Università degli Studi di Milano - Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di laurea in Informatica

ChatGPT a supporto dello sviluppo software: Un caso di studio in ambito mobile

Relatore: Daniela Micucci

Correlatore: Maria Teresa Rossi

Relazione della prova finale di:
Montrasio Filippo
Matricola 875551

Anno Accademico 2023 - 2024

Indice

1 Introduzione

Negli ultimi anni, i modelli di linguaggio di grandi dimensioni (LLM), come ChatGPT, hanno acquisito un'importanza crescente in vari ambiti della tecnologia e dell'innovazione. Questi sofisticati strumenti, capaci di comprendere e generare testo in linguaggio naturale, offrono supporto in numerosi settori, dalle comunicazioni aziendali alla ricerca scientifica, fino allo sviluppo di applicazioni e soluzioni tecnologiche. In particolare, i LLM stanno assumendo un ruolo sempre più rilevante nello sviluppo software.

1.1 LLM e ChatGPT

I LLM rappresentano uno degli sviluppi più avanzati nel campo dell'intelligenza artificiale. Essi utilizzano tecniche di machine learning e deep learning per elaborare enormi quantità di dati testuali tratti da diverse fonti, come libri, articoli e siti web; ciò consente ai modelli di apprendere e comprendere il linguaggio naturale con alta precisione. Una delle caratteristiche più rilevanti dei LLM è la loro capacità di generare testi che siano corretti sotto il profilo grammaticale, coerenti dal punto di vista semantico e adatti al contesto. Questo è reso possibile grazie all'uso di reti neurali e tecniche di apprendimento non supervisionato, che permettono ai modelli di assimilare strutture linguistiche e significati in modo efficace.

L'adozione dei modelli di linguaggio di grandi dimensioni nello sviluppo software e nella programmazione sta diventando sempre più significativa. Questi modelli possono assistere gli sviluppatori nella generazione di codice, nella redazione della documentazione, nella risoluzione di problemi tecnici e nella creazione di test. Tali capacità contribuiscono a rendere più rapidi ed efficienti i processi di sviluppo. Tuttavia, nonostante il loro valore considerevole, LLM presentano ancora alcune limitazioni. Questi modelli possono produrre contenuti errati o fuorvianti e richiedono supervisione umana per garantire la sicurezza e l'accuratezza delle informazioni generate, soprattutto se il modello non riceve un contesto chiaro o sufficiente. Il continuo progresso dei modelli e il perfezionamento delle tecniche di addestramento saranno fondamentali per massimizzare le potenzialità dei LLM.

ChatGPT, in particolare, si basa su un meccanismo noto come *modello di linguaggio* e utilizza l'architettura dei **trasformatori**, per prevedere la parola successiva in una sequenza di testo in base a un input dell'utente. Quando ChatGPT riceve una richiesta, il modello applica le conoscenze acquisite durante l'addestramento per generare la parola più probabile che segue l'input fornito. Questo processo, analogo a quello del completamento automatico impiegato nei motori di ricerca, viene chiamato **self-attention** [1] e consente di esaminare il contesto delle parole all'interno di una frase. Inoltre, ChatGPT è progettato per eseguire compiti con **few-shot learning** [2],una capacità che consente al modello di adattarsi e performare bene anche con pochi esempi, senza la necessità di addestrare una rete specifica per ogni applicazione. Questo approccio permette a ChatGPT di rispondere a richieste specifiche senza richiedere un addestramento aggiuntivo per ciascuna nuova funzione. ChatGPT non ha però accesso a fonti in tempo reale durante l'addestramento e per questo motivo non ha una comprensione umana, ma apprende solo

schemi linguistici e strutture grammaticali. I modelli di apprendimento automatico, come ChatGPT, sono costituiti da una serie di numeri, chiamati **pesi** o **parametri**, e da codice che interpreta ed elabora questi numeri. Durante l'addestramento, i pesi vengono modificati per migliorare la capacità del modello di fare previsioni accurate. Tuttavia, i modelli non memorizzano direttamente i testi che leggono; piuttosto, l'apprendimento influisce sui pesi, che vengono regolati per riflettere le relazioni linguistiche identificate dal modello. Di conseguenza, il modello non conserva o copia frasi specifiche, ma affina la sua capacità di generare risposte pertinenti basandosi sulle informazioni assimilate.[3].

