Object

Design Document

Team:

Alberico Angelo

Landolfi Gianvincenzo

Rocchino Luigi



[1. Introduzione 2](#_Toc1691448032)

[1.1 Trade off 2](#_Toc811739715)

[1.2 Linee Guida 2](#_Toc1969348956)

[1.3 Definizione,Acronimi,abbrevazioni 3](#_Toc827509203)

[1.4 Reference 3](#_Toc513699316)

[2. Package 3](#_Toc961368009)

[2.1 Package com.GottaBattleEmAll.GottaBattleEmAll 4](#_Toc312222254)

[2.1.1 Package config 4](#_Toc1186437234)

[2.1.2 Package controller 4](#_Toc1714902174)

[2.1.3 Package entity 4](#_Toc979849510)

[2.1.4 Package repository 4](#_Toc988119826)

[2.1.5 Package service 5](#_Toc1534899896)

[2.2 Package resources 5](#_Toc1408611851)

[2.2.1 Package static.assets 5](#_Toc482373288)

[2.2.2 Package templates 5](#_Toc1010186024)

[3.Interfacce delle classi 6](#_Toc2062097855)

[3.1 GuestService 6](#_Toc701226020)

[3.2 UtenteService 6](#_Toc1707124115)

[3.3 TorneoService 6](#_Toc1832580591)

[3.4 ModeratoreService 7](#_Toc1756609117)

[4.Object Constrain Language 7](#_Toc695870840)

[4.1 GuestService 8](#_Toc452568471)

[4.2 UtenteService 8](#_Toc1777621032)

[4.3 TorneoService 10](#_Toc191104858)

[4.4 ModeratoreService 13](#_Toc552158096)

# 1. Introduzione

## 1.1 Trade off

* Performance vs Usabilità

Il sistema è progettato per privilegiare le performance, adottando il rendering lato server tramite pagine Thymeleaf. Questo approccio mira a migliorare tempi di risposta e carico del server, ottimizzando l'esecuzione delle operazioni sul lato server. Tuttavia, l’interattiva viene compromessa siccome le pagine devono essere caricate ad ogni interazione.

* Sicurezza vs Accessibilità

Il sistema è progettato per prediligere l’accessibilità, andando a diminuire la complessità di accesso da parte dell’utente. Ciò comporta l’aumento del rischio di vulnerabilità, siccome per la sicurezza si implementeranno dei filtri e non ci saranno meccanismi di conferma dell’autenticazione come 2FA o MFA.

* Sviluppo rapido vs funzionalità

Il sistema è orientato a uno sviluppo rapido concentrandosi sulle funzionalità essenziali per garantire una rapida implementazione. Questo trade-off mira a fornire tempestivamente un prodotto funzionale, sacrificando funzionalità non strettamente necessarie. L'approccio si basa sulla priorità delle caratteristiche cruciali, consentendo una rapida consegna e successivi aggiornamenti incrementali per migliorare e arricchire le funzionalità.

## 1.2 Linee Guida

Per lo sviluppo saranno utilizzate le seguenti convenzioni:

* Gli oggetti e i metodi saranno nominati a seconda della convenzione CamelCase.
* Gli oggetti entità saranno chiamati semplicemente con il nome dell'entità.
* Gli oggetti saranno nominati a seconda dell'entità di cui si occupano e del loro ruolo nell' applicazione per esempio TorneoService, UtenteController ecc...
* Le pagine html in Thymeleaf saranno segnate completamente in minuscolo.
* I nomi dei package saranno scritte in minuscolo.

