



Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

*Elaborato di Software Architecture
Design - StudyApp*

Anno Accademico 2023/2024

Candidato

Adriano Coppola matr M63001303

Andrea Olandese matr M63001304

Salvatore Guerrisi matr M63001390

Contents

1	Introduzione	1
1.1	Unified Process	1
1.2	Prima Iterazione	3
1.3	Seconda Iterazione	5
1.4	Tecnologie di supporto	7
1.4.1	Vaadin	7
1.4.2	Spring Boot	7
1.4.3	Eclipse	8
1.4.4	Firefox	8
1.4.5	Discord	9
1.4.6	Visual Paradigm	9
1.4.7	Github	10
2	Visione	11
2.1	Descrizione testuale dei requisiti	11
2.2	Obiettivi	12
2.3	Portatori di interesse	12
2.4	Glossario	14
2.5	Stima dei tempi	14

3 Requisiti funzionali e analisi di dominio	19
3.1 Casi d'uso	20
3.1.1 Registrazione	20
3.1.2 Login	21
3.1.3 Recupero Password	22
3.1.4 Visualizzare il proprio profilo	23
3.1.5 Modificare il proprio profilo	24
3.1.6 Cambiare Password	25
3.1.7 Visualizzare notifiche	26
3.1.8 Accettare invito gruppo/sessione	27
3.1.9 Visualizzare calendario	28
3.1.10 Creare nuovo evento	29
3.1.11 Creare gruppo	30
3.1.12 Modificare gruppo	31
3.1.13 Visualizzare i propri gruppi	32
3.1.14 Visualizzare i membri di un gruppo	33
3.1.15 Visualizzare le proprie sessioni	33
3.1.16 Visualizzare i membri di una sessione	34
3.1.17 Creare sessione	35
3.1.18 Creare una sessione con i membri di un gruppo	36
3.1.19 Modificare sessione	37
3.1.20 Visualizzare dashboard	38
3.2 Context Diagram	39
3.3 System Domain Model	40
3.4 Entity-Relationship Diagram	41

4 Requisiti non funzionali	42
4.1 Scenari	43
4.1.1 Modifiability	43
4.1.2 Robustness	44
4.1.3 Security	45
4.1.4 Usability	47
5 Architettura del sistema	49
5.1 Front-End	49
5.2 Back-End	50
5.2.1 Spring	50
5.3 Package Diagram	54
5.3.1 Entity Class Diagram	54
5.3.2 Repository Class Diagram	55
5.3.3 Service Class Diagram	56
5.3.4 Security Class Diagram	57
5.3.5 Presenter Class Diagram	58
5.3.6 View Class Diagram	59
5.4 Component Diagram	60
5.5 API	61
5.5.1 Session	61
5.5.2 Profile	64
5.5.3 Layout	65
5.5.4 Group	67
5.5.5 Calendar	69
5.5.6 Authentication	72

6 Diagrammi comportamentali	76
6.1 Sequence Diagram	77
6.1.1 leaveGroup	77
6.1.2 leaveSession	78
6.1.3 createGroup	79
6.1.4 createSession	80
6.1.5 createNotification	81
6.1.6 onStudentGroupRemoved	82
6.1.7 OnSessionRemoved	83
6.1.8 createUser	84
6.1.9 sendEmail	85
6.1.10 changePassword	86
6.1.11 upadateCalendar	87
6.1.12 onGroupClick	88
6.1.13 onSessionClick	89
6.2 Flowchart	90
6.3 Activity Diagram	93
7 Rilascio e Testing	100
7.1 Deployment Diagram	100
7.1.1 Install view	102
7.2 Testing d'unità	102
7.3 Testing d'integrazione	112
7.3.1 Testing Backend	112
7.3.2 Test intero sistema tramite GUI	120

8 Manuale Utente	124
8.1 Pagina login (/login)	124
8.2 Navbar	126
8.3 Dashboard (/dashboard)	126
8.4 Profilo personale (/profile/me)	128
8.5 Profilo amico (/profile/username)	128
8.6 Gruppi (/groups)	131
8.7 Sessioni (/sessions)	134
8.8 Notifiche e gestione richieste	137

Chapter 1

Introduzione

StudyApp è una web-app progettata per facilitare l'organizzazione di sessioni di studio tra studenti con interessi accademici simili. L'applicazione offre un ambiente virtuale in cui gli utenti possono trovare vecchi e nuovi compagni di studio, pianificare e partecipare a sessioni di studio collettive, con lo scopo di condividere conoscenze e risorse pertinenti alle loro materie di studio.

1.1 Unified Process

StudyApp è stata concepita e sviluppata seguendo le linee guida del Modello di Processo Software Unified Process (UP). Si è ritenuto essenziale adottare un processo di sviluppo iterativo al fine di creare un'applicazione in grado di adattarsi prontamente a eventuali modifiche nei requisiti funzionali. Inoltre, era fondamentale avere feedback

regolari per il team di sviluppo al fine di monitorare lo stato del progetto.

Una scelta altrettanto significativa è stata quella di seguire un modello di sviluppo guidato dalla valutazione del rischio. Questo approccio implica di concentrarsi sulle funzionalità di base inizialmente, mentre le funzionalità accessorie sono state considerate in una fase successiva. Poiché StudyApp è un'applicazione web intrinsecamente interattiva, è stato adottato un modello basato sui casi d'uso, concentrandoci sulle sequenze di azioni e sulle modalità di interazione tra gli utenti finali e il sistema.

Un'altra scelta strategica è stata quella di fare uso del linguaggio UML per definire e realizzare i diagrammi, al fine di garantire una documentazione uniforme e conforme agli standard. Il processo UP si è dimostrato adatto a questi scopi.

Lo sviluppo incrementale è stato realizzato suddividendo il progetto in due iterazioni della durata di 16 giorni ciascuna. Ogni iterazione aveva come obiettivo la produzione di un incremento valido, ovvero un sottosistema (o il sistema finale) eseguibile, testato, funzionante correttamente e in linea con i requisiti specificati. Questa tecnica ha permesso raffinamenti sia nelle specifiche dei requisiti che nell'architettura del sistema.

Essendo stato adottato un modello evolutivo, anziché prototipale, è stato inizialmente creato un nucleo di base dell'architettura del sis-

tema nella prima iterazione. Successivamente, nell'iterazione finale, sono state apportate modifiche e raffinamenti concentrandosi sulle funzionalità meno critiche.

Sono stati pianificati due workshop per la definizione degli obiettivi dell'iterazione successiva e un workshop di riepilogo durante le iterazioni per esaminare passo dopo passo i risultati ottenuti. Indipendentemente da questi workshop, il team ha collaborato quotidianamente per coordinare le attività e lavorare insieme su compiti specifici come la stesura della documentazione e il testing di alcune funzionalità.

1.2 Prima Iterazione

Durante la prima iterazione, l'attenzione del team è stata concentrata sulle attività più critiche e fondamentali dell'applicazione StudyApp. Prima di tutto è stata condotta un'analisi approfondita del dominio applicativo, identificando le entità concettuali del progetto e delineando i requisiti funzionali e non funzionali. Questa fase ha preparato il terreno per la progettazione dell'architettura di base del sistema.

Nel corso della prima iterazione, abbiamo focalizzato i nostri sforzi sulla realizzazione di funzionalità chiave, tra cui il sistema di registrazione, login e autenticazione, la gestione del proprio profilo e la creazione di sessioni e gruppi con la possibilità di aggiungervi utenti. Un principio guida importante è stato il "security by design", ovvero l'attenzione alla sicurezza sin dalle prime fasi di progettazione.

Durante questa fase, sono stati generati una serie di documenti cruciali, tra cui:

- Documento di visione che ha delineato la direzione del progetto.
- Use Case Diagram
- Scenari di alto livello legati ai casi d'uso implementati nella prima iterazione.
- Entity-Relationship Diagram
- Class Diagram
- Package Diagram
- Component Diagram
- Sequence Diagram

I casi d'uso identificati e implementati nella prima iterazione includono:

- Registrazione
- Login
- Visualizzare il proprio profilo
- Creare un gruppo
- Modificare un gruppo

- Visualizzare i propri gruppi
- Visualizzare i membri di un gruppo
- Creare una sessione
- Creare una sessione con i membri di un gruppo
- Modificare una sessione
- Visualizzare le proprie sessioni
- Visualizzare i membri di una sessione
- Visualizzare Dashboard

Per ulteriori dettagli sulla descrizione di questi casi d'uso verranno approfonditi nel capitolo relativo ai requisiti funzionali dell'applicazione. Inoltre, durante la prima iterazione, sono stati svolti i primi test di unità e integrazione.

1.3 Seconda Iterazione

Nella seconda iterazione, l'obiettivo principale era l'espansione dell'architettura software con l'introduzione di funzionalità accessorie. Queste funzionalità includono un sistema di notifica per visualizzare richieste di partecipazione ad un gruppo o una sessione, una gestione

tramite email di token nel caso di password dimenticata e un calendario per pianificare eventi personali. Durante questa fase, sono stati prodotti i seguenti documenti:

- Sequence Diagram di dettaglio relativi ai casi d'uso implementati nella prima iterazione.
- Scenari di alto livello legati ai casi d'uso implementati nella seconda iterazione.
- Sequence Diagram di dettaglio relativi ai casi d'uso implementati nella seconda iterazione.
- Deployment Diagram
- Install View

I casi d'uso affrontati durante la seconda iterazione includono:

- Recupero Password
- Modificare il proprio profilo
- Cambiare password
- Visualizzare notifiche
- Accettare invito gruppo/sessione
- Visualizzare calendario

- Creare nuovo evento

Durante questa fase, sono stati condotti i test di unità e integrazione finali, e la documentazione del progetto è stata completata.

1.4 Tecnologie di supporto

1.4.1 Vaadin



Vaadin è un framework HTML, CSS e Javascript per lo sviluppo di applicazioni web, in particolare per il front-end. Fornisce componenti per l'interfaccia grafica, come pulsanti, moduli di compilazione, navbar e cards.

1.4.2 Spring Boot



Spring è un framework open-source per lo sviluppo di applicazioni web

che supporta il linguaggio Java. In particolare, si è utilizzato Spring Boot, un micro-framework derivato da Spring.

1.4.3 Eclipse



Eclipse è un ambiente di sviluppo integrato (IDE) per il linguaggio di programmazione Java, sviluppato da Eclipse Foundation e disponibile in licenza gratuita. L'IDE è stato fondamentalmente utilizzato per lo sviluppo sia del back-end che front-end, grazie alla familiarità del team con l'ambiente e alle numerose funzionalità offerte.

1.4.4 Firefox



Firefox è un browser web sviluppato da Mozilla. È stato utilizzato dal team per testare l'interfaccia utente e le funzionalità del client.

1.4.5 Discord



Discord è una piattaforma di comunicazione e collaborazione utilizzata dal team per pianificare riunioni e condividere materiale.

1.4.6 Visual Paradigm



Visual Paradigm è uno strumento di supporto alla progettazione dei diagrammi UML. È stato utilizzato dal team per realizzare i diagrammi, con l'utilizzo della versione gratuita (Community Edition).

1.4.7 Github



GitHub è una piattaforma di sviluppo collaborativo basata su Git, utilizzata dal team per lavorare contemporaneamente sullo stesso codice, condividendo file e modifiche reciproche.

Chapter 2

Visione

2.1 Descrizione testuale dei requisiti

StudyApp è una web application che offre la possibilità agli utenti di registrarsi e creare sessioni di studio.

Queste permetteranno agli utenti di stabilire incontri in base a orari e luoghi specifici. Le funzionalità dell'applicazione saranno disponibili solo ad utenti registrati. Gli utenti registrati avranno accesso a una dashboard personalizzata in cui potranno visualizzare il proprio calendario di impegni passati e futuri, ricevendo notifiche pertinenti. La piattaforma consentirà anche di stabilire connessioni con altri utenti mediante i gruppi, semplificando l'invito a sessioni di studio.

2.2 Obiettivi

StudyApp ha come obiettivo fornire all’utenza le seguenti funzionalità:

- Creare gruppi invitando altri studenti con il fine di studiare assieme
- Organizzare sessioni di studio selezionando luogo, orario e studenti da invitare
- Controllare sessioni ed eventi futuri mediante un calendario ed un sistema di notifiche
- Visualizzare statistiche e informazioni utili mediante una dashboard

2.3 Portatori di interesse

L’utente indicato per la piattaforma è uno studente con interesse nell’organizzarsi con altri utenti per studiare assieme. L’organizzazione delle sessioni di studio può essere semplificata e velocizzata mediante i gruppi, che racchiudono studenti interessati ad organizzarsi più volte assieme. L’applicazione è utilizzabile unicamente da utenti registrati, i quali possono:

- Accedere al sistema
- Visualizzare la propria Dashboard

- Modificare il proprio profilo
- Controllare i gruppi di cui fa parte
- Creare, modificare ed eliminare i propri gruppi come admin
- Osservare gli eventi prossimi mediante un calendario
- Visualizzare sessioni a cui è invitato, con sistema di notifiche
- Creare, modificare ed eliminare le proprie sessioni come admin

Oltre ad un sistema di registrazione, vi è un meccanismo di Password Recovery, il quale manderà una email all'indirizzo compilato in fase di registrazione con un link che permetterà di inserire una nuova password.

2.4 Glossario

Servendosi della descrizione testuale, sono stati identificati i termini corrispondenti alle entità concettuali del dominio.

Termine	Significato
Group	Gruppo di studenti che vogliono organizzare sessioni assieme
Session	Appuntamento con orario e luogo finalizzato allo studio
Student	Coincidente con l'utente, persona che vuole organizzare il proprio studio
CalendarEntry	Evento creato dall'utente visibile solo da esso
Notification	Notifica di info, evento prossimo o invito ad un gruppo o sessione

2.5 Stima dei tempi

Per fare una stima dei tempi di realizzazione del progetto, è stato adottato il metodo degli Use Case Points. La prima fase di questo processo implica il calcolo degli Unadjusted Use Case Weight (UUCW). In seguito, si procede attribuendo differenti pesi a ciascun Use Case in base al numero delle transazioni associato a ciascuno di essi. Questa assegnazione di pesi fa riferimento alla tabella seguente:

Complessità del caso d'uso	Transazioni	Peso
Bassa	3 o meno	5
Media	da 4 a 7	10
Alta	più di 7	15

I risultati delle stime effettuate per il conteggio dei pesi dei casi d'uso sono stati raccolti in una tabella. È importante notare che i casi d'uso di estensione sono stati omessi in quanto sono conteggiati insieme ai casi d'uso base.

Caso d'uso	Transazioni	Peso
Crea un gruppo	6	10
Gestisci gruppi	3	5
Visualizza gruppi	3	5
Elimina un gruppo	3	5
Crea una sessione	6	10
Gestisci sessioni	3	5
Visualizza sessioni	3	5
Elimina una sessione	3	5
Login	3	5
Registrati	3	5
Gestisci profilo	3	5
Visualizza profilo	3	5
Visualizza calendario	3	5
Crea evento su calendario	6	10
Gestisci calendario	3	5
Visualizza notifiche	3	5
Gestisci notifiche	3	5
Invia richiesta join gruppo	3	5
Invia richiesta join sessione	3	5
Visualizza dashboard	3	5
Totale		115

Il totale degli Unadjusted Use Case Weight (UUCW) è di 115. Prossimo

passo è il calcolo degli Unadjusted Actor Weight (UAW), utilizzando come riferimento la seguente tabella:

Tipo d'attore	Descrizione	Peso
Semplice	Sistema esterno che interagisce tramite API	1
Medio	Sistema esterno che interagisce attraverso un protocollo o una persona che interagisce attraverso un'interfaccia testuale	2
Complesso	Una persona che interagisce attraverso una GUI	3

Nel contesto del sistema analizzato, si identificano due categorie di attori: l’Utente e l’Utente registrato, entrambi rappresentanti di attori complessi che interagiscono con il nostro sistema tramite un’interfaccia grafica. Nonostante le distinzioni nominali, entrambi gli utenti interagiscono in modo analogo con il sistema, consentendo di trattarli come un singolo attore.

Di conseguenza, si ottiene un valore di UAW pari a 3. L’Unadjusted Use Case Point (UUCP) viene quindi calcolato come la somma degli UUCW e dell’UAW, risultando in un valore di UUCP pari a 118.

Il procedimento successivo prevede il calcolo dei fattori di aggiustamento, TCF (Technical Complexity Factor) ed EF (Environmental Factor), che contribuiranno al calcolo del valore degli Use Case Point (UCP).

Nella tabella successiva sono riportati i dettagli del calcolo del TFactor specifico per il sistema in esame.

Fattore	Descrizione	Peso	Valutazione	Prodotto
T1	Sistema distribuito	2	0	0
T2	Obiettivi di performance	2	1	2
T3	Efficienza per l'utente finale	1	2	2
T4	Complessità di elaborazione	1	1	1
T5	Riusabilità del codice	1	1	1
T6	Facilità di installazione	0.5	2	1
T7	Facilità d'uso	0.5	2	1
T8	Portabilità	2	1	2
T9	Modificabilità	1	2	2
T10	Calcolo parallelo	1	0	0
T11	Sicurezza	1	2	2
T12	Accesso per terze parti	1	0	0
T13	Necessità di training per l'utente finale	1	0	0
TFactor				14

Quindi:

$$TCF = 0.6 + (0.01 * TFactor) = 0.74$$

Nella tabella successiva, invece, è riportato il calcolo dell'EFactor:

Fattore	Descrizione	Peso	Valutazione	Prodotto
E1	Familiarità col processo di sviluppo	1.5	4	6
E2	Esperienza applicativa	0.5	4	2
E3	Esperienza object oriented	1	4	4
E4	Capacità di analisi	0.5	4	2
E5	Motivazione	1	5	5
E6	Stabilità dei requisiti	2	5	10
E7	Personale part-time	-1	0	0
E8	Complessità del linguaggio di programmazione	-1	2	-2
EFactor				27

Quindi:

$$EF = 1.4 + (-0.03 * EFactor) = 0.59$$

Complessivamente possiamo calcolare gli UCP complessivi come:

$$UCP = UUCP * TCF * ECF = 118 * 0.74 * 0.59 = 51$$

Andando a considerare ogni UCP pari a 15 ore di lavoro, si ha un totale di 765 ore di lavoro. Distribuendo queste ore tra i tre componenti del team e considerando le giornate di lavoro di 8 ore, è stata prevista una durata del progetto di circa 32 giorni

Chapter 3

Requisiti funzionali e analisi di dominio

In questa sezione della documentazione, vengono esposti i risultati dell'analisi del dominio, che includono il diagramma UML dei casi d'uso e i relativi scenari. Inoltre, viene presentato il modello di dominio del sistema, il quale visualizza le entità concettuali del dominio e le associazioni che le collegano. Infine, il modello dei dati è dettagliato tramite l'utilizzo del diagramma Entity-Relationship.

3.1 Casi d'uso

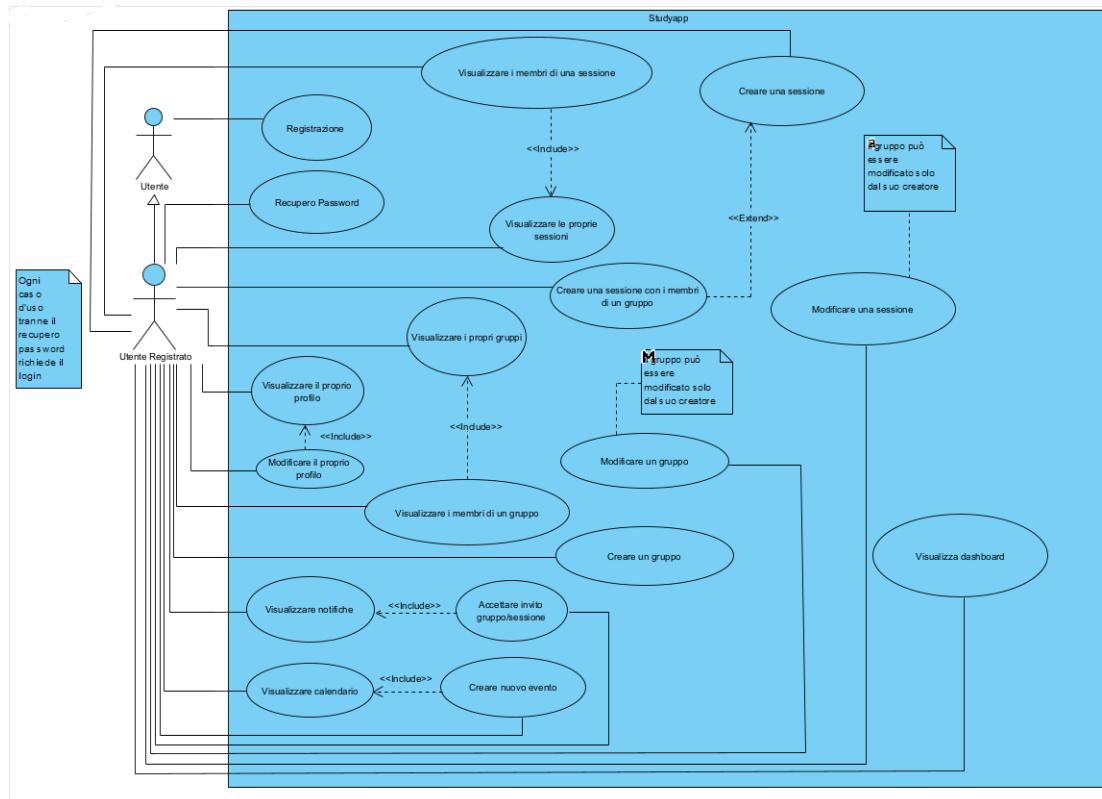


Figure 3.1: Use Case Diagram

Di seguito sono esplicitati i casi d'uso individuati dal diagramma.

3.1.1 Registrazione

- Nome: Registrazione
- Scopo: permettere ad un utente di registrarsi.
- Precondizioni: l'utente non risulta registrato.
- Flusso Principale:

1. L'utente inserisce username, nome, cognome, email, password, campo di studi, anno di frequentazione e data di nascita nella pagina di Registrazione.
 2. Il sistema valida i campi, in particolare controllando che non ci sia un altro utente associato alla stessa e-mail o allo stesso username.
 3. Il sistema crea l'account e restituisce un messaggio di avvenuta registrazione.
 4. L'utente viene reindirizzato alla pagina di login.
- Flusso Secondario: il fallimento della validazione al secondo punto genera un messaggio di errore e l'account non viene creato.
 - Post-condizioni: L'account relativo ai dati inseriti dall'utente viene memorizzato.
 - Attori: Utente

3.1.2 Login

- Nome: Login
- Scopo: permettere ad un utente registrato di autenticarsi ed accedere alle funzionalità dell'applicazione
- Precondizioni: L'utente deve essere registrato.

- Flusso Principale:
 1. L'utente fornisce Username e Password al sistema attraverso la pagina di accesso.
 2. Il sistema verifica la validità e corrispondenza delle credenziali.
 3. Il sistema autentica e reindirizza l'utente sulla main page.
- Flusso Secondario: La verifica al punto 2 fallisce e il sistema notifica l'utente con un messaggio di errore
- Post-Condizioni: L'utente è autenticato.
- Attori: Utente Registrato

3.1.3 Recupero Password

- Nome: Recupero Password
- Scopo: permettere ad un utente di scegliere una nuova password per un account legato ad un email del quale è in possesso.
- Precondizioni: L'utente deve essere registrato e avere accesso alla email legata all'account sul quale si tenta il recupero password
- Flusso Principale:
 1. L'utente inserisce l'email legata all'account.

2. Il sistema da un messaggio di successo, invitando l'utente a controllare la propria inbox.
 3. Il sistema verifica che l'email sia effettivamente legata ad un account esistente.
 4. Il sistema genera un token dal quale ricava un link che viene inviato all'email indicata.
 5. L'utente naviga al link trovato nella propria inbox e inserisce una nuova password mediante un apposito form.
 6. Il sistema distrugge il token precedentemente generato e cambia la password legata all'account.
- Flusso Secondario: la verifica al punto 3 fallisce e il sistema non fa nulla.
 - Post-Condizioni(1): La password relativa all'account viene cambiata.
 - Attori: Utente Registrato.

3.1.4 Visualizzare il proprio profilo

- Nome: Visualizzare il proprio profilo
- Scopo: permettere all'utente di vedere i dati relativi al proprio account

- Precondizioni: L'utente deve essere registrato ed aver effettuato con successo il login.
- Flusso Principale:
 1. L'utente naviga sulla pagina del profilo
 2. Il sistema recupera le informazioni relative al profilo con una ricerca sulla base dell'username.
 3. Il sistema restituisce le informazioni trovate inserendole nella view apposita.
- Attori: Utente Registrato
- Casi d'uso inclusi: Login

3.1.5 Modificare il proprio profilo

- Nome: Modificare il proprio profilo
- Scopo: permettere all'utente di modificare uno o più campi informativi legati al proprio account.
- Precondizioni: L'utente deve essere registrato ed aver effettuato con successo il login.
- Flusso Principale:
 1. L'utente modifica uno o più campi del proprio profilo

2. Il sistema valida i campi e abilita il pulsante di modifica
 3. L'utente clicca il pulsante di modifica
 4. Il sistema modifica nel database i campi modificati dall'utente sostituendo i vecchi dati con i nuovi e notifica l'utente del successo dell'operazione
- Flusso Secondario: la validazione dei campi fallisce e il cliccare il pulsante di modifica non provoca cambi, mostrando all'utente un messaggio di errore per ogni campo errato.
 - Post-Condizioni(1): i campi modificati dall'utente vengono aggiornati nel database.
 - Attori: Utente registrato
 - Casi d'uso inclusi: Login, Visualizza il proprio profilo.

