Laurea Triennale in informatica-Università di Salerno
Corso di *Ingegneria del Software*- Prof. C. Gravino

System Design Document

SDD

Riferimento	
Versione	0.6
Data	19/12/2025
Destinatario	Docente Ingegneria del Software Prof. Carmine Gravino
Presentato da	Gaetano Aprile, Luigi Artuso, Alessandro De Bonis & Marco Galdi
Approvato da	

Revision History

Data	Versione	Descrizione	Autori
01/11/2025	0.1	Introduzione del nostro sistema, specificando il sistema attuale.	Team
02/11/2025	0.2	Decomposizione del sistema in sottosistemi e Component Diagram	Team
17/11/2025	0.3	Mapping Hardaware/Software	Team
26/11/2025	0.4	Gestione dei Dati Persistenti	Team
12/12/2025	0.5	Controllo degli Accessi e Controllo Globale	Team
19/12/2025	0.6	Reiterazione ed aggiunta delle Condizioni Limite	Team

Indice

1. Introduzione

1.1 Scopo del Sistema

Il sistema MatchPoint è concepito per superare il persistente ostacolo logistico e la difficoltà nel coordinamento che caratterizzano l'organizzazione di eventi sportivi amatoriali. L'attuale panorama è afflitto dalla difficoltà di riunire, in tempi rapidi, un numero adeguato di partecipanti.

La visione di MatchPoint è quella di fornire a tutti gli sportivi italiani un'applicazione web-based dinamica e geolocalizzata che non solo faciliti la ricerca di eventi vicini, ma che risolva il problema di fondo della mancanza di affidabilità, ad esempio quando partecipanti non si presentano all'incontro. Questo viene affrontato introducendo un sistema di feedback basato su rating che funge da metrica di reputazione per la community.

1.2 Obiettivi di Design (Design Goals)

Nella presente sezione si andranno a presentare i Design Goals, ovvero le qualità sulle quali il sistema deve essere focalizzato, formalizzati esplicitamente così che qualsiasi importante decisione di design può essere fatta consistentemente seguendo lo stesso insieme di design goal.

I design goals sono suddivisi nelle seguenti categorie:

- **Performance**, requisiti di spazio e velocità.
- **Dependability**, determinano lo sforzo da spendere per minimizzare i fallimenti e le loro conseguenze.
- **Maintenance**, determina lo sforzo necessario per modificare il sistema dopo il rilascio.
- **End User**, includono qualità desiderabili dall'utente, non coperte da Performance e Dependability.

ID	Nome	Categoria	Descrizione	RNF di origine
DG1	Bassa latenza di transazione	Performance	Le operazioni che modificano lo stato del sistema (come l'iscrizione o l'annullamento) devono avere una risposta quasi istantanea, massimo 1 secondo.	RNF7
DG2	Scalabilità geografica	Performance	L'architettura deve essere progettata per supportare l'intera base di utenti su tutto il territorio italiano e gestire un aumento potenziale degli eventi e degli utenti, assicurando che la capacità di memorizzazione sia alta.	RNF8
DG3	Precisione funzionale e affidabilità	Dependability	Il sistema deve garantire il 100% di accuratezza nella logica transazionale (iscrizione, annullamento, gestione del quorum). Tutte le operazioni avvengono con successo.	RNF4
DG4	Separazione rigorosa dei privilegi	Dependability	Il design deve implementare una separazione netta delle operazioni in base ai ruoli degli utenti, garantendo che le azioni sensibili siano eseguite solo dagli attori.	RNF5

DG5	Aggiornabilità continua	Maintenance	L'architettura deve supportare l'aggiornamento del sistema (deployment) in modo incrementale, riducendo al minimo le interruzioni prolungate del servizio.	RNF9
DG6	Portabilità e compatibilità browser	Maintenance	Il front-end deve essere sviluppato per garantire la piena compatibilità e funzionalità con i principali browser (Chrome, Firefox, Edge, Safari)	RNF10
DG7	Interfaccia intuitiva e immediata	End User	L'intero sistema deve permettere l'utilizzo in modo immediato, consentendo all'utente di completare le azioni senza consultare documentazione.	RNF3
DG8	Chiarezza Visiva	End User	Il design dell'interfaccia deve rendere ogni elemento visuale chiaro e semplice, esplicitando la sua funzionalità all'utente	RNF2
DG9	Accessibilità e comprensione	End User	L'interfaccia deve essere progettata per essere utilizzabile da un'utenza meno esperta e deve risultare di facile comprensione per tutti gli sportivi.	RNF1
DG10	Design responsivo	End User	L'interfaccia grafica deve essere progettata per essere responsive, garantendo un'esperienza coerente su dispositivi con diverse dimensioni dello schermo	RNF6

Trade-off	Descrizione
Bassa latenza vs. Scalabilità geografica	Per garantire il tempo di risposta massimo di 1 secondo su query geolocalizzate, è necessario ricorrere a soluzioni di memoria veloci (es. caching). Questo riduce la latenza, ma aumenta notevolmente il costo di hosting e l'infrastruttura di archiviazione.
Affidabilità vs. Interfaccia responsiva	Per garantire la completa coerenza e l'affidabilità delle transazioni, il sistema deve usare processi sincroni e blocchi che assicurano il quorum o l'assegnazione dei posti. Questo garantisce l'integrità dei dati, ma rallenta leggermente il feedback utente, compromettendo la percezione di immediatezza e fluidità

1.3 Definizioni, acronimi, e abbreviazioni

Vengono riportati di seguito alcune definizioni presenti nel documento corrente:

- **Sottosistema**: un sottoinsieme dei servizi del dominio applicativo, formato da servizi legati da una relazione funzionale.
- **Design Goal**: le qualità sulle quali il sistema deve essere focalizzato.
- **Dati Persistenti**: dati che sopravvivono all'esecuzione del programma che li ha creati e che dunque vengono salvati.
- **Mapping Hardware/Software**: studio della connessione tra parti fisiche e logiche di cui si compongono il sistema.
- **SDD**: System Design Document
- **RAD**: Requirements Analysis Document

1.4 Organizzazione del documento

Il presente documento di System Design è articolato in tre sezioni principali, come segue:

- **Introduzione**: viene descritto in generale lo scopo del sistema, gli obiettivi di design che il sistema propone di raggiungere.

