Design Document UniMeet

Versione 1.0.0 del 25/01/26

Versione	Modifiche
1.0.0	Compilazione

Bregolin Gabriele
Leggeri Leonardo
Vecchi Samuele

1. Architettura del Software (Software Architecture)

Per UniMeet abbiamo adottato un'architettura Multilivello, Spring Boot. Appunto a causa del Framework scelto è stata sviluppata un'architettura basata su molteplici servizi @Service in esecuzione parallela.

L'architettura si articola in quattro livelli principali: (punto 2.1)

- **Interfaccia grafica:** Realizzato con Vaadin. Abbiamo costruito viste Java che compongono l'interfaccia senza dover scrivere HTML/JavaScript, questo ha reso molto più semplice la realizzazione dell'interfaccia grafica.
- **Service Layer:** Contiene i componenti @Service di Spring. Qui risiede la logica decisionale: questo layer garantisce la comunicazione fra tutti i componenti di UniMeet.
- **Repository Layer:** Utilizza Spring Data JPA. Questo livello crea le query SQL, permettendoci di interagire con il database H2 tramite interfacce Java (Repository).
- **Entity:** Hibernate + H2. Questa scelta ci ha permesso di avere un database embedded senza dover effettivamente scrivere una riga di DML o DDL.

1.2 Utilizzo di stan4j: Abbiamo usato stan4j per la generazione delle metriche e per ottenere una visione completa del progetto attraverso i grafici forniti dal tool. Abbiamo generato il package diagram in modo da fornire la visione del funzionamento e delle dipendenze dei package dando importanza alla valutazione dei possibili cicli, che avrebbero portato all'indebolimento alla struttura del progetto, cosa che come si può vedere in seguito abbiamo evitato.

2. Software Design e Design Patterns

Nella progettazione di UniMeet non si è avuto come limite la stesura di un codice senza pensare alla sua ottimizzazione ,invece sono stati applicati dei pattern atti a migliorare la logica implementata per garantire una qualità del codice maggiore.

2.1 Pattern Model-View-Controller Spring(Entity-Repository-Service-VaadinUI)

- **Entity:** Le classi che rappresentano le @entity vengono usate nel per la creazione di tabelle nel database(H2) es. Studente, Tavolo e Prenotazione sono create con la propria chiave primaria e i propri attributi, l'entità definisce anche le tabelle nel database per quanto riguarda le relazioni, dando il nome alle colonne e gestendo il tipo di fetch adottato.

- **Repository**: Interfacce standardizzate dal framework spring che consentono l'interazione diretta con il database, con esse è possibile scrivere query sql personalizzate o attenersi a quelle generate automaticamente in base al nome del metodo.

- **Service:** I servizi in spring sono il core logico dell'applicazione. Avendo utilizzato Vaadin essi fanno letteralmente da API tra database e UI.
  In essi vi è la logica di controllo di integrità negli alterare / inserire /eliminare oggetti dell'applicazione.

- **VaadinUI**: Interfaccia grafica web, qui la logica sta strettamente nel visualizzare gli elementi in maniera consistente, dovrebbe richiamare esclusivamente i @Service e gestire le eccezioni lanciate da quest'ultimi mostrando una notifica all'utente in caso di comportamento errato.

2.2 Dependency Injection (DI)

Fondamentale per il nostro progetto è stato l'uso del Constructor Injection un elemento fondamentale in Spring Boot. Non si istanziano manualmente i servizi, si lascia che Spring "inietti" le dipendenze necessarie. Questo è molto utile in quanto l'architettura è basata su @Bean, ovvero microservizi singleton quali i vari servizi.

2.4 Singleton Pattern

Tutti i nostri @Service e @Repository sono gestiti da Spring come @Bean Singleton in quanto standard del framework.
Questo garantisce che esista un'unica istanza di un oggetto servizio per l'applicazione che si occuperà però delle richieste di ogni singolo utente connesso.