## 1.3 Definizione,Acronimi,abbrevazioni

Acronimi:

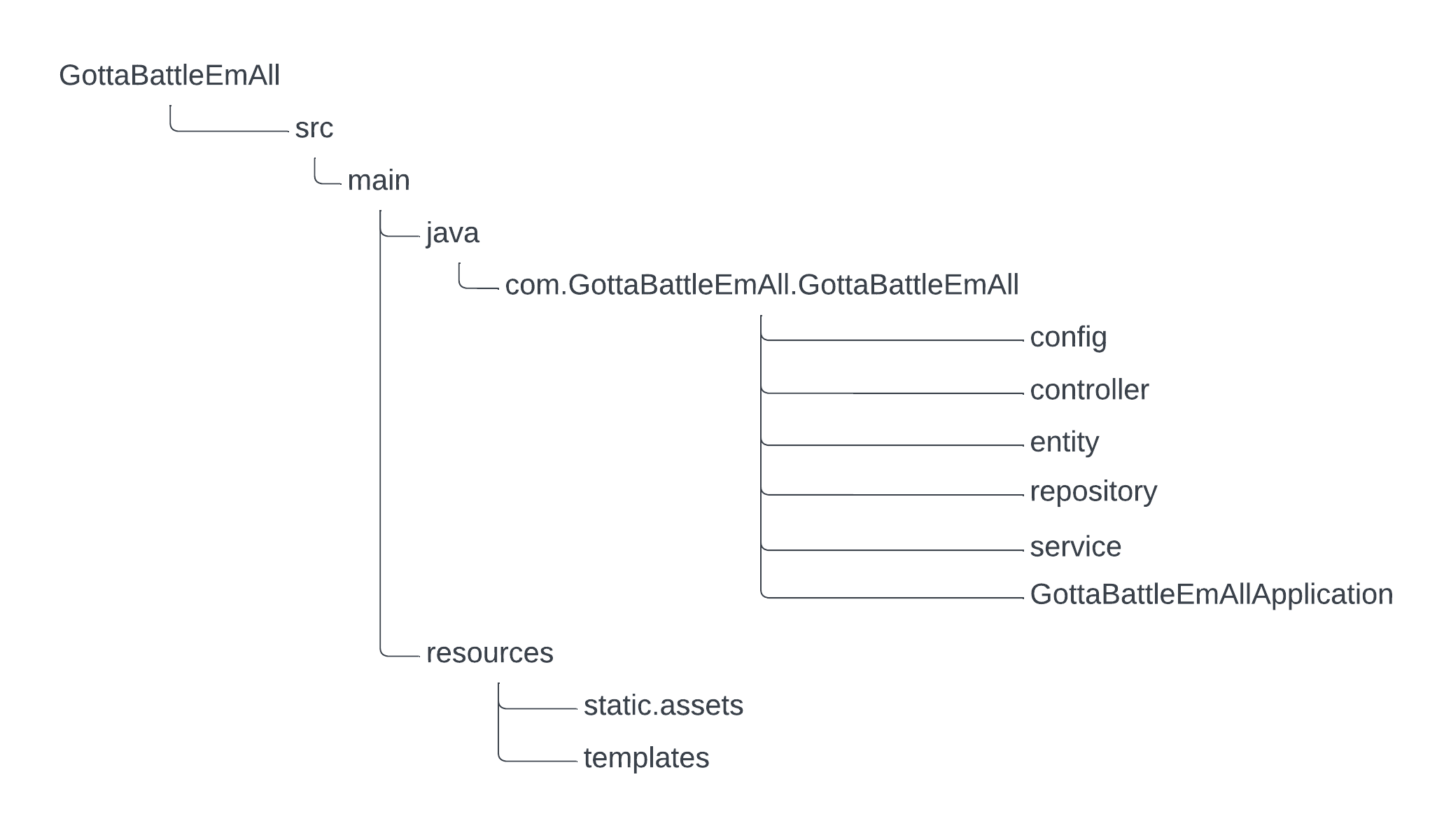
* RAD: Requirements Analysis Document.
* SDD: System Design Document.
* ODD: Object Design Document.