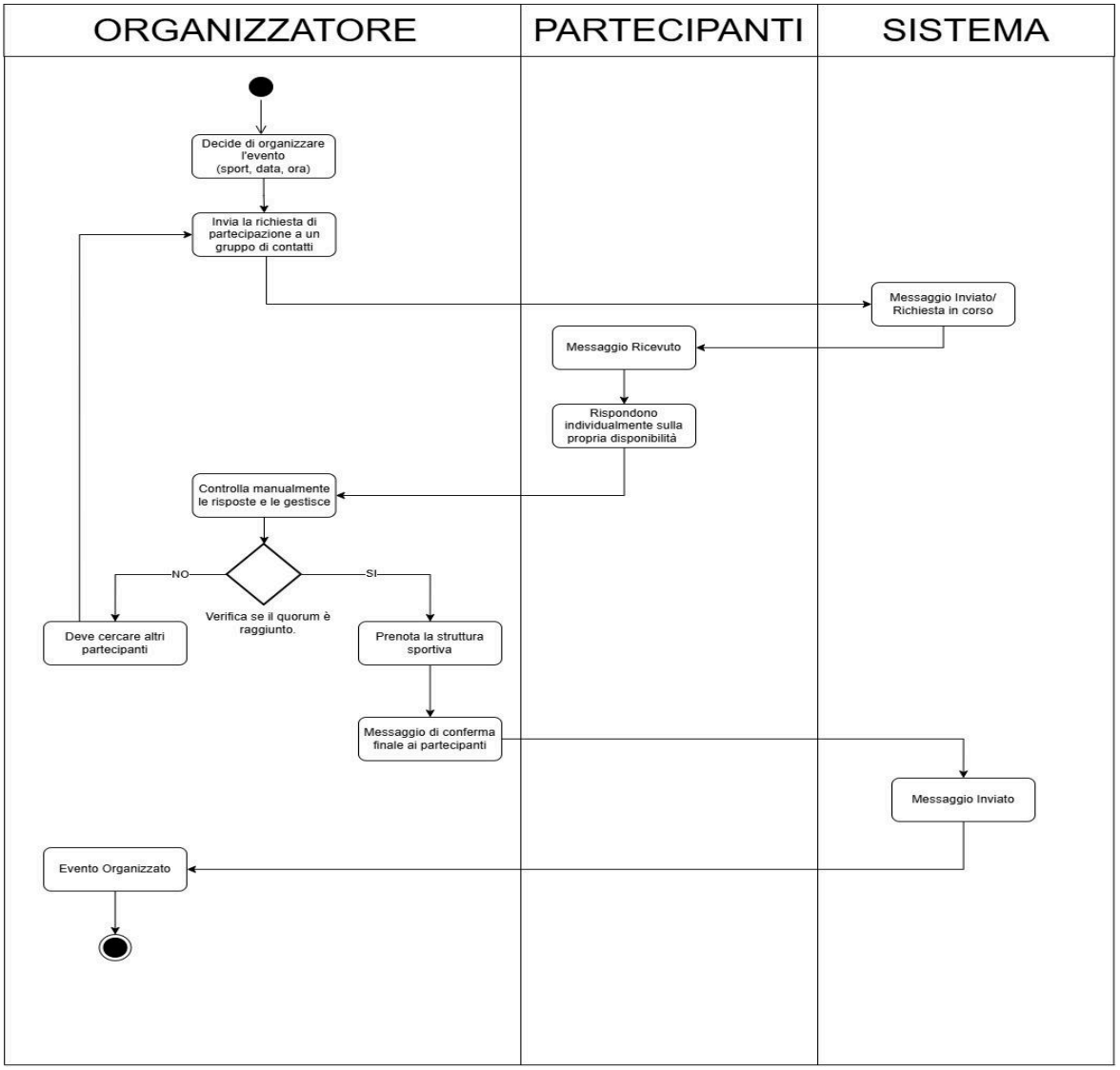
- **Architettura Software Attuale**: in questa parte, viene illustrato lo stato attuale dell'architettura del software già esistente (se presente).

- **Architettura Software Proposta**: viene descritto come il sistema sarà definito e partizionato in sottosistemi, il loro mapping Hardware/Software, la gestione dei dati

persistenti. Verranno poi presentate la struttura dei singoli sottosistemi e le boundary conditions riguardanti l'intero sistema.

2. Architettura del sistema corrente

Attualmente, l'organizzazione di eventi sportivi amatoriali come partite di calcetto o pallavolo è gestita attraverso metodi informali e non integrati. Il sistema esistente si basa prevalentemente sull'interazione diretta e non centralizzata tra gli individui.



VANTAGGI	SVANTAGGI
L'uso di canali di comunicazione già noti (chat) permette di avviare il reclutamento in modo estremamente rapido.	L'Organizzatore deve gestire manualmente risposte, rinunce e calcolo del quorum, sprecando tempo e spesso fallendo nell'aggregare i partecipanti.

