System document design

GottaBattleEmAll

Sommario

[1.Introduzione 2](#_Toc22147821)

[1.1 Scopo del sistema 2](#_Toc901584330)

[1.2 Design goals 2](#_Toc807988704)

[1.2.1 Criteri di Performance 2](#_Toc656370791)

[1.2.2 Criteri di Affidabilità 2](#_Toc841322933)

[1.2.3 Criteri di Manutenzione 3](#_Toc2023442571)

[1.2.4 Criteri dell'utente finale 3](#_Toc447634800)

[1.3 Definizioni, acronimi ed abbreviazioni 3](#_Toc1039358839)

[1.4 Riferimenti 3](#_Toc390108955)

[1.5 Panoramica 3](#_Toc1274845026)

[2. Architettura del Sistema 3](#_Toc1274148724)

[2.1 Panoramica 3](#_Toc1886353884)

[2.2 Decomposizione del sistema 4](#_Toc587306910)

[2.2.1 Design pattern 4](#_Toc621013090)

[2.3 Mapping hardware/software 4](#_Toc303171994)

[2.4 Gestione dati persistenti 5](#_Toc1365644436)

[2.5 Controllo e sicurezza degli accessi 5](#_Toc1696129218)

[2.5.1 Matrice degli accessi GestioneUtente 5](#_Toc558956519)

[2.5.2 Matrice degli accessi GestioneTornei 5](#_Toc1258614545)

[2.5.3 Matrice degli accessi GestioneModeratore 5](#_Toc847333321)

[2.6 Controllo software globale 5](#_Toc149338432)

[2.7 Boundary conditions 5](#_Toc525734375)

[2.7.1 Start-up 6](#_Toc537276724)

[2.7.2 Shutdown 6](#_Toc854538790)

[2.7.3 Error Behavior of the system 6](#_Toc1541613691)

[3. Servizi del sottosistema Glossario 6](#_Toc1087239156)

[3.1 Servizi della Gestione Utente 6](#_Toc1313303931)

[3.2 Servizi della Gestione Torneo 6](#_Toc2074044208)

[3.3 Servizi della Gestione Moderatore 7](#_Toc1132383619)

# 1.Introduzione

## 1.1 Scopo del sistema

Lo scopo del sistema è fornire un supporto completo agli appassionati di Pokémon per organizzare e partecipare a tornei. Si tratta di un sito web dedicato alla gestione efficiente degli aspetti organizzativi e partecipativi dei tornei Pokémon. Per i partecipanti, il sistema consente di iscriversi ai tornei, costruire il proprio player, confrontarsi con altri giocatori, visualizzare e iscriversi a tornei, gestire il proprio profilo e tenere traccia delle partite passate. Gli scontri tra giocatori possono avvenire attraverso software online esterno o utilizzando qualsiasi console nello stesso luogo. Per gli organizzatori, il sistema fornisce strumenti di amministrazione per definire regole del torneo, gestire iscrizioni, monitorare l'avanzamento delle competizioni e pubblicare risultati, consentendo loro di concentrarsi sulla logistica e sulle dinamiche del torneo.

## 1.2 Design goals

### 1.2.1 Criteri di Performance

#### 1.2.1.a Tempi di risposta

Il sistema deve garantire un tempo massimo di risposta di 6 secondi per soddisfare qualsiasi funzionalità per garantire una navigazione fluida.

#### 1.2.1.b Throughput

Il sistema deve garantire l’accesso simultaneo di 96 utenti, garantendo tempi di risposta accettabili.

### 1.2.2 Criteri di Affidabilità

#### 1.2.2.a Robustezza

Il sistema è progettato per garantire la gestione affidabile di input utente non validi, fornendo una risposta robusta e prevenendo comportamenti imprevisti.

#### 1.2.2.b Persistenza dati

Il sistema è progettato per garantire una persistenza affidabile dei dati, assicurando che le informazioni cruciali siano conservate in modo sicuro e recuperabili in caso di necessità. La struttura di persistenza dei dati è orientata a fornire coerenza e integrità nell'archiviazione delle informazioni.

#### 1.2.2.c Disponibilità

Il sistema è progettato per assicurare un'elevata percentuale di disponibilità, consentendo agli utenti di svolgere normalmente le attività senza interruzioni significative.

#### 1.2.2.d Sicurezza

Il sistema è progettato per resistere a potenziali attacchi dannosi, proteggendo l'integrità dei dati e garantendo la sicurezza globale del sistema contro intrusioni malevoli.

### 1.2.3 Criteri di Manutenzione

#### 1.2.3.a Modificabilità

Il sistema è progettato con l'obiettivo di offrire facilità nella modifica della funzionalità, consentendo adeguamenti efficienti per soddisfare requisiti in evoluzione senza compromettere la sua stabilità.