3.1.6 Cambiare Password

- Nome: Cambiare Password
- Scopo: permettere ad un utente che ha effettuato il login di cambiare la propria password
- Precondizioni: L'utente deve essere registrato ed aver effettuato con successo il login.
- Flusso Principale:

1. L'utente inserisce la nuova password e la conferma in un campo apposito.
 2. Il sistema valida la password, controllandone forma e corrispondenza con il campo di conferma password
 3. Il sistema aggiorna la password e reindirizza l'utente alla pagina del profilo.
- Flusso Secondario: La validazione al punto 2 fallisce e il sistema da un messaggio di errore.
 - Post-Condizioni(1): La password relativa all'account dell'utente viene aggiornata.
 - Attori: Utente Registrato
 - Casi d'uso inclusi: Login, Visualizza il proprio profilo.
 - Casi d'uso estesi: Modifica il proprio profilo.

3.1.7 Visualizzare notifiche

- Nome: Visualizzare notifiche
- Scopo: permettere all'utente di visualizzare notifiche su informazioni utili ed inviti a gruppi o sessioni.
- Precondizioni: L'utente deve essere registrato ed aver effettuato con successo il login.

- Flusso Principale:
 1. L'utente clicca sul pulsante delle notifiche
 2. Il sistema mostra le notifiche correnti.
- Flusso Secondario: non ci sono notifiche, non vi è seguito al passo 1.
- Attori: Utente Registrato
- Casi d'uso inclusi: Login

3.1.8 Accettare invito gruppo/sessione

- Nome: Accettare invito gruppo/sessione
- Scopo: permettere all'utente di accettare un invito alla partecipazione
- Precondizioni: L'utente deve essere registrato, aver effettuato con successo il login e aver ricevuto almeno un invito a partecipare ad un gruppo o sessione
- Flusso Principale:
 1. L'utente clicca sul pulsante delle notifiche
 2. Il sistema mostra l'invito a partecipare con un relativo pulsante per accettare.

3. L'utente clicca sul pulsante
 4. Il sistema aggiunge l'utente alla sessione/gruppo
- Post-Condizione: Al gruppo o alla sessione viene aggiunto l'utente che ha accettato.
 - Attori: Utente Registrato
 - Casi d'uso inclusi: Login, Visualizzare Notifiche

3.1.9 Visualizzare calendario

- Nome: Visualizzare calendario
- Scopo: permettere all'utente di visualizzare una vista sugli eventi e sessioni già schedulati
- Precondizioni: L'utente deve essere registrato ed aver effettuato con successo il login.
- Flusso Principale:
 1. L'utente naviga alla pagina del calendario
 2. Il sistema carica la vista del calendario con gli eventi presenti in memoria.
- Attori: Utente Registrato
- Casi d'uso inclusi: Login

3.1.10 Creare nuovo evento

- Nome: Creare nuovo evento
- Scopo: permettere all'utente di creare eventi visualizzabili sul calendario e soggetti a notifiche.
- Precondizioni: L'utente deve essere registrato ed aver effettuato con successo il login.
- Flusso Principale:
 1. L'utente clicca su una casella del calendario.
 2. Il sistema mostra un dialog con dei form caratterizzanti l'evento.
 3. L'utente riempie i form.
 4. Il sistema valida i campi inseriti.
 5. Il sistema inserisce l'evento nel database e notifica l'utente del successo dell'operazione.
- Flusso Secondario: La validazione al punto 4 fallisce e il sistema lo mostra con un messaggio di errore per ogni campo sul quale è fallita la validazione.
- Post-Condizione(1): Nel database viene salvato l'evento creato dall'utente.
- Attori: Utente Registrato
- Casi d'uso inclusi: Login, Visualizza calendario

3.1.11 Creare gruppo

- Nome: Creare gruppo
- Scopo: permettere all’utente di creare un gruppo con altri utenti.
- Precondizioni: L’utente deve essere registrato, aver effettuato con successo il login e conoscere l’username degli utenti con il quale formare il gruppo.
- Flusso Principale:
 1. L’utente naviga sulla view dei gruppi e clicca sul bottone per aggiungere un gruppo.
 2. Il sistema mostra un dialog.
 3. L’utente compila i form inserendo il nome del gruppo ed eventuali utenti invitati.
 4. Il sistema valida i campi inseriti.
 5. Il sistema crea il gruppo e invia notifiche di invito a tutti gli utenti indicati dal creatore del gruppo.
- Flusso Secondario: La validazione al punto 4 fallisce e il sistema lo mostra con un messaggio di errore per ogni campo sul quale è fallita la validazione.
- Post-Condizione(1): Il sistema crea il gruppo mettendo l’utente come admin e unico membro, manda notifiche di invito a tutti

gli altri utenti.

- Attori: Utente Registrato
- Casi d'uso inclusi: Login

3.1.12 Modificare gruppo

- Nome: Modificare gruppo.
- Scopo: permettere all'utente di modificare il gruppo che ha creato in precedenza.
- Precondizioni: L'utente deve essere registrato, aver effettuato con successo il login e deve essere l'admin del gruppo.
- Flusso Principale:
 1. L'utente seleziona il gruppo che vuole modificare.
 2. Il sistema mostra un dialog.
 3. L'utente compila i form aggiornando nome del gruppo ed eventuali nuovi utenti invitati.
 4. Il sistema valida i campi inseriti.
 5. Il sistema aggiorna le informazioni del gruppo e invia notifiche di invito a tutti gli utenti nuovi indicati dal creatore del gruppo.

- Flusso Secondario: La validazione al punto 4 fallisce e il sistema lo mostra con un messaggio di errore per ogni campo sul quale è fallita la validazione.
- Post-Condizione(1): Il sistema aggiorna il gruppo ed eventualmente manda notifiche di invito a tutti gli altri utenti.
- Attori: Utente Registrato
- Casi d'uso inclusi: Login

3.1.13 Visualizzare i propri gruppi

- Nome: Visualizzare i propri gruppi
- Scopo: permettere all'utente di vedere i gruppi di cui fa parte
- Precondizioni: L'utente deve essere registrato, aver effettuato con successo il login e fare parte almeno di un gruppo.
- Flusso Principale:
 1. L'utente naviga sulla pagina di visualizzazione dei gruppi
 2. Il sistema recupera i gruppi di cui l'utente è parte e li mostra.
- Attori: Utente Registrato
- Casi d'uso inclusi: Login

3.1.14 Visualizzare i membri di un gruppo

- Nome: Visualizzare i membri di un gruppo
- Scopo: permettere all'utente di vedere i membri di un gruppo
- Precondizioni: L'utente deve essere registrato, aver effettuato con successo il login e deve far parte di almeno un gruppo.
- Flusso Principale:
 1. L'utente naviga sulla pagina di visualizzazione dei gruppi
 2. Il sistema recupera i gruppi di cui l'utente è parte e li mostra
 3. L'utente seleziona il gruppo
 4. Il sistema recupera i membri del gruppo e li mostra
- Attori: Utente Registrato
- Casi d'uso inclusi: Login, Visualizzare i propri gruppi

3.1.15 Visualizzare le proprie sessioni

- Nome: Visualizzare le proprie sessioni
- Scopo: permettere all'utente di vedere le sessioni al quale partecipa
- Precondizioni: L'utente deve essere registrato, aver effettuato con successo il login e partecipare ad almeno una sessione.

- Flusso Principale:
 1. L'utente naviga sulla pagina di visualizzazione delle sessioni
 2. Il sistema recupera le sessioni di cui l'utente è parte e le mostra.
- Attori: Utente Registrato
- Casi d'uso inclusi: Login

3.1.16 Visualizzare i membri di una sessione

- Nome: Visualizzare i membri di una sessione
- Scopo: permettere all'utente di vedere i membri di una sessione.
- Precondizioni: L'utente deve essere registrato, aver effettuato con successo il login e deve far parte di almeno una sessione.
- Flusso Principale:
 1. L'utente naviga sulla pagina di visualizzazione delle sessioni
 2. Il sistema recupera le sessioni di cui l'utente è parte e le mostra.
 3. L'utente seleziona una sessione
 4. Il sistema recupera i membri della sessione e li mostra
- Attori: Utente Registrato

- Casi d'uso inclusi: Login, Visualizzare le proprie sessioni

3.1.17 Creare sessione

- Nome: Creare sessione
- Scopo: permettere all'utente di creare una sessione per prendere appuntamento con altri utenti.
- Precondizioni: L'utente deve essere registrato, aver effettuato con successo il login e conoscere l'username degli utenti da invitare.
- Flusso Principale:
 1. L'utente naviga sulla view delle sessioni e clicca sul bottone per aggiungere una sessione.
 2. Il sistema mostra un form.
 3. L'utente compila il form inserendo la materia di studio, il luogo d'incontro, la data ed eventuali utenti invitati.
 4. Il sistema valida i campi inseriti.
 5. Il sistema crea la sessione e invia notifiche di invito a tutti gli utenti indicati dal creatore della sessione.
- Flusso Secondario: La validazione al punto 4 fallisce e il sistema lo mostra con un messaggio di errore per ogni campo sul quale è fallita la validazione.

- Post-Condizione(1): Il sistema crea la sessione mettendo l'utente come admin e unico partecipante, manda notifiche di invito a tutti gli altri utenti e crea un evento sul calendario.
- Attori: Utente Registrato
- Casi d'uso inclusi: Login

3.1.18 Creare una sessione con i membri di un gruppo

- Nome: Creare una sessione con i membri di un gruppo
- Scopo: permettere all'utente di creare una sessione invitando i membri di un gruppo preesistente
- Precondizioni: L'utente deve essere registrato, aver effettuato con successo il login e fare parte di un gruppo.
- Flusso Principale:
 1. L'utente naviga sulla pagina di visualizzazione dei gruppi.
 2. Il sistema recupera i gruppi di cui l'utente è parte e li mostra.
 3. L'utente seleziona il gruppo.
 4. L'utente clicca sul bottone per aggiungere una sessione.
 5. Il sistema mostra un form.

6. L'utente compila il form inserendo la materia di studio, il luogo d'incontro, la data ed eventuali utenti invitati.
 7. Il sistema valida i campi inseriti.
 8. Il sistema crea la sessione e invia notifiche di invito a tutti i membri dal gruppo.
- Flusso Secondario:
 - Attori: Utente Registrato
 - Casi d'uso inclusi: Login
 - Casi d'uso estesi: Creare una sessione

3.1.19 Modificare sessione

- Nome: Modificare sessione.
- Scopo: permettere all'utente di modificare la sessione che ha creato in precedenza.
- Precondizioni: L'utente deve essere registrato, aver effettuato con successo il login e deve essere l'admin della sessione.
- Flusso Principale:
 1. L'utente seleziona la sessione che vuole modificare.
 2. Il sistema mostra un form.

3. L'utente compila il form aggiornando materia di studio, il luogo d'incontro, la data ed eventuali nuovi utenti invitati.
 4. Il sistema valida i campi inseriti.
 5. Il sistema aggiorna le informazioni della sessione e invia notifiche di invito a tutti gli utenti nuovi indicati dall'admin della sessione.
- Flusso Secondario: La validazione al punto 4 fallisce e il sistema lo mostra con un messaggio di errore per ogni campo sul quale è fallita la validazione.
 - Post-Condizione(1): Il sistema aggiorna la sessione ed eventualmente manda notifiche di invito a tutti gli altri utenti.
 - Attori: Utente Registrato
 - Casi d'uso inclusi: Login

3.1.20 Visualizzare dashboard

- Nome: Visualizzare dashboard
- Scopo: permettere all'utente di guardare informazioni utili su una dashboard
- Precondizioni: L'utente deve essere registrato ed aver effettuato con successo il login

- Flusso Principale:
 1. L'utente visita la pagina principale
 2. Il sistema recupera le informazioni da caricare nella dashboard
 3. Il sistema mostra la dashboard con i dati aggiornati
- Attori: Utente Registrato
- Casi d'uso inclusi: Login

3.2 Context Diagram

Ecco il System Context Diagram, il quale delinea chiaramente i confini del sistema, considerato come una scatola nera, e l'ambiente esterno.

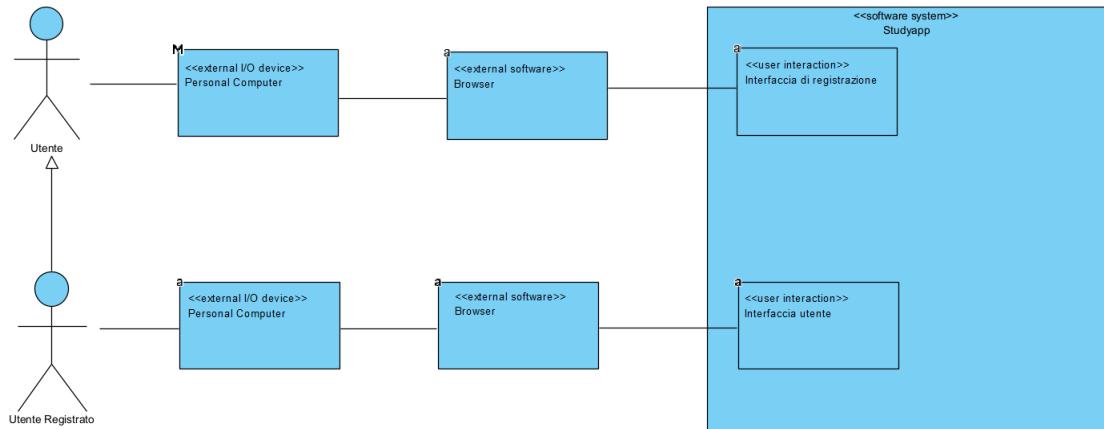


Figure 3.2: System Context Diagram

3.3 System Domain Model

Di seguito è presentato il System Domain Model derivato dall'analisi testuale. Il diagramma illustra le entità concettuali del dominio e le relative associazioni/relazioni.

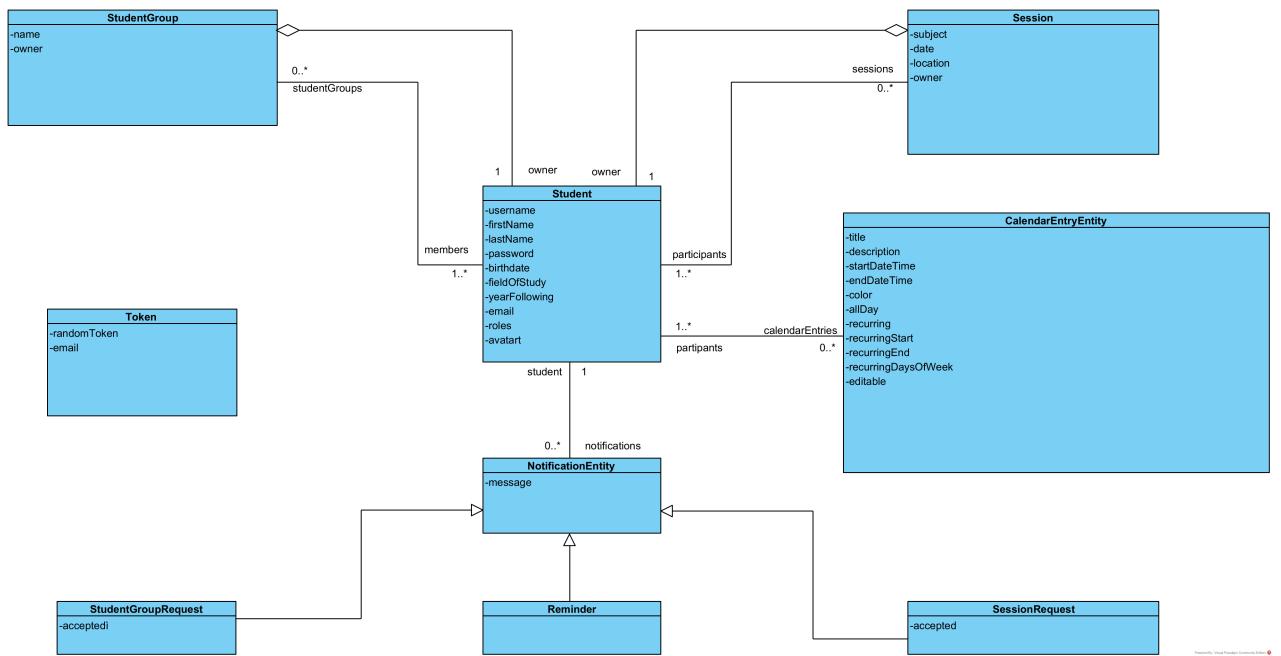


Figure 3.3: System Domain Model

3.4 Entity-Relationship Diagram

Di seguito è riportato l'Entity-Relationship Diagram, che mette in risalto le associazioni tra le tabelle del database relazionale. Si noti che ogni tabella corrisponde ad una ed una sola entità concettuale del Domain Model, fatta eccezione per la *notification entity*.

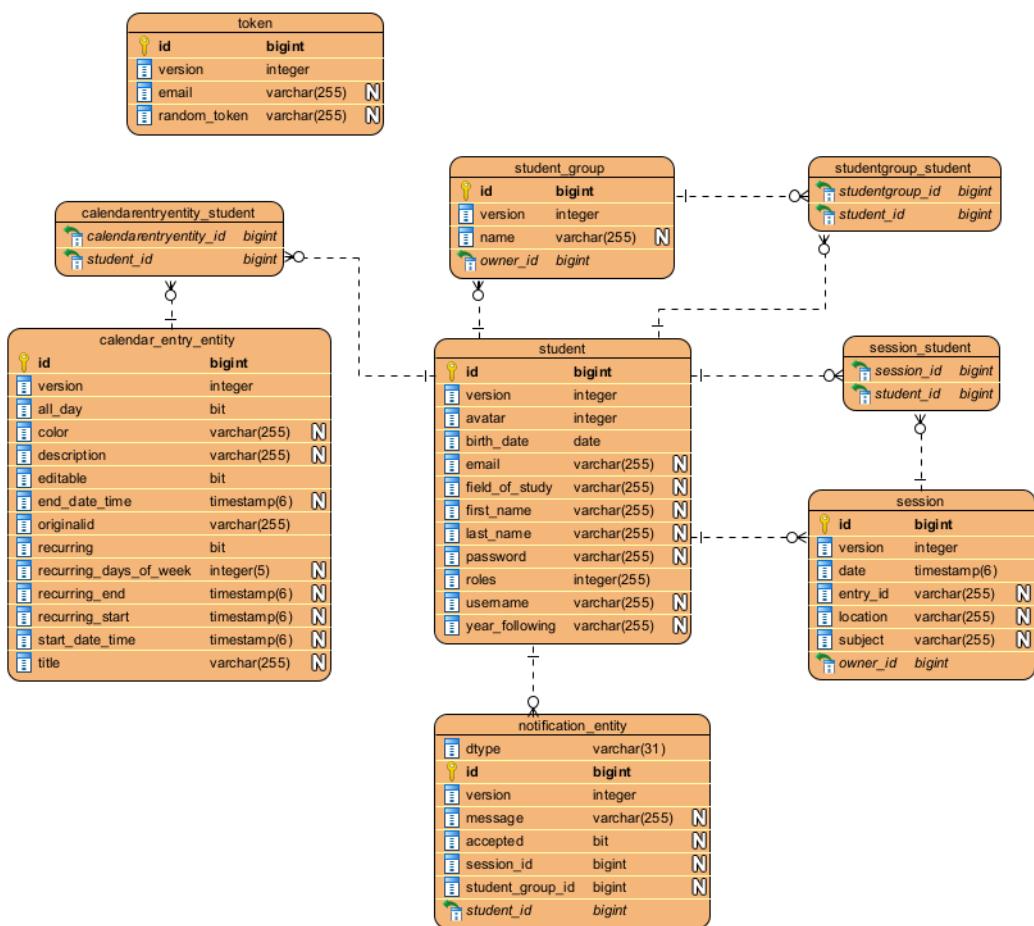


Figure 3.4: Entity-Relationship Diagram

Chapter 4

Requisiti non funzionali

Nella seguente sezione, sono elencati i principali attributi di qualità dell'applicazione, accompagnati dalla loro relativa descrizione scenistica. Un tipico scenario comprende i seguenti elementi chiave:

- Origine (Source): L'entità o l'evento che innesca l'avvio dello scenario.
- Stimolo (Stimulus): L'evento o l'azione specifica che dà inizio allo scenario, solitamente associato a un cambio di stato o a una richiesta specifica.
- Artefatto (Artifact): La parte specifica del sistema coinvolta nell'attuazione dello scenario, evidenziando i componenti o i moduli coinvolti.
- Ambiente (Environment): Il contesto operativo in cui si verifica lo scenario, spesso indicando se il sistema sta operando in una

situazione di utilizzo normale o in circostanze particolari.

- Risposta (Response): La reazione del sistema all'evento o all'azione scatenante, inclusi i cambiamenti di stato o le azioni intraprese.
- Misura della risposta (Response measure): Una metrica quantitativa utilizzata per valutare oggettivamente l'efficacia e l'adeguatezza della risposta del sistema nell'ambito di uno scenario specifico.

4.1 Scenari

4.1.1 Modifiability

La capacità di apportare modifiche è sostenuta sia dall'architettura client/server che dalla struttura a livelli del sistema. L'architettura client/server comporta un basso accoppiamento tra i componenti del sistema poiché il server è scarsamente vincolato a quest'ultimo. Inoltre, non vi è alcun accoppiamento tra i client, poiché operano in modo indipendente e non comunicano tra di loro. Client e server possono essere modificati in modo indipendente, a condizione che le interfacce concordate durante la fase di progettazione rimangano invariate. La struttura a livelli, inoltre, promuove sia un'alta coesione all'interno dei moduli, poiché ciascun livello ha responsabilità ben definite, sia un basso accoppiamento tra i moduli, dato che ogni livello comunica solo con quello sottostante. L'elevata coesione e il basso accoppiamento,

entrambi derivanti da questa struttura, influenzano positivamente la possibilità di apportare modifiche al sistema software realizzato.

- Origine (Source): il team di sviluppo
- Stimolo (Stimulus): aggiunta delle funzionalità relativa alla modifica del profilo
- Artefatto (Artifact): classi associate a Studente, Gruppi, Sessioni ed Eventi
- Ambiente (Environment): fase di progettazione
- Risposta (Response): funzionalità aggiunta correttamente
- Misura della risposta (Response measure): occorrono sulle 5 ore per apportare questa modifica

4.1.2 Robustness

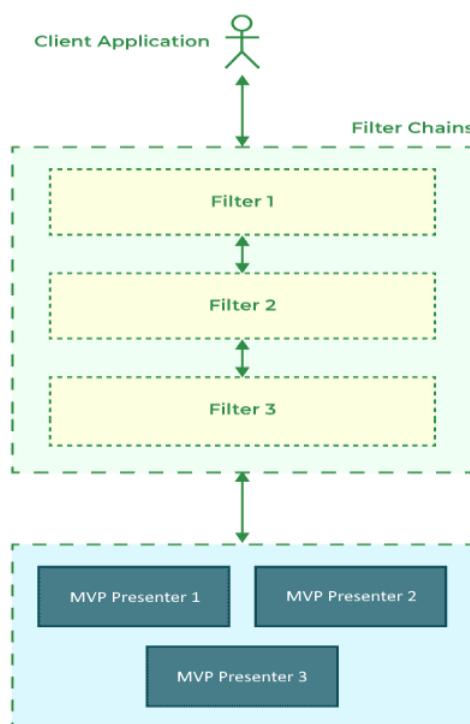
Il sistema è robusto poichè valida ogni input compilabile dall'utente.

- Source: l'utente non registrato
- Stimulus: compilazione del form di registrazione
- Artifact: classi associate a Studente e form di registrazione
- Environment: funzionamento normale del sistema

- Response: il sistema permette la registrazione solo se i campi sono tutti riempiti, la data di nascita è relativa ad un utente di almeno 10 anni, l'username non è stato già scelto e l'email è unica e in formato corretto.
- Response measure: -

4.1.3 Security

Per garantire la sicurezza del nostro progetto, abbiamo adottato il principio della "Security by Design" utilizzando le funzionalità di sicurezza di Spring Security. Nel nostro approccio, focalizziamo l'attenzione sulla configurazione della catena di filtri di sicurezza di Spring, una componente fondamentale per gestire l'autenticazione e l'autorizzazione degli utenti.



La catena di filtri di sicurezza è configurata attraverso la classe SecurityConfig, la quale aggiunge filtri alla catena di default di Spring Security. Questa classe definisce una serie di filtri che vengono applicati in sequenza ad ogni richiesta ricevuta dal sistema. I filtri sono progettati per essere eseguiti in un ordine specifico. In caso di una richiesta ad un endpoint protetto da parte di un utente non autenticato o non autorizzato, la catena di filtri di sicurezza negherà l'accesso, garantendo così che solo gli utenti autenticati e autorizzati possano interagire con le risorse protette. L'ordine e la configurazione dei filtri nella catena sono fondamentali per garantire un sistema sicuro e controllato.