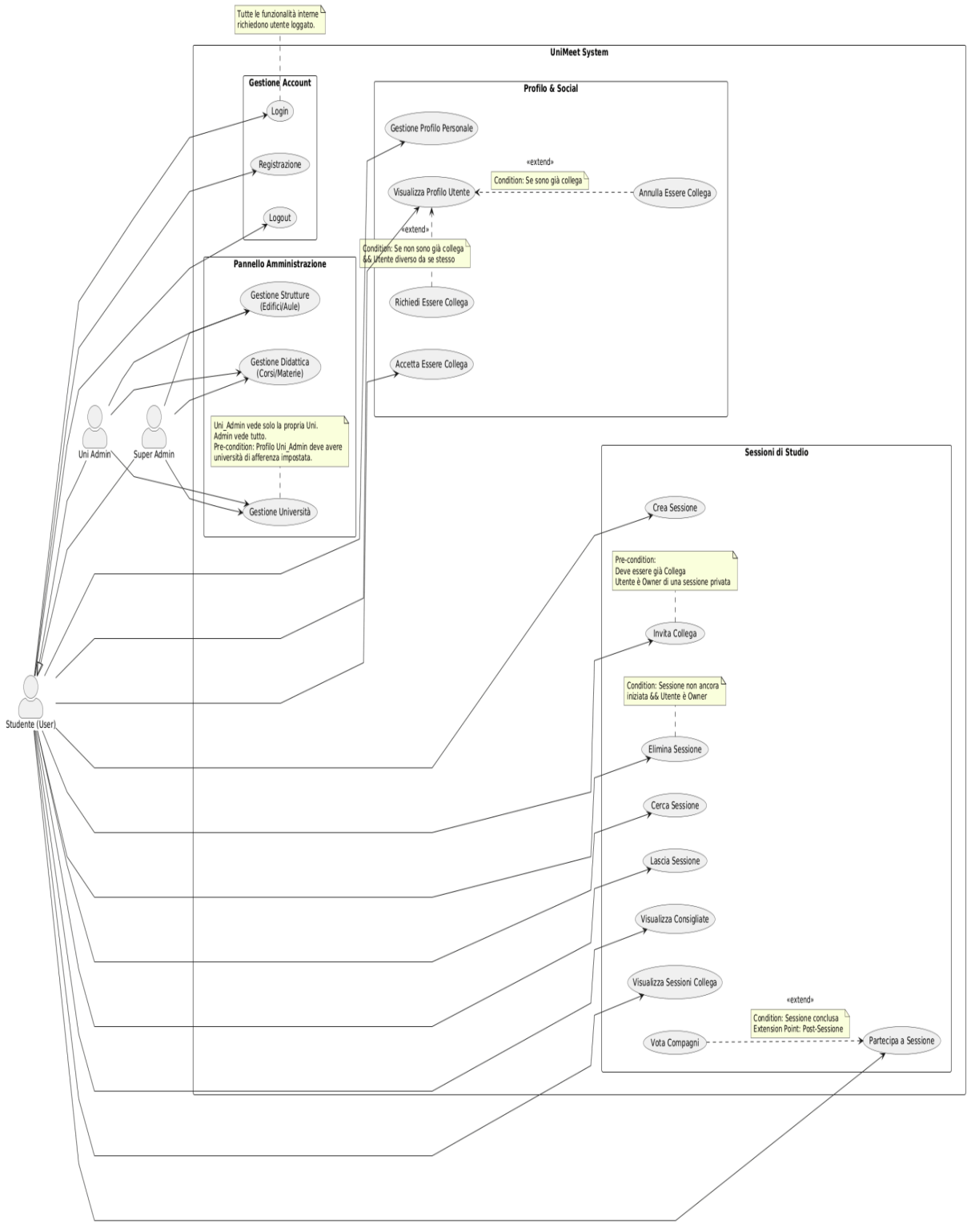
3. Modellazione

Il sistema è stato rappresentato tramite diagrammi UML che descrivono il comportamento e la struttura di UniMeet per darne una comprensione generale della sua struttura.