#### 1.2.3.b Leggibilità

Il sistema pone enfasi su un codice chiaro e comprensibile, migliorando la leggibilità per agevolare la comprensione e la manutenzione da parte degli sviluppatori

### 1.2.4 Criteri dell'utente finale

#### 1.2.4.a Usabilità

Il sistema è progettato con un'interfaccia intuitiva e user-friendly, mirando a garantire un'esperienza utente agevole e accessibile. La chiarezza nella navigazione e nell'utilizzo delle funzionalità promuove un ambiente interattivo e di facile comprensione per gli utenti.

#### 1.2.4.b Accessibilità

Il sistema è progettato per essere accessibile da qualsiasi dispositivo dotato di un web browser, facilitato da un layout responsive che si adatta automaticamente alle dimensioni dello schermo per garantire un'esperienza utente ottimale su diverse piattaforme.

## 1.3 Definizioni, acronimi ed abbreviazioni

* RAD: Requirement Analysis Document
* DBMS: Database Management System

## 1.4 Riferimenti

RAD \_GottaBattleEmAll

## 1.5 Panoramica

Nel seguente documento verrà descritto inizialmente l’architettura software del sistema e successivamente dei servizi offerti dai vari sottosistemi.

# 2. Architettura del Sistema

## 2.1 Panoramica

Il sistema si basa sul modello Three-tier caratterizzato dalla separazione di 3 livelli, ciascuno con responsabilità distinte:

* Presentation Layer:

Le viste sono responsabili di presentare i dati all'utente in un formato comprensibile, e interagiscono con l’application layer per inviare richieste e ricevere dati.

* Application Layer:

I controller si interfacciano con i servizi applicativi per coordinare le azioni richieste dagli utenti e delegare la logica di business appropriata.

* Data Access Layer:  
  Questo livello si occupa dell'accesso ai dati e della persistenza.

## 2.2 Decomposizione del sistema

I sottosistemi individuali con le relative funzionalità:

* GUI\_GestioneUtente: si occupa della gestione delle interfacce dell’utente.
* GUI\_GestioneModeratore: si occupa della gestione delle interfacce del moderatore.
* GestioneUtente: garantisce le funzionalità relative all’autenticazione, registrazione, modifica di un profilo utente: giocatore e organizzatore
* GestioneModeratore: garantisce le funzionalità relative al moderatore
* GestioneTornei: garantisce le funzionalità di creazione,iscrizione,terminazione ed altre funzionalità del torneo.
* UserStorage: si occupa di conservare i dati degli user.
* ModeratoreStorage: si occupa di conservare i dati del moderatore.

//inserire l’immagine Subsystem Diagram

### 2.2.1 Design pattern

I Design Pattern utilizzati sono:

* + Inversion of control

L'Inversion of Control (IoC) è un pattern che ribalta il tradizionale flusso di controllo in un'applicazione. In questo contesto, un componente di livello applicativo riceve il controllo da un componente appartenente a una libreria riusabile, anziché dettarlo direttamente. Questo approccio è spesso implementato mediante il pattern Dependency Injection, che utilizza metadati per istruire i contenitori su come gestire le dipendenze tra i vari componenti del sistema. Questo modo di operare favorisce una maggiore modularità e facilità di manutenzione del codice.

## 2.3 Mapping hardware/software

//inserire l’immagine Deployment Diagram

Il sistema è suddiviso in tre parti abbiamo:

1. **Utenti/Desktop e Mobile**: Gli utenti interagiscono con il sistema attraverso un WebClient. Questa interazione avviene tramite il protocollo HTTP, che facilita la comunicazione tra il client e il server.
2. **Server**: Al centro del sistema c’è il server, che ospita il Web Server Apache Tomcat. Questa applicazione gestisce le richieste degli utenti e si interfaccia con lo storage del sistema.
3. **Storage**: Lo storage del sistema contiene un DBMS MySQL. Il server accede a questo DBMS tramite il protocollo JDBC per recuperare o memorizzare i dati necessari.

## 2.4 Gestione dati persistenti

La gestione dei dati persistenti è affidata a un sistema di gestione di basi di dati relazionali. Per garantire una corretta separazione delle responsabilità e favorire la modularità del sistema, si adotta un approccio che incapsula l'accesso ai dati. Questo è realizzato attraverso l'utilizzo di un Object-Relational Mapping (ORM), che semplifica le operazioni di accesso ai dati e consente di rappresentare gli oggetti dell'applicazione in modo più diretto nella base di dati relazionale. L'ORM agisce come uno strato intermedio tra le entità del sistema e la base di dati, facilitando la gestione delle operazioni CRUD (Create, Read, Update, Delete).

## 2.5 Controllo e sicurezza degli accessi

Di seguito riportiamo la matrice degli accessi, che serve per determinare quali oggetti sono condivisi tra i vari utenti.