- Source: utente che non può fornire credenziali valide
- Stimulus: richiesta ad un endpoint protetto del sistema
- Artifact: server, JwtAuthorizationFilter, SecurityConfig
- Environment: funzionamento normale del sistema
- Response: al momento della richiesta, il sistema nega l'autorizzazione all'utente per effettuare tale richiesta, non inviandola.
- Response measure: -

4.1.4 Usability

L’usabilità è assicurata dall’impiego di un browser come client, eliminando la necessità di dipendenze o programmi esterni che richiedano installazioni da parte dell’utente. La registrazione all’applicazione e l’utilizzo del browser sono sufficienti per consentire all’utente di beneficiare di tutte le funzionalità offerte.

- Source: utente registrato
- Stimulus: creare un gruppo
- Artifact: interfaccia grafica
- Environment: funzionamento normale del sistema
- Response: si può accedere alla funzionalità navigando alla pagina corretta, premendo un bottone e compilando il form del gruppo
- Response measure: Per il calcolo della misura di risposta, vengono identificate tre categorie di utenti.
 1. Utente Esperto: si tratta di un utente con esperienza, in grado di navigare in modo agevole su un sito web.
 2. Utente Medio: si riferisce a un utente capace di raggiungere una soluzione dopo un numero limitato di tentativi.
 3. Utente Poco Capace e Poco Esperto: indica un utente con limitata competenza e poca esperienza nella navigazione su

un sito web.

La metrica selezionata per il calcolo si fonda su una media pesata dei tempi di risoluzione del problema. La media è direttamente influenzata dalla frequenza delle diverse classi di utenti:

$$T = \frac{0.25 \cdot T_a + 0.5 \cdot T_b + 0.25 \cdot T_c}{T_a + T_b + T_c}$$

Chapter 5

Architettura del sistema

In questa sezione, vengono forniti dettagli approfonditi sull'architettura del sistema software, in linea con quanto anticipato precedentemente, ossia che il software adotta il paradigma client/server.

5.1 Front-End

Il front-end dell'applicazione è costituito da un sito web dinamico che espone pagine HTML. La definizione del comportamento e dell'aspetto avviene attraverso il codice Javascript, beneficiando dell'integrazione di un potente framework grafico: Vaadin.

Il framework Vaadin offre una solida base per la progettazione responsiva delle pagine, facilitando la creazione di un'interfaccia utente moderna e visivamente accattivante. Vaadin fornisce una serie di componenti UI avanzati e semplifica la gestione dinamica degli elementi

dell’interfaccia.

Vaadin, con il suo approccio basato su Java, offre la possibilità di costruire interfacce utente complesse e altamente interattive senza richiedere una conoscenza approfondita di Javascript. Questo è particolarmente utile per sviluppatori Java che desiderano creare applicazioni web dinamiche senza dover gestire intricati dettagli lato client.

5.2 Back-End

Il back-end è composto da un server e un database, con il server responsabile della logica di business dell’applicazione e il database gestisce lo storage delle tabelle, come studenti e gruppi, garantendo la persistenza dei dati. L’implementazione del server è stata realizzata attraverso l’utilizzo del framework Spring.

5.2.1 Spring

Spring è ampiamente adottato dagli sviluppatori Java per la generazione di codice altamente testabile, riutilizzabile e performante. Questo framework minimizza l’impatto sulla memoria della web app e si basa su un’architettura modulare, consentendo agli sviluppatori di concentrarsi sulla logica di business senza doversi preoccupare eccessivamente del codice necessario per gestire il web server. Inoltre, Spring supporta diverse tecnologie, come Hibernate per l’Object Relational

Mapping (ORM) e Tomcat come JavaServer Pages (JSP).

Spring Boot

Spring Boot, derivato da Spring, è un micro-framework che semplifica ulteriormente lo sviluppo di applicazioni Spring, configurando automaticamente gli strumenti sottostanti. Include il nucleo di Spring, Hibernate come ORM e Tomcat come JSP, garantendo anche il supporto per qualsiasi Relational Database Management System (RDBMS).

L'architettura di Spring Boot segue il paradigma a livelli, con quattro livelli distinti:

- Il controller layer, che gestisce le richieste HTTP inviate dal client.
- Il business layer, contenente la logica di business dell'applicazione.
- Il repository layer, che implementa la logica di memorizzazione e manipola gli oggetti del dominio di business nelle tabelle del database.
- Il database layer, responsabile delle operazioni effettive sulle tabelle, come le query.

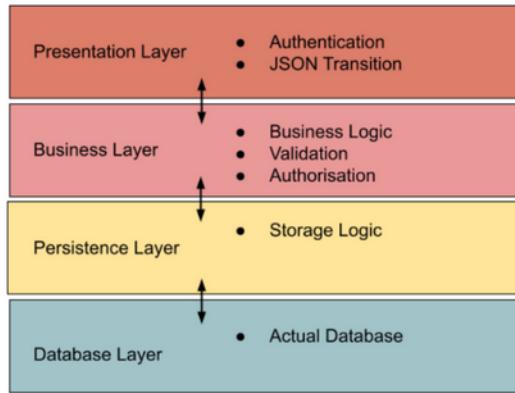


Figure 5.1: Schema a livelli Architettura Spring Boot

Spring Security

Il progetto si avvale delle potenzialità di Vaadin e Spring Security per instaurare un sistema solido di autenticazione e autorizzazione. La configurazione di base di Spring Security è stata mantenuta, includendo filtri di sicurezza essenziali e impostazioni predefinite, fornendo una base robusta in linea con le best practice di sicurezza. Un aspetto fondamentale della personalizzazione risiede nelle implementazioni custom di UserDetails e UserDetailsManager. L'interfaccia UserDetails è stata adattata per includere le informazioni specifiche dell'utente utili all'applicazione, mentre UserDetailsManager è stato personalizzato per aggiungere la persistenza alla gestione degli utenti.

Inoltre, la versione di default di Spring Security gestisce in modo trasparente il meccanismo di sessione. Questo meccanismo consente di tenere traccia delle sessioni degli utenti, garantendo un controllo sicuro dell’accesso e delle attività all’interno dell’applicazione. Le autorizzazioni sono gestite in modo dichiarativo attraverso Spring Security,

consentendo l'applicazione di annotazioni a livello di metodo o classe per definire chi può accedere a determinate risorse o eseguire azioni specifiche.

L'integrazione di Vaadin con Spring Security garantisce un flusso di lavoro di autenticazione e autorizzazione senza soluzione di continuità, e l'interfaccia utente di Vaadin è stata personalizzata per gestire scenari diversificati di autenticazione, garantendo un'esperienza utente coerente e sicura. L'accesso alle viste e alle risorse UI è gestito in modo coeso attraverso le regole di sicurezza di Spring, garantendo che solo gli utenti autorizzati possano interagire con parti specifiche dell'applicazione.

Nel contesto di StudyApp, sono state adottate alcune funzionalità di Spring Security per garantire la sicurezza dell'applicazione. L'interfaccia PasswordEncoder è utilizzata per eseguire una trasformazione unidirezionale delle password, mentre l'autenticazione degli utenti avviene attraverso Security Filter Chain e Authentication Manager. L'uso di BCryptPasswordEncoder garantisce l'hash sicuro delle password.

5.3 Package Diagram

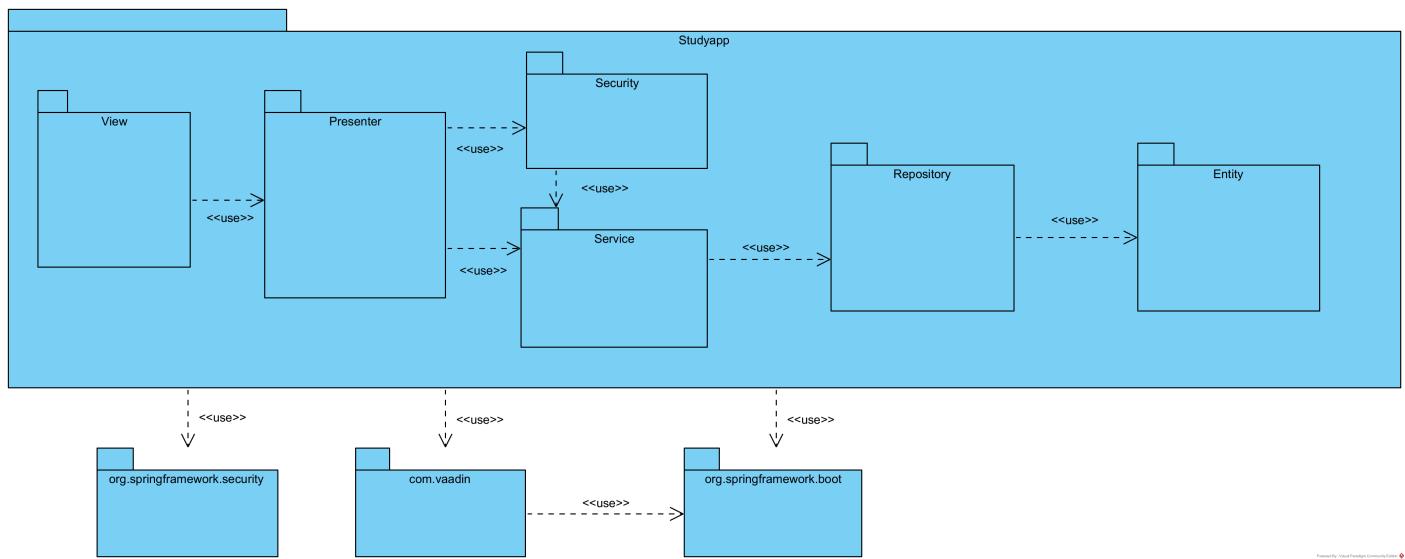


Figure 5.2: Package Diagram

5.3.1 Entity Class Diagram

Presentato di seguito è il diagramma delle classi relativo al pacchetto "Entity", risultante dalla successiva elaborazione del System Domain Model.

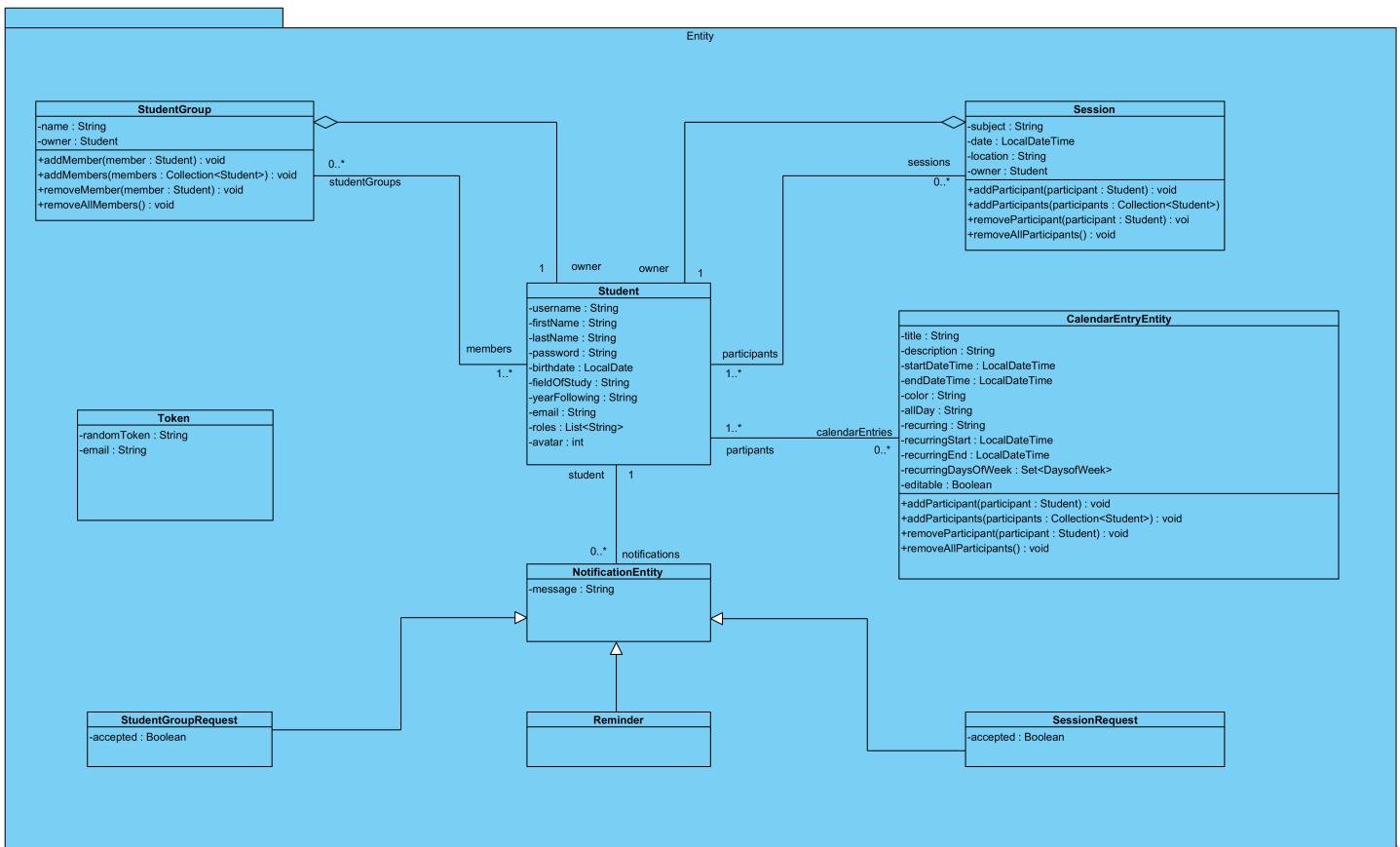


Figure 5.3: Entity Class Diagram

5.3.2 Repository Class Diagram

Viene presentato di seguito il diagramma delle classi relativo al pacchetto "Repository", il quale ospita le interfacce JPA destinate alla gestione delle comunicazioni con il database.

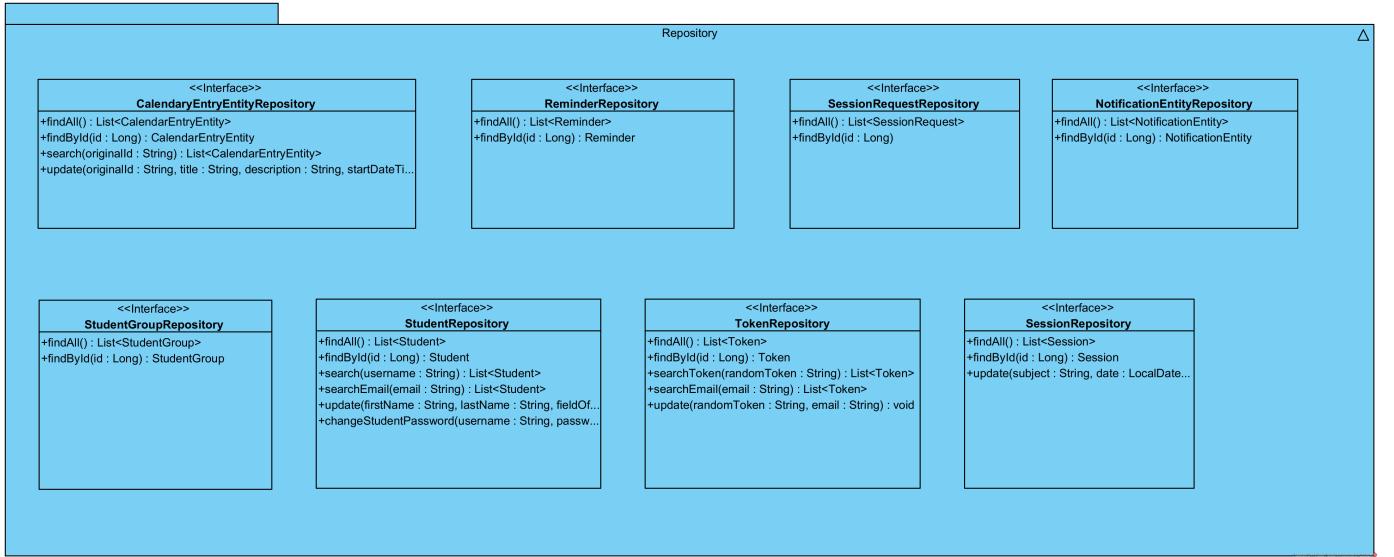


Figure 5.4: Repository Class Diagram

5.3.3 Service Class Diagram

Viene qui illustrato il diagramma delle classi relativo al pacchetto "Service", il quale assume l'incarico della logica di business all'interno dell'applicazione web.

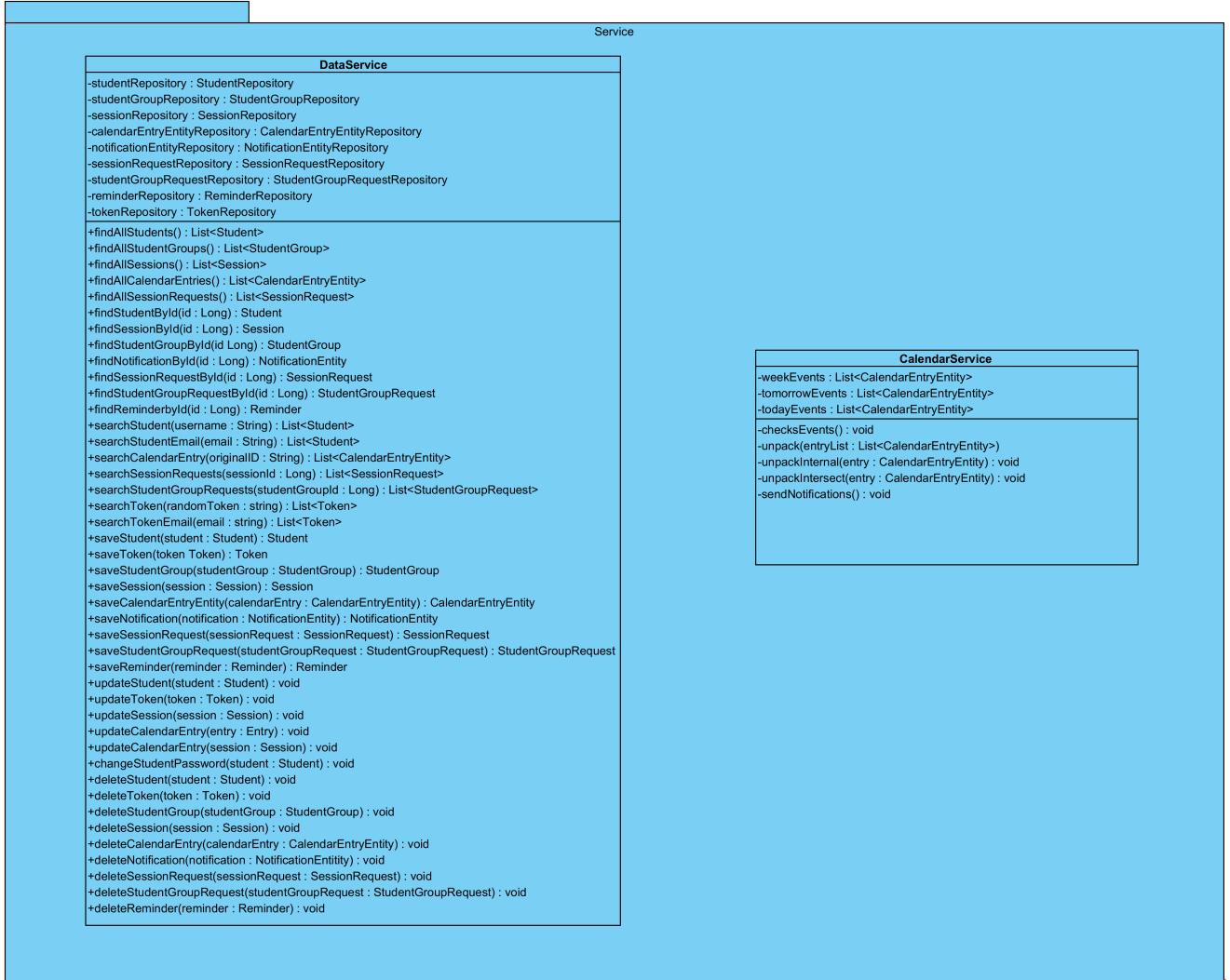


Figure 5.5: Service Class Diagram

5.3.4 Security Class Diagram

Ecco il diagramma delle classi relativo al pacchetto "Security", focalizzato sulla gestione dell'autenticazione all'interno del sistema. Questo pacchetto riveste un ruolo cruciale nell'implementazione delle misure di sicurezza, garantendo un adeguato controllo e verifica dell'identità degli utenti.

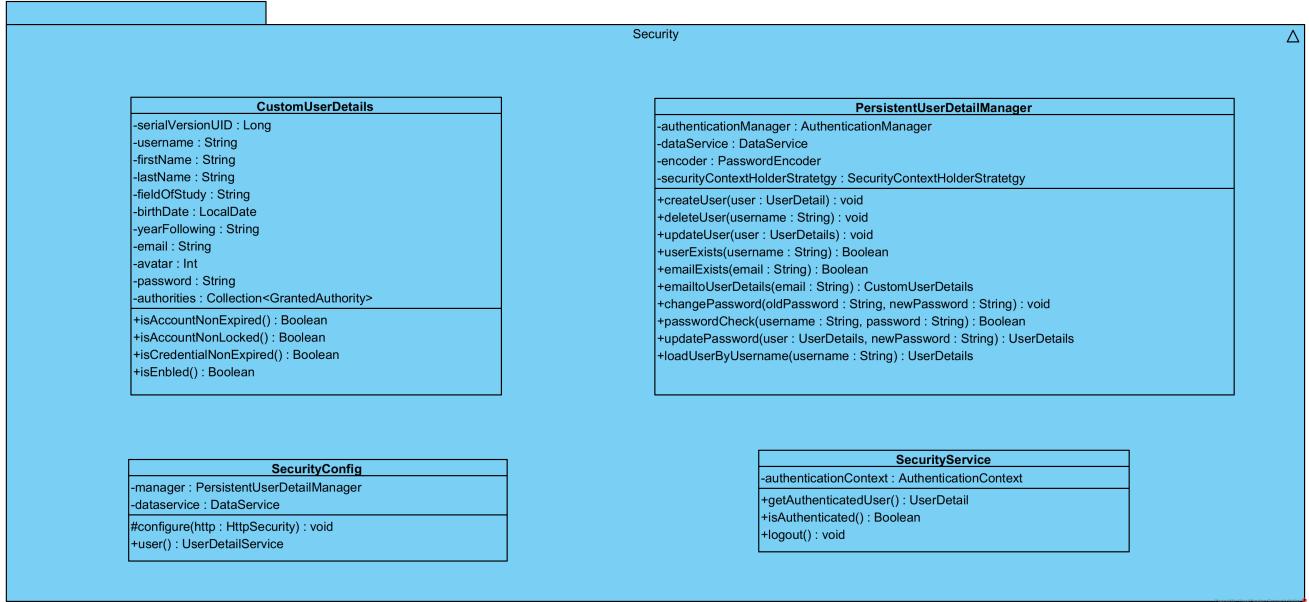


Figure 5.6: Security Class Diagram

5.3.5 Presenter Class Diagram

Di seguito viene presentato il diagramma delle classi relativo al pacchetto "Presenter", il quale svolge un ruolo fondamentale nella struttura del pattern Model-View-Presenter (MVP). Questo pacchetto gestisce la logica di presentazione dell'interfaccia utente, separandola dalla logica di business.

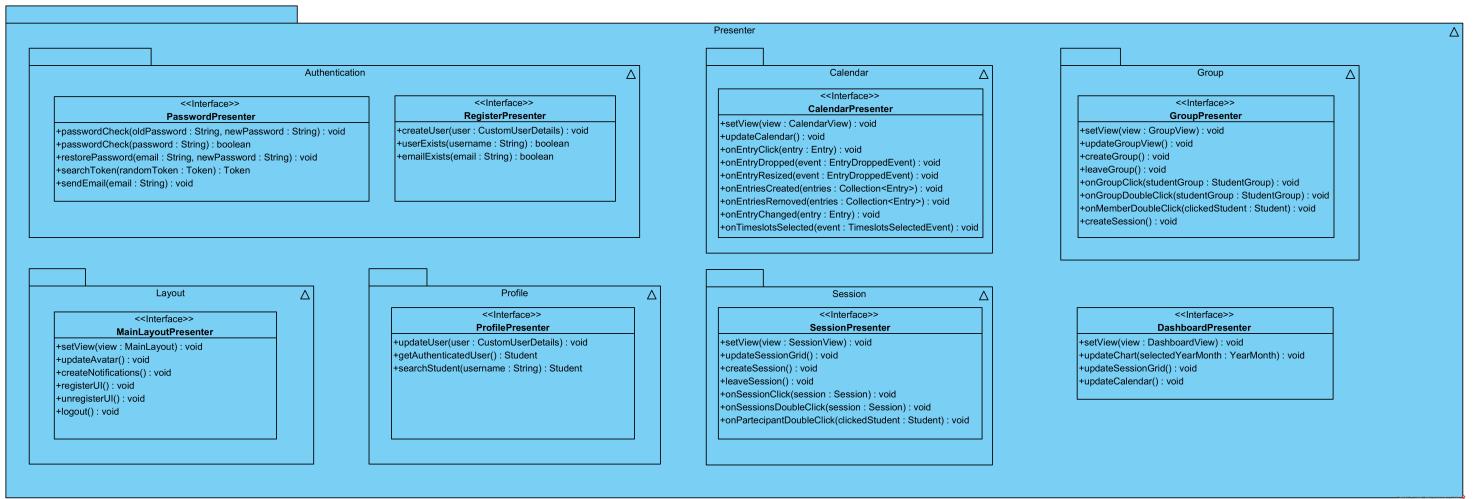


Figure 5.7: Presenter Class Diagram

5.3.6 View Class Diagram

Il diagramma delle classi per il pacchetto "View" è presentato di seguito, evidenziando la sua importanza nella struttura complessiva dell'applicazione. Questo pacchetto si occupa della gestione dell'interfaccia utente, definendo la presentazione visiva e l'interazione dell'utente con il sistema.