3.1 Diagramma dei Casi d'Uso (Use Case Diagram)

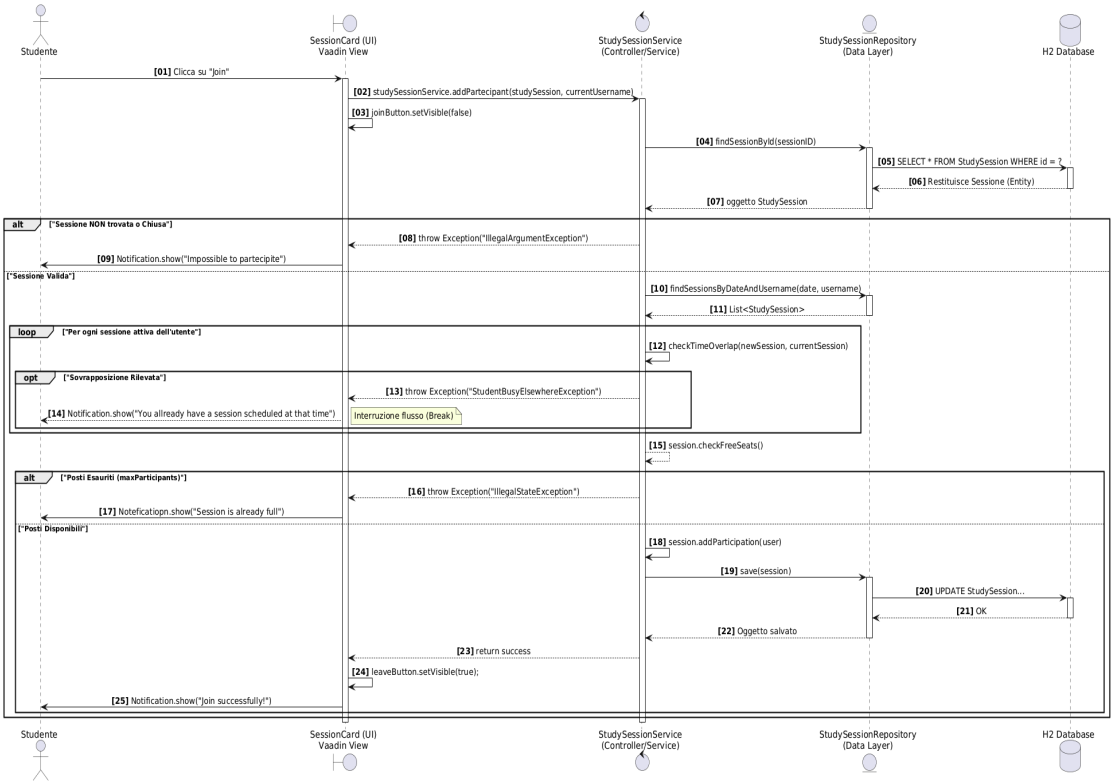
Attori:

- Studente
- Uni Admin (admin locale di una certa università)
- Super Admin (admin generale)



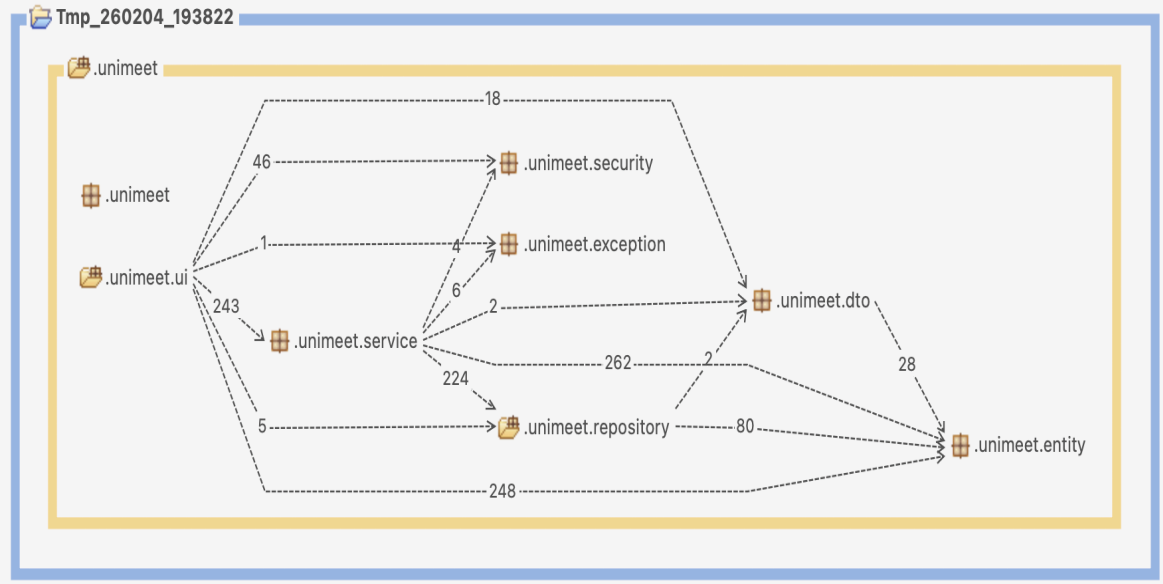
3.3 Diagramma di Sequenza (Sequence Diagram)

Nel sequence diagram si mostra il processo di Join in una studySession pubblica, con relative chiamate a catena da parte dei layer Spring, si noti il lancio e la gestione di vari tipi di eccezioni in base all'errore riscontrato.



