Object Design Document

GottaBattleEmAll

[1.Introduzione 1](#_Toc130460945)

[1.1. Trade off 1](#_Toc1831398113)

[1.2. Linee Guida 2](#_Toc580242359)

[1.3.Definizione,Acronimi,abbrevazioni 2](#_Toc193994362)

[1.4 Reference 2](#_Toc1499677108)

[2. Package 3](#_Toc9632077)

[2.1 Package gottaBattleEmAll 3](#_Toc392752904)

[2.1.1 Package config 4](#_Toc238692248)

[2.1.2 Package controller 4](#_Toc3312925)

[2.1.3 Package entity 4](#_Toc2074860145)

[2.1.4 Package repository 4](#_Toc574142277)

[2.1.5 Package service 4](#_Toc898457249)

[2.2 Package resources 4](#_Toc889372605)

[2.2.1 Package fragment 4](#_Toc1908382499)

[2.2.2 Package static 4](#_Toc2040307945)

[3.Interfacce delle classi 4](#_Toc702086016)

[3.1 GuestService 4](#_Toc325053668)

[3.2 UtenteService 4](#_Toc466084007)

[3.3 GiocatoreService 4](#_Toc267984687)

[3.4 TorneoService 4](#_Toc1976249202)

[3.5 PartitaService 5](#_Toc1836119923)

[3.6 ModeratoreService 5](#_Toc1808309589)

# 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 ricaricate ad ogni interazione.

* Sicurezza vs Accessibilità

Il sistema è progettato per avere un giusto equilibrio tra il rischio di vulnerabilità e la complessità di accesso da parte dell’utente. Andando ad implementare i filtri ma non inserendo meccanismi di 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 metodo saranno nominati a seconda della convenzione CamelCase
* Gli oggetti entita saranno chiamati semplicemente con il nome dell'entita
* Gli oggetti saranno nominati a seconda dell' entita di cui si occupano e del loro ruolo nell' applicazione per esempio TorneoService, UtenteController ecc...
* le pagine html in thydeconposizionemeleaf saranno segnate completamente in minuscolo
* i nomi dei package sono in minuscolo

## 1.3.Definizione,Acronimi,abbrevazioni

Acronimi:

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

## 1.4 Reference

- R.A.D.

- S.D.D.

# 2. Package

La decomposizione in package è la seguente:

<eventuale disegno>

il package gottaBattleEmAll , è un package in cui sono contenuti altri package che riguardano le classi Service, Repository, Controller, Configuration(config), Entity e si trova anche un file di startup per lo start del sistema.

il package configuration (per convenzione si chiamerà config) contiene le configurazioni di sistema per operazioni che vengono fatte durante lo startup o la creazione di determinati Bean.

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 di CRUD per ogni singola entità.

Il package service contiene l’interfaccia e l’implementazione dei servizi.

il package resources è composto da: le pagine html, il package fragment e il package static. Rappresentano le view del sistema.

il package fragment contiene le sottopagine html che possibile ripetere su piu pagine

il package static contiene le risorse statiche per le pagine html

Tali package nascono dall’utilizzo della tecnologia Spring all’interno dell’architettura su 3 livelli:

1. Presentation Layer:
   * Controller: responsabile di gestire le richieste della view utilizzando i controller.
   * View: layer responsabile di visualizzare i dati forniti dal model, utilizzando componenti come Thymeleaf.
2. Application Layer:

* Model: contiene la logica di business con l’utilizzo di service e component, inoltre si occupa della gestione dei dati con l’ausilio degli entity bean e i repository.

1. Data Access Layer:  
    Comunica con il Model con Spring data JPA che facilita la gestione della persistenza e delle operazioni.

## 2.1 Package gottaBattleEmAll

### 2.1.1 Package config

### 2.1.2 Package controller

### 2.1.3 Package entity

### 2.1.4 Package repository

### 2.1.5 Package service

## 2.2 Package resources

### 2.2.1 Package fragment

2.2.2 Package static

# 3.Interfacce delle classi

## 3.1 GuestService

|  |  |
| --- | --- |
| **GuestService** | |
| **Metodo** | **Descrizione** |
| Public boolean registrazioneGiocatore(Giocatore giocatore) | Servizio che consente all’utente di registrarsi come giocatore con le credenziali fornite. |
| Public boolean registrazioneOrganizzatore(Organizzatore organizzatore) | Servizio che 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) | Consente all’utente registrato di modificare il proprio profilo utente. |
| Public void logout(Utente utente) | Consente all’utente autenticato di uscire dal sistema |
| public Utente findByUsername(String username,String ruolo) | Consente di ricercare un utente dato l’username e il suo ruolo |