## 1.4 Reference

* RAD\_GottaBattleEmAll
* SDD \_GottaBattleEmAll

# 2. Package

La decomposizione in package è la seguente:



Tutto il codice si trova all’interno del package GottaBattleEmAll, al suo interno troviamo:

* Il package com.GottaBattleEmAll.GottaBattleEmAll , è un package in cui sono contenuti sia altri package che riguardano le classi Configuration, Controller, Entity, Repository e Service, sia un file di startup per lo start del sistema(GottaBattleEmAllApplication).
* Il package configuration (config) contiene le configurazioni di sistema per operazioni che vengono fatte durante lo startup o la creazione di determinati Bean(Component e Configuration),.
* Il package controller contiene tutti gli oggetti Controller che si occupano delle richieste Http da parte dell'utente.
* Il package entity conterrà le entità del sistema con eventuale mapping in sql tramite le annotazioni di Spring.
* Il package repository contiene tutte le interfacce per la creazione delle repository per le operazioni CRUD per ogni singola entità.
* Il package service contiene l’interfaccia e l’implementazione dei servizi.
* Il package resources è composto dal package static.assets in cui ci sono i package relativi a: css, fonts, images, js, scss, vendors. Inoltre è presente il package templates in cui ci sono le pagine html e il package fragment.
* il package fragment contiene le sottopagine html che è possibile ripetere su piu pagine.

## 2.1 Package com.GottaBattleEmAll.GottaBattleEmAll

### 2.1.1 Package config

Nel package config sono presente tutte le classi Filter che si occupano di gestire l’accesso degli utenti a determinate pagine. Inoltre è presente una classe Configuration che permette di creare un Bean, che può essere iniettato all’interno dei servizi, per codificare e decodificare le password in modo sicuro utilizzando l'algoritmo di hashing Bcrypt.

* GiocatoreFilter
* ModeratoreFilter
* OrganizzatoreFilter
* SecurityConfig
* StartupRunner
* GlobalExceptionHandler

### 2.1.2 Package controller

Nel package controller sono presenti le classi Controller che sono responsabili di gestire le richieste HTTP dai client e di elaborarle, prendendo decisioni su cosa fare con la richiesta e generando una risposta appropriata.

Le classi Controller interagiscono con il modello per ottenere dati necessari per generare la risposta da inviare al client o per aggiornare lo stato dell'applicazione in base alla richiesta ricevuta.  
Inoltre gestiscono gli errori che si verificano durante l'elaborazione delle richieste tramite la restituzione di risposte di errore appropriate al client.

* GuestController
* ModeratoreController
* TorneoController
* UtenteController

### 2.1.3 Package entity

Nel package entity sono presenti le Entità che modellano gli oggetti del dominio dell’applicazione, ogni istanza di una classe entity rappresenta una riga di dati all'interno della tabella corrispondente nel database.

* Giocatore
* Moderatore
* Organizzatore
* Richiesta
* Torneo
* Utente

### 2.1.4 Package repository

Nel package repository sono presenti interfacce responsabili di fornire un'interfaccia per l'interazione con il database e l'esecuzione di operazioni CRUD (Create, Read, Update, Delete) sulle entità (classi entity) all'interno dell'applicazione.

* GiocatoreRepository
* ModeratoreRepository
* OrganizzatoreRepository
* RichiestaRepository
* TorneoRepository

### 2.1.5 Package service

Nel package service sono presenti sia le interfacce di servizio che le classi di servizio. Le prime si occupano di definire i servizi che verranno implementate dalle seconde, le quali vanno a incapsulare la logica di business dell’applicazione. È stata definita un'interfaccia di servizio per ogni classe di servizio.

* GuestService
* GuestServiceImpl
* ModeratoreService
* ModeratoreServiceImpl
* TorneoService
* TorneoServiceImpl
* UtenteService
* UtenteServiceImpl

## 2.2 Package resources

Nel package resources oltre i package static.assets e templates, è presente il file application.properties, nel quale viene definita la connessione al database.

### 2.2.1 Package static.assets

Nel package static.assets sono presenti file css, scss, js, immagini.