3. Architettura del sistema proposto

3.1 Panoramica sulla sezione

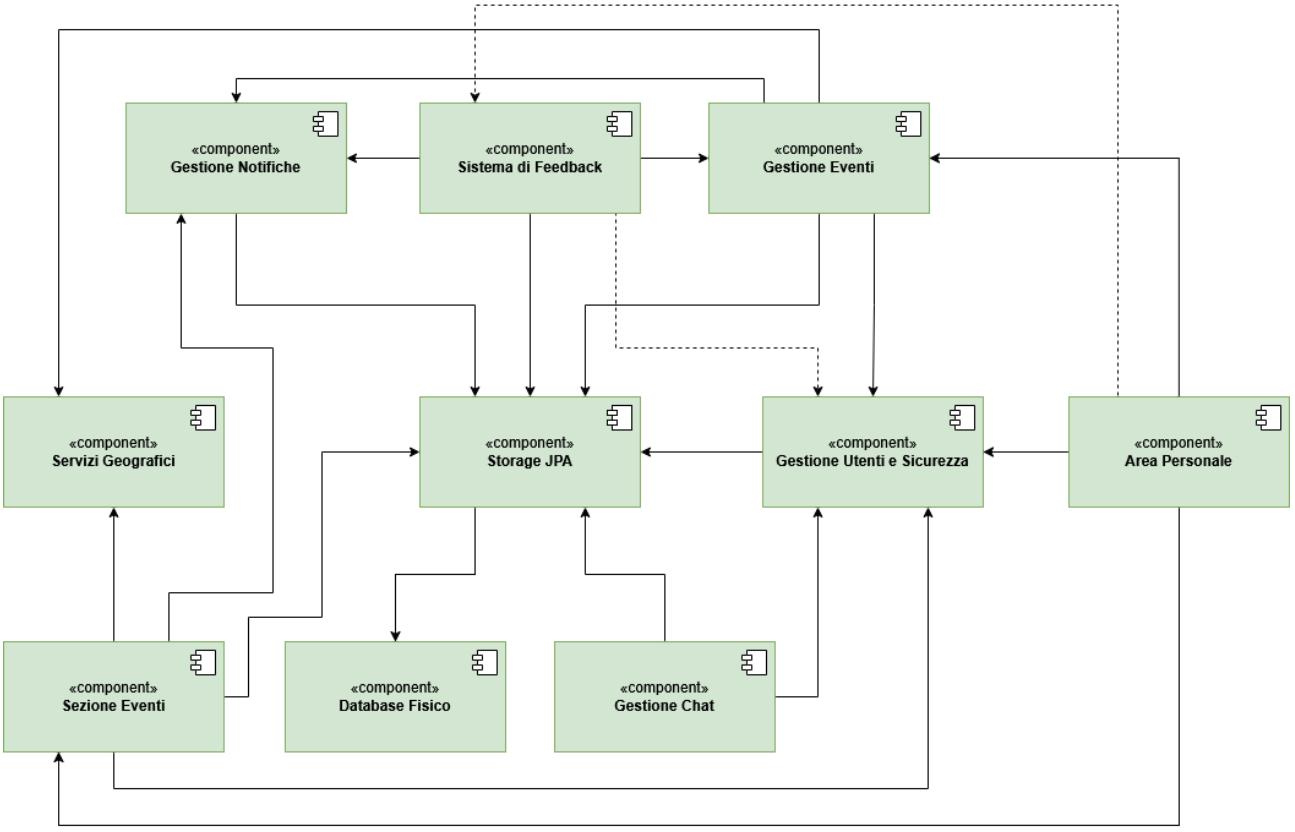
Il sistema proposto è basato sullo stile architetturale Three Tier. Il motivo della presente scelta è che tale architettura è perfetta per lo sviluppo di web application come il nostro sistema, poiché la separazione della logica di presentazione da quella di elaborazione, migliora una serie di qualità, tra le quali: leggibilità, manutenzione e riuso.

3.2 Decomposizione in sottosistemi

I sottosistemi individuati sono:

- **Gestione Utenti e Sicurezza:** gestisce il ciclo di vita dell'account utente, l'autenticazione, la sicurezza delle sessioni,la gestione dei profili e la gestione delle funzioni amministrative.
- **Sezione eventi**: è responsabile della visualizzazione dei vari eventi a cui poter partecipare, della funzione di ricerca e filtro, della partecipazione all'evento e del ritiro dell'iscrizione.
- **Gestione eventi**: si occupa della creazione da parte di un utente, della modifica e cancellazione di un evento, e della gestione del suo stato.
- **Sistema di Feedback e Rating:** si occupa di coordinare la raccolta dei feedback e delle valutazione tra i giocatori e del ricalcolo della valutazione totale mostrata nel profilo di ciascun utente.
- **Gestione Notifiche:** si occupa dell'invio e della gestione di tutte le comunicazioni verso l'utente (email di conferma, avvisi di cancellazione).
- **Gestione Chat:** gestisce il servizio di messaggistica istantanea dedicato ai gruppi degli eventi.
- **Servizi Geografici:** fornisce funzionalità legate alla posizione, calcolo distanze e integrazione con mappe esterne.
- **Area Personale**: si occupa di visualizzare la sezione contenente tutti gli eventi organizzati o a cui si è iscritto, e di rendere accessibili i dati relativi allo storico di partite giocate.
- **Storage JPA**: si interpone tra i tra i vari sottosistemi e il sottosistema di Persistenza.
- **Database Fisico**: si occupa di gestire la persistenza dei dati con un database, garantendo l'affidabilità e la consistenza delle transizioni.

Sono mostrate di seguito le dipendenze tra i sottosistemi attraverso un **Component Diagram** UML.



Le linee tratteggiate sono usate "a scanso di equivoci", in caso di accavallamento tra loro.