## 3.3 GiocatoreService

|  |  |
| --- | --- |
| GiocatoreService | |
| Metodo | Descrizione |
| public String sostituireMembroTeam(Pokemon pokemon, int posizione, Giocatore giocatore) | Consente al giocatore di sostituire un membro del suo team Pokemon |

## 3.4 TorneoService

|  |  |
| --- | --- |
| **TorneoService** | |
| **Metodo** | **Descrizione** |
| Public List<Torneo> cercareTorneo(String nome) | Consente al giocatore di cercare dei tornei dato un nome. |
| Public String iscrizioneTorneo(Giocatore giocatore, Torneo torneo) | Consente al giocatore di iscriversi ad un torneo |
| public boolean seguireOrganizzatore(Giocatore giocatore, Organizzatore organizzatore) | Consente al giocatore di seguire un utente organizzatore |
| Public List<Torneo> getTorneoIscritto(Giocatore giocatore) | Consente al giocatore di ottenere la lista dei tornei a cui è iscritto |
| 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 Torneo findByName(String nome) | Consente di ottenere il torneo avente un determinato nome |

## 3.5 PartitaService

|  |  |
| --- | --- |
| PartitaService | |
| Metodo | Descrizione |
| public boolean aggiungereRisultato(String risultato, Partita partita, Torneo torneo, Organizzatore organizzatore) | Consente all’organizzatore di poter inserire il risultato finale al termine di una partita di un suo torneo |

## 3.6 ModeratoreService

|  |  |
| --- | --- |
| **ModeratoreService** |  |
| **Metodo** | **Descrizione** |
| public String login(Moderatore moderatore) | Consente al moderatore di effettuare l’autenticazione |
| public boolean logout(Moderatore moderatore) | Consente al moderatore di uscire dal sistema |
| 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 Moderatore findByUsername(String username) | Consente di cercare un moderatore dato il suo username |

|  |  |
| --- | --- |
| GuestService | Questa classe raggruppa i servizi dedicati all'utente guest |
| Pre-Condizione | Post-Condizione |
| context GuestService::registrazioneGiocatore(giocatore: Giocatore)    pre: giocatore.username <> null and  UtenteService.findByUsername(giocatore.username, 'giocatore') = null    pre: giocatore.nome <> null    pre: giocatore.cognome <> null    pre: giocatore.email <> null and giocatore.email.re('^\\w+([-+.]\\w+)\*@\\w+([-.]\\w+)\*\\.\\w+([-.]\\w+)\*$')    pre: giocatore.password <> null and giocatore.passwordmatch = giocatore.password | post: UtenteService.findByUsername(@pre.giocatore.username, ‘giocatore’) <> null  post: UtenteService.findByUsername(@pre.giocatore.username, 'GIOCATORE').stato = Stato.ATTIVO  post: result=true |
| **context context** GuestService::registrazioneOrganizzatore(organizzatore: Organizzatore)    pre: organizzatore.username <> null and  UtenteService.findByUsername(organizzatore.username, 'organizzatore') = null    pre: organizzatore.nome <> null    pre: organizzatore.cognome <> null    pre: organizzatore.email <> null and organizzatore.email.re('^\\w+([-+.]\\w+)\*@\\w+([-.]\\w+)\*\\.\\w+([-.]\\w+)\*$')    pre: organizzatore.password <> null and organizzatore.passwordmatch = organizzatore.password | post: UtenteService.findByUsername(@pre.organizzatore.username, 'organizzatore') <> null    post: UtenteService.findByUsername(@pre.organizzatore.username, 'organizzatore').stato = Stato.INVERIFICA  //TODO gestore delle richieste  **post:** result=true |