### 2.2.2 Package templates

Nel package templates sono presenti tutte le pagine html:

* 404.htlm
* admin.html
* centroNotifiche.html
* creaTorneo.html
* homeGuest.html
* homeOrganizzatore.html
* homePlayer.html
* impostazioni.html
* impostazioniGiocatore.html
* impostazioniOrganizzatore.html
* iscrizioniTorneo.html
* listaTornei.html
* listaUtenti.html
* login.html
* loginAdmin.html
* profiloUtente.html
* registrazione.html
* ricercaTornei.html
* torneoFinitoBro.html
* torneoInAttesa.html
* torneoInAttesaGiocatore.html
* torneoInCorso.html
* torneoIscrizioneChiuseGiocatore.html

# 3.Interfacce delle classi

## 3.1 GuestService

|  |  |
| --- | --- |
| **GuestService** | |
| **Metodo** | **Descrizione** |
| Public String **registrazioneGiocatore**(Giocatore giocatore, String confermaPassword) | Consente all’utente di registrarsi come giocatore con le credenziali fornite. |
| Public String **registrazioneOrganizzatore**(Organizzatore organizzatore, confermaPassword) | Consente all’utente di registrarsi come organizzatore con le credenziali fornite. |

## 3.2 UtenteService

|  |  |
| --- | --- |
| **UtenteService** | |
| **Metodo** | **Descrizione** |
| Public String **login**(Utente utente, String ruolo) | Consente all’utente registrato di effettuare l’autenticazione a secondo del ruolo |
| Public String **modificaProfilo**(Utente utente, String confermaPassword, String ruolo, String username) | Consente all’utente registrato di modificare il proprio profilo utente. |
| public Giocatore **findGiocatoreByUsername**(String username) | Consente di ricercare un giocatore dato l’username |
| public Organizzatore **findOrganizzatoreByUsername**(String username) | Consente di ricercare un giocatore dato l’username |
| public List<Organizzatore> **findOrganizzatoriPaged**(int number, int size) | Consente di ricercare tutti gli organizzatori paginandoli |
| public List<Giocatore> **findGiocatoriPaged**(int number, int size); | Consente di ricercare tutti i giocatori paginandoli |
| public int **countGiocatoriTotali**() | Consente di ottenere il numero dei giocatori totali |
| public int **countOrganizzatoriTotali**() | Consente di ottenere il numero degli organizzatori totali |
| public int **countUtentiTotali**() | Consente di ottenere il numero degli utenti totali |

## 3.3 TorneoService

|  |  |
| --- | --- |
| **TorneoService** | |
| **Metodo** | **Descrizione** |
| public String **creaTorneo**(Torneo torneo, Organizzatore organizzatore) | Consente all’organizzatore di creare il proprio torneo |
| public boolean **iniziareTorneo**(Torneo torneo,Organizzatore organizzatore) | Consente all’organizzatore di far iniziare un proprio torneo |
| public boolean **terminareTorneo**(Torneo torneo,Organizzatore organizzatore) | Consente all’organizzatore di far terminare un proprio torneo |
| public String **toglierePartecipanti**(Torneo torneo, Giocatore giocatore, Organizzatore organizzatore) | Consente all’organizzatore di togliere un giocatore iscritto ad un suo torneo |
| public Giocatore **visualizzaProfiloUtente**(Torneo torneo,Giocatore giocatore, Organizzatore organizzatore) | Consente all’organizzatore di visualizzare il profilo dei giocatori iscritti ad un suo torneo |
| public List<Giocatore> **getPartecipanti(**Torneo torneo) | Consente di ottenere la lista dei partecipanti di un dato torneo |
| public Torneo **findByName**(String nome) | Consente di ottenere il torneo avente un determinato nome |
| Public List<Torneo> **getTorneoIscritto**(Giocatore giocatore) | Consente al giocatore di ottenere la lista dei tornei a cui è iscritto |
| Public String **iscrizioneTorneo**(Giocatore giocatore, Torneo torneo) | Consente al giocatore di iscriversi ad un torneo |
| Public List<Torneo> **cercareTorneo**(String nome) | Consente al giocatore di cercare dei tornei dato un nome. |
| public int **countTorneiTotali**() | Consente di ottenere il numero dei tornei totali |