1.2 L'applicazione BicoccaHelp

Sfruttando le funzionalità di ChatGPT nel generare codice e fornire suggerimenti specifici, ho sviluppato un'applicazione di tutoraggio interno per l'Università, **BicoccaHelp**. Attraverso l'applicazione, gli studenti possono prenotare lezioni private in modo rapido e personalizzato. Solo gli iscritti all'Università Bicocca possono accedere all'applicazione, autenticandosi tramite la mail istituzionale. Questa misura assicura un elevato livello di sicurezza e affidabilità per gli utenti registrati.

Una volta completata la fase di autenticazione tramite l'email di ateneo, lo studente ha la possibilità di personalizzare il proprio profilo. Durante questo processo, l'utente può inserire informazioni relative al suo corso di studi, creando così un'identità digitale utile non solo per accedere ai servizi di tutoraggio, ma anche per facilitare la ricerca di tutor più affini alle sue necessità. Gli studenti possono aggiornare e modificare i dettagli del proprio profilo in qualsiasi momento. Gli utenti che desiderano offrire tutoraggio possono attivare il profilo tutor, specificando le proprie competenze accademiche e la disponibilità oraria. Questo duplice sistema di profili (studente e tutor) rende l'app versatile e adatta a chiunque voglia sia ricevere che offrire supporto accademico.

La funzionalità chiave dell'applicazione è la possibilità di prenotare lezioni di tutoraggio. Gli studenti possono cercare i tutor disponibili utilizzando diversi filtri, come il nome, il corso di studi frequentato, le competenze specifiche e le disponibilità orarie. Questa flessibilità di ricerca consente di trovare il tutor più adatto alle proprie esigenze di studio. Una volta selezionato il tutor, lo studente può prenotare una lezione scegliendo la data e l'orario in cui desidera svolgerla. Il sistema garantisce una gestione trasparente e facile delle prenotazioni: ogni lezione confermata viene automaticamente aggiunta alla home page dell'utente, dove è possibile visualizzare, modificare o cancellare la prenotazione. Tuttavia, le lezioni possono essere modificate o annullate solo fino a 24 ore prima dell'orario previsto.

L'applicazione prevede anche una sezione dedicata alla gestione del profilo, dove gli utenti possono modificare il nome, la password e caricare una foto profilo. Quest'ultima funzionalità contribuisce a migliorare l'esperienza d'uso e la personalizzazione della piattaforma, facilitando al contempo l'identificazione degli utenti all'interno della comunità universitaria. In questo modo l'interazione tra studenti e tutor diventa più riconoscibile e umanizzata.

Un ulteriore elemento che valorizza l'applicazione è il sistema di recensioni. Gli studenti hanno la possibilità di valutare i tutor con cui hanno avuto esperienza, assegnando una valutazione da 1 a 5 stelle. Gli utenti possono rilasciare una recensione per ciascun tutor solo una volta, garantendo che le valutazioni siano autentiche. Grazie alla sua interfaccia semplice e alle sue funzionalità, l'app potrà diventare uno strumento prezioso per l'ateneo, ottimizzando l'accesso ai servizi di tutoraggio e migliorando la qualità complessiva dell'esperienza accademica.

L'applicazione vede inoltre un ampio margine di sviluppo. Tra le future implementazioni, si prevede un'espansione delle funzionalità lato tutor, permettendo loro una gestione più dettagliata delle proprie sessioni, così come una sezione dedicata alla gestione delle lezioni prenotate da altri studenti. Queste integrazioni non solo aumenteranno l'efficienza del sistema, ma miglioreranno ulteriormente l'esperienza d'uso, offrendo un maggiore controllo e personalizzazione sia agli studenti che ai tutor. Inoltre, sarà introdotta una funzionalità di accumulo punti per gli studenti che offriranno il servizio di tutoraggio. Ogni lezione completata consentirà di guadagnare punti, che potranno essere utilizzati presso esercizi convenzionati situati nei pressi dell'università.