### 2.5.1 Matrice degli accessi GestioneUtente

|  |  |  |
| --- | --- | --- |
| Attori | Classe | Funzioni |
| Guest | GuestService | registrazioneGiocatore() |
| Utente(Generico) | UtenteService | login()  modificaProfilo()  logout() |
| Giocatore(Utente) | GiacatoreService | sostituireMembroTeam() |

### 2.5.2 Matrice degli accessi GestioneTornei

|  |  |  |
| --- | --- | --- |
| Attori | Classe | Funzioni |
| Giocatore(Utente) | TorneoService | cercareTorneo()  iscrizioneTorneo()  seguireOrganizzatore()  getTorneoIscritto() |
| Organizzatore(Utente) | TorneoService | creaTorneo()  iniziareTorneo()  terminareTorneo()  toglierePartecipanti()  visualizzaProfiloUtente() |
| Organizzatore(Utente) | PartitaService | aggiungereRisultato() |

### 2.5.3 Matrice degli accessi GestioneModeratore

|  |  |  |
| --- | --- | --- |
| Attori | Classe | Funzioni |
| Moderatore | ModeratoreService | login()  logout()  bannare/sbannare()  accettare/rifiutare() |

## 2.6 Controllo software globale

Le richieste sono gestite attraverso il Web Server, che funge da intermediario per il sistema. Inizialmente, vengono sottoposte a filtri dedicati per operazioni di autenticazione e gestione delle sessioni. Successivamente le richieste vengono smistate ai rispettivi controller. Questi controller, a loro volta, interagiscono con i servizi (services) che contengono la logica di business del sistema. Una volta completate le elaborazioni, le risposte vengono inviate dal controller al client.

## 2.7 Boundary conditions

### 2.7.1 Start-up

Per l’inizializzazione del sistema occorre avviare il DBMS, in particolare MySQL per accedere ai dati persistenti. L'apertura della connessione al database MySQL è gestita attraverso JDBC. Successivamente bisogna avviare il Web Server, nonché Tomcat dove avverrà il deploy della nostra applicazione, la espone le funzionalità del sistema in modo trasparente agli utenti.

Il primo Start-up fa eccezione in quanto deve prevedere l’inserimento degli n account di tipo moderatore richiesti, l’inserimento di tali account è a carico dell’Admin del sistema.

### 2.7.2 Shutdown

In questa fase avviene lo spegnimento del DBMS e del Web Server, prima però l’applicazione si assicura che tutte le attività in corso siano completate prima della terminazione, minimizzando l'impatto sugli utenti e garantendo una chiusura senza perdita di dati.

### 2.7.3 Error Behavior of the system

In caso di problemi di connessione al database, il sistema ritenta la connessione.

In caso di problemi di alimentazione viene effettuato il ripristino all’ultimo stato persistente.

# 3. Servizi del sottosistema Glossario

## 3.1 Servizi della Gestione Utente

|  |  |
| --- | --- |
| **Servizio** | **Descrizione** |
| registrazioneGiocatore() | Servizio che consente all’utente di registrarsi come giocatore |
| registrazioneOrganizzatore() | Servizio che consente all’utente di registrarsi come organizzatore |
| login() | Consente all’utente registrato di effettuare l’autenticazione |
| modificaProfilo() | Consente all’utente registrato di modificare il proprio profilo utente. |
| logout() | Consente all’utente autenticato di uscire dal sistema |
| sostituireMembroTeam() | Consente al giocatore di sostituire un membro del suo team Pokemon |

## 3.2 Servizi della Gestione Torneo

|  |  |
| --- | --- |
| **Servizio** | **Descrizione** |
| cercareTorneo() | Consente al giocatore di cercare un torneo |
| iscrizioneTorneo() | Consente al giocatore di iscriversi ad un torneo |
| seguireOrganizzatore() | Consente al giocatore di seguire un utente organizzatore |
| getTorneoIscritto() | Consente al giocatore di ottenere la lista dei tornei a cui è iscritto |
| creaTorneo() | Consente all’organizzatore di creare il proprio torneo |
| iniziareTorneo() | Consente all’organizzatore di far iniziare un proprio torneo |
| terminareTorneo() | Consente all’organizzatore di far terminare un proprio torneo |
| toglierePartecipanti() | Consente all’organizzatore di togliere un n giocatori iscritti ad un suo torneo |
| visualizzaProfiloUtente() | Consente all’organizzatore di visualizzare il profilo dei giocatori iscritti ad un suo torneo |
| aggiungereRisultato() | Consente all’organizzatore di poter inserire il risultato finale al termine di una partita |

## 3.3 Servizi della Gestione Moderatore

|  |  |
| --- | --- |
| **Servizio** | **Descrizione** |
| login() | Consente al moderatore di effettuare l’autenticazione |
| logout() | Consente al moderatore di uscire dal sistema |
| bannare/sbannare() | Consente al moderatore di poter bannare o sbannare un giocatore o un organizzatore |
| accettare/rifiutare() | Consente al moderatore di poter accettare o rifiutare una richiesta di creazione di un profilo organizzatore |