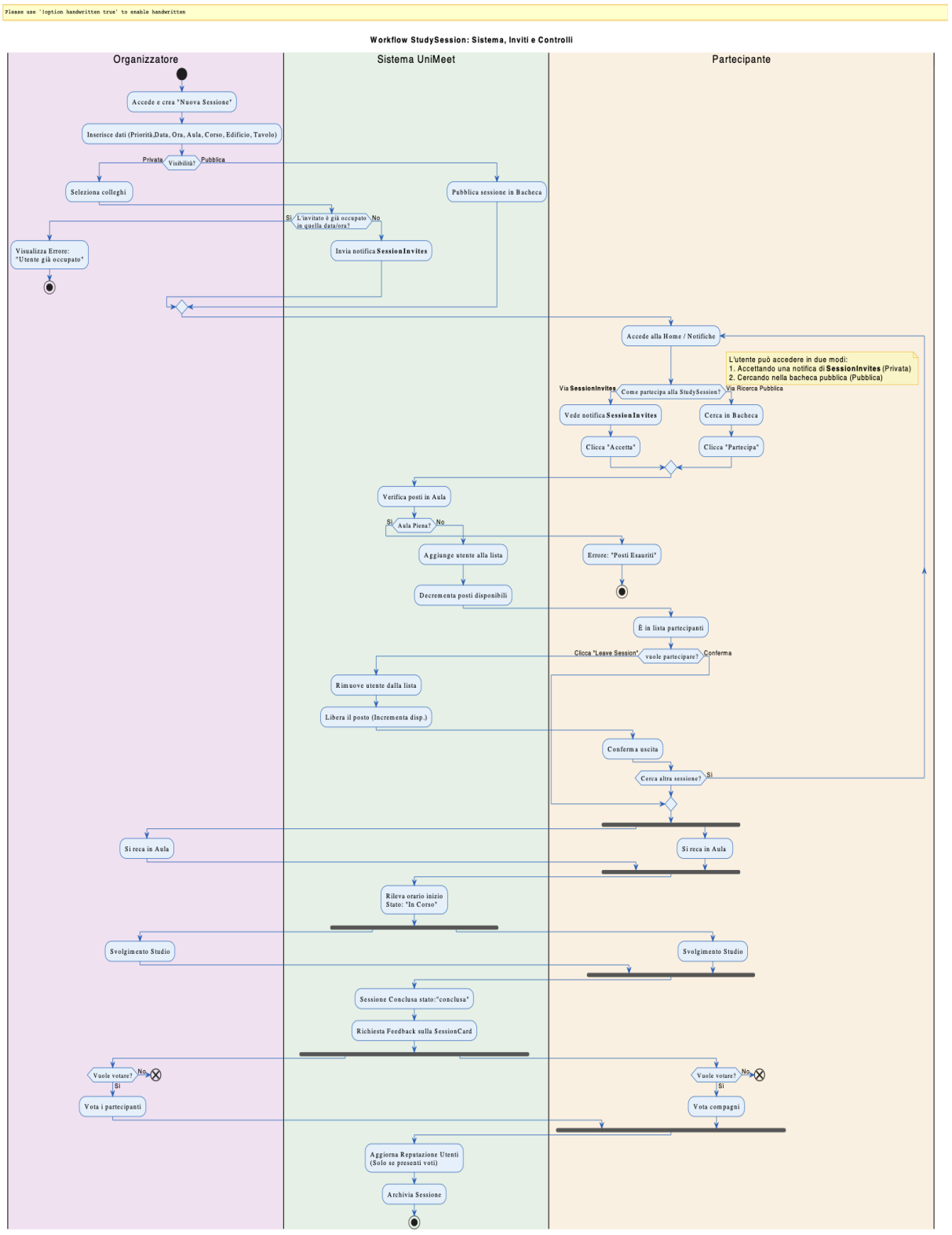
3.5 package diagram

Visualizzazione dell'accoppiamento efferente tra i vari packages in cui le classi del sistema si dividono, grazie a stan4J è stato possibile rilevare un ciclo di interdipendenza e correggerlo, giungendo quindi a questo risultato:



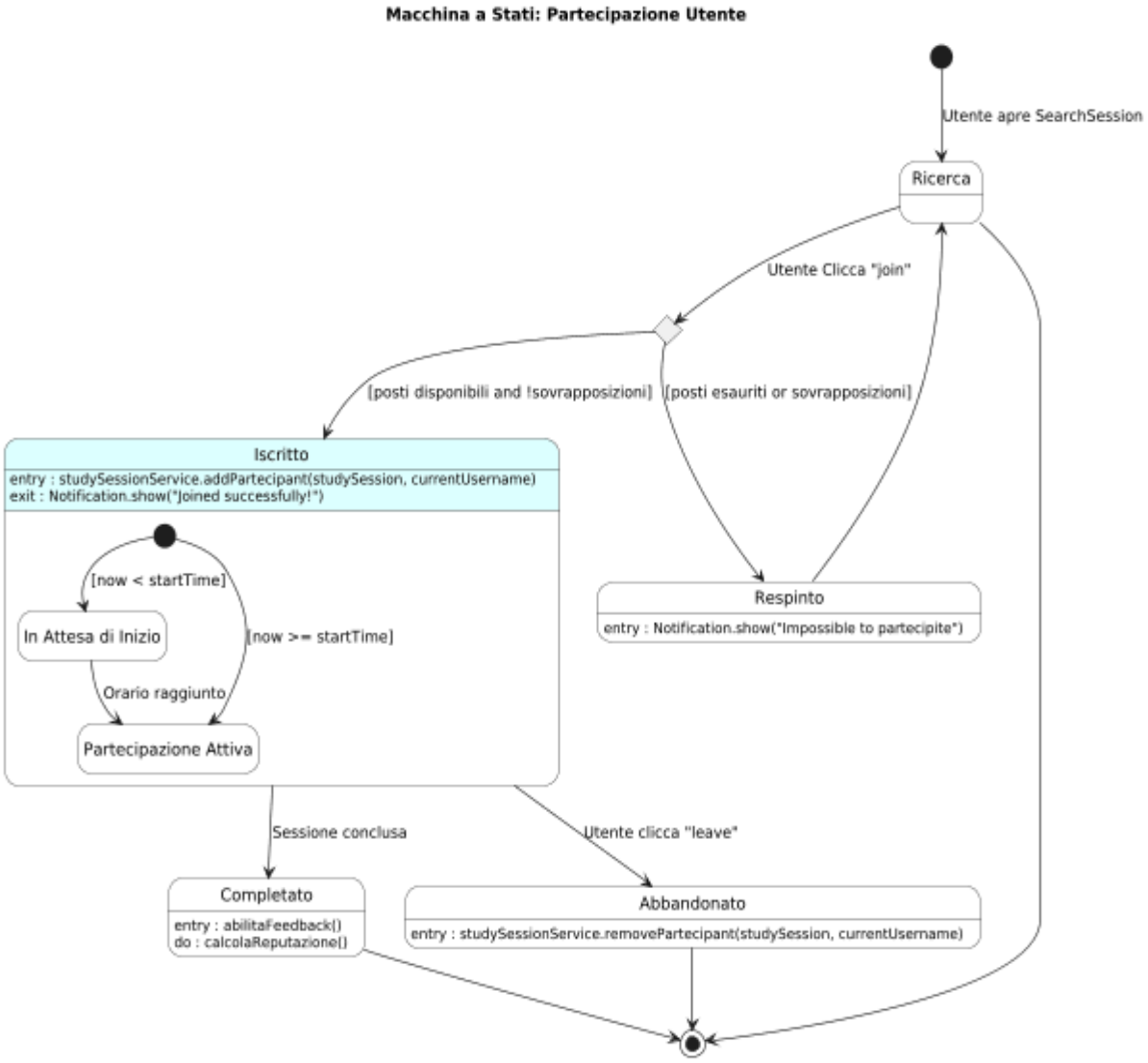
3.6 activity diagram

In questo diagramma viene mostrato il processo di uno User nella creazione di una sessione, la parte di notifica (invito) agli utenti fino allo svolgimento della sessione conseguente valutazione o meno dei partecipanti.



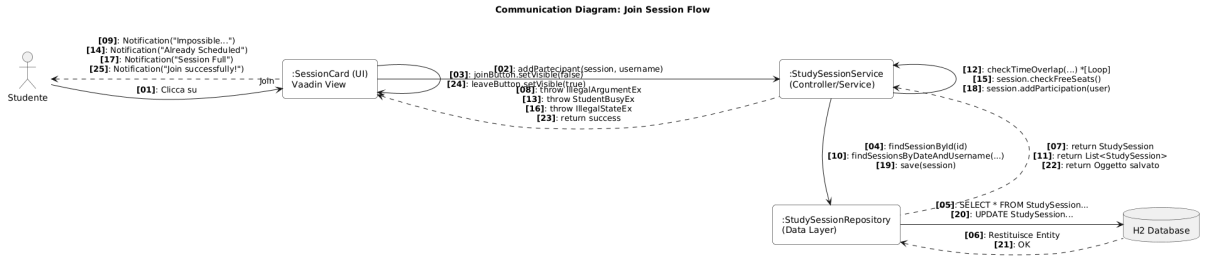
3.7 State machine Diagram

Descrive il ciclo di vita della relazione tra un utente e una specifica sessione.



