| **Nome Servlet:** | **RemoveServlet** |
| **Descrizione:** | Permette ad un Amministratore di eliminare un Giocatore dal sito |
| **Pre Condizioni:** | **Context:** RemoveServlet::processRequest(request, response)  **Pre:** request.getParameter(“name\_rm”) != null; |
| **Post Condizioni:** | **Context:** RemoveServlet::processRequest(request, response)  **Post:** Il giocatore viene rimosso dal sito. |

|  |  |
| --- | --- |
| **Nome Servlet:** | **RegistrationServlet** |
| **Descrizione:** | Permette all’utente finale di creare un nuovo account e di accedere all’area della creazione/selezione personaggio. |
| **Pre Condizioni:** | **Context:** RegistrationServlet::processRequest(request, response)  **Pre:** request.getParameter(“username”) != null && request.getParameter(“password”) != null && request.getParameter(“repassword”) != null && request.getParameter(“email”) != null |
| **Post Condizioni:** | **Context:** RegistrationServlet::processRequest(request, response)  **Post:** Il nuovo utente viene inserito all’interno del database. |

|  |  |
| --- | --- |
| **Nome Servlet:** | **OnlineListServelt** |
| **Descrizione:** | Restituisce una collezione che contiene i nomi dei personaggi online. |
| **Pre Condizioni:** | **Context:** OnlineListServlet::processRequest(request, response)  **Pre:** |
| **Post Condizioni:** | **Context:** OnlineListServlet::processRequest(request, response)  **Post:** Restituisce una collezione di Personaggi online. |

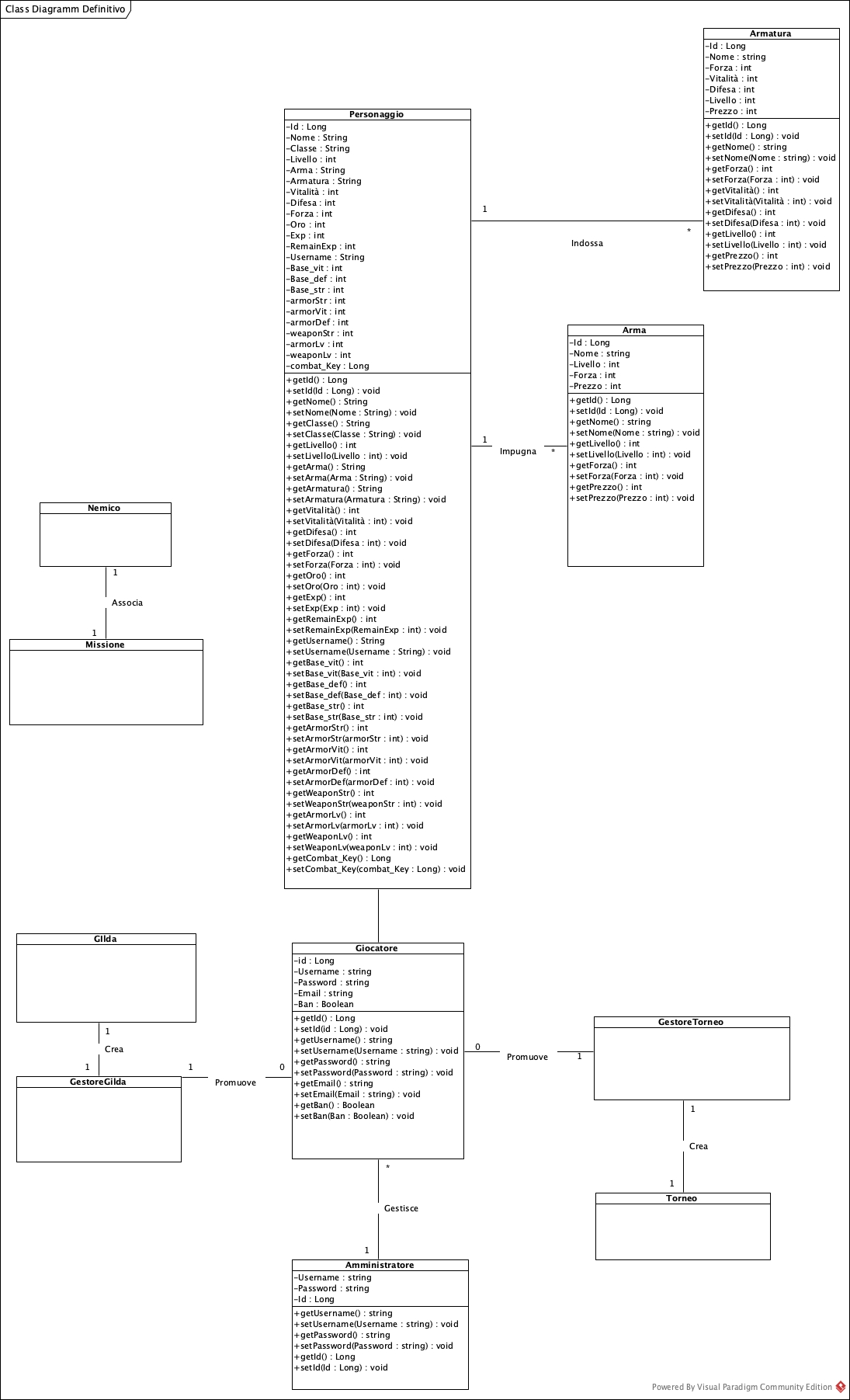
|  |  |
| --- | --- |
| **Nome Servlet:** | **NotifyServlet** |
| **Descrizione:** | Crea una nuova notifica tra due Personaggi. |
| **Pre Condizioni:** | **Context:** NotifyServlet::processRequest(request, response)  **Pre:** session.getAttribute(“character”) != null &&request**.**getParameter(“id\_enemy”) != null; |
| **Post Condizioni:** | **Context:** NotifyServlet::processRequest(request, response)  **Post:** Aggiunge una notifica che è scambiata tra due Personaggi. |

|  |  |
| --- | --- |
| **Nome Servlet:** | **NotifyDisplayServlet** |
| **Descrizione:** | Restituisce in formato JSON una collezione che rappresenta tutte le notifiche che un personaggio ha ricevuto. |
| **Pre Condizioni:** | **Context:** NotifyDisplayServlet::processRequest(request, response)  **Pre:** session.getAttribute(“character”) != null |
| **Post Condizioni:** | **Context:** NotifyDisplayServlet::processRequest(request, response)  **Post:** Restituisce una lista che contiene le notifiche del Personaggio in sessione. |

|  |  |
| --- | --- |
| **Nome Servlet:** | **LogoutServlet** |
| **Descrizione:** | Permette di eseguire il logout in maniera sicura, distruggendo la sessione e salvando i dati di progressione del personaggio. |
| **Pre Condizioni:** | **Context:** LogoutServlet::processRequest(request, response)  **Pre:** session.getAttribute(“character”) != null |
| **Post Condizioni:** | **Context:** LogoutServlet::processRequest(request, response)  **Post:** Salva i progressi del personaggio ed elimina in modo sicuro la sessione. |

|  |  |
| --- | --- |
| **Nome Servlet:** | **LoginServlet** |
| **Descrizione:** | Permette l’accesso al sito se un utente e registrato. |
| **Pre Condizioni:** | **Context:** LoginServlet::processRequest(request, response)  **Pre: request.getParameter(“username”) != null && request.getParameter(“password”) != null** |
| **Post Condizioni:** | **Context:** LoginServlet::processRequest(request, response)  **Post:** Registra in sessione il giocatore e gli permette di accedere al sito. |

**Class Diagram Finale:**

****