## 3.4 ModeratoreService

|  |  |
| --- | --- |
| **ModeratoreService** |  |
| **Metodo** | **Descrizione** |
| public String **login (**Moderatore moderatore) | Consente al moderatore di effettuare l’autenticazione |
| public Moderatore **findByUsername**(String username) | Consente di cercare un moderatore dato il suo username |
| public boolean **bannare (**Moderatore moderatore, Utente utente, String ruolo) | Consente al moderatore di poter bannare un utente con il ruolo di giocatore o di organizzatore |
| public boolean **sbannare**(Moderatore moderatore, Utente utente, String ruolo) | Consente al moderatore di poter sbannare un utente con il ruolo di giocatore o di organizzatore |
| public boolean **accettare (**Moderatore moderatore, Organizzatore organizzatore) | Consente al moderatore di poter accettare una richiesta di creazione di un profilo organizzatore |
| public boolean **rifiutare (**Moderatore moderatore, Organizzatore organizzatore) | Consente al moderatore di poter rifiutare una richiesta di creazione di un profilo organizzatore |
| public List<Richiesta> **notifiche**() | Consente di ottenere tutte le richieste degli organizzatori |

# 4.Object Constrain Language

## 4.1 GuestService

|  |  |
| --- | --- |
| **GuestService** | Questa classe raggruppa i servizi dedicati all'utente guest |
| **Pre-Condizione** | **Post-Condizione** |
| **context**  GuestService::registrazioneGiocatore(giocatore: Giocatore,confermaPassword: String)    **pre:**  giocatore.username <> null and  giocatore.password <> null and  giocatore.nome <> null and  giocatore.cognome <> null and giocatore.email <> null and  giocatore.email.re("^[a-zA-Z0-9\_+&\*-]+(?:\\.[a-zA-Z0-9\_+&\*-]+)\*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,}$") and  confermaPassword <> null  **pre:**  UtenteService.findGiocatoreByUsername (giocatore.username) = null    **pre:**  giocatore.password <> null and confermaPassword = giocatore.password | **post:**  UtenteService.findGiocatoreByUsername (@pre.giocatore.username) <> null    **post:**  UtenteService.findGiocatoreByUsername (@pre.giocatore.username).stato = Stato.ATTIVO  **post:**  result=true |
| **context** GuestService::registrazioneOrganizzatore(organizzatore: Organizzatore, confermaPassword: String)    **pre:**  organizzatore.username <> null and  organizzatore.password <> null and  organizzatore.nome <> null and  organizzatore.cognome <> null and organizzatore.email <> null and  organizzatore.email.re("^[a-zA-Z0-9\_+&\*-]+(?:\\.[a-zA-Z0-9\_+&\*-]+)\*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,}$") and  confermaPassword <> null    **pre:**  UtenteService::findOrganizzatoreByUsername (organizzatore.username) = null    **pre:**  giocatore.password <> null and confermaPassword = giocatore.password | **post:**  UtenteService::findOrganizzatoreByUsername (@pre.organizzatore.username) <> null    **post:**  UtenteService::findOrganizzatoreByUsername (@pre.organizzatore.username) .stato = Stato.INVERIFICA  **post:**  result=true |