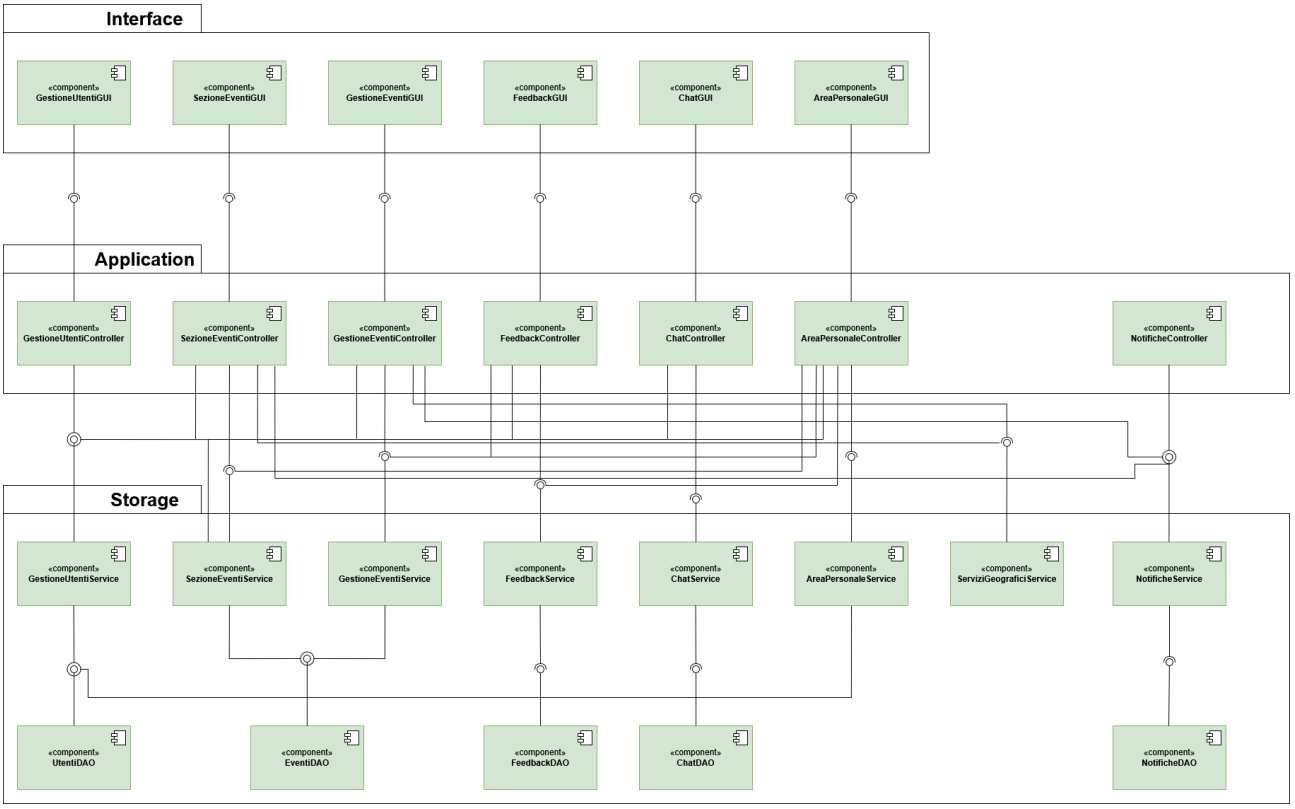
Alcuni sottosistemi saranno gestiti da componenti COTS (Commercial off the shelf), e framework open source:

- **Framework Applicativo**: Il sistema è basato su **Spring Boot**, che gestisce l'architettura, l'Inversion of Control (IoC) e l'iniezione delle dipendenze.
- **Storage JPA**: L'interazione con il database è gestita dal modulo **Spring Data JPA** (che incapsula Hibernate ORM), permettendo l'astrazione delle query SQL.
- **Web Server**: L'applicazione è ospitata all'interno del container **Apache Tomcat** (embedded), gestito direttamente da Spring Boot.
- **Persistenza**: Il salvataggio fisico dei dati avviene sul DBMS relazionale **MySQL**.

Di seguito una vista dettagliata di ciascun sottosistema evidenziando le componenti principale:

- **GUI**: Graphic User Interface, che contiene le varie view che saranno renderizzate per creare le pagine web da mostrare al cliente.
- **Controller**: si occupa della logica per il controllo del sistema.
- **Service**: si occupa della logica di business.
- **DAO**: Data Access Object, che si occupa di fornire accesso ai dati persistenti.

Diagramma architetturale



3.3 Mapping hardware/software

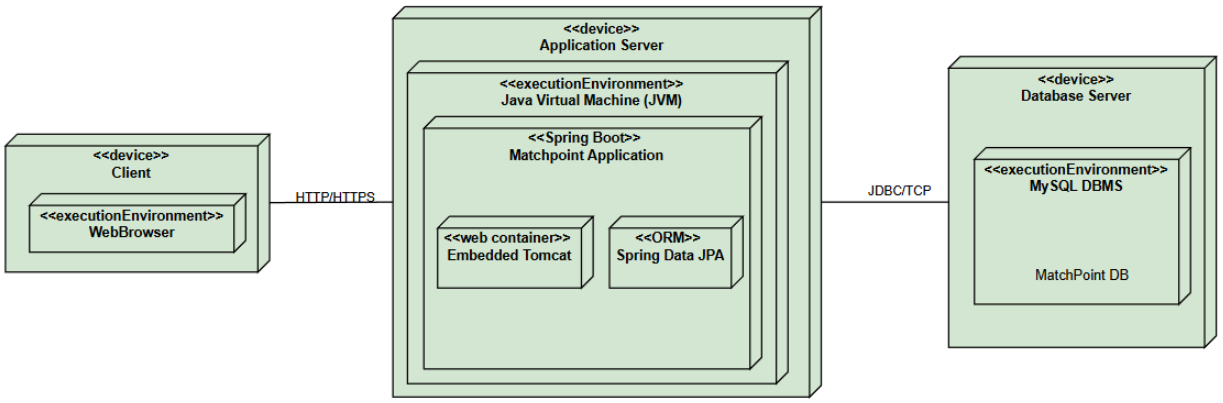
L'applicazione web che verrà sviluppata si basa su una piattaforma hardware costituita da un server che risponderà ai servizi richiesti dai client da una qualsiasi macchina (PC o dispositivo mobile) dotata di un browser web ed una connessione ad Internet.