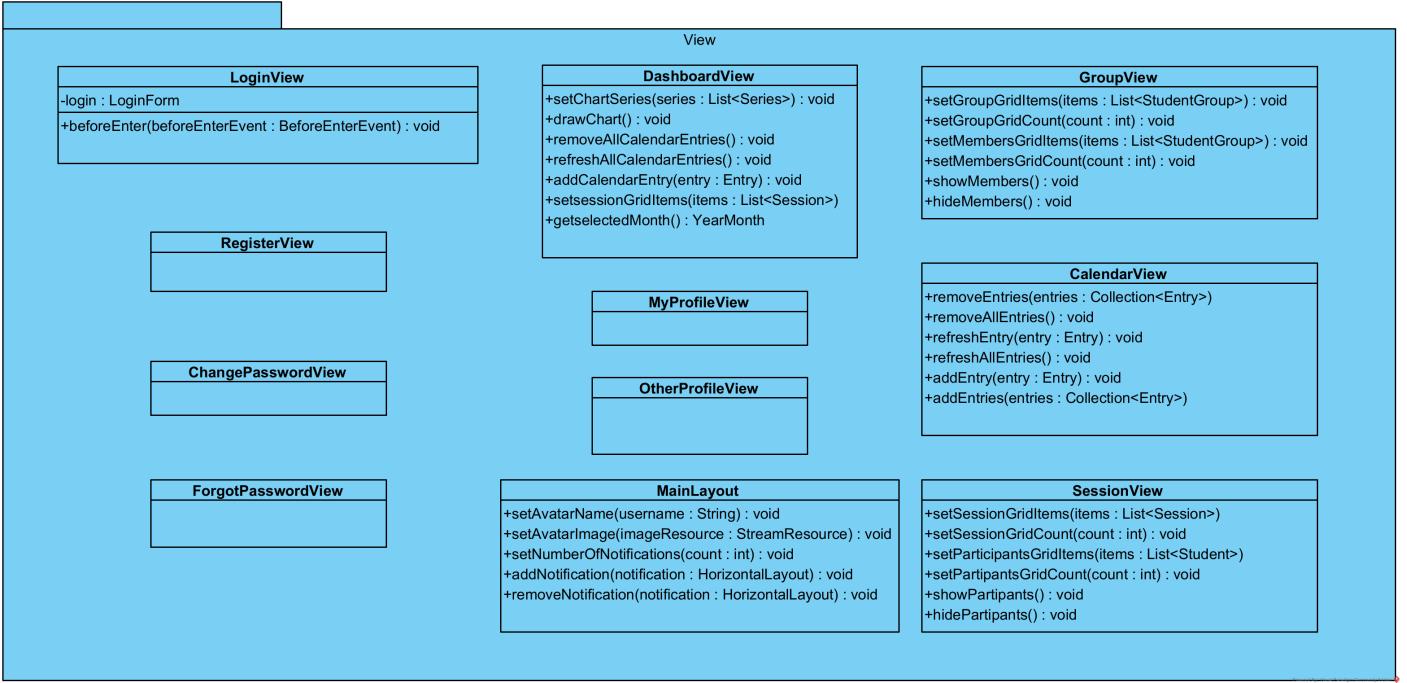


Figure 5.8: View Class Diagram

5.4 Component Diagram

Di seguito è illustrato il diagramma dei componenti, in cui il client e il server interagiscono attraverso il protocollo HTTP, mentre il server comunica con il database tramite il protocollo JDBC. È importante notare che esiste una relazione N a 1 tra i moduli del sistema e i componenti. Questo significa che i file Java associati alle classi che costituiscono il server si consolidano durante l'esecuzione in un unico componente denominato WebApplicationServer. Allo stesso modo, i file HTML, risorse e Javascript (che compongono la logica di presentazione del client) si combinano durante l'esecuzione in un unico com-

ponente denominato WebApplicationClient.



Figure 5.9: Component Diagram

5.5 API

Di seguito si è deciso di documentare tutti i metodi facenti parte delle API. Per ognuno di essi, viene riportato il nome, i parametri di input, output ed il comportamento atteso.

5.5.1 Session

setView

- Input: SessionView view
- Output: -
- Comportamento: Imposta l'istanza di SessionView associata a questo presenter. Questa istanza rappresenta l'interfaccia utente che verrà manipolata e aggiornata da questo presenter.

updateSessionGrid

- Input: -
- Output: -
- Comportamento: Recupera le sessioni dello studente autenticato dal servizio dati. Ordina le sessioni in base alla data e le aggiorna nella SessionView, riflettendo così eventuali modifiche nell’interfaccia utente.

createSession

- Input: -
- Output: -
- Comportamento: Apre una finestra di dialog per consentire all’utente di creare una nuova sessione di studio.

leaveSession

- Input: Session session
- Output: -
- Comportamento: Rimuove lo studente autenticato dai partecipanti della sessione corrente. Se lo studente è il proprietario della sessione, imposta il nuovo proprietario come il secondo partecipante (se presente), altrimenti rimuove la sessione ed infine aggiorna la vista e la griglia dei partecipanti.

onSessionClick

- Input: Session session
- Output: -
- Comportamento: Quando una sessione viene cliccata, recupera la lista dei partecipanti e li mostra nella griglia dei partecipanti nella SessionView.

onSessionDoubleClick

- Input: Session session
- Output: -
- Comportamento: Se la sessione è di proprietà dell'utente autenticato, apre una finestra di dialog che consente all'utente di modificare o rimuovere la sessione.

onParticipantDoubleClick

- Input: Student clickedStudent
- Output: -
- Comportamento: Se l'utente clicca su un partecipante, naviga alla pagina del profilo dell'utente cliccato.

5.5.2 Profile

updateUser

- Input: CustomUserDetails user
- Output: -
- Comportamento: Utilizza il gestore di sicurezza per aggiornare le informazioni dell’utente. Questo metodo può essere chiamato quando sono necessarie modifiche alle informazioni dell’utente autenticato.

getAuthenticatedUser

- Input: -
- Output: Student
- Comportamento: Recupera l’utente attualmente autenticato attraverso il servizio di sicurezza e successivamente utilizza il servizio dati per cercare e restituire l’entità Student associata all’utente autenticato.

searchStudent

- Input: String username
- Output: Student

- Comportamento: Utilizza il servizio dati per cercare uno studente in base al nome utente fornito. Restituisce la prima corrispondenza trovata nella lista di studenti. Se non viene trovato alcuno studente, restituisce null.

5.5.3 Layout

setView

- Input: MainLayout view
- Output: -
- Comportamento: Imposta l'istanza di MainLayout associata a questo presenter. Questa istanza rappresenta l'interfaccia utente che verrà manipolata e aggiornata da questo presenter.

updateAvatar

- Input: -
- Output: -
- Comportamento: Aggiorna l'immagine dell'avatar nella vista con il nome utente e l'immagine associate all'utente autenticato.

createNotifications

- Input: -
- Output: -

- Comportamento: Recupera le notifiche dell’utente autenticato e crea layout per ciascuna notifica. Aggiunge pulsanti di chiusura e gestisce la logica per rimuovere le notifiche dalla view quando chiuse ed inoltre aggiorna il conteggio delle notifiche nel counter di notifica posto nella navBar.

registerUI

- Input: -
- Output: -
- Comportamento: Registra l’interfaccia utente corrente associata all’utente autenticato. Questo è utile per inviare notifiche o aggiornamenti all’interfaccia utente.

unregisterUI

- Input: -
- Output: -
- Comportamento: Annulla la registrazione dell’interfaccia utente corrente associata all’utente autenticato. Viene chiamato quando l’utente esegue il logout o chiude la finestra del browser.

logout

- Input: -

- Output: -
- Comportamento: Esegue il logout dell'utente, invalidando la sessione corrente.

5.5.4 Group

setView

- Input: GroupView view
- Output: -
- Comportamento: Imposta l'istanza di GroupView associata a questo presenter. Questa istanza rappresenta l'interfaccia utente che verrà manipolata e aggiornata da questo presenter.

updateGroupGrid

- Input: -
- Output: -
- Comportamento: Aggiorna la griglia dei gruppi nella view.

createGroup

- Input: -
- Output: -

- Comportamento: Apre un dialog per creare un nuovo gruppo ed imposta i consumer per gestire gli eventi di salvataggio del gruppo creato.

leaveGroup

- Input: -
- Output: -
- Comportamento: Rimuove lo studente autenticato dai partecipanti della sessione corrente. Se lo studente è il proprietario della sessione, imposta il nuovo proprietario come il secondo partecipante (se presente), altrimenti rimuove la sessione ed infine aggiorna la vista e la griglia dei membri.

onGroupClick

- Input: StudentGroup studentGroup
- Output: -
- Comportamento: Mostra la sezione dei membri nella view quando un gruppo viene cliccato ed aggiorna la griglia dei membri con i membri del gruppo selezionato.

onGroupDoubleClick

- Input: StudentGroup studentGroup

- Output: -
- Comportamento: Apre un dialog quando un gruppo viene cliccato due volte e se l'utente autenticato è il proprietario del gruppo, imposta i consumer per gestire gli eventi di aggiornamento e rimozione del gruppo.

onMemberDoubleClick

- Input: Student clickedStudent
- Output: -
- Comportamento: Se l'utente autenticato fa doppio clic su se stesso, naviga alla pagina del profilo personale; altrimenti, naviga alla pagina del profilo dell'utente cliccato.

5.5.5 Calendar

setView

- Input: CalendarView view
- Output: -
- Comportamento: Imposta l'istanza di CalendarView associata a questo presenter. Questa istanza rappresenta l'interfaccia utente che verrà manipolata e aggiornata da questo presenter.

updateCalendar

- Input: -
- Output: -
- Comportamento: Aggiorna il calendario nella view aggiungendo e visualizzando le nuove voci del calendario.

onEntryClick

- Input: Entry entry
- Output: -
- Comportamento: Gestisce il click su una voce del calendario. Se la voce è modificabile, apre una finestra di dialog per la modifica o la cancellazione della voce.

onEntryDropped

- Input: EntryDroppedEvent event
- Output: -
- Comportamento: Gestisce l'evento di trascinamento di una voce nel calendario.

onEntryResized

- Input: EntryResizedEvent event
- Output: -

- Comportamento: Gestisce l'evento di ridimensionamento di una voce nel calendario.

onEntriesCreated

- Input: Collection<Entry> entries
- Output: -
- Comportamento: Gestisce la creazione di nuove voci nel calendario. Aggiunge le nuove voci alla view e le salva nel servizio dei dati come voci di calendario.

onEntriesRemoved

- Input: Collection<Entry> entries
- Output: -
- Comportamento: Gestisce la rimozione di voci dal calendario. Rimuove le voci dalla view e dal servizio dei dati.

onEntryChanged

- Input: Entry entry
- Output: -
- Comportamento: Gestisce le modifiche a una voce del calendario. Aggiorna la view con le modifiche e salva le modifiche nel servizio dei dati.

onTimeslotsSelected

- Input: TimeslotsSelected
- Output: -
- Comportamento: Gestisce la selezione di uno slot temporale nel calendario. Apre una finestra di dialog per la creazione di una nuova voce di calendario basata sugli slot temporali selezionati.

5.5.6 Authentication

Register

createUser

- Input: CustomUserDetails user
- Output: -
- Comportamento: Responsabile della creazione di un nuovo utente nel sistema, progettato per essere chiamato quando un nuovo utente si registra nel sistema.

userExists

- Input: String username
- Output: boolean

- Comportamento: Verifica se esiste già un utente con lo username fornito, utile per evitare la registrazione di utenti con lo stesso username.

emailExists

- Input: String email
- Output: boolean
- Comportamento: Verifica se esiste già un utente con l'email fornita, utile per evitare la registrazione di utenti con la stessa email.

Password

changePassword

- Input: String oldPassword, String newPassword
- Output: -
- Comportamento: Gestisce il cambio della password.

passwordCheck

- Input: String password
- Output: boolean

- Comportamento: Verifica se la password fornita è corretta per l'utente autenticato, utile per effettuare controlli sulla validità della password durante le operazioni che richiedono autenticazione.

restorePassword

- Input: String email, CustomUserDetails user, String newPassword
- Output: -
- Comportamento: Gestisce il ripristino della password in caso di smarrimento ove se almeno un token è presente, imposta l'username dell'utente associato all'email fornita, aggiorna la password dell'utente con quella fornita (cifrata) utilizzando l'Authentication Manager, e infine elimina il token associato.

searchToken

- Input: String randomToken
- Output: Token
- Comportamento: Cerca un token nel sistema basandosi sul suo valore e restituisce il primo token trovato (se presente) o null se nessun token è stato trovato.

sendEmail

- Input: String email
- Output: -
- Comportamento: Responsabile dell'invio di un'email contenente un token per il ripristino della password.

Chapter 6

Diagrammi comportamentali

In questo capitolo verranno presentati diagrammi comportamentali dettagliati del software, i quali forniranno una panoramica delle operazioni eseguite in Java nel caso di utilizzo delle varie API.

6.1 Sequence Diagram

Riportiamo in questa sezione i sequence diagram realizzati per le varie API.

6.1.1 leaveGroup

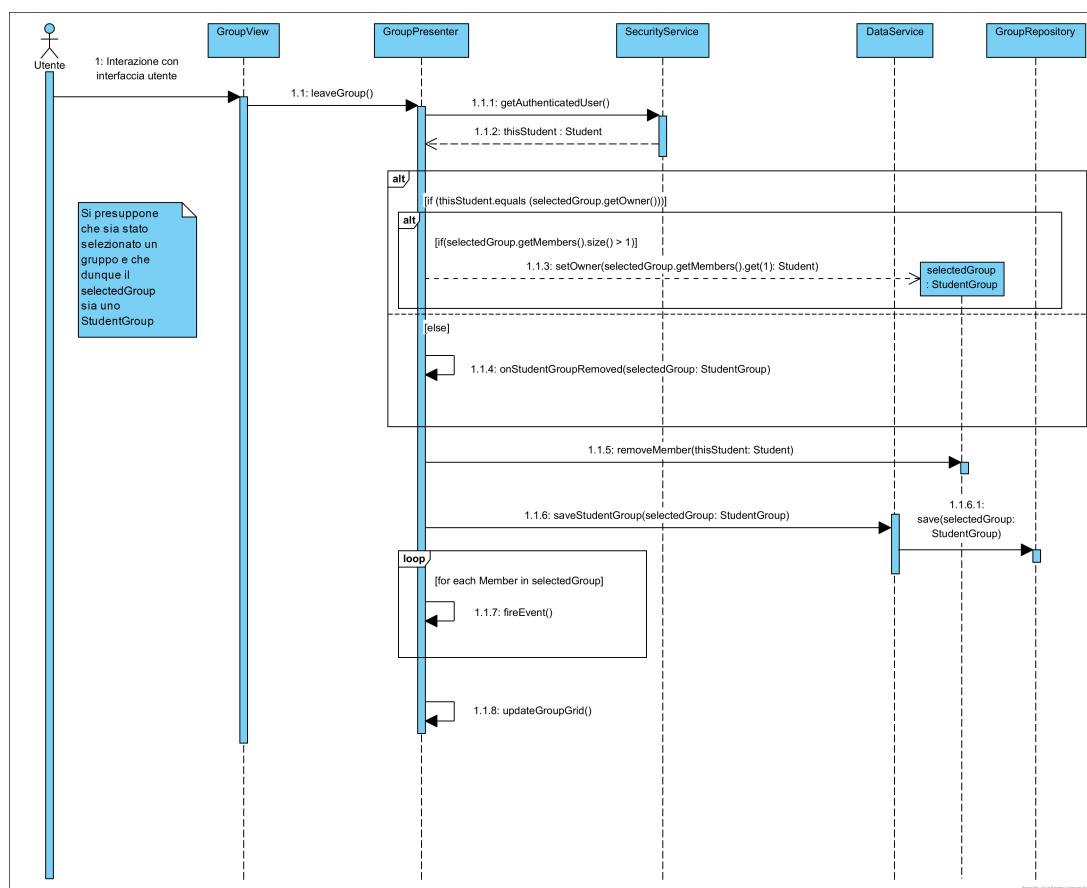


Figure 6.1: Sequence diagram della funzione `leaveGroup()`

6.1.2 leaveSession

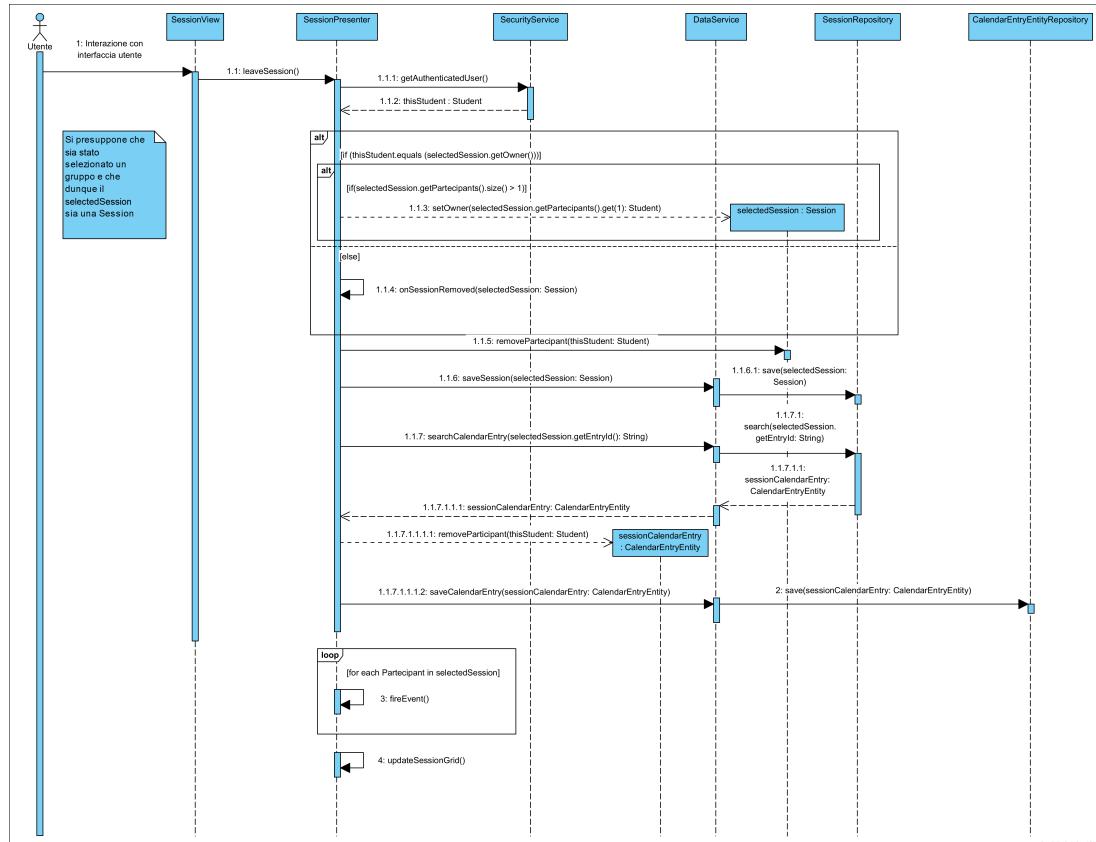


Figure 6.2: Sequence diagram della funzione `leaveSession()`

6.1.3 createGroup

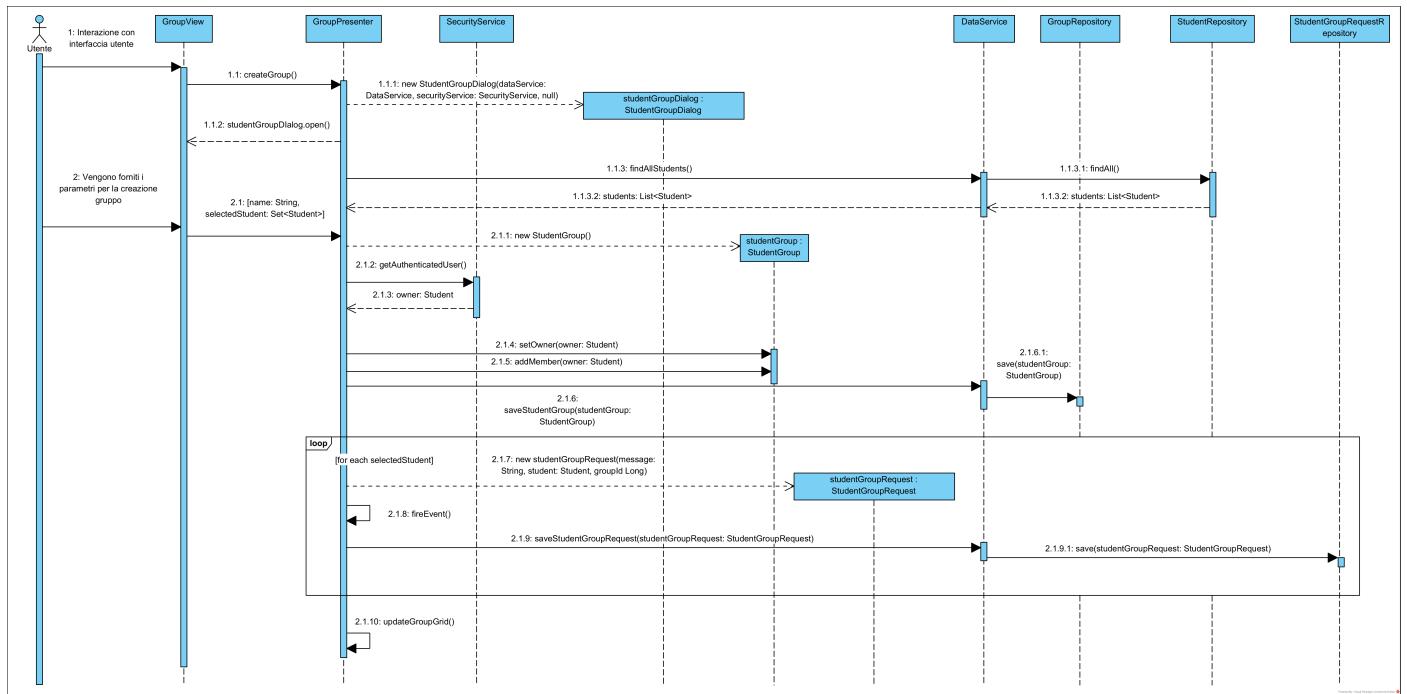


Figure 6.3: Sequence diagram della funzione createGroup()

6.1.4 createSession

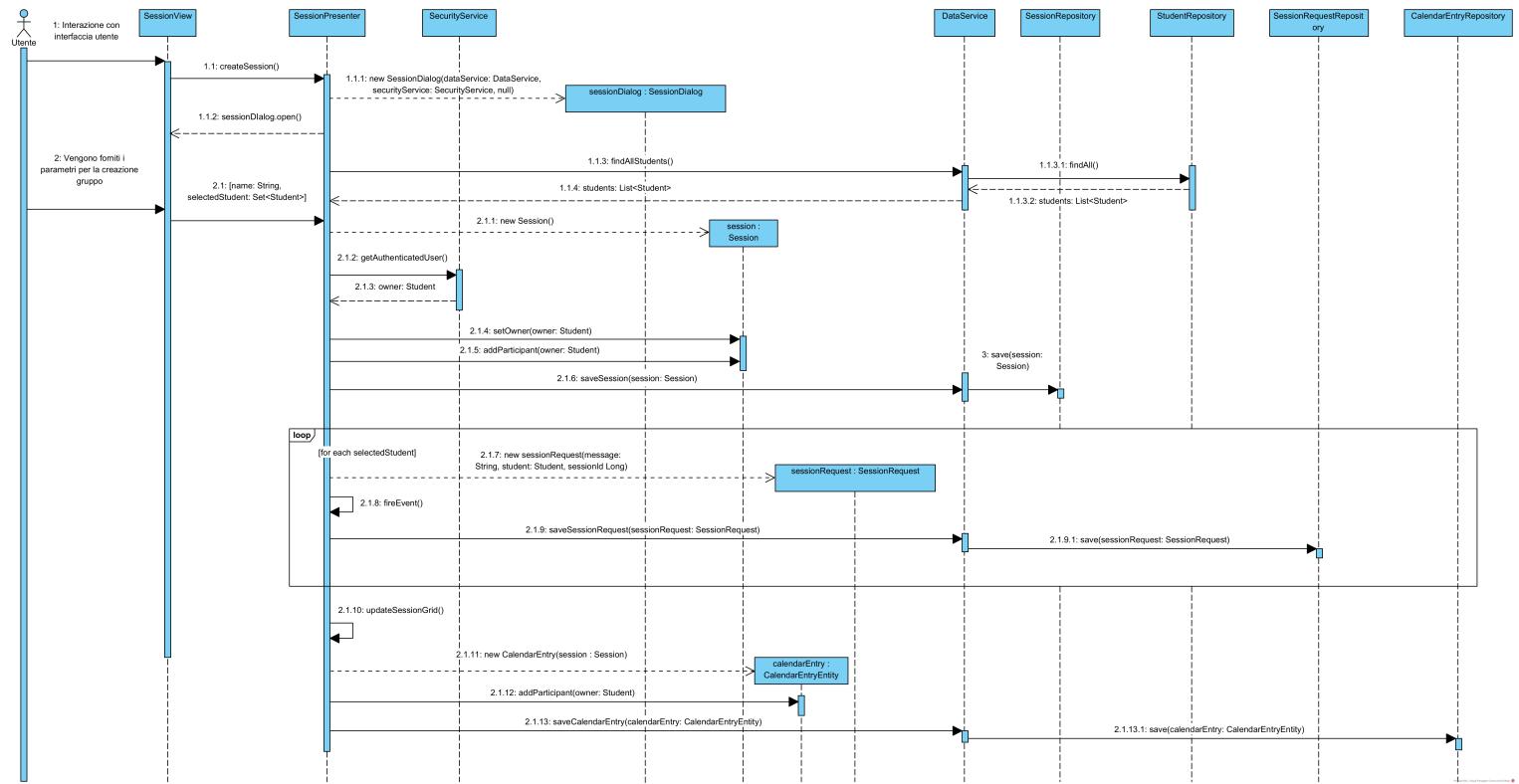


Figure 6.4: Sequence diagram della funzione createSession()