## 4.2 UtenteService

|  |  |
| --- | --- |
| **UtenteService** | Questo servizio fornisce i servizi che accumanano i servizi dei giocatori e organizzatori |
| **Pre-Condizione** | **Post-Condizione** |
| **context**  context UtenteService::login(utente: Utente, ruolo: String)    **pre:**  utente.username <> null and  utente.password <> null  **pre:**  ruolo <> null    **pre:**  UtenteService.findGiocatoreByUsername (utente.username, ruolo) <> null    **pre:**  Utente.password = UtenteService.findGiocatoreByUsername (utente.username, ruolo).password    **pre:**  UtenteService.findGiocatoreByUsername (utente.username, ruolo).stato = Stato.ATTIVO | **post:**  sessions.get(utente.username) <> null  **post:**  result = 'account bannato' if UtenteService.findGiocatoreByUsername (utente.username, ruolo).stato = Stato.BANNATO    **post:**  result = 'account ancora non accettato' if UtenteService.findGiocatoreByUsername (utente.username, ruolo).stato = Stato.INVERIFICA  **post:**  result = 'credenziali sbagliate' se anche una sola condizione sulle variabile non è rispettata  **post:**  result=true |
| **context**  UtenteService::modificaProfilo(utente: Utente, confermaPassword: String)    **pre:**  utente.username <> null and  utente.password <> null and  utente.nome <> null and  utente.cognome <> null and utente.email <> null and  utente.email.re("^[a-zA-Z0-9\_+&\*-]+(?:\\.[a-zA-Z0-9\_+&\*-]+)\*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,}$") and  confermaPassword <> null  **pre:**  UtenteService.findGiocatoreByUsername (utente.username, ruolo) <> null    **pre:**  utente.password = utente.repassword | **post:**  Result = ‘password non corrispondono’ if  UtenteService.findGiocatoreByUsername (utente.username, ruolo).password <> confermaPassword  **post:**  Result = ‘username già esistente’ if  UtenteService.findGiocatoreByUsername (utente.username, ruolo).username == utente.username  **post**:  result=true  **post:**  utente.@pre.nome <> utente.nome  or utente.@pre.cognome <> utente.cognome  or utente.@pre.email <> utente.email  or utente.@pre.password <> utente.password  or utente.@pre.username <> utente.username |
| **context**  UtenteService.findGiocatoreByUsername (username: String)  **pre:**  username <> null and <> ‘ ‘ | **post:**  result <> null implies result.username = username |
| **context**  UtenteService::findOrganizzatoreByUsername (username: String)  **pre:**  username <> null and <> ‘ | **post:**  result <> null implies result.username = username |
| **context**  UtenteService::findOrganizzatoriPaged (number: int, size: int) | **post:**  result->size() <= size |
| **context**  UtenteService:: findGiocatoriPaged (number: int, size: int) | **post:**  result->size() <= size |
| **context**  UtenteService:: countGiocatoriTotali () | **post:**  result = Giocatore.allInstances()-> size() |
| **context**  UtenteService:: countOrganizzatoriTotali () | **post:**  result = Organizzatore.allInstances()-> size() |
| **context**  UtenteService:: countUtentiTotali () | **post:**  result = Giocatore.allInstances()-> size() +  Organizzatore.allInstances()-> size() |