Essendo il sistema una web application basata su un'architettura Three Tier, la logica applicativa risiede sul server, mentre la gestione dei dati è affidata a un servizio database esterno

Nello specifico, la dislocazione dei componenti è la seguente:

- **Client**: il dispositivo dell'utente finale che esegue il browser (Chrome, Firefox, Safari, Edge) per visualizzare l'interfaccia grafica.
- **Server Applicativo**: il nodo ospita l'applicazione basata su Spring Boot, che utilizza un container Apache Tomcat per l'esecuzione. Il framework gestisce l'iniezione delle dipendenze e il ciclo di vita dei componenti.
- **Database Server**: il nodo che ospita il DBMS relazionale MySQL. Questo componente gestisce tutte le operazioni di lettura e scrittura dei dati garantendo l'integrità referenziale.

Di seguito un UML Deployment Diagram che descrive il mapping hardware/software:



3.4 Gestione dei dati persistenti

Per la gestione del salvataggio dei dati persistenti del sistema si è deciso di utilizzare un database relazionale, al fine di gestire agevolmente l'accesso concorrente ai dati e contemporaneamente garantire la consistenza dei dati tramite l'utilizzo di un DBMS.

La scelta di utilizzo di un DBMS relazionale (MySQL) è stata presa al fine di mantenersi quanto più possibile coerenti con i design goals stabiliti, potendo contare su:

- **Imposizioni di vincoli di integrità sui dati**: un DBMS permette di specificare diversi tipi di vincoli per mantenere l'integrità dei dati (ad esempio, un'iscrizione non può esistere se non esiste l'evento associato) e controlla che tali vincoli siano soddisfatti quando la base di dati cambia.
- **Privatezza dei dati**: garantita dal fatto che il DBMS permette un accesso protetto ai dati sensibili degli utenti registrati.
- **Affidabilità dei dati**: il DBMS offre metodi per salvare copie dei dati e per ripristinare lo stato della base di dati in caso di guasti.
- **Atomicità delle operazioni**: data la natura critica della prenotazione dei posti negli eventi sportivi, è necessario che l'intera sequenza di operazioni (controllo posti disponibili -> assegnazione posto -> aggiornamento contatore) venga eseguita come una transazione atomica.

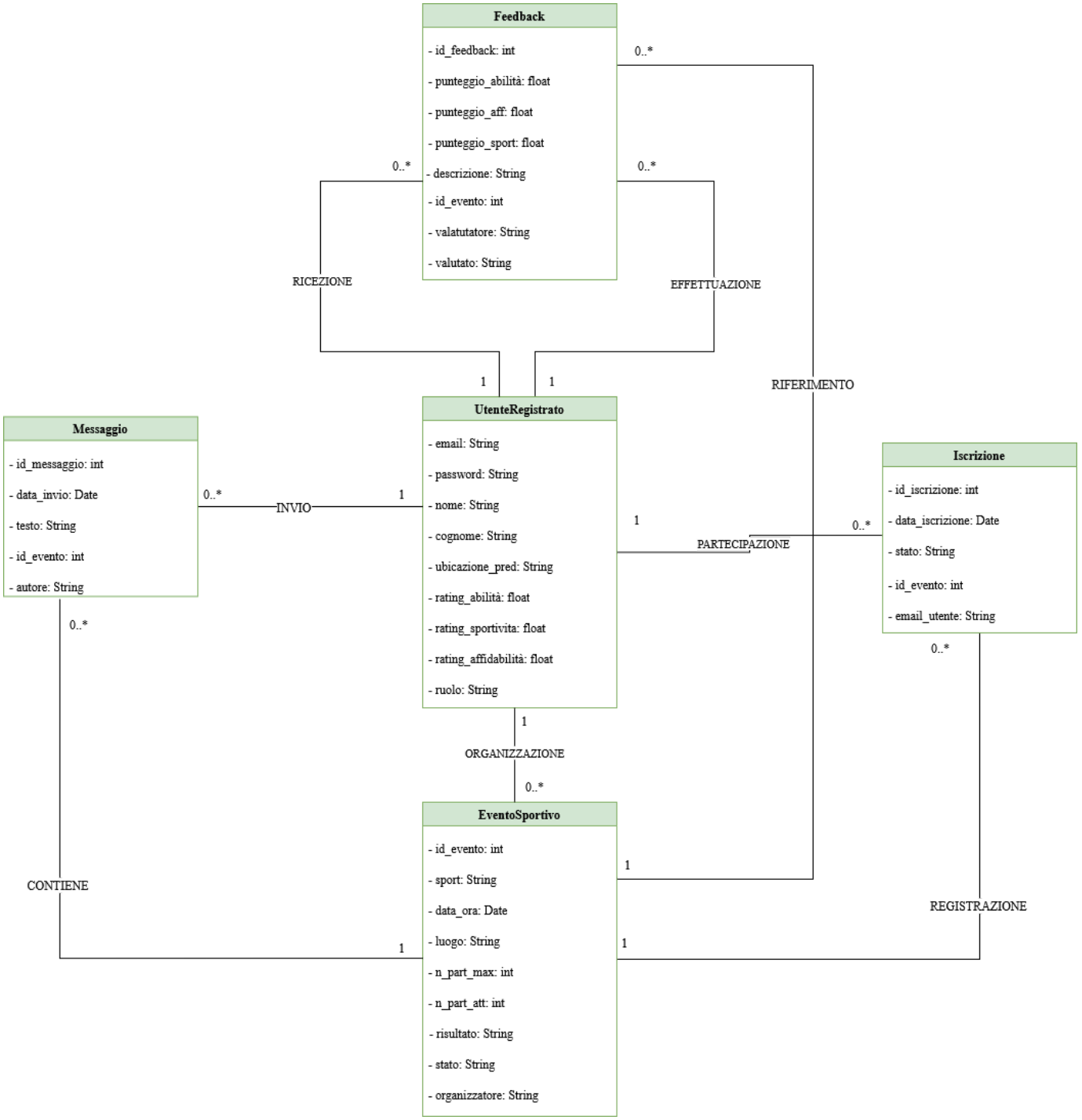
L'applicazione, in esecuzione su Apache Tomcat, interagirà con il database MySQL attraverso il framework Spring Data JPA. Questo permette di astrarre le operazioni CRUD (Create, Read, Update, Delete) tramite l'uso di interfacce Repository, demandando al framework la generazione automatica delle query SQL e la gestione delle transazioni, riducendo la possibilità di errori manuali e garantendo la sicurezza