3.8 Communication Diagram

Nel communication diagram si mostra il processo di Join in una studySession pubblica, con relative chiamate a catena da parte dei layer Spring.



4. Scelte Progettuali Specifiche

le scelte progettuali eseguite sono le seguenti:

- **Admin/User:** Gli admin non sono differenziati da user in quanto entità, vengono gestiti invece tramite un “flag” o meglio un enum e sono anch’essi studenti.
  Godono infatti delle stesse funzionalità abilitate per gli utenti comuni, nel caso di admin locale potranno anche gestire la struttura dell’università assegnata, e nel caso di admin globale qualunque struttura.

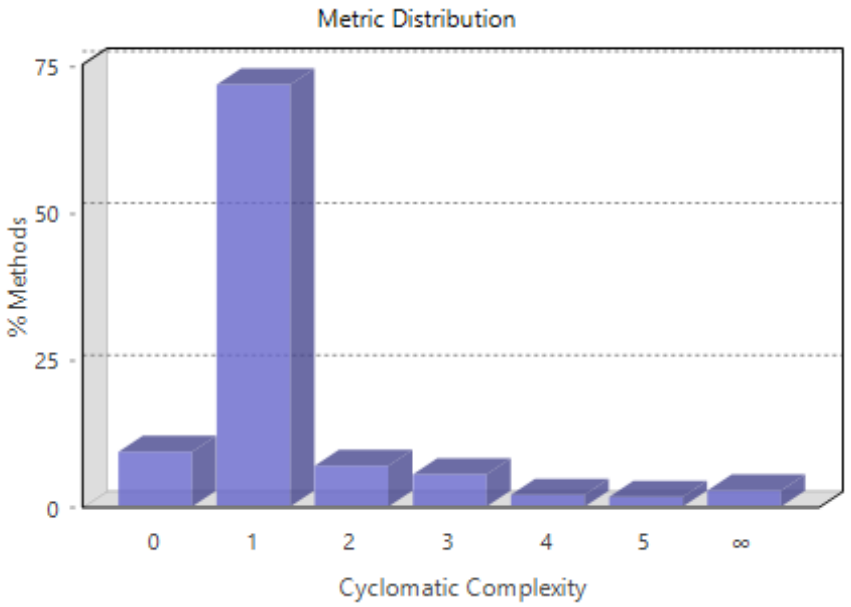
- **Reputazione:** A fine sessione un utente può (dovrebbe) valutare gli altri membri delle sessione a cui ha preso parte, le valutazioni vanno da 1 a 5 stelle.
  Se un utente partecipa a diverse sessione in cui è presente un secondo utente ricorrente, in ogni occasio si avrà l’opportunità di valutare quest'ultimo.
  Le valutazioni sono inoltre anonime. il valutato non sa direttamente quanto ha ricevuto da quale valutante.

- **Inviti a sessioni:** Uno studente nel caso crei una sessione ha come unico meccanismo per far aderire altri utenti gli inviti. è possibile fare un overloading di inviti: ossia invitare 6 persone a una sessione su un tavolo da 4 posti, qualora tutti cliccasserro “partecipa” sull’invito, solo i primi 3 verrebbero effettivamente iscritti (4 posti, 1 owner 3 partecipanti aggiuntivi), mentre agli ultimi tre verrebbe comunicato che la sessione è ormai full.