## 4.3 TorneoService

|  |  |
| --- | --- |
| **TorneoService** | Questo servizio raggruppa tutti i servizi che riguardano la gestione dei tornei |
| **Pre-Condizione** | **Post-Condizione** |
| **context**  TorneoService::creaTorneo(torneo: Torneo, organizzatore: Organizzatore)    **pre:**  Torneo <> null and  Organizzatore <> null and  Torneo.nome <> null and <> ‘ ‘ and  Organizzatore.username <> null and <> ‘ ‘  **pre:**  TorneoService.findByName(torneo.nomeTorneo) = null    **pre:**  torneo.capienza <> null and torneo.capienza  compreso tra (4 a 16 divisibile per 4) + 2. | **post:**  TorneoService.findByName(torneo.nome) <> null    **post:**  TorneoService.findByName(torneo.nome).stato = StatoTorneo.ATTESAISCRIZIONI  **post:**  result =true |
| **context**  TorneoService::iniziareTorneo(torneo: Torneo, organizzatore: Organizzatore)    **pre:**  Torneo <> null and  Organizzatore <> null and  Torneo.nome <> null and <> ‘ ‘ and  Organizzatore.username <> null and <> ‘ ‘  **pre:**  torneo.stato = StatoTorneo.ISCRIZIONICOMPLETATE  **pre:**  torneo.organizzatore = organizzatore    **pre:**  torneo.giocatori->size() = torneo.capienza | **post:**  TorneoService.findByName(torneo.nome).stato = StatoTorneo.INCORSO  **post:**  result =true |
| **context**  TorneoService::terminareTorneo(torneo: Torneo, organizzatore: Organizzatore)    **pre:**  Torneo <> null and  Organizzatore <> null and  Torneo.nome <> null and <> ‘ ‘ and  Organizzatore.username <> null and <> ‘ ‘    **pre:**  torneo.stato = StatoTorneo.INCORSO    **pre:**  torneo.organizzatore = organizzatore | **post:**  TorneoService.findByName(torneo.nome).stato = statoTorneo.TORNEOCOMPLETATO  **post:**  result =true |
| **context**  TorneoService::toglierePartecipanti(torneo: Torneo, giocatore: Giocatore, organizzatore: Organizzatore)    **pre:**  Torneo <> null and  Giocatore <> null and  Organizzatore <> null and  Torneo.nome <> null and <> ‘ ‘ and  Giocatore.username <> null and <> ‘ ‘ and  Organizzatore.username <> null and <> ‘ ‘  **pre:**  torneo.organizzatore = organizzatore    **pre:**  torneo.stato = StatoTorneo.ATTESAISCRIZIONI ||  torneo.stato = StatoTorneo.ISCRIZIONICOMPLETATE    **pre:**  torneo.giocatori->includes(giocatore) | **post:**  torneo.giocatori = @pre.torneo.giocatori->excluding(giocatore)  TorneoService.findByName(torneo.nome).stato = statoTorneo.ATTESAISCRIZIONI  **post:**  result = true |
| **context**  TorneoService::visualizzaProfiloUtente(torneo: Torneo, giocatore: Giocatore, organizzatore: Organizzatore)    **pre:**  Torneo <> null and  Giocatore <> null and  Organizzatore <> null and  Torneo.nome <> null and <> ‘ ‘ and  Giocatore.username <> null and <> ‘ ‘ and  Organizzatore.username <> null and <> ‘ ‘  **pre:**  torneo.organizzatore = organizzatore    **pre:**  torneo.giocatori->includes(giocatore) | **post:**  result = Utente. findGiocatoreByUsername (giocatore.username, 'Giocatore') |
| Context  TorneoService::getPartecipanti(torneo: Torneo)  Pre: torneo <> null and Torneo.nome <> null and <> ‘ ‘ | **post:**  result = torneo.getPartecipanti() |
| **context**  TorneoService::findByName(nome: String)    **pre:**  nome <> null and nome <> ‘ ‘ | **post:**  result <> null implies result.nome = nome |
| **context**  TorneoService::getTorneoIscritto(giocatore: Giocatore)    **pre:**  giocatore <> null and  Giocatore.username <> null and <> ‘ ‘ | **post:**  result = giocatore.getTornei() limitato a 10 |
| **context**  Torneo::iscrizioneTorneo(giocatore: Giocatore, torneo: Torneo)    **pre:**  Torneo <> null and  Giocatore <> null and  Torneo.nome <> null and ‘ ‘ and  Giocatore.username <> null and ‘ ‘  **pre:**  torneo.stato = StatoTorneo.ISCRIZIONICOMPLETATE  **pre:**  giocatore.tornei->includes(torneo)  **pre**:  Torneo.giocatori -> size() < torneo.capienza  **pre:**  giocatore.tornei.includes(torneo) -- Il giocatore è già stato presente in passato nel torneo | **post:**  torneo.giocatori=@pre.torneo.giocatori + giocatore  **post:**  giocatore.tornei=@pre.giocatore.tornei + giocatore  **post:**  result = true  **post:**  result = 'iscrizioni piene' if torneo.giocatori->size() >= torneo.capienza  **post:**  result = 'sei stato rimosso dal torneo, non puoi iscriverti' if torneo.giocatori->excludes(giocatore) |
| **context**  TorneoService::cercareTorneo(String name)  **pre:**  nome <> null and nome <> ‘ ‘ | **post:**  result = Torneo->select(t | (t.nome <> null and t.nome->includes(name)) or (t.nome = name) or (name = null) or (name = '')) limitato a 10 |