6.1.5 createNotification

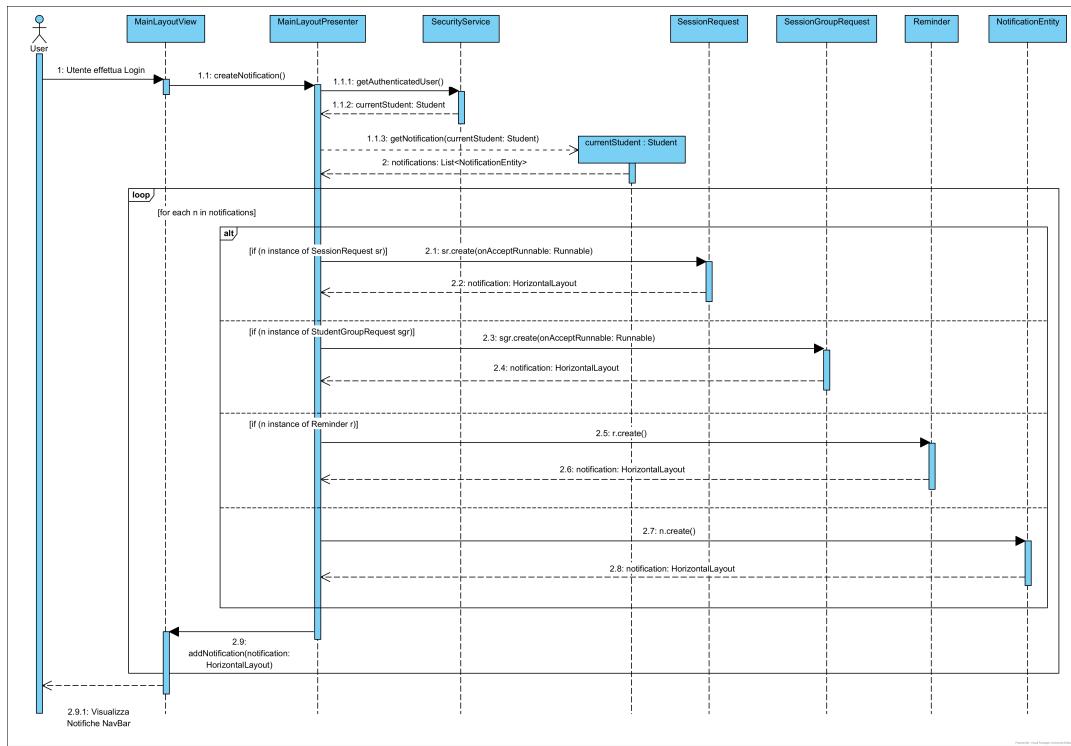


Figure 6.5: Sequence diagram della funzione createNotification()

6.1.6 onStudentGroupRemoved

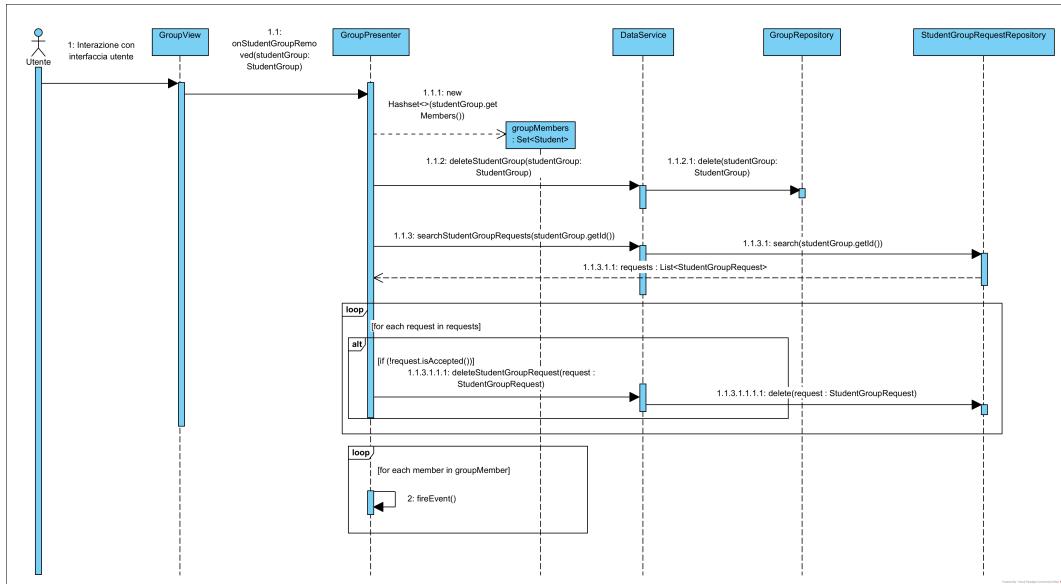


Figure 6.6: Sequence diagram della funzione `onStudentGroupRemoved()`

6.1.7 OnSessionRemoved

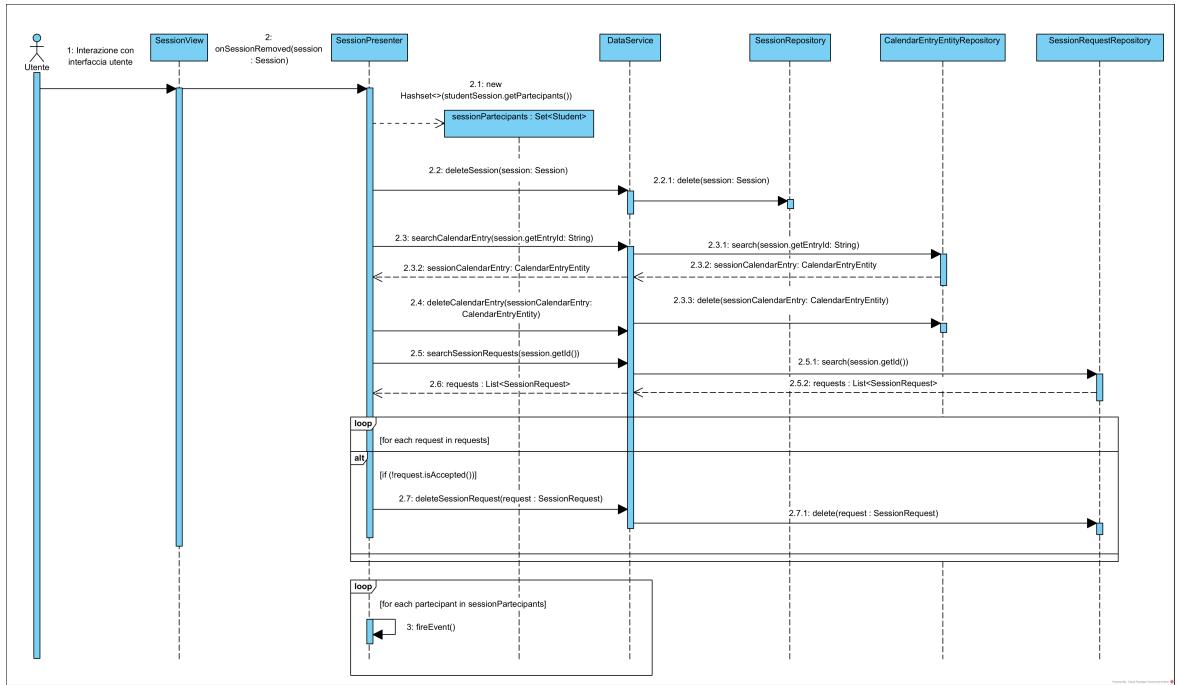


Figure 6.7: Sequence diagram della funzione `OnSessionRemoved()`

6.1.8 createUser

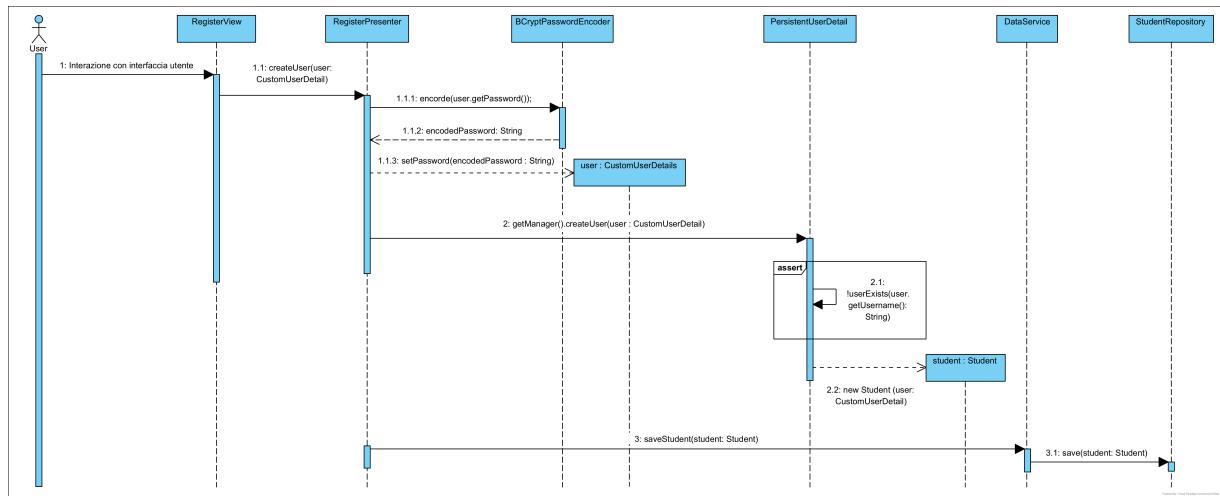


Figure 6.8: Sequence diagram della funzione createUser()

6.1.9 sendEmail

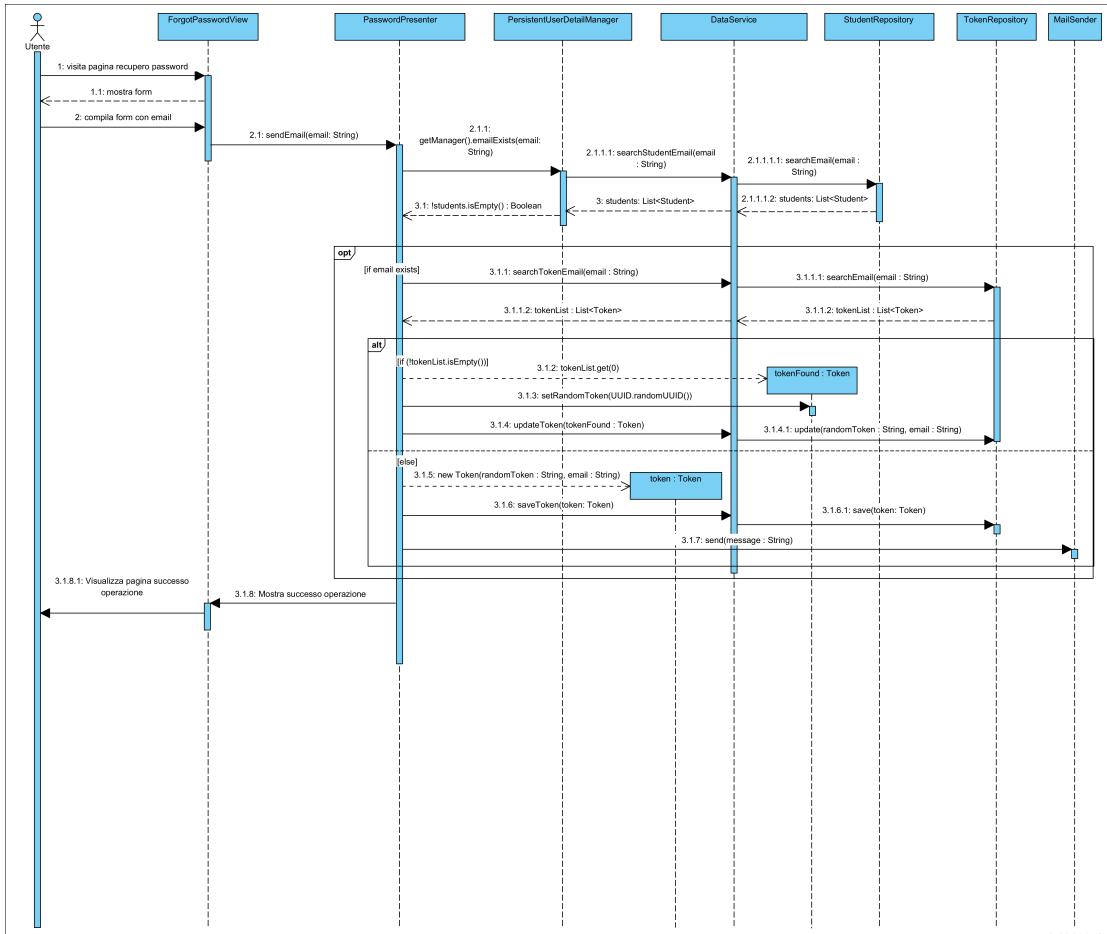


Figure 6.9: Sequence diagram della funzione `sendEmail()`

6.1.10 changePassword

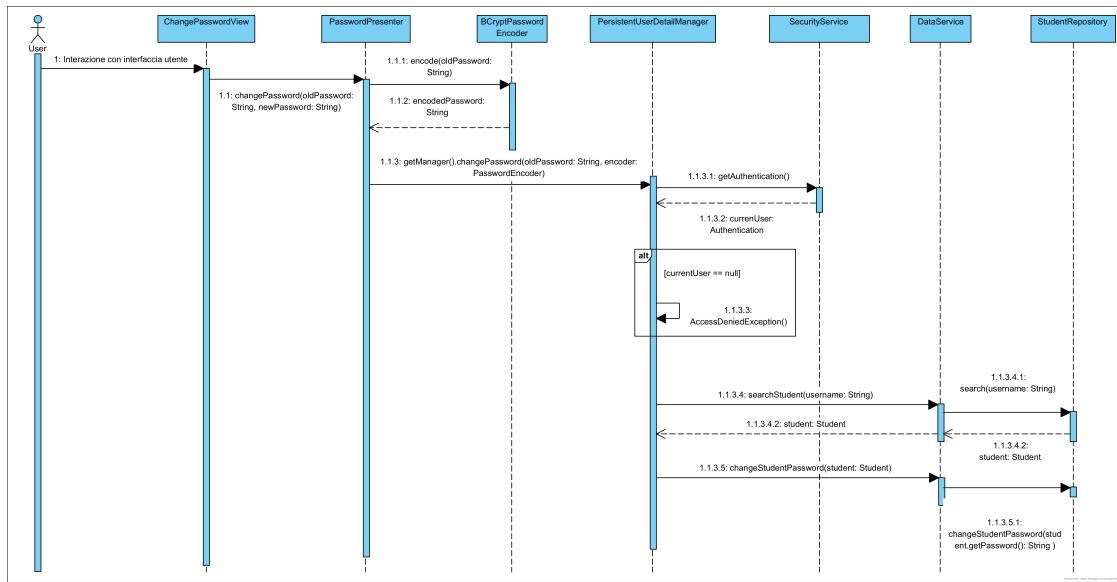


Figure 6.10: Sequence diagram della funzione `changePassword()`

6.1.11 updateCalendar

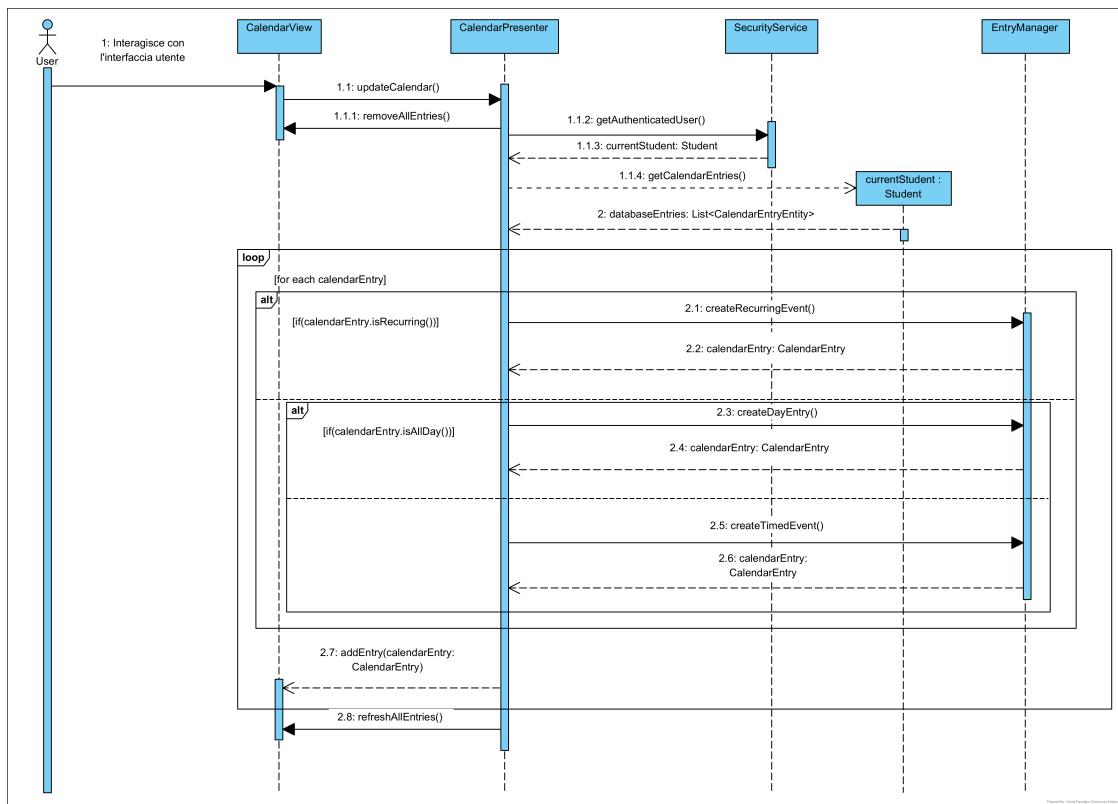


Figure 6.11: Sequence diagram della funzione `updateCalendar()`

6.1.12 onGroupClick

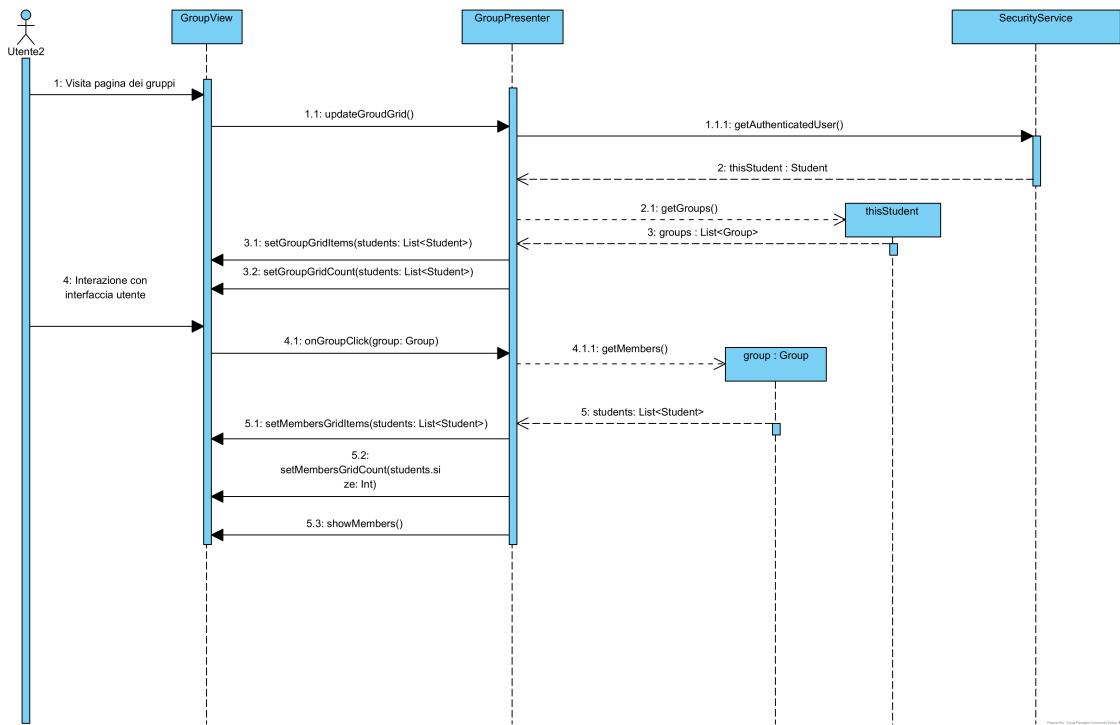


Figure 6.12: Sequence diagram della funzione OnGroupClick()

6.1.13 onSessionClick

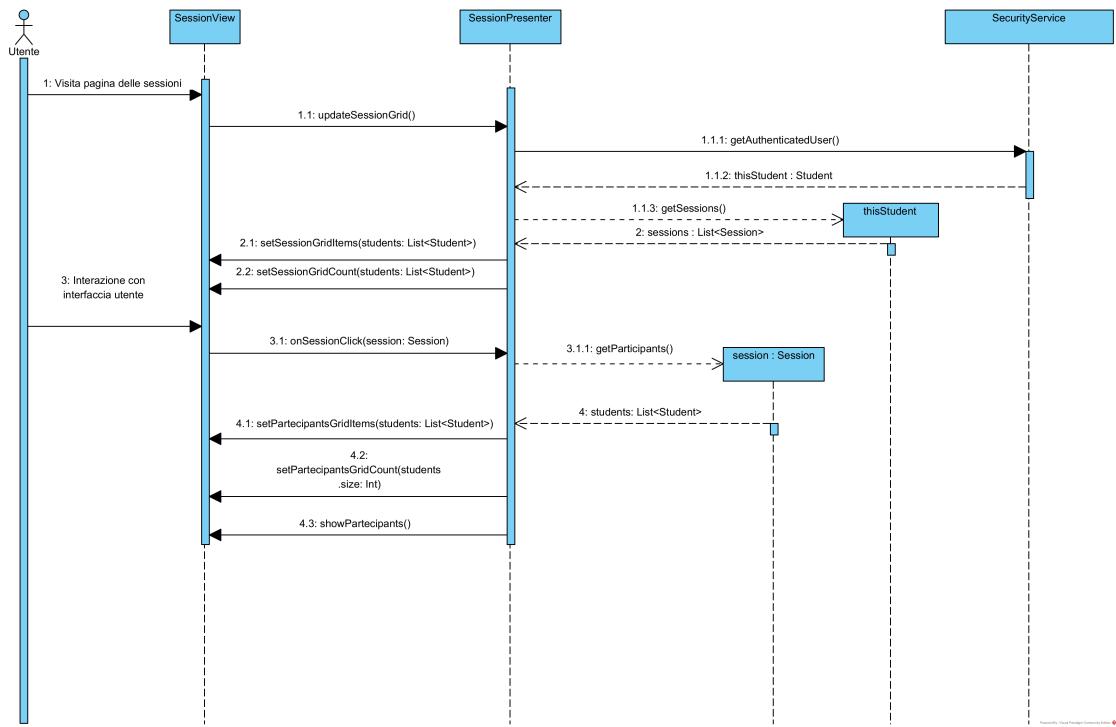


Figure 6.13: Sequence diagram della funzione `onSessionClick()`

6.2 Flowchart

In questa sezione si andrà invece a riportare i diagrammi di flusso di alcune tipiche operazioni svolte da un utente che racchiudono in esse più casi d'uso.