1.3 Obiettivi e valutazioni

Il presente lavoro di stage si propone di indagare l'efficacia di ChatGPT nel contesto dello sviluppo di un'applicazione mobile. L'obiettivo principale è valutare in che misura questo strumento possa realmente contribuire e facilitare il processo di sviluppo, implementazione e testing di un'applicazione mobile, mettendo a confronto la sua utilità con le aspettative e le esigenze tipiche di un progetto di sviluppo software. In particolare, il lavoro si concentrerà su tre aspetti fondamentali:

1. **Descrivere le funzionalità dell'applicazione:** Questa sezione sarà dedicata a un'analisi approfondita delle caratteristiche principali dell'applicazione sviluppata, includendo una descrizione di ogni fase del ciclo di vita dello sviluppo. Verranno analizzate l'analisi dei requisiti, la progettazione e l'implementazione delle funzionalità chiave. Ogni fase sarà esaminata in dettaglio, documentando le decisioni tecniche adottate, le tecnologie utilizzate e come si è proceduto nell'integrazione dei vari componenti dell'applicazione.

2. **Valutare il supporto offerto da ChatGPT:** In questa sezione verrà analizzato in che modo ChatGPT è stato integrato nel processo di sviluppo. Sarà valutato il suo contributo nella generazione di codice, nella progettazione dell'interfaccia utente (UI), nella gestione della logica di business e nella risoluzione di problematiche tecniche. Si esplorerà come ChatGPT ha facilitato la comunicazione e la comprensione di concetti complessi, automatizzato attività ripetitive e aiutato nel debugging del codice.

3. **Esaminare l'utilità e le limitazioni:** In questa parte del lavoro verrà fornita una valutazione critica del contributo complessivo di ChatGPT durante lo sviluppo. Saranno evidenziati i punti di forza dello strumento, come la sua capacità di accelerare il processo di sviluppo, offrendo soluzioni rapide e supporto continuo, riducendo così i tempi di progettazione e testing. Saranno però anche identificate le limitazioni riscontrate, come eventuali mancanze di contesto, proposte di codice non ottimali o problemi legati alla comprensione di specifiche tecniche complesse.

Durante lo sviluppo dell'applicazione, ChatGPT ha fornito supporto in diverse fasi del progetto, contribuendo alla definizione dei requisiti, all'implementazione del codice e alla risoluzione dei bug, dimostrando la versatilità dei LLM nel contesto dello sviluppo software.

In sintesi, questo lavoro non si limita a esplorare le funzionalità e l'efficacia di ChatGPT nel supportare lo sviluppo di un'applicazione mobile, ma mira anche a evidenziarne i limiti e le potenzialità, al fine di fornire una valutazione completa dell'utilizzo dei modelli di linguaggio di grandi dimensioni nello sviluppo software.

2 Analisi Dei Requisiti

L'analisi dei requisiti rappresenta una fase fondamentale nello sviluppo di qualsiasi applicazione software, poiché consente di delineare in modo preciso le funzionalità che il sistema dovrà offrire e di identificare i bisogni degli utenti finali. In questo capitolo verrà illustrata l'analisi dei requisiti condotta per l'applicazione **BicoccaHelp**.

L'obiettivo di questa fase è definire le funzionalità chiave che l'applicazione deve supportare, come la prenotazione delle lezioni, l'autenticazione degli utenti e la gestione dei profili. Nel corso di questo capitolo verranno presentati i requisiti funzionali e i requisiti di sistema, seguiti da una descrizione dei principali attori coinvolti. Il diagramma dei casi d'uso fornirà una panoramica dell'interazioni tra utenti e sistema. In seguito verrà presentata un'analisi dettagliata di ciascun caso d'uso.