## 4.4 ModeratoreService

|  |  |
| --- | --- |
| **ModeratoreService** | Questo servizio riguarda tutti i servizi che servono per gestire le operazioni del moderatore |
| **Pre-Condizione** | **Post-Condizione** |
| **context**  ModeratoreService::login(moderatore: Moderatore)    **pre**  moderatore.username <> null and <> ‘ ‘    **pre:**  moderatore.password <> null and <> ‘ ‘    **pre:**  Moderatore.findByUsername(moderatore.username) <> null    **pre:**  moderatore.password = ModeratoreService.findByUsername(moderatore.username).passwordmoderatore.password = Moderatore Service.findByUsername (moderatore.username).password | **post:**  sessions.get(moderatore.username) <> null  **post:**  result=true |
| **context**  ModeratoreService::findByUsername(username String)    **pre:**  username <> null and <> ‘ ‘ | **post:**  result <> null implies result.username = username |
| **context**  ModeratoreService::bannare(moderatore: Moderatore, utente: Utente, ruolo: String)    **pre**  moderatore.username <> null and <> ‘ ‘ and  moderatore.password <> null and <> ‘ ‘ and  Utente.username <> null and <> ‘ ‘ and  Ruolo <> null and <> ‘ ‘  **pre:** ModeratoreService.findByUsername(moderatore.username) <> null    **pre:**  UtenteService.findGiocatoreByUsername (utente.username, ruolo) <> null    **pre:**  UtenteService.findGiocatoreByUsername (utente.username, ruolo).stato = Stato.ATTIVO | **post:**  utente.stato = Stato.BANNATO  **post:**  result= true |
| **context**  ModeratoreService::sbannare(moderatore: Moderatore, utente: Utente, ruolo: String)    **pre**  moderatore.username <> null and <> ‘ ‘ and  moderatore.password <> null and <> ‘ ‘ and  Utente.username <> null and <> ‘ ‘ and  Ruolo <> null and <> ‘ ‘  **pre:** ModeratoreService.findByUsername(moderatore.username) <> null    **pre:**  UtenteService.findGiocatoreByUsername (utente.username, ruolo) <> null    **pre:**  UtenteService.findGiocatoreByUsername (utente.username, ruolo).stato = Stato.BANNATO | **post:**  utente.stato = Stato.ATTIVO  **post:**  result= true |
| **context**  ModeratoreService::accettare(moderatore: Moderatore, organizzatore: Organizzatore)    Pre:  moderatore.username <> null and <> ‘ ‘ and  organizzatore.username <> null and <> ‘ ‘  **pre:** ModeratoreService.findByUsername(moderatore.username) <> null    **pre:** UtenteService::findOrganizzatoreByUsername (organizzatore.username, 'Organizzatore').stato = Stato.INVERIFICA    **pre:** UtenteService::findOrganizzatoreByUsername (organizzatore.username, 'Organizzatore') <> null | **post:**  organizzatore.stato = Stato.ATTIVO  **post:**  result= true |
| **context**  ModeratoreService::rifiutare(moderatore: Moderatore, organizzatore: Organizzatore)    Pre:  moderatore.username <> null and <> ‘ ‘ and  organizzatore.username <> null and <> ‘ ‘  **pre:** ModeratoreService.findByUsername(moderatore.username) <> null    **pre:** UtenteService::findOrganizzatoreByUsername (organizzatore.username, 'Organizzatore').stato = Stato.InVerifica    **pre:** UtenteService::findOrganizzatoreByUsername (organizzatore.username, 'Organizzatore') <> null | **post:**  organizzatore.stato = Stato.RIFIUTATO  **post:**  result= true |
| Context  ModeratoreService::notifiche() | Post:  result <> null implies result.findAll |

# 5 Class Diagram