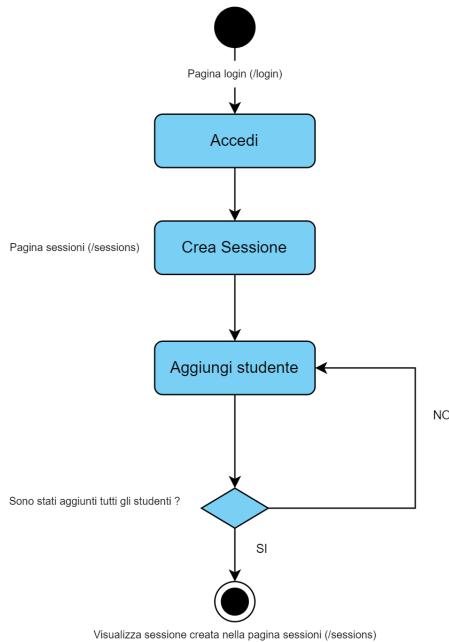


Figure 6.14: Creazione di una sessione

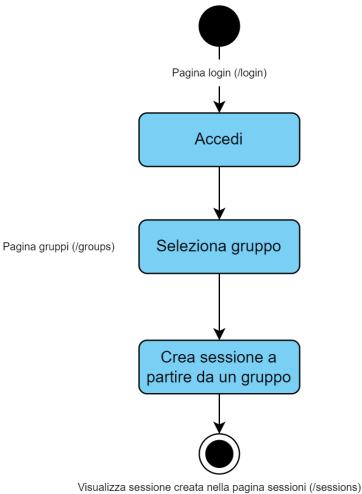


Figure 6.15: Creazione di una sessione a partire da un gruppo

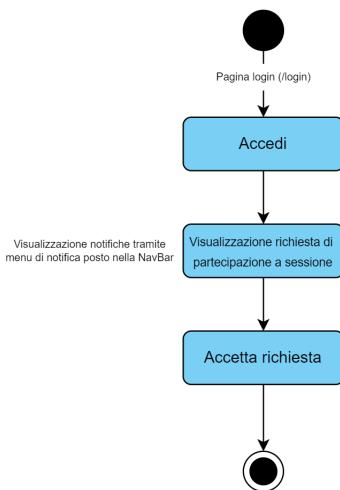


Figure 6.16: Accettazione richiesta partecipazione a sessione

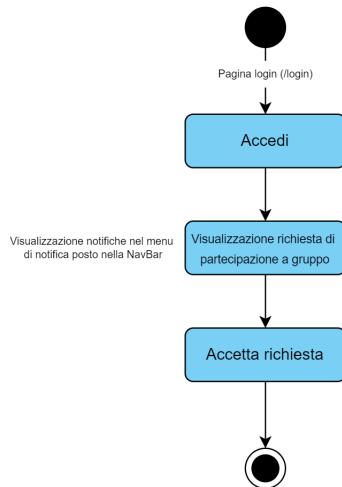


Figure 6.17: Accettazione richiesta partecipazione a gruppo

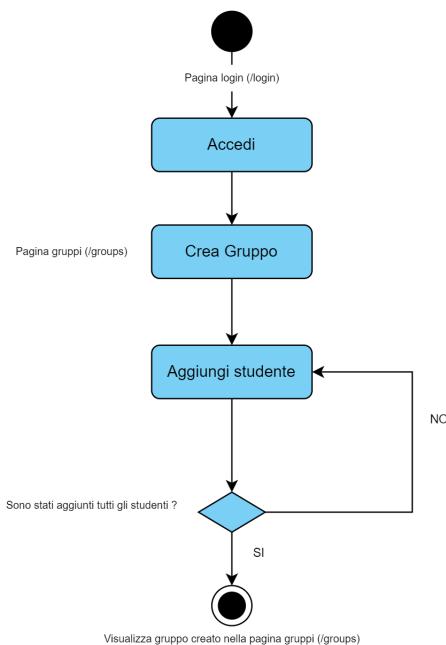


Figure 6.18: Creazione di un gruppo



Figure 6.19: Creazione evento calendario

6.3 Activity Diagram

Andiamo in questa sezione a descrivere tali operazioni più nel dettaglio con degli Activity Diagram, che riportano anche i parametri inseriti in input:

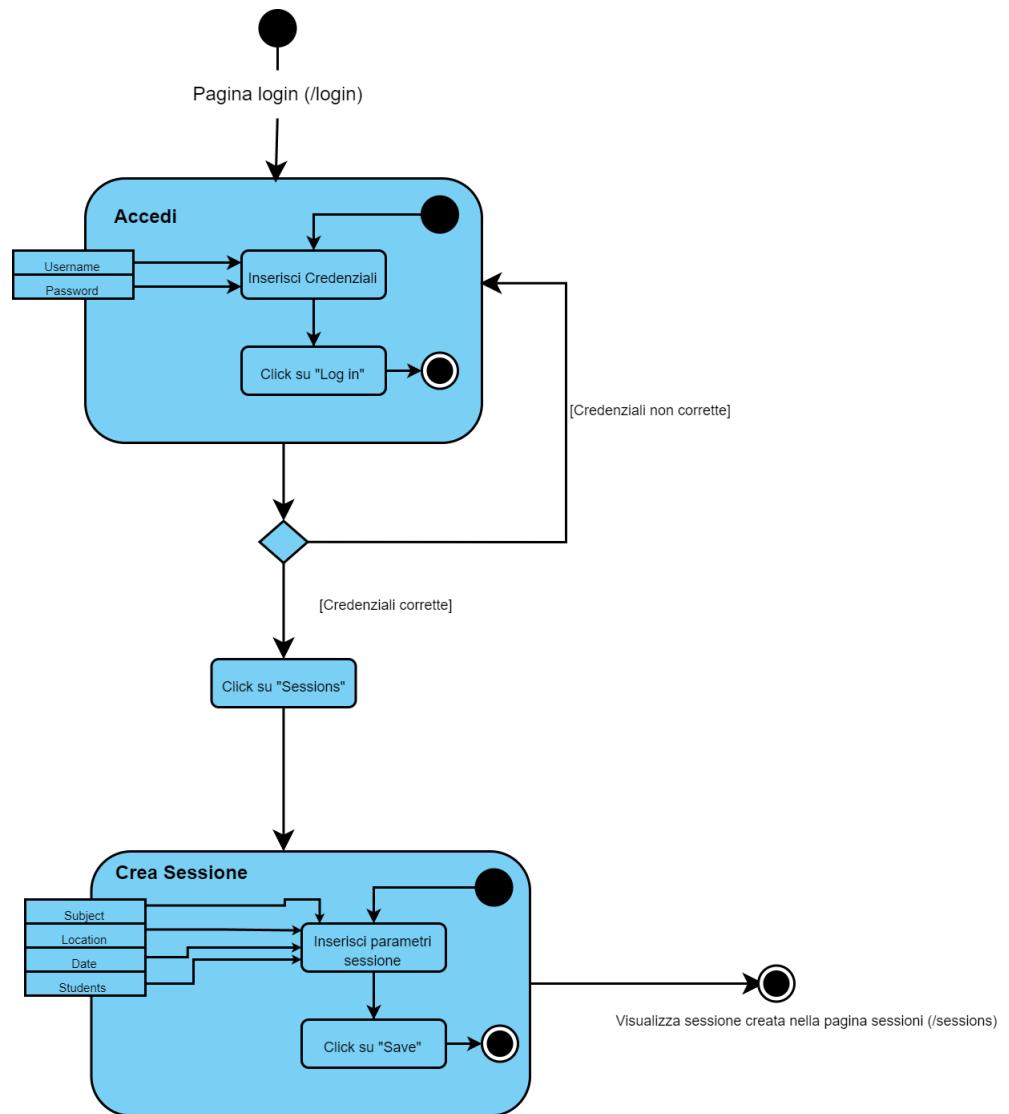


Figure 6.20: Creazione di una sessione

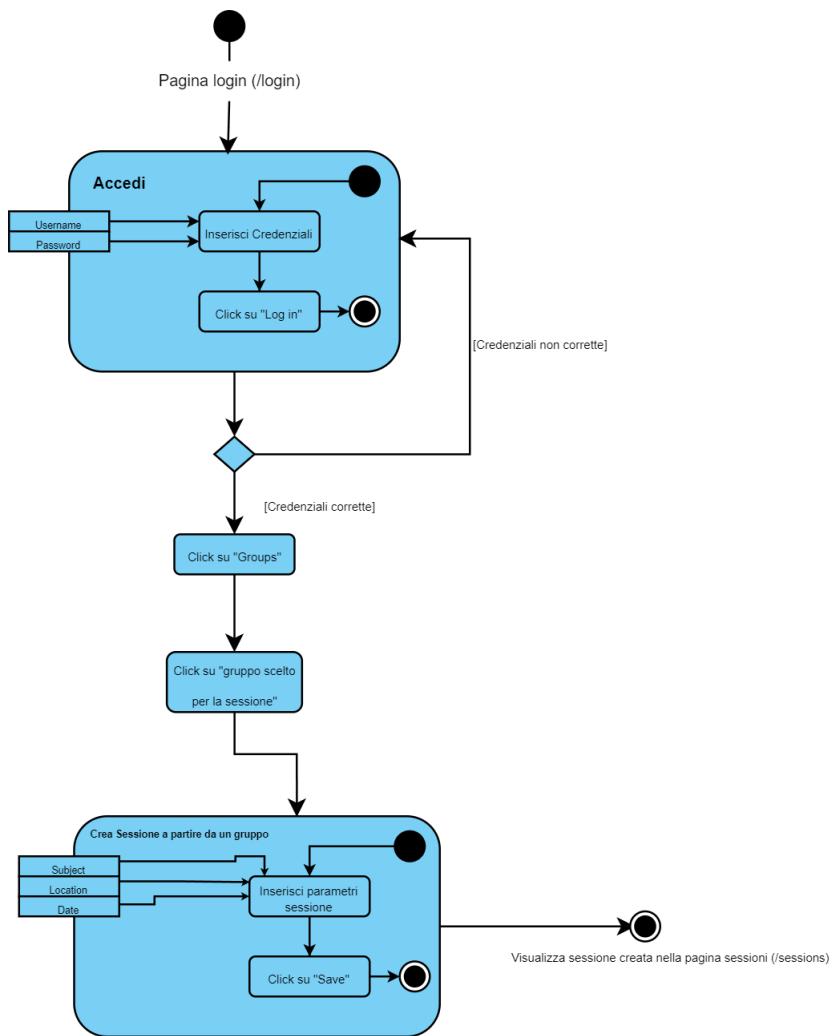


Figure 6.21: Creazione di una sessione a partire da un gruppo

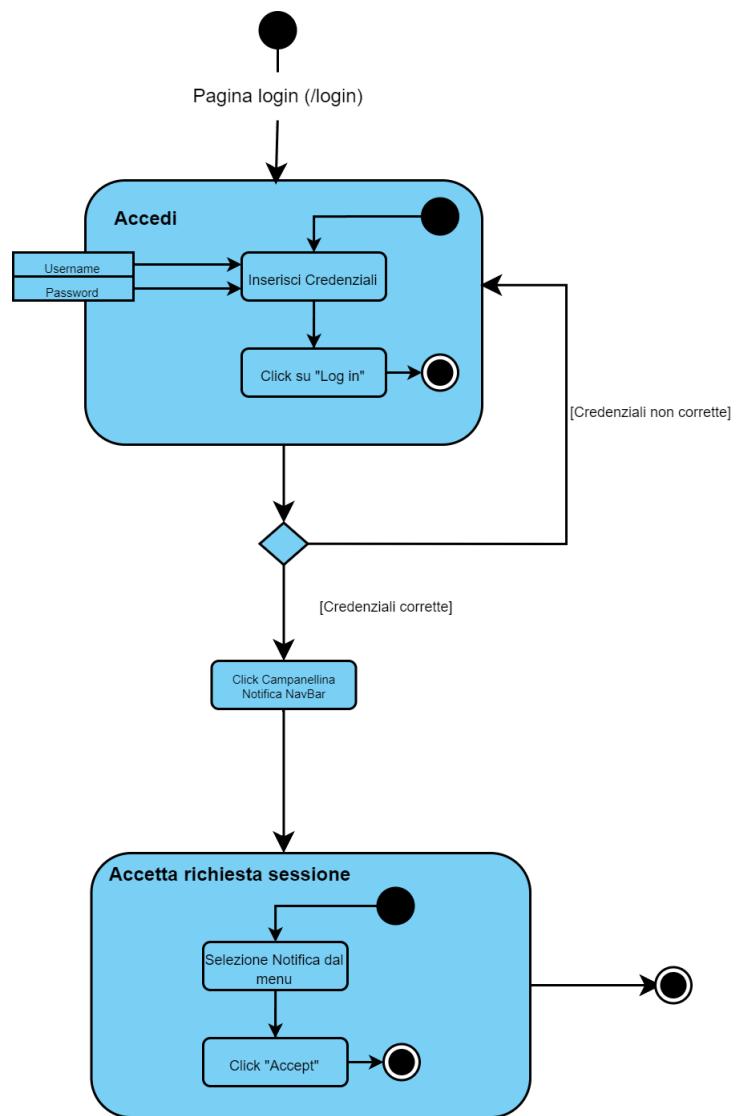


Figure 6.22: Accettazione richiesta partecipazione a sessione

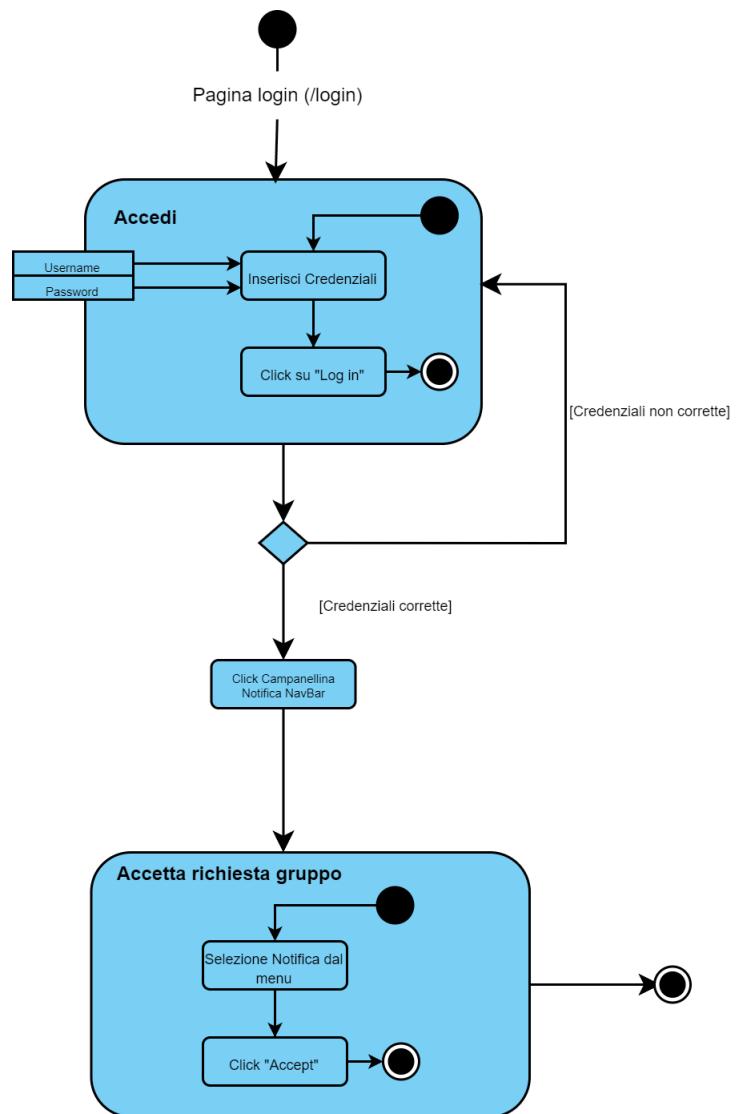


Figure 6.23: Accettazione richiesta partecipazione a gruppo

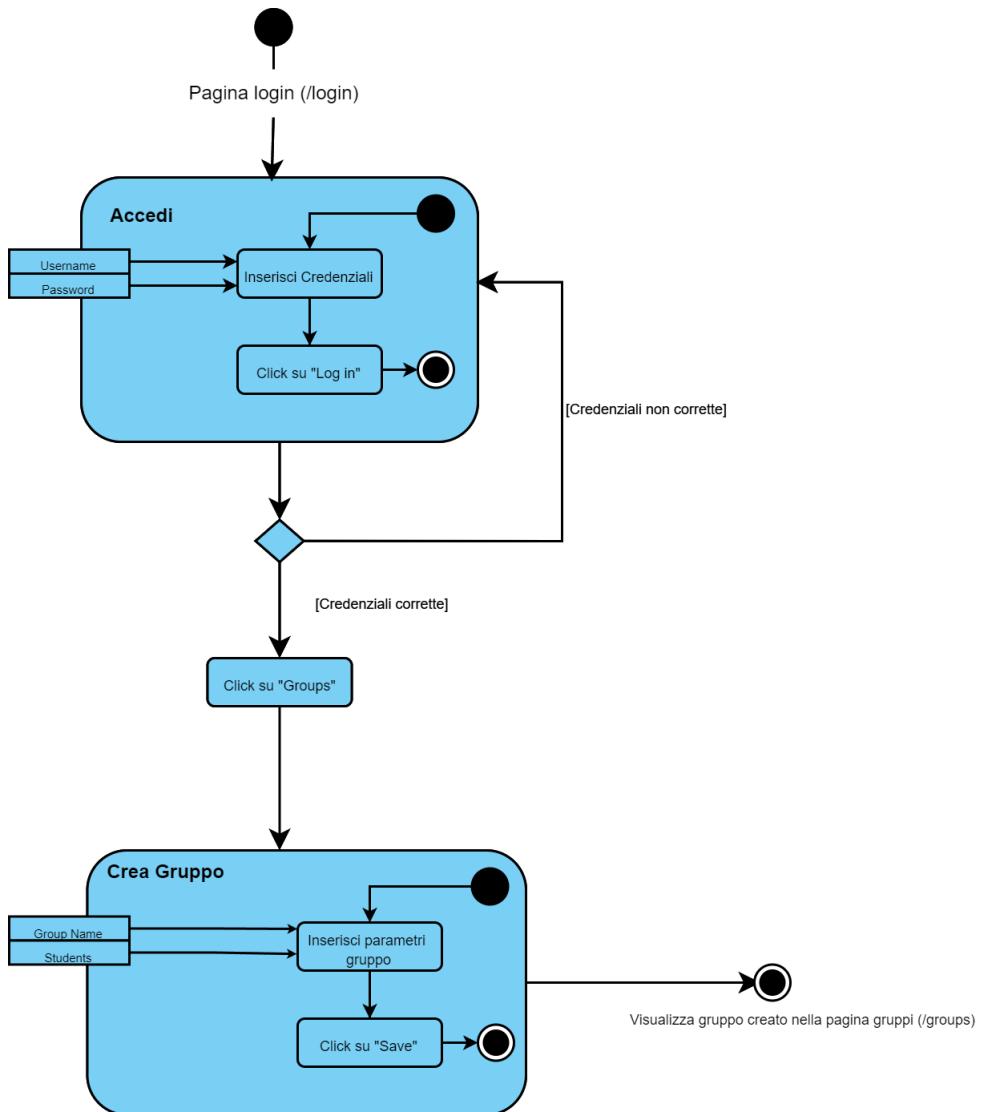


Figure 6.24: Creazione di un gruppo

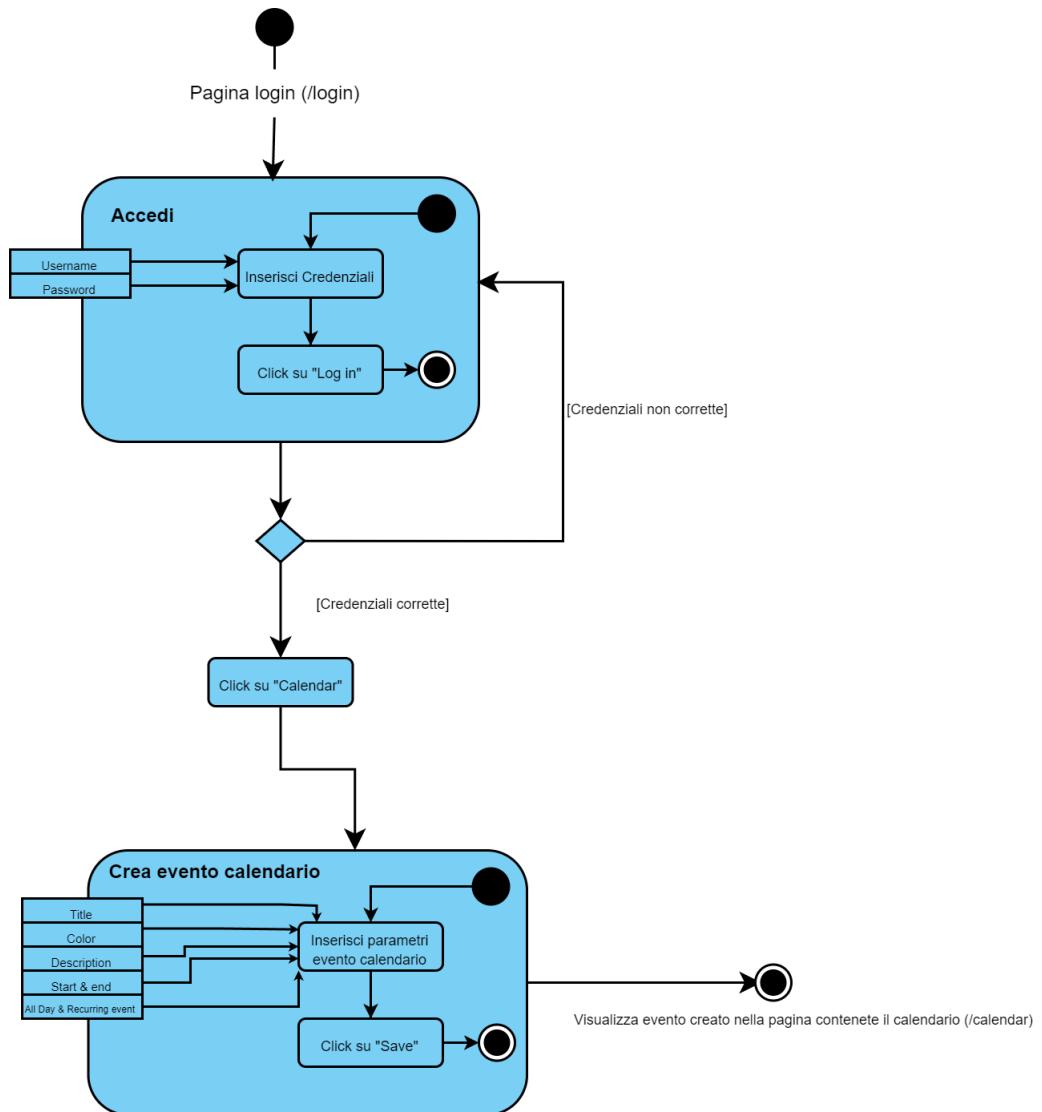


Figure 6.25: Creazione evento calendario

Chapter 7

Rilascio e Testing

La fase finale del progetto include il rilascio dell'applicazione e il completamento dei test. Tramite l'ambiente di sviluppo Eclipse è stata eseguita una *build* di produzione di Maven, che produce l'archivio *.jar*. Questo file è un archivio che racchiude il codice compilato dell'applicazione e dei framework Vaadin e Spring Boot, file HTML, CSS e JS per il client, librerie esterne in formato jar e file di configurazione.

7.1 Deployment Diagram

Di seguito si riporta il deployment diagram dell'applicazione. A causa di mancanza di risorse, il web server e il database sono eseguiti sulla stessa macchina server su sistema operativo Ubuntu 20. A fini esemplificativi, si è immaginato di eseguire database e web server su due macchine diverse. Il client è rappresentato da un web browser che

comunica tramite protocollo HTTP con il web server. Quest'ultimo è implementato dal JSP Tomcat e comunica con il database tramite protocollo JDBC. L'artefatto **studyapp-1.0.jar** svolge il ruolo di server a runtime, contenendo il necessario per avviare Vaadin e Spring Boot (e quindi il server Tomcat incluso in esso). Esso contiene anche le pagine HTML e i file Javascript utilizzati dal client e ottenuti tramite richieste HTTP. L'artefatto `postgresql.service` si manifesta come RDBMS ed è utilizzato per avviare il database PostgreSQL 16. Si noti che l'applicazione è stata dispiegata sull'URL radice:

studyapp.northeasteuropa.cloudapp.azure.com:8080/.

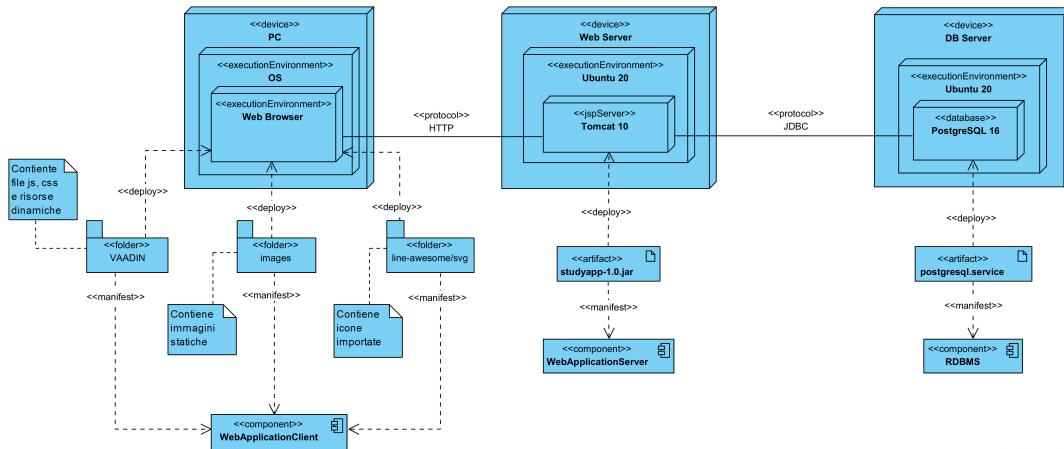


Figure 7.1: Deployment Diagram dell'applicazione

7.1.1 Install view

Si riporta qui il contenuto del file **studyapp-1.0.jar**.