|  |  |
| --- | --- |
| 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    pre: utente.password <> null    pre: utente.ruolo <> null    pre: UtenteService.findByUsername(utente.username, ruolo) <> null    pre: utente.password = UtenteService.findByUsername(utente.username, ruolo).password    pre: UtenteService.findByUsername(utente.username, ruolo).stato = Stato.Attivo  bisognerebbe distinguere i vari casi(ban e in verifica, rifiutato) | post: sessions.get(utente.username) <> null  //FIXME la sintassi non e perfettamente corretta  post: result = 'utente bannato' if UtenteService.findByUsername(utente.username, ruolo).stato = 'bannato'    post: result = 'utente in verifica' if UtenteService.findByUsername(utente.username, ruolo).stato = 'in verifica'  **post:** result = 'credenziali sbagliate' se anche una sola condizione sulle variabile non vale  **post:** result=true  //HACK non so se va bene scritto cosi |
| context UtenteService::modificaProfilo(utente: Utente)    pre: utente.username <> null    pre: utente.password <> null    pre: utente.email <> null and utente.email.re('^\\w+([-+.]\\w+)\*@\\w+([-.]\\w+)\*\\.\\w+([-.]\\w+)\*$')    pre: utente.nome <> null    pre: utente.cognome <> null    pre: UtenteService.findByUsername(utente.username, ruolo) <> null    pre: utente.password = utente.repassword | post: UtenteService.findByUsername(utente.username, @pre.ruolo).nome = utente.nome  and UtenteService.findByUsername(utente.username, @pre.ruolo).cognome = utente.cognome  and UtenteService.findByUsername(utente.username, @pre.ruolo).email = utente.email  and UtenteService.findByUsername(utente.username, @pre.ruolo).password = utente.password  and UtenteService.findByUsername(utente.username, @pre.ruolo).username = utente.username  **post**: result=true |
| context UtenteService::logout(Utente: Utente)    pre: Utente.username <> null  pre: sessions.get(Utente.username) <> null | post: sessions = @pre.sessions->excluding (Utente.username)  //HACK una possibile soluzione |
| context UtenteService::findByUsername(username: String, ruolo: String)    pre: username <> null and ruolo <> null | post: result <> null implies (result.username = username and result.ruolo = ruolo)  //FIXME da sistemare |

|  |  |
| --- | --- |
| TorneoService | questo servizio raggruppa tutti i servizi che riguardano la gestione dei tornei |
| Pre-Condizione | Post-Condizione |
| context TorneoService::findByName(nome: String)    pre: nome <> null  //NOTE questa funzione deve essere usata internamente e solo per la pagina torneo lato utente , lato organizzatore ci sara un'altra con il controllo dell'organizzatore | post: result <> null implies result.nome = nome  //FIXME da sistemare |
| **context** TorneoService::cercareTorneo(String name) | 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 |
| context Torneo::iscrizioneTorneo(giocatore: Giocatore, torneo: Torneo)    pre: giocatore <> null  pre: torneo <> null  pre: torneo.stato = Stato.AttesaIscrizioni  pre: not 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::seguireOrganizzatore(giocatore: Giocatore, organizzatore: Organizzatore)    pre: giocatore <> null  pre: organizzatore <> null | post: giocatore.organizzatoriSeguiti = @pre.giocatore.organizzatoriSeguiti->including (organizzatore)  **post:** result =true |
| context TorneoService::getTorneoIscritto(giocatore: Giocatore)    pre: giocatore <> null | post: result = giocatore.getTornei() limitato a 10 |
| context TorneoService::creaTorneo(torneo: Torneo, organizzatore: Organizzatore)    pre: torneo.nomeTorneo <> null    pre: TorneoService.findByName(torneo.nomeTorneo) = null    pre: torneo.capienza <> null and torneo.capienza <= 16 and torneo.capienza % 2 = 0    pre: organizzatore <> null | post: TorneoService.findByName(torneo.nome) <> null    post: TorneoService.findByName(torneo.nome).stato = AttesaIscrizioni  **post:** result =true |
| context TorneoService::iniziareTorneo(torneo: Torneo, organizzatore: Organizzatore)    pre: torneo <> null    pre: torneo.stato = AttesaIscrizioni    pre: torneo.organizzatore = organizzatore    pre: torneo.giocatori->size() = torneo.capienza | post: TorneoService.findByName(torneo.nome).stato = Incorso  //TODO qua bisognerebbe mettere delle cose per fare creare le partite in un determinato ordine //post: torneo.PartitaService.creaPartita(torneo)  **post:** result =true |
| context TorneoService::terminareTorneo(torneo: Torneo, organizzatore: Organizzatore)    pre: torneo <> null    pre: torneo.stato = InCorso    pre: torneo.organizzatore = organizzatore | post: TorneoService.findByName(torneo.nome).stato = TorneoCompletato  **post:** result =true |
| context TorneoService::toglierePartecipanti(torneo: Torneo, giocatore: Giocatore, organizzatore: Organizzatore)    pre: torneo.organizzatore = organizzatore    pre: torneo <> null    pre: torneo.stato = AttesaIscrizioni    pre: torneo.giocatori->includes(giocatore) | post: torneo.giocatori = @pre.torneo.giocatori->excluding(giocatore)  **post:** result =true |
| context TorneoService::visualizzaProfiloUtente(torneo: Torneo, giocatore: Giocatore, organizzatore: Organizzatore)    pre: organizzatore <> null and torneo.organizzatore = organizzatore    pre: torneo <> null    pre: torneo.giocatori->includes(giocatore) | post: result = UtenteService.findByUsername(giocatore.username, 'Giocatore') |

