**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**

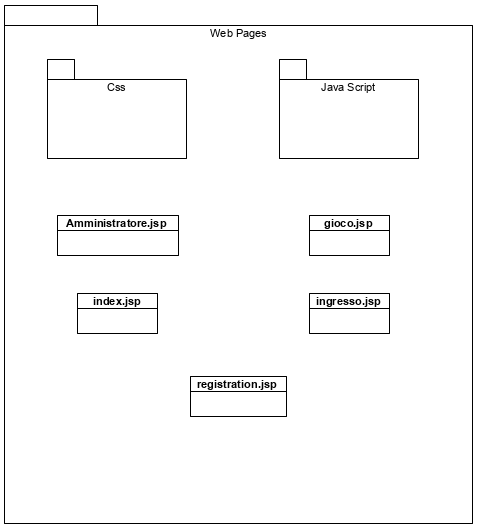
Componenti che andremmo a utilizzare sono:

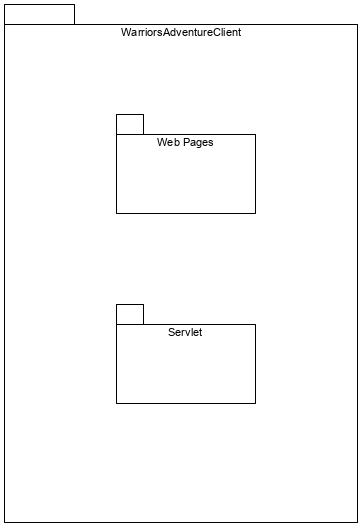
* Libreria open-source GSON distribuita da Google per linguaggio Java.
* Libreria jQuery per linguaggio Java Script.
* Libreria Open- Source AJAX per garantire aggiornamenti asincroni tra client e server.

**1.4 Definizioni, acronimi e abbreviazioni**

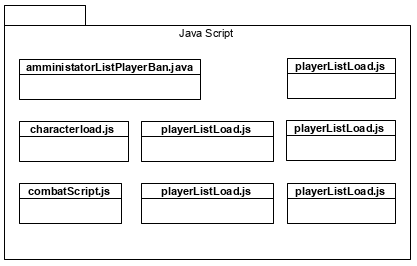
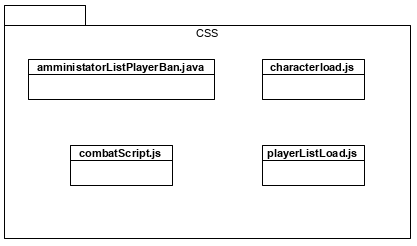
* DB = Database;
* DERBY = Apache Derby;
* RAD = Requirement Analysis Document;
* SDD = System Design Document;
* ODD = Object Design Document;
* PvP = Player vs Player;
* PvE = Player vs Environment;
* Player/s = Giocatore/I;
* JDBC = Java Data Base Connectivity;
* HTML = Linguaggio di Mark-Up per pagine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

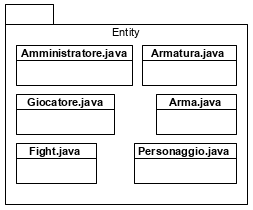
**2. Package**

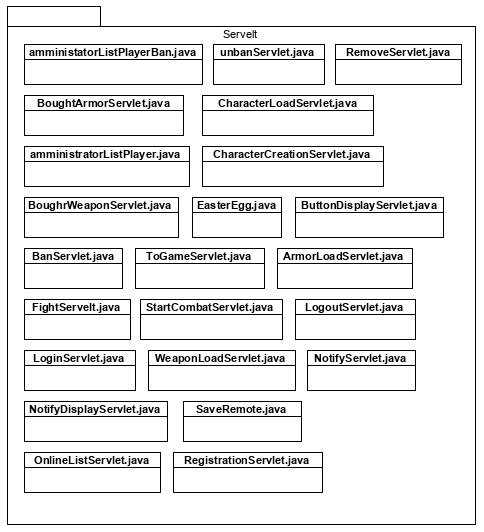
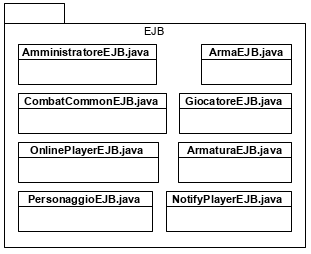


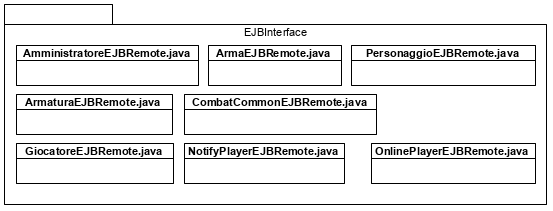
**2.1 Package Generale:**

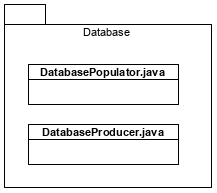
**2.2** **Package CSS e Java Script:**



**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 |

**3.2.6.1 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. |

**3.2.6.2 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. |

**3.2.6.3 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. |

**3.2.6.4 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. |

**3.2.6.5 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. |

**3.2.6.6 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. |

**3.2.6.7 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. |

**3.2.6.8 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. |

**3.3 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. |