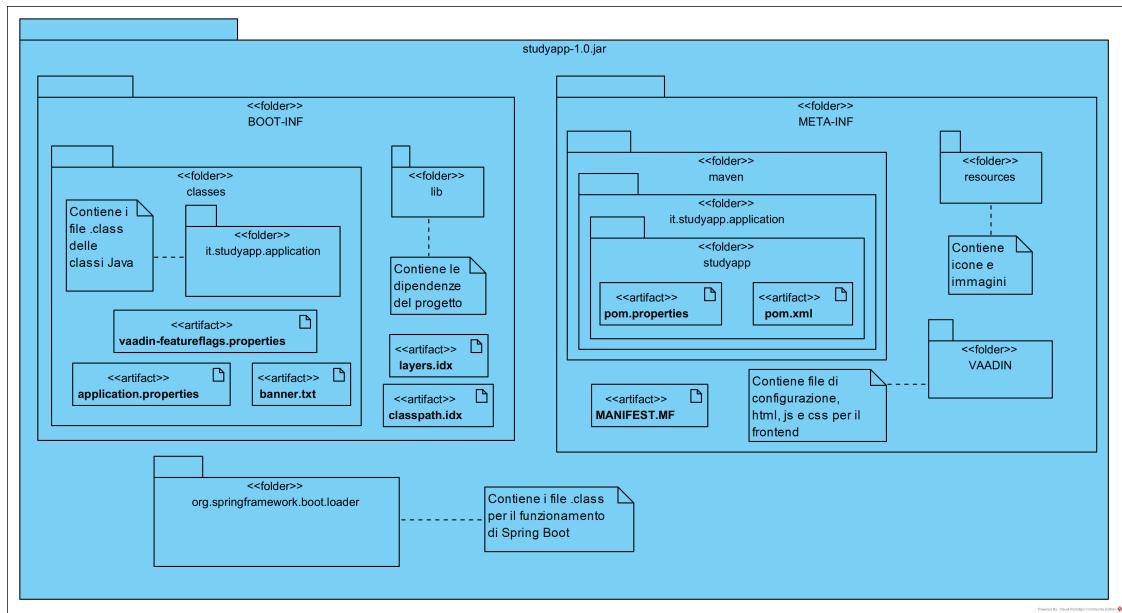


Figure 7.2: Install View

7.2 Testing d'unità

Per il testing d'unità, è stato adottato l'utilizzo di JUnit, un modulo che consente la verifica individuale dei metodi presenti in tutte le classi dell'applicazione. Nei presenti casi, si identificano come oggetto di interesse per i test i seguenti metodi:

- Session Session::addParticipant(Student participant)
- Session Session::removeParticipant(Student participant)

- CalendarEntryEntity CalendarEntryEntity::addParticipant(Student participant)
- CalendarEntryEntity CalendarEntryEntity::removeParticipant(Student participant)
- StudentGroup StudentGroup::addMember(Student member)
- StudentGroup StudentGroup::removeMember(Student member)

Session::addParticipant

Il metodo addParticipant accetta in input uno studente, inteso come un nuovo partecipante alla sessione. L'invocazione del metodo addParticipant su tale sessione, comporterà che la sessione conterrà il partecipante in input. Il controllo condizionale nel metodo assicura che l'utente venga aggiunto solo se non è già presente nell'elenco dei partecipanti.

Session::removeParticipant

Il metodo removeParticipant accetta in ingresso uno studente, considerato come un partecipante della sessione, e procede a rimuoverlo dall'elenco dei partecipanti. L'invocazione del metodo removeParticipant comporterà la modifica della sessione che non conterrà più il partecipante in input.

```

1 public class SessionTest {
2
3     private Student owner;
4
5     @BeforeEach
6     public void setup() {
7         owner = new Student("mrossi", "Mario", "Rossi", "password",
8                             LocalDate.now(),
9                             "Ingegneria Informatica", "2 Anno
10                            Magistrale", "mrossi@gmail.com", 0,
11                            Arrays.asList("ROLE_USER"));
12     }
13
14     @Test
15     public void addParticipant() {
16         // Sessione con solo l'owner come partecipante
17         Session testSession = new Session("SAD", LocalDateTime.
18                                         now(), "Aula studio", owner, new ArrayList<>(Arrays.
19                                         asList(owner)));
20
21         // Nuovo partecipante
22         Student student = new Student("gespo", "Giuseppe", "Esposito",
23                                         "password", LocalDate.now(),
24                                         "Ingegneria Informatica", "2 Anno
25                                         Magistrale", "gespo@gmail.com", 0,
26                                         Arrays.asList("ROLE_USER"));
27
28         testSession.addParticipant(student);
29     }
30 }
```

```
21
22     Session expectedSession = new Session("SAD",
23         LocalDateTime.now(), "Aula studio", owner, Arrays.
24         asList(owner, student));
25
26
27     assertEquals(expectedSession.getParticipants(),
28         testSession.getParticipants());
29 }
30
31
32     @Test
33     public void removeParticipant() {
34         // Sessione con solo l'owner come partecipante
35         Session testSession = new Session("SAD", LocalDateTime.
36             now(), "Aula studio", owner, new ArrayList<>(Arrays.
37             asList(owner)));
38
39         // Nuovo partecipante
40         Student student = new Student("gespo", "Giuseppe", "Esposito",
41             "password", LocalDate.now(),
42             "Ingegneria Informatica", "2 Anno
43             Magistrale", "gespo@gmail.com", 0,
44             Arrays.asList("ROLE_USER"));
45
46         testSession.addParticipant(student);
47         testSession.removeParticipant(student);
48
49         Session expectedSession = new Session("SAD",
50             LocalDateTime.now(), "Aula studio", owner, Arrays.
```

```
40
41     assertEquals(expectedSession.getParticipants(),
42                   testSession.getParticipants());
43
44 }
45 }
```

CalendarEntryEntity::addParticipant

Il metodo addParticipant accetta in input uno studente, considerato come un nuovo partecipante all'evento di calendario. L'invocazione del metodo addParticipant su tale evento di calendario, comporterà che l'evento di calendario corrisponderà ad un evento contenente il partecipante fornito in input. Il controllo condizionale nel metodo assicura che lo studente venga aggiunto solo se non è già presente nell'elenco dei partecipanti.

CalendarEntryEntity::removeParticipant

Il metodo removeParticipant accetta in ingresso uno studente, considerato come un partecipante dell'evento di calendario, e procede a rimuoverlo dall'elenco dei partecipanti. L'invocazione del metodo removeParticipant comporterà la modifica dell'evento di calendario che corrisponderà ad un evento senza il partecipante fornito in input.

```

1 public class CalendarEntryEntityTest {
2
3     @Test
4     public void addParticipant() {
5         // Entry del calendario senza partecipanti
6         CalendarEntryEntity testEntry = new CalendarEntryEntity(
7             UUID.randomUUID().toString(), "Ricevimento SAD", "",
8             LocalDateTime.now(), LocalDateTime.now().
9                 plusHours(2), "dodgerblue", false,
10                false, null, null, null, true,
11                new ArrayList<>());
12
13         // Nuovo partecipante
14         Student student = new Student("gespo", "Giuseppe", "Esposito",
15                                         "password", LocalDate.now(),
16                                         "Ingegneria Informatica", "2 Anno
17                                         Magistrale", "gespo@gmail.com", 0,
18                                         Arrays.asList("ROLE_USER"));
19
20         testEntry.addParticipant(student);
21
22         CalendarEntryEntity expectedEntry = new
23             CalendarEntryEntity(UUID.randomUUID().toString(), "Ricevimento SAD",
24                 "", LocalDateTime.now(), LocalDateTime.now().
25                     plusHours(2), "dodgerblue", false,
26                     false, null, null, null, true,
27                     Arrays.asList(student));
28     }
29
30 }
```

```
19  
20     assertEquals(expectedEntry.getParticipants(), testEntry.  
21         getParticipants());  
22  
23 @Test  
24 public void removeParticipant() {  
25     // Entry del calendario senza partecipanti  
26     CalendarEntryEntity testEntry = new CalendarEntryEntity(  
27         UUID.randomUUID().toString(), "Ricevimento SAD", "",  
28         LocalDateTime.now(), LocalDateTime.now().  
29             plusHours(2), "dodgerblue", false,  
30             false, null, null, null, true,  
31             new ArrayList<>());  
32  
33     // Nuovo partecipante  
34     Student student = new Student("gespo", "Giuseppe", "  
35         Esposito", "password", LocalDate.now(),  
36         "Ingegneria Informatica", "2 Anno  
37         Magistrale", "gespo@gmail.com", 0,  
38         Arrays.asList("ROLE_USER"));  
39  
40     testEntry.addParticipant(student);  
41     testEntry.removeParticipant(student);  
42  
43     CalendarEntryEntity expectedEntry = new  
44         CalendarEntryEntity(UUID.randomUUID().toString(), "  
45             Ricevimento SAD", "",  
46             LocalDateTime.now(), LocalDateTime.now().  
47                 plusHours(2), "dodgerblue", false,  
48                 false, null, null, null, true,  
49                 new ArrayList<>());  
50  
51     assertEquals(expectedEntry.getParticipants(), testEntry.  
52         getParticipants());  
53 }
```

```
38         LocalDateTime.now() , LocalDateTime.now() .  
39             plusHours(2) , "dodgerblue" , false ,  
40             false , null , null , null , true ,  
41             new ArrayList<>()) ;  
42     }  
43 }  
44 }
```

StudentGroup::addMember

Il metodo addMember accetta in input uno studente, considerato come un nuovo membro del gruppo. L'invocazione del metodo addMember su tale gruppo, comporterà che il gruppo corrisponderà ad un gruppo contenente il membro fornito in input. Il controllo condizionale nel metodo assicura che lo studente venga aggiunto solo se non è già presente nell'elenco dei membri.

StudentGroup::removeMember

Il metodo removeMember accetta in ingresso uno studente, considerato come un membro del gruppo, e procede a rimuoverlo dall'elenco dei membri. L'invocazione del metodo removeMember corrisponderà ad un gruppo senza il membro fornito in input.

```

1 public class StudentGroupTest {
2
3     private Student owner;
4
5     @BeforeEach
6     public void setup() {
7         owner = new Student("mrossi", "Mario", "Rossi", "password",
8             LocalDate.now(),
9             "Ingegneria Informatica", "2 Anno
10            Magistrale", "mrossi@gmail.com", 0,
11            Arrays.asList("ROLE_USER"));
12     }
13
14     @Test
15     public void addMember() {
16         // Gruppo con solo 1'owner come membro
17         StudentGroup testGroup = new StudentGroup("Progetto SAD",
18             owner, new ArrayList<>(Arrays.asList(owner)));
19
20         // Nuovo membro
21         Student student = new Student("gespo", "Giuseppe", "Esposito",
22             "password", LocalDate.now(),
23             "Ingegneria Informatica", "2 Anno
24            Magistrale", "gespo@gmail.com", 0,
25             Arrays.asList("ROLE_USER"));
26
27         testGroup.addMember(student);
28     }
29
30 }
```

```

22     StudentGroup expectedGroup = new StudentGroup("Progetto
23         SAD", owner, Arrays.asList(owner, student));
24
25     assertEquals(expectedGroup.getMembers(), testGroup.
26         getMembers());
27 }
28
29 @Test
30 public void removeMember() {
31     // Gruppo con solo l'owner come membro
32     StudentGroup testGroup = new StudentGroup("Progetto SAD",
33         owner, new ArrayList<>(Arrays.asList(owner)));
34
35     // Nuovo membro
36     Student student = new Student("gespo", "Giuseppe", "
37         Esposito", "password", LocalDate.now(),
38         "Ingegneria Informatica", "2 Anno
39         Magistrale", "gespo@gmail.com", 0,
40         Arrays.asList("ROLE_USER"));
41
42     testGroup.addMember(student);
43     testGroup.removeMember(student);
44
45     StudentGroup expectedGroup = new StudentGroup("Progetto
46         SAD", owner, Arrays.asList(owner));
47
48     assertEquals(expectedGroup.getMembers(), testGroup.
49         getMembers());

```

```
42 }  
43  
44 }
```

7.3 Testing d'integrazione

Nel processo di testing di integrazione, si esamina il funzionamento dei moduli che interagiscono tra di loro. Inizialmente, verrà condotto un test sul backend dell'applicazione, seguito dalla valutazione dell'intero sistema attraverso l'interfaccia grafica fornita dal client.

7.3.1 Testing Backend

Per il testing del server, sono stati creati e eseguiti test utilizzando JUnit, focalizzandosi su funzioni che interagiscono con il database. I test verificano che le entità, salvate in memoria dopo l'esecuzione delle funzioni testate, corrispondano a quanto atteso. In particolare, è stato verificato il corretto funzionamento di tutte le funzionalità principali fornite dal server. Di seguito vi è un esempio di test che verifica il corretto funzionamento di funzioni relative al cambio della password.

```
1 @SpringBootTest
2 @AutoConfigureMockMvc
3 @TestExecutionListeners(listeners =
4     DependencyInjectionTestExecutionListener.class,
5     mergeMode = TestExecutionListeners.MergeMode.MERGE_WITH_DEFAULTS)
6 @DirtiesContext(classMode = DirtiesContext.ClassMode.
7     AFTER_EACH_TEST_METHOD)
8 public class PasswordPresenterTest {
9
10    static {
11        // Prevent Vaadin Development mode to launch browser
12        // window
13        System.setProperty("vaadin.launch-browser", "false");
14    }
15
16    @Autowired
17    private PasswordPresenterImpl presenter;
18
19    @Autowired
20    private DataService dataService;
21
22    private Student user;
23
24    @BeforeEach
25    public void setup() {
26        user = new Student("mrossi", "Mario", "Rossi", "{bcrypt}")
27            "$2a$10$GRLdNijSQMUvl/au9ofL.eDwmoohzzS7.rmNSJZ.0FxO/"
28            "BTk76klW",
29    }
```

```
24             LocalDate.now() , "Ingegneria Informatica"
25             , "2 Anno Magistrale" , "mrossi@gmail.
26             com" , 0 , Arrays.asList("ROLE_USER"));
27
28
29     @Test
30
31     @WithMockUser(username = "mrossi" , password = "password" , roles =
32         "USER")
33
34     public void password_cambiata_correttamente() {
35
36
37         presenter.changePassword("password" , "newpassword");
38
39
40         String expectedPassword = "newpassword";
41
42
43         String persistentPassword = dataService.searchStudent("mrossi").get(0).getPassword();
44
45
46         // Verifichiamo che la password sia stata cambiata
47         // correttamente
48
49         assertEquals(expectedPassword , persistentPassword);
50
51
52     }
53
54
55     @Test
56
57     @WithMockUser(username = "mrossi" , password = "password" , roles =
58         "USER")
```

```

45    public void controllo_password() {
46
47        // Controlli sull'utente autenticato (mrossi)
48        Boolean checkTrueResult = presenter.passwordCheck("password");
49
50        Boolean checkFalseResult = presenter.passwordCheck("notpassword");
51
52        // Verifichiamo che sia riconosciuta la password corretta
53        assertEquals(true, checkTrueResult);
54
55        // Verifichiamo che la password errata non sia
56        // riconosciuta
57        assertEquals(false, checkFalseResult);
58    }
59
60    @Test
61    @WithMockUser(username = "mrossi", password = "password", roles =
62        "USER")
63    public void password_cambiata_tramite_token() {
64        UUID uuid = UUID.randomUUID();
65        String token_string = uuid.toString().replaceAll("-", "");
66
67        String userEmail = "mrossi@gmail.com";
68
69        Token token = new Token(token_string, userEmail);
70        dataService.saveToken(token);

```

```
68
69     String expectedPassword = "newpassword";
70     presenter.restorePassword(userEmail, expectedPassword);
71
72     String persistentPassword = dataService.searchStudent("mrossi").get(0).getPassword();
73
74     assertEquals(expectedPassword, persistentPassword);
75 }
76
77 @Test
78 @WithMockUser(username = "mrossi", password = "password", roles =
79     "USER")
80
81     public void token_correttamente_eliminato() {
82
83         UUID uuid = UUID.randomUUID();
84         String token_string = uuid.toString().replaceAll("-", "");
85
86         String userEmail = "mrossi@gmail.com";
87
88         Token token = new Token(token_string, userEmail);
89         token = dataService.saveToken(token);
90
91
92         Token persistentToken = presenter.searchToken(
93             token_string);
```

```
91         // Verifichiamo che il token sia stato salvato e trovato
92         // correttamente
93         assertEquals(token, persistentToken);
94
95         presenter.restorePassword(userEmail, "newpassword");
96
97         persistentToken = presenter.searchToken(userEmail);
98
99         // Verifichiamo che il token sia stato eliminato dopo
100        // aver resettato la password
101
102        assertEquals(null, persistentToken);
103    }
104
105
106
107    private void assertPasswordEquals(String plainPassword, String
108                                     cryptPassword) {
109
110        PasswordEncoder encoder = PasswordEncoderFactories.
111            createDelegatingPasswordEncoder();
112
113        assertEquals(true, encoder.matches(plainPassword,
114                                         cryptPassword));
115    }
116
117
118    private void assertTokenEquals(Token expectedToken, Token
119                                 actualToken) {
120
121        assertEquals(expectedToken.getRandomToken(), actualToken.
122                    getRandomToken());
123
124        assertEquals(expectedToken.getEmail(), actualToken.
125                    getEmail());
126    }
127
```

I risultati dei test sono dettagliati nella seguente tabella:

Test	Funzionalità	API	Tipo variabili di ingresso	Tipo di ritorno	Esito
1	Creare evento di calendario	onEntriesCreated	Collection<Entry>	void	OK
2	Rimuovere evento di calendario	onEntriesRemoved	Collection<Entry>	void	OK
3	Modificare evento di calendario	onEntriesChanged	Collection<Entry>	void	OK
4	Creare Gruppo	OnStudentGroupCreated	StudentGroup, Set<Student>	void	OK
5	Rimuovere Gruppo	OnStudentGroupRemoved	StudentGroup	void	OK
6	Modificare Gruppo	OnStudentGroupUpdated	StudentGroup, Set<Student>	void	OK
7	Visualizzare Gruppo	onGroupClick	StudentGroup	void	OK
8	Abbandonare Gruppo	leaveGroup		void	OK
9	Cambiare Password	changePassword	String, String	void	OK
10	Verifica Password	checkPassword	String	boolean	OK
11	Recuperare Password	restorePassword	String, String	void	OK
12	Ricerca Token	searchToken	String	Token	OK
13	Aggiornare profilo	updateUser	CustomUserDetails	void	OK

Test	Funzionalità	API	Tipo variabili di ingresso	Tipo di ritorno	Esito
14	Ottenere utente autenticato	getAuthenticatedUser		Student	OK
15	Ricerca Studente	searchStudent	String	Student	OK
16	Creazione utente	createUser	CustomUserDetails	void	OK
17	Verificare se utente esiste	userExists	String	boolean	OK
18	Verificare se un email è associata ad utente esistente	emailExists	String	void	OK
19	Creare sessione	OnSessionCreated	StudentGroup, Set<Student>	void	OK
20	Rimuovere sessione	OnSessionRemoved	StudentGroup	void	OK
21	Modificare sessione	OnSessionUpdated	Session, Set<Student>	void	OK
22	Visualizzare sessione	onGroupClick	Session	void	OK
23	Abbandonare sessione	leaveGroup		void	OK

7.3.2 Test intero sistema tramite GUI

Sono riportati nella seguente tabella i test effettuati attraverso l'interfaccia grafica fornita dal client:

Test	Funzionalità	Azione	Input	Output	Esito
1	Visualizzare Profilo	Click su icona e "Profile"		Dati del profilo	OK
2	Modificare Profilo	Cambiare uno dei campi del profilo e click su "Submit"	Dati validi	Dati del profilo aggiornati	OK
3	Modificare Profilo	Cambiare uno dei campi del profilo e click su "Submit"	"08/12/2022" su data	Errore	OK
4	Visualizzare Dashboard	Click su "Dashboard" sul drawer laterale		Componenti OK Dashboard	
5	Recupero Password	Click su "Forgot Password"	Email associata ad un account	Email inviata	OK
6	Recupero Password	Click su "Forgot Password"	Email non associata ad un account esistente		OK
7	Creare Evento Calendario	Click su una casella del calendario	Titolo, Colore, Descrizione, Tipo di evento, Inizio e Fine Evento	Evento Creato	OK

8	Modificare Evento Calendario	Click sull'evento del calendario, modifica campi e click su "Save"	Titolo, Colore, Descrizione, Tipo di evento, Inizio e Fine Evento	Evento Modificato	OK
9	Eliminare Evento Calendario	Click sull'evento del calendario e click su "Remove"		Evento eliminato	OK
10	Visualizzare Gruppi	Click su "Groups" dal drawer laterale		Lista Gruppi	OK
10	Visualizzare lista membri di un gruppo	Click su gruppo di cui si è membro		Lista membri del gruppo	OK
11	Creare gruppo	Click su "Add Group"	Nome, Lista di Studenti	Gruppo creato	OK
12	Modificare Gruppo	Doppio Click su gruppo di cui si è owner, modifica campi e click su "Save"	Nome, Lista di Studenti	Gruppo modificato	OK
13	Eliminare Gruppo	Doppio Click su gruppo di cui si è owner e click su "Remove"		Gruppo eliminato	OK
14	Lasciare Gruppo	Click su gruppo di cui si è membro e click su "Leave"		Utente rimosso dal gruppo	OK

15	Visualizzare Sessioni	Click su "Sessions" sul drawer laterale		Lista sessioni	OK
16	Visualizzare lista partecipanti di una sessione	Click su sessione alla quale si partecipa		Lista partecipanti della sessione	OK
17	Creare sessione	Click su "Add Session"	Materia, Luogo, Orario, Lista di Studenti	Sessione creata	OK
18	Creare sessione da un gruppo	Click su un gruppo di cui si è parte e click su "Add Session"	Materia, Luogo, Orario	Sessione creata	OK
19	Modificare sessione	Doppio Click su sessione del quale si è owner, modifica campi e click su "Save"	Materia, Luogo, Orario, Lista di Studenti	Sessione modificata	OK
20	Eliminare sessione	Doppio Click su sessione del quale si è owner e click su "Remove"		Sessione eliminata	OK
21	Lasciare sessione	Click su sessione di cui si è partecipante e click su "Leave"		Utente rimosso dalla sessione	OK

22	Accettare richiesta di partecipazione a sessione	Click sull'icona delle notifiche e click su "Accetta"		Utente aggiunto alla sessione	OK
23	Accettare richiesta di aggiunta a gruppo	Click sull'icona delle notifiche e click su "Accetta"		Utente aggiunto al gruppo	OK

Chapter 8

Manuale Utente

Il presente manuale ha lo scopo di introdurre i concetti fondamentali dell'utilizzo dell'applicazione agli utenti.

8.1 Pagina login (/login)

Su questa pagina, gli utenti hanno la possibilità di completare il processo di registrazione per la prima volta inserendo un username, un indirizzo email e una password, oppure di effettuare l'accesso se sono già registrati.

Login now

Log in

Username •

Password •

Log in

[Forgot password](#)

[Registrati Ora](#)

Figure 8.1: Pagina Login (/login)

8.2 Navbar

L'elemento cruciale per la navigazione all'interno della web app è la NavBar, che consente di accedere al drawer situato a sinistra in qualsiasi momento. Questo è accessibile attraverso un pulsante che richiama un hamburger menu. Inoltre, sarà sempre visibile il notification center, il quale avviserà tramite un'icona della presenza o assenza di notifiche in arrivo. In aggiunta, è disponibile un pop-up menu del profilo utente, dalla quale è possibile visualizzare il proprio profilo o effettuare il logout.



Figure 8.2: Screenshot di dettaglio NavBar

8.3 Dashboard (/dashboard)

Un elemento fondamentale per monitorare i progressi relativi alle sessioni, ai gruppi creati e agli eventi del calendario è la dashboard, la quale funge da pagina iniziale dopo aver effettuato l'accesso. All'interno di essa, è implementata una visualizzazione compatta del calendario che consente di esaminare gli eventi giornalieri. Inoltre, sono presenti grafici che mostrano le statistiche mensili delle sessioni e una griglia

che riporta le informazioni dettagliate sulle varie sessioni e sui gruppi. La dashboard offre una panoramica immediata e intuitiva delle attività in corso, fornendo all'utente un modo efficiente per tenere sotto controllo i propri impegni, monitorare i progressi e gestire le interazioni nei gruppi creati. La combinazione di elementi visivi come i plot mensili e la griglia delle sessioni fornisce una visione completa e dettagliata delle informazioni essenziali per l'utente.