Entity Class Diagram ristrutturato

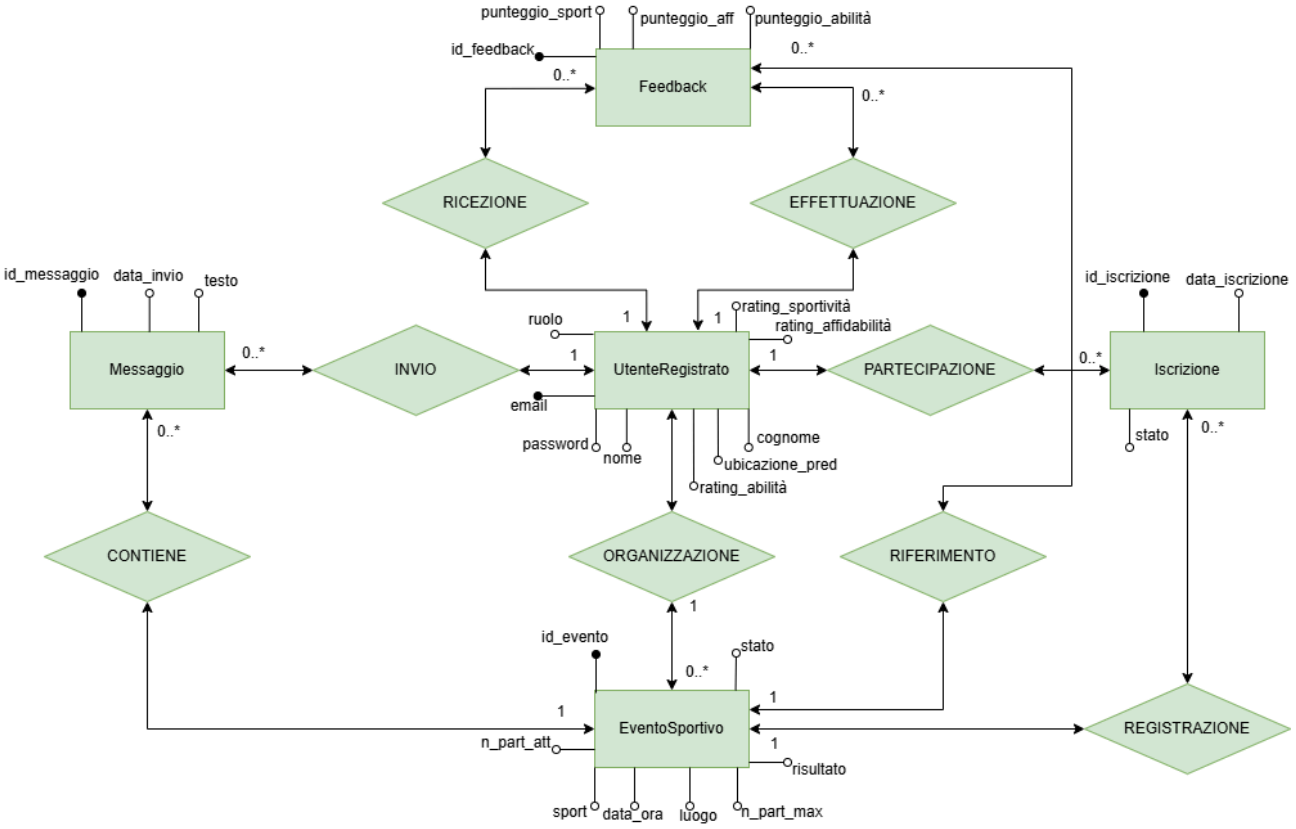
Rispetto al diagramma delle classi presentato nel documento di Analisi dei Requisiti (RAD), in questa fase di System Design sono state apportate delle modifiche strutturali per adattare il modello alle esigenze di persistenza su database relazionale **MySQL**.

La principale ristrutturazione è stata

- Definizione delle Chiavi (PK/FK): sono stati introdotti gli attributi identificativi (ID univoci auto-incrementali o Email come Primary Key) e i vincoli di integrità referenziale (Foreign Key) necessari per il mantenimento della consistenza dei dati nel DBMS.