5. Metriche del codice:

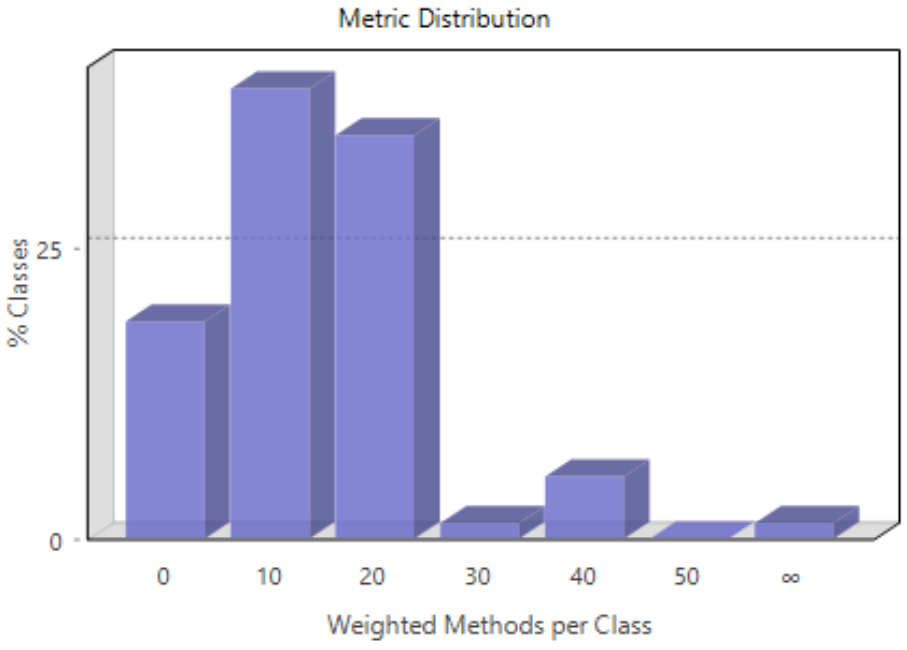
Complessità Ciclomatica

Quantifica il numero di percorsi linearmente indipendenti attraverso il flusso di controllo di un modulo software



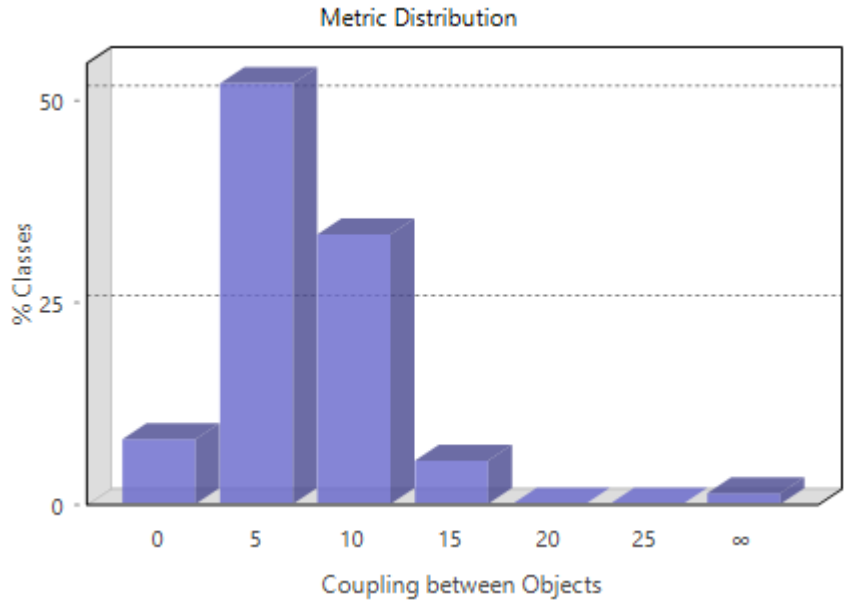
Weighted Method per Class

Rappresenta la somma della complessità di tutti i metodi definiti all'interno di una singola classe.