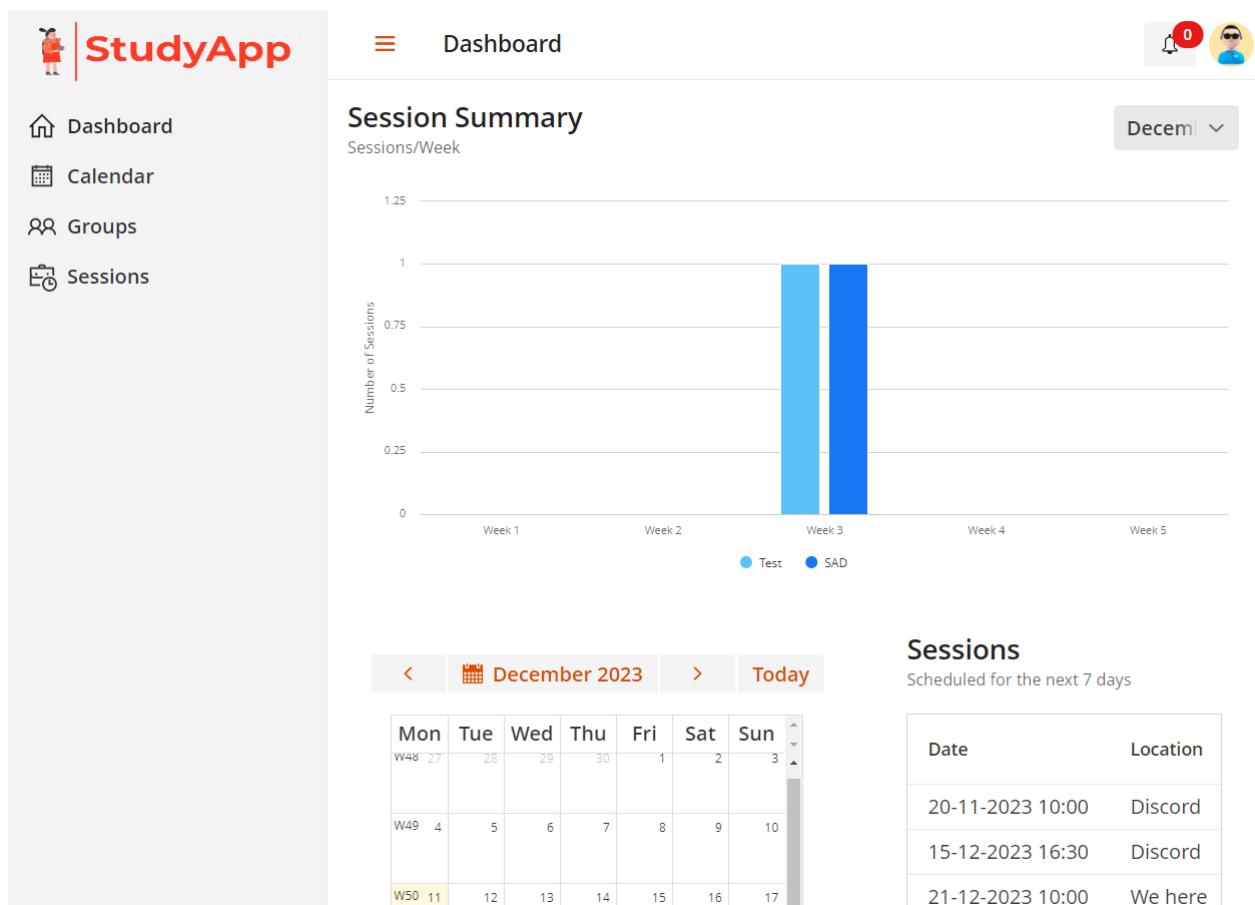


Figure 8.3: Pagina Dashboard (/dashboard)

8.4 Profilo personale (/profile/me)

La pagina del profilo personale, accessibile tramite l'avatar nella Navbar in alto a destra, consente agli utenti di visualizzare e modificare le informazioni personali, quali nome, cognome, corso di studi, anno accademico ed email. Inoltre, offre la possibilità di cambiare la password per garantire la sicurezza dell'account.

8.5 Profilo amico (/profile/username)

La pagina del profilo amico è visibile successivamente all'atto di creazione di gruppi o sessioni. Gli utenti possono esplorare le informazioni personali del loro amico, come nome, cognome, corso di studi, anno accademico ed email, offrendo dettagli rilevanti durante la formazione di gruppi o sessioni e nella gestione dei partecipanti.

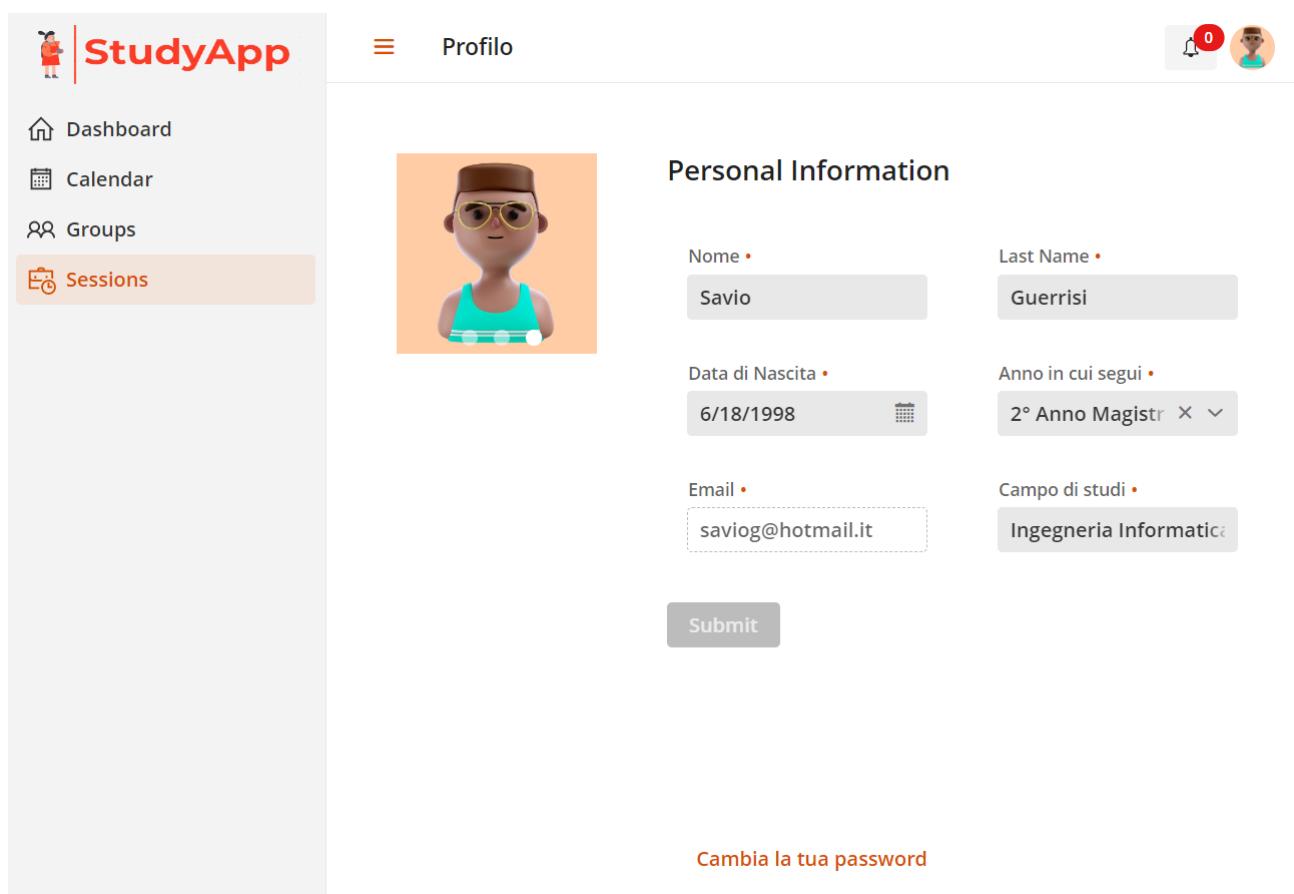


Figure 8.4: Pagina profilo personale (/profile/me)

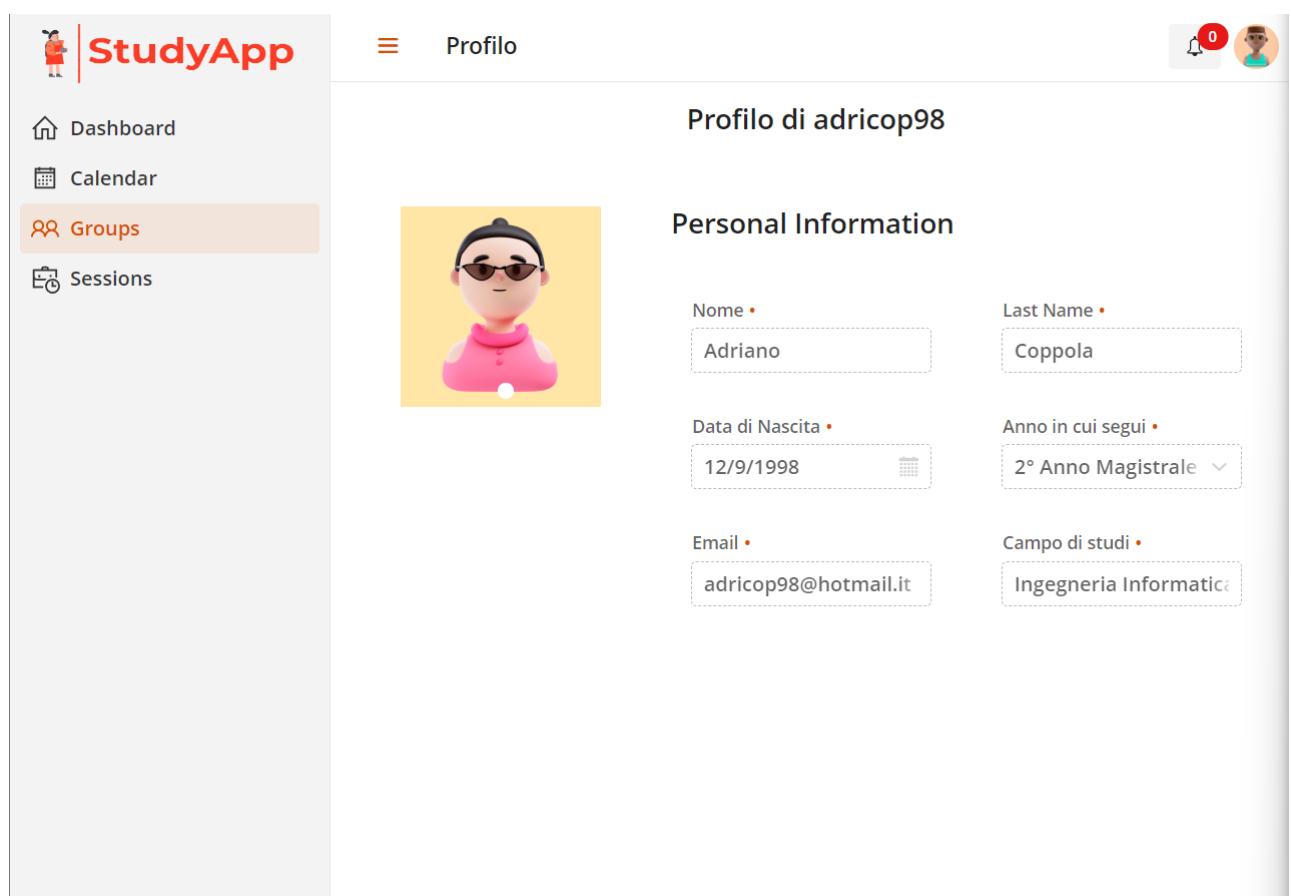


Figure 8.5: Pagina profilo amico (/profile/username)

8.6 Gruppi (/groups)

Nella sezione dedicata alla gestione dei gruppi è possibile esplorare i diversi gruppi a cui si appartiene, ottenendo dettagli sulle loro caratteristiche e sui membri partecipanti. Utilizzando il pulsante "Add group", è possibile creare un nuovo gruppo, definirne un nome e selezionare gli studenti da invitare. Nella parte inferiore della pagina, una griglia fornisce informazioni dettagliate sugli utenti membri del gruppo selezionato dalla griglia superiore denominata "Student Groups". Inoltre, selezionando uno studente membro, sarà possibile visualizzare il suo profilo. Questa pagina offre una gestione completa e intuitiva dei gruppi, consentendo agli utenti di esplorare, creare e gestire facilmente le interazioni all'interno di gruppi specifici.

The screenshot shows the 'Groups' section of the StudyApp web interface. On the left, a sidebar menu includes 'Dashboard', 'Calendar', 'Groups' (which is highlighted in orange), and 'Sessions'. The main content area has a header 'Groups' with a bell icon showing '0' notifications and a user profile icon.

Student Groups (1)

Name	Admin	# Members
Progetto SAD	andreaol	3

Members (3)

Username	Corso di Studi	Anno frequentante
andreaol	Ingegneria Informatica	2° Anno Magistrale
adricop98	Ingegneria Informatica	2° Anno Magistrale
saviog	Ingegneria Informatica	2° Anno Magistrale

Figure 8.6: Pagina gestione gruppi (/groups)

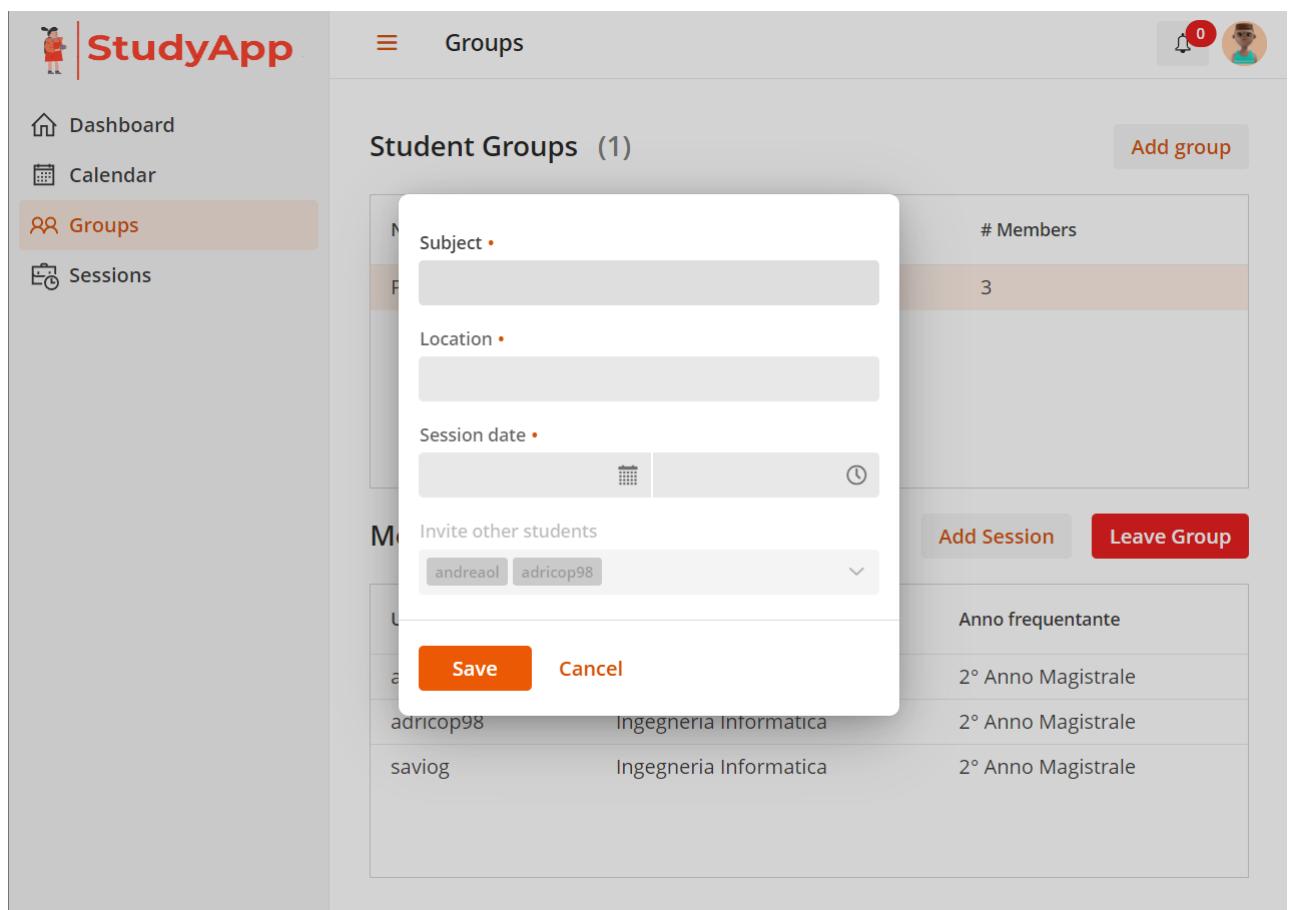


Figure 8.7: Dialog aggiunta gruppo (/groups)

8.7 Sessioni (/sessions)

Nella sezione dedicata alla gestione delle sessioni, è possibile esaminare le sessioni programmate, ottenendo dettagli informativi su di esse e sugli studenti o gruppi partecipanti. Utilizzando il pulsante "Add session", è possibile creare una nuova sessione specificando la materia, il luogo, la data della sessione e selezionando gli studenti da invitare. Analogamente alla pagina di gestione dei gruppi, si troverà una griglia denominata "Participants", tramite la quale sarà possibile visualizzare i profili degli studenti che partecipano alla sessione. Questa pagina offre un controllo completo e dettagliato sulle sessioni, consentendo agli utenti di esplorare, creare e gestire facilmente le loro attività di studio. La possibilità di visualizzare i dettagli delle sessioni e i profili degli studenti partecipanti fornisce una panoramica completa delle dinamiche delle sessioni di studio e delle relazioni tra gli utenti coinvolti.

CHAPTER 8. MANUALE UTENTE

The screenshot shows the 'Sessions' section of the StudyApp web interface. On the left sidebar, 'Sessions' is highlighted. The main area displays a single session entry:

Date	Location	Subject	Admin	# Participants
20-11-2023 10:00	Discord	SAD	andreaol	3

Below this, a 'Participants' section lists three users:

Username	Corso di Studi	Anno frequentante
andreaol	Ingegneria Informatica	2° Anno Magistrale
adricop98	Ingegneria Informatica	2° Anno Magistrale
saviog	Ingegneria Informatica	2° Anno Magistrale

A red 'Leave Session' button is visible in the top right corner of the participant table.

Figure 8.8: Pagina gestione sessioni (/sessions)

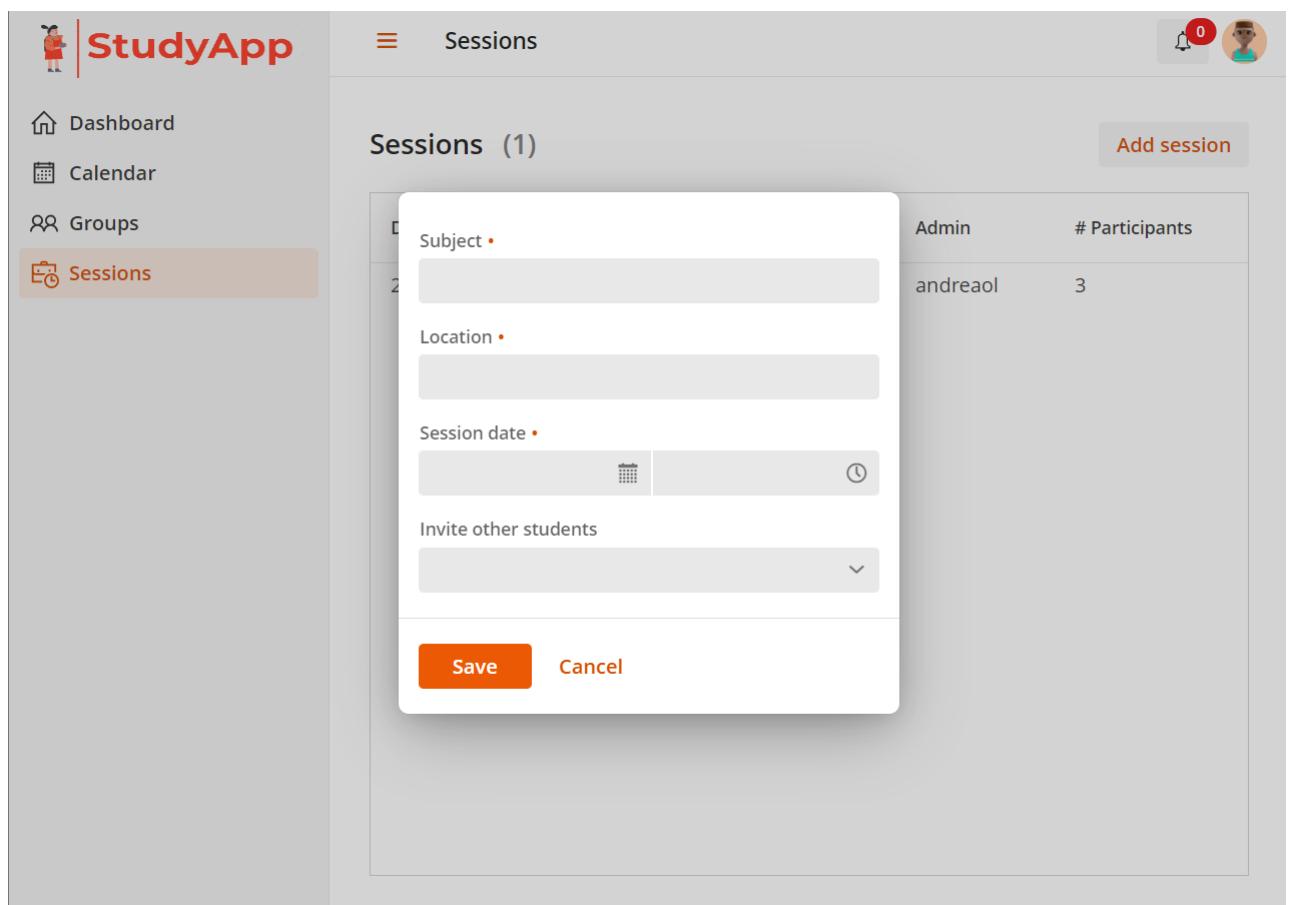


Figure 8.9: Dialog aggiunta sessione (/sessions)

8.8 Notifiche e gestione richieste

Come precedentemente indicato nella sezione riguardante la Navbar, è essenziale sottolineare che questo elemento grafico persistente offre un accesso immediato al notification center, posizionato in alto a destra. Grazie a questa disposizione, gli utenti possono visualizzare le notifiche in arrivo, tra cui inviti per partecipare a sessioni o gruppi, da qualsiasi sezione della web app. La presenza costante della Navbar garantisce un accesso diretto alle informazioni cruciali, mantenendo le notifiche sempre sotto controllo per gli utenti durante la navigazione.

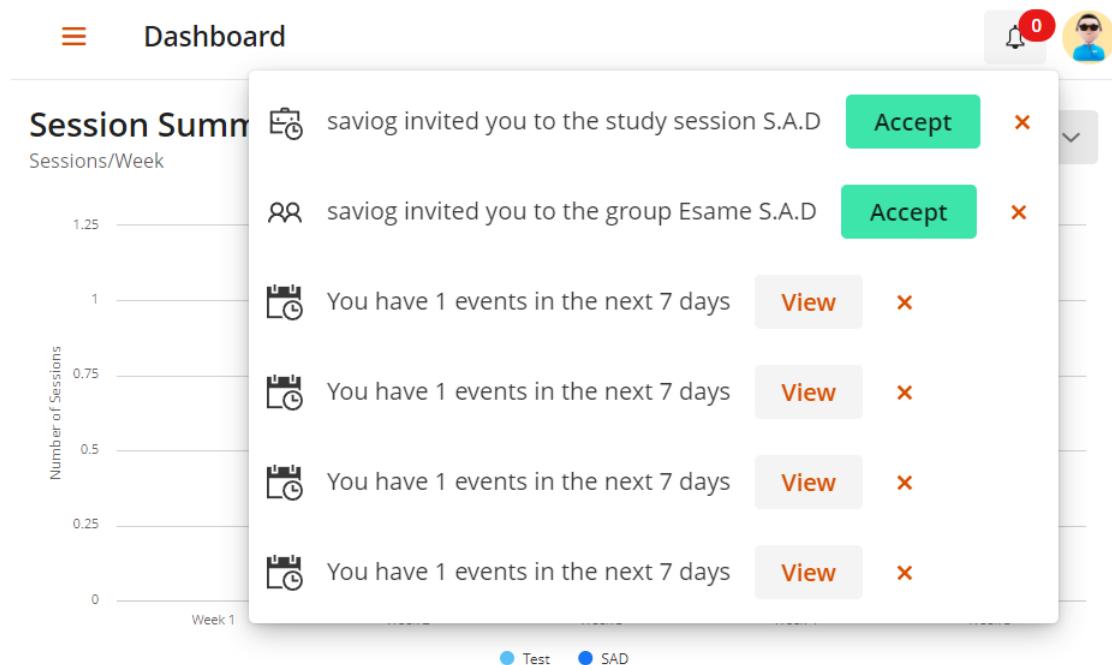


Figure 8.10: Notification center contenente varie tipologie di notifiche