Feedback
- id_feedback: int
- punteggio_abilità: float
- punteggio_aff: float
- punteggio_sport: float
- descrizione: String
- id_evento: int
- valatutatore: String
- valutato: String
0..*
0..*
0..*
RICEZIONE
EFFETTUAZIONE
1
1
RIFERIMENTO
Messaggio
- id_messaggio: int
- data_invio: Date
- testo: String
- id_evento: int
- autore: String
0..*
INVIO
1
UtenteRegistrato
- email: String
- password: String
- nome: String
- cognome: String
- ubicazione_pred: String
- rating_abilità: float
- rating_sportivita: float
- rating_affidabilità: float
- ruolo: String
1
PARTECIPAZIONE
0..*
Iscrizione
- id_iscrizione: int
- data_iscrizione: Date
- stato: String
- id_evento: int
- email_utente: String
0..*
0..*
ORGANIZZAZIONE
1
0..*
EventoSportivo
- id_evento: int
- sport: String
- data_ora: Date
- luogo: String
- n_part_max: int
- n_part_att: int
- risultato: String
- stato: String
- organizzatore: String
CONTIENE
1
1
REGISTRAZIONE
1

Modello Concettuale dei Dati



Dizionario dei dati

Di seguito si mostrano gli attributi per ogni entità individuata.

Nome Entità	UtenteRegistrato		
Descrizione	Contiene i dati relativi ad un utente registrato		
Nome campo	Tipo	Vincolo di chiave	Altri vincoli
email	Varchar(320)	PRIMARY KEY	NOT NULL

password	Byte(32)		NOT NULL
nome	Varchar(50)		NOT NULL
cognome	Varchar(50)		NOT NULL
ubicazione_predefinita	Varchar(100)		
rating_affidabilità	Decimal(2,1)		DEFAULT 0.0
rating_abilità	Decimal(2,1)		DEFAULT 0.0
rating_sportività	Decimal(2,1)		DEFAULT 0.0
ruolo	Varchar(20)		DEFAULT “utente”

Nome Entità	EventoSportivo		
Descrizione	Contiene i dati relativi ad un evento sportivo		
Nome campo	Tipo	Vincolo di chiave	Altri vincoli
id_evento	Int(32)	PRIMARY KEY	NOT NULL AUTO-INCREMENT
sport	Varchar(30)		NOT NULL
data_ora	DateTime		NOT NULL
luogo	Varchar(255)		NOT NULL
n_part_max	Int(20)		NOT NULL
n_part_attuali	Int(20)		DEFAULT 0
risultato	Varchar(50)		NULLABLE
stato	Enum(...)		NOT NULL Può assumere 5 valori: 1. Evento Creato 2. In attesa di partecipanti 3. Confermato 4. Terminato 5. Annullato

organizzatore	Varchar(320)	FOREIGN KEY (UtenteRegistrato)	

Nome Entità	Feedback		
Descrizione	Contiene i dati relativi ai feedback rilasciati e ricevuti dagli utenti		
Nome campo	Tipo	Vincolo di chiave	Altri vincoli
id_feedback	Int(32)	PRIMARY KEY	AUTO-INCREMENT
punteggio_abilità	Decimal(2, 1)		NOT NULL (Check 1-5)
punteggio_aff	Decimal(2, 1)		NOT NULL (Check 1-5)
punteggio_sport	Decimal(2, 1)		NOT NULL (Check 1-5)
descrizione	Varchar(320)		
id_evento	Int(32)	FOREIGN KEY (EventoSportivo)	
valutatore	Varchar(320)	FOREIGN KEY (UtenteRegistrato)	
valutato	Varchar(320)	FOREIGN KEY (UtenteRegistrato)	

Nome Entità	Iscrizione		
Descrizione	Contiene i dati relativi all'iscrizione ad un evento sportivo		
Nome campo	Tipo	Vincolo di chiave	Altri vincoli
id_iscrizione	Int(32)	PRIMARY KEY	NOT NULL AUTO-INCREMENT
id_evento	Int(32)	FOREIGN KEY (EventoSportivo)	
data_iscrizione	DateTime		NOT NULL
email_utente	Varchar(320)	FOREIGN KEY (UtenteRegistrato)	
stato	Enum(...)		Può assumere 3 valori: 1. In attesa 2. Confermato 3. Annullato

Nome Entità	Messaggio		
Descrizione	Contiene i dati relativi ad i messaggi di un evento		
Nome campo	Tipo	Vincolo di chiave	Altri vincoli
id_messaggio	Int(32)	PRIMARY KEY	NOT NULL AUTO-INCREMENT
testo	Text		NOT NULL
data_invio	DateTime		NOT NULL
id_evento	Int(32)	FOREIGN KEY (EventoSportivo)	
autore	Varchar(320)	FOREIGN KEY (UtenteRegistrato)	

3.5 Controllo degli accessi e sicurezza

Di seguito viene mostrata la matrice degli accessi per poter tenere traccia di quali attori possono accedere a quali dei servizi offerti dal sistema.

Attori Oggetti	Utente Registrato	Organizzatore Evento	Partecipante Evento
Gestione Utente e Sicurezza	Login, Logout, ModificaDati, CancellazioneAccount	Login, Logout, ModificaDati, CancellazioneAccount	Login, Logout, ModificaDati, CancellazioneAccount
Gestione Eventi (Organizzatore)	CreaEvento VisualizzaDatiEvento	CreaEvento, ModificaEvento, AnnullaEvento, VisualizzaIscritti	CreaEvento VisualizzaDatiEvento
Sezione Eventi (Partecipante)	RicercaEventi, IscrizioneEvento	RicercaEventi, IscrizioneEvento	AbbandonaEvento, IscrizioneEvento
Feedback & Rating	Nessun Accesso	RilasciaFeedback,	RilasciaFeedback
Chat & Messaggistica	Nessun Accesso	InviaMessaggio	InviaMessaggio
Area Personale	VisualizzaStorico, VisualizzaRating, VisualizzaFeedback	VisualizzaStorico, VisualizzaRating, VisualizzaFeedback	VisualizzaStorico, VisualizzaRating, VisualizzaFeedback

Attori Oggetti	Utente non Registrato	Amministratore
Gestione Utente e Sicurezza	Registrazione	Login, Logout, Modifica Dati, CancellazioneAccount, BanUtente
Gestione Eventi (Organizzatore)	VisualizzaDatiEvento	RimuoviEvento
Sezione Eventi (Partecipante)	RicercaEventi	VisualizzaEventi
Feedback & Rating	Nessun Accesso	RimozioneFeedback
Chat & Messaggistica	Nessun Accesso	RimuoviMessaggio, VisualizzaMessaggi
Area Personale	Nessun Accesso	Accesso completo a tutti i profili e storici per fini di moderazione.