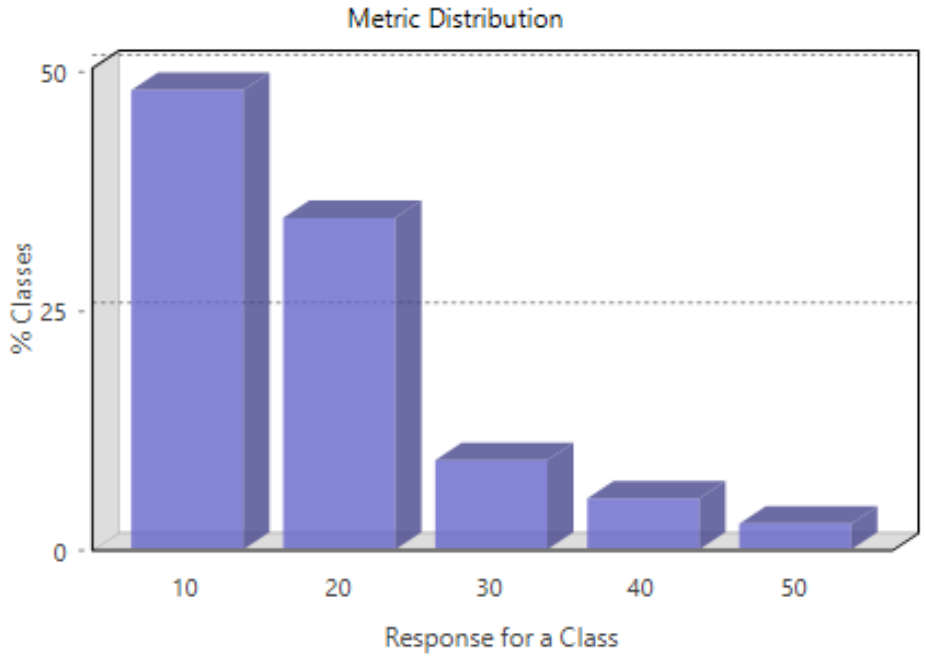
Coupling Between Objects

Misura il numero di classi a cui una determinata classe è collegata tramite l'uso di metodi o variabili di istanza.



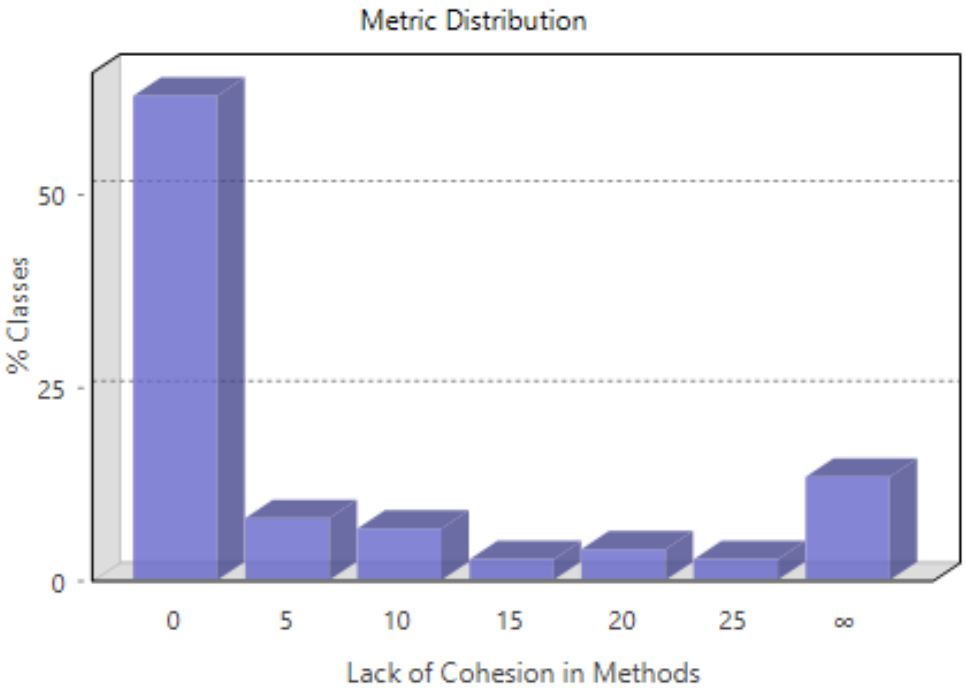
Response For a Class

Indica il numero totale di metodi che possono essere eseguiti in risposta a un messaggio ricevuto da un oggetto della classe.



Lack of Cohesion in Methods

Valuta il grado di coesione interna misurando quanto i metodi di una classe condividano le medesime variabili di istanza.



Considerazioni finali

Il progetto evidenzia un'ottima qualità granulare, caratterizzata da metodi semplici e un accoppiamento contenuto che favorisce la manutenibilità locale.

Tuttavia, la presenza di outlier con valori estremi nelle metriche WMC e LCOM rivela criticità strutturali concentrate in poche "God Classes" prive di coesione interna.

Sarebbe necessario del refactoring per distribuire meglio certe logiche di gestioni in più servizi, o più sottoclassi per l'UI

Complessivamente, la base di codice è discreta se non solida.

6. Considerazioni Finali sul Design

Il design di UniMeet riflette un compromesso ingegneristico tra robustezza e semplicità. L'uso di Spring Boot ci ha fornito una base solida, mentre Vaadin ha eliminato la complessità del frontend, garantendo la consegna di un prodotto funzionale e coerente con i requisiti definiti nel relativo documento.