|  |  |
| --- | --- |
| PartitaService | Questo servizio riguarda tutti i servizi che servono per gestire la singola partita |
| Pre-Condizione | Post-Condizione |
| context PartitaService::aggiungereRisultato(risultato: String, partita: Partita, torneo: Torneo, organizzatore: Organizzatore)    pre: partita <> null    pre: organizzatore <> null and organizzatore = torneo.organizzatore    pre: torneo <> null    pre: torneo.partite->includes(partita) | post: partita.idGiocatore = risultato |

|  |  |
| --- | --- |
| 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    pre: moderatore.password <> null    pre: ModeratoreService.findByUsername(moderatore.username) <> null    pre: moderatore.password = ModeratoreService.findByUsername(moderatore.username).password | post: sessions.get(moderatore.username) <> null  //FIXME la sintassi non e perfettamente corretta  **post:** result=true |
| **context** ModeratoreService::logout(Moderatore moderatore)  **pre:** moderatore.username!=null  **pre:** sessions.get(moderatore.username)!=null | post: sessions = @pre.sessions->excluding (Moderatore.username)  //HACK una possibile soluzione |
| context ModeratoreService::bannare(moderatore: Moderatore, utente: Utente, ruolo: String)    pre: ModeratoreService.findByUsername(moderatore.username) <> null    pre: UtenteService.findByUsername(utente.username, ruolo) <> null    pre: UtenteService.findByUsername(utente.username, ruolo).stato = Stato.ATTIVO | post: utente.stato = Stato.BANNATO  **post:** result= true |
| context ModeratoreService::sbannare(moderatore: Moderatore, utente: Utente, ruolo: String)    pre: ModeratoreService.findByUsername(moderatore.username) <> null    pre: UtenteService.findByUsername(utente.username, ruolo) <> null    pre: UtenteService.findByUsername(utente.username, ruolo).stato = Stato.BANNATO | post: utente.stato = Stato.ATTIVO  **post:** result= true |
| context ModeratoreService::accettare(moderatore: Moderatore, organizzatore: Organizzatore)    pre: ModeratoreService.findByUsername(moderatore.username) <> null    pre: UtenteService.findByUsername(organizzatore.username, 'Organizzatore').stato = Stato.INVERIFICA    pre: UtenteService.findByUsername(organizzatore.username, 'Organizzatore') <> null | post: organizzatore.stato = Stato.ATTIVO  **post:** result= true |
| context ModeratoreService::rifiutare(moderatore: Moderatore, organizzatore: Organizzatore)    pre: ModeratoreService.findByUsername(moderatore.username) <> null    pre: UtenteService.findByUsername(organizzatore.username, 'Organizzatore').stato = Stato.InVerifica    pre: UtenteService.findByUsername(organizzatore.username, 'Organizzatore') <> null | post: organizzatore.stato = Stato.RIFIUTATO  **post:** result= true |
| context ModeratoreService::findByUsername(username String)    pre: username <> null | post: result <> null implies result.username = username |

|  |  |
| --- | --- |
| GiocatoreService | questo servizio fornisce i servizi specifici per i giocatori |
| Pre-Condizione | Post-Condizione |
| context GiocatoreService::sostituireMembroTeam(pokemon: Pokemon, posizione: int, giocatore: Giocatore)    //check  pre: giocatore <> null and UtenteService.findByUsername(giocatore.username, 'Giocatore') <> null  pre: posizione >= 0 and posizione <= 5 | post: UtenteService.findByUsername(giocatore.username, 'RuoloGiocatore').pokemons->at(posizione) = pokemon  **post:** result=true |

|  |  |
| --- | --- |
|  |  |
| Pre-Condizione | Post-Condizione |