2.1 Requisiti di Sistema

Di seguito sono riportati i requisiti principali che il sistema deve soddisfare per garantire un'esperienza efficace e completa:

- Il sistema deve consentire all'utente di accedere mediante l'inserimento di una e-mail e una password.
- Il sistema deve fornire un'interfaccia intuitiva e facile da usare.
- Il sistema deve salvare il profilo utente, creato dopo il primo accesso.
- Il sistema deve consentire di prenotare una lezione di tutoring all'utente studente.
- Il sistema deve consentire di modificare/cancellare una lezione di tutoring all'utente.
- Il sistema deve consentire agli utenti studenti di recensire l'utente insegnante.

2.2 Requisiti Utente

I seguenti requisiti descrivono le funzionalità che il sistema deve offrire agli utenti per garantire una gestione efficace delle lezioni di tutoring. Questi requisiti sono essenziali per assicurare una esperienza utente completa e soddisfacente

- L'utente deve poter accedere al sistema mediante l'inserimento di una e-mail, obbligatoriamente e-mail universitaria (dominio `@campus.unimib.it`).
- L'utente deve poter prenotare una lezione di tutoring.
- L'utente deve poter modificare/cancellare una lezione di tutoring, entro 24 ore dal suo svolgimento.
- Gli utenti devono poter visualizzare lezioni future e le lezioni passate.
- Gli utenti devono poter recensire l'insegnante.

2.3 Attori

Gli attori sono entità che interagiscono con il sistema e sono essenziali per il suo funzionamento. Ogni attore ha un ruolo specifico. Identificare e comprendere gli attori è cruciale per definire le funzionalità e i requisiti del sistema in modo che rispondano efficacemente alle loro esigenze. Di seguito sono elencati gli attori principali del sistema e le loro responsabilità.

2.3.1 Attori Primari e Attori Finali

- **Studente**:
  - **Ruolo**: Utilizza l'applicazione per prenotare lezioni di tutoring al fine di migliorare le proprie competenze accademiche.
  - **Obiettivo**: Ottenere assistenza dagli insegnanti per risolvere dubbi e approfondire argomenti di studio.
  - **Interazione con il sistema**: Prenota, modifica e cancella lezioni, visualizza il proprio storico di lezioni e fornisce recensioni sugli insegnanti.
- **Tutor**:
  - **Ruolo**: Fornisce supporto e assistenza agli studenti in difficoltà attraverso lezioni di tutoring.
  - **Obiettivo**: Offrire aiuto agli studenti e gestire il proprio programma di lezioni per massimizzare l'efficacia del supporto fornito.
  - **Interazione con il sistema**: L'interazione tra sistema e tutor è descritta nel paragrafo relativo agli sviluppi futuri (paragrafo 1.2) del capitolo 1