3.6 Controllo globale del software

Il sistema MatchPoint è un'applicazione web interattiva, basata su un'architettura Three-Tier e caratterizzata da un meccanismo di controllo del flusso di tipo *event-driven*.

L'interazione con il sistema avviene tramite l'interfaccia grafica (GUI) su browser, utilizzata sia dagli Sportivi che dagli Amministratori per impartire comandi e avviare le funzionalità (es. Creazione Evento, Iscrizione).

Quando un utente interagisce con la GUI, l'azione genera una richiesta HTTP (evento) che viene intercettata dal Controller corrispondente. L'handler specifico gestisce l'evento e trasferisce il controllo del flusso al sottosistema responsabile della logica di controllo. Questo, a sua volta, utilizza i Service per la logica applicativa e infine i DAO per la persistenza dei dati.

3.7 Condizioni Limite

Nel presente paragrafo vengono presentate le *boundary conditions* (condizioni limite) inerenti all'avvio del sistema, spegnimento del sistema, fallimento del sistema ed errore di accesso ai dati persistenti.

UCBC_1: Avvio del sistema

Identificativo	**UCBC_1**: Avvio del sistema
Descrizione	Lo UC permette l'avvio del sistema MatchPoint e la messa in linea dei servizi.
Attore Principale	Amministratore
Attori Secondari	N/A
Entry Condition	L'Amministratore accede al server web (Apache Tomcat).
Exit Condition **On Success**	Il sistema viene avviato correttamente e il database è raggiungibile.
Exit Condition **On Failure**	Il sistema non viene avviato.

Flusso di eventi principale (1):

1. **Amministratore**: Esegue il comando di avvio dell'application server (Tomcat).
2. **Sistema**: Inizializza i componenti, verifica la connessione al Database (MySQL) e ai servizi esterni (Mappe/Email). Se i controlli passano, apre le porte per le connessioni HTTP.

Flusso alternativo (2):

1. **Sistema**: Rileva che il Database non sia raggiungibile o dati sono corrotti.
2. **Sistema**: Notifica l'errore all'amministratore e abortisce l'avvio.
3. **Amministratore**: Risolve il problema di connettività o corregge i dati.
4. **Amministratore**: Esegue il flusso di eventi principale (1).

UCBC_2: Spegnimento del sistema

Identificativo	**UCBC_2**: Spegnimento del sistema
Descrizione	Lo UC permette lo spegnimento controllato del sistema
Attore Principale	Amministratore
Attori Secondari	N/A
Entry Condition	Il Sistema è avviato AND L'amministratore ha accesso alla console del server
Exit Condition **On Success**	Il sistema viene spento correttamente senza perdita di dati.
Exit Condition **On Failure**	Il sistema forza la chiusura o rimane appeso.

Flusso di eventi principale (1):

1. **Amministratore:** Invia un segnale di spegnimento (shutdown) al Sistema.
2. **Sistema:** Controlla che non ci siano transazioni critiche in corso (es. creazione evento o iscrizione) e termina l'esecuzione.

Flusso alternativo (2):

1. **Sistema:** Rileva connessioni attive o query in scrittura.
2. **Sistema:** Attende il completamento delle operazioni in corso, rifiutando nuove connessioni in entrata.
3. **Sistema:** Termina l'esecuzione una volta completate le code.
4. **Sistema:** Notifica l'avvenuto spegnimento.

UCBC_3: Fallimento del sistema

Identificativo	**UCBC_3**: Fallimento del sistema
Descrizione	L'UC definisce il comportamento in caso di crash o terminazione imprevista.
Attore Principale	Amministratore
Attori Secondari	N/A
Entry Condition	Il Sistema viene terminato inaspettatamente (Crash).
Exit Condition **On Success**	Il Sistema viene riavviato e i dati rimangono consistenti.
Exit Condition **On Failure**	Il sistema richiede manutenzione manuale.

Flusso di eventi principale (1):

1. **Amministratore:** Rileva il disservizio e include UCBC_1 (Avvio del Sistema).
2. **Sistema:** Durante l'avvio, esegue il rollback di eventuali transazioni non completate e interrotte dal crash per garantire l'integrità dei dati.

UCBC_4: Errore Accesso Dati

Identificativo	**UCBC_4**: Errore Accesso Dati
Descrizione	Descrive il comportamento se il Database non è raggiungibile.
Attore Principale	Amministratore
Attori Secondari	N/A
Entry Condition	Il Sistema perde la connessione con il Database Server o i dati sono corrotti.
Exit Condition **On Success**	Il Sistema riprende la connessione.
Exit Condition **On Failure**	Il sistema entra in modalità manutenzione.

Flusso di eventi principale (1):

1. **Sistema:** Rileva eccezioni SQL o timeout di connessione JPA.
2. **Sistema:** Notifica l'amministratore dell'errore critico.
3. **Sistema:** Mostra una pagina di cortesia ("Manutenzione") agli utenti, cessando di processare richieste funzionali.
4. **Amministratore:** Ripristina la connettività o il servizio Database.
5. **Amministratore:** Riavvia o riconnette il sistema (Include UCBC_1).