2.3.2 Attori esterni

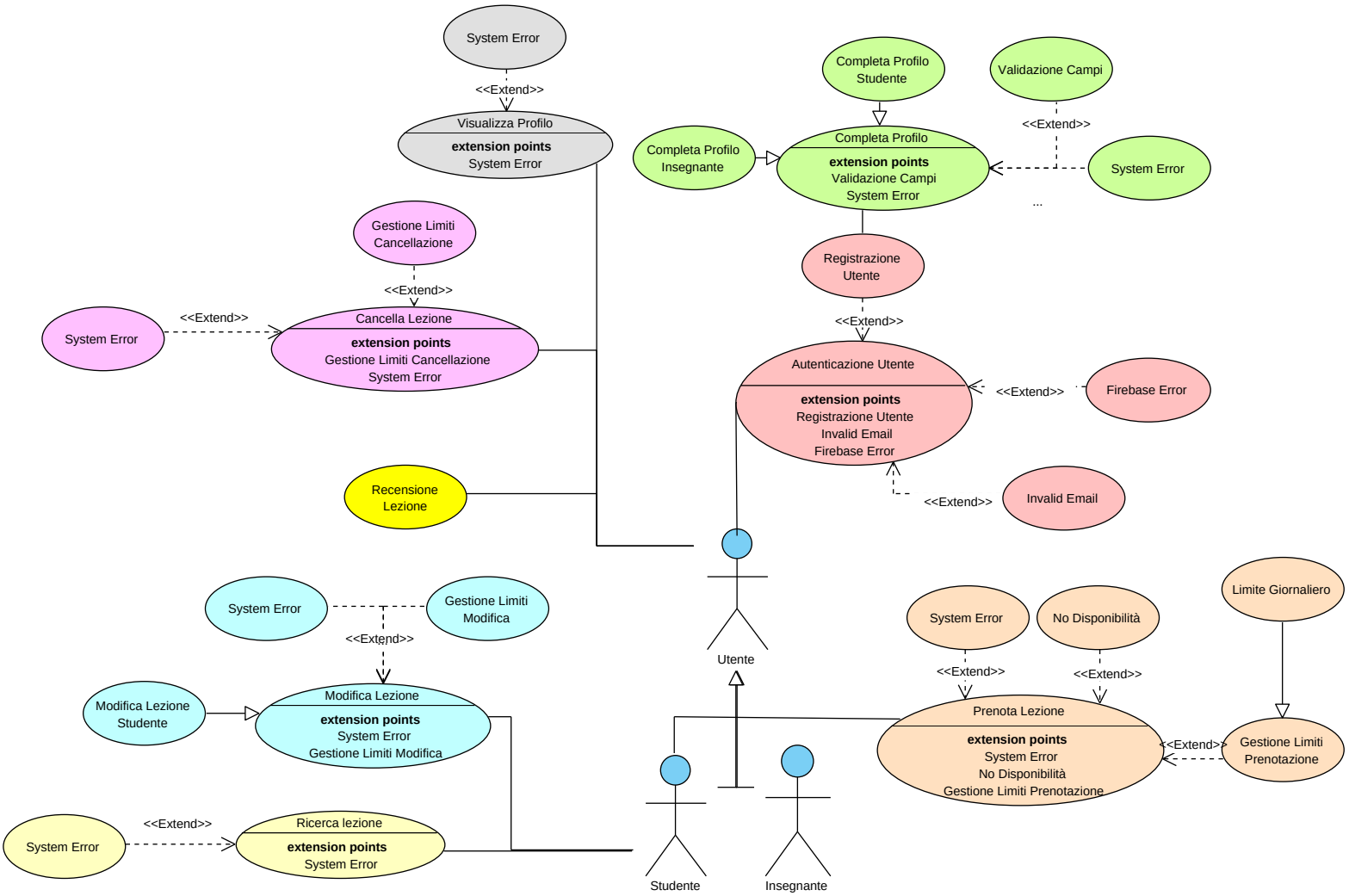
- **Firebase Authentication**: Servizio esterno per gestire l'accesso degli utenti. L' applicazione delega l'autenticazione a Firebase, che si occupa di verificare le credenziali degli utenti. Restituisce un token di autenticazione valido all'applicazione se le credenziali sono corrette. È un attore esterno in quanto non fa parte direttamente del sistema di prenotazioni delle lezioni, ma fornisce un servizio esterno per la gestione di un'attività specifica.
- **Firestore DB**: Servizio esterno per gestire il database dell'applicazione. Gestisce la memorizzazione e la lettura delle informazioni legate agli utenti, come i profili, le prenotazioni e le recensioni. È un attore esterno in quanto non fa parte direttamente del sistema, il quale interagisce con Firestore per reuperare e aggiornare i dati in tempo reale. Il suo funzionamento e la sua struttura verrà analizzata nel capitolo relativo all'implementazione (3)
- **Firebase Storage**: È responsabile del caricamento e della gestione di file, come le foto profilo degli utenti, che vengono salvati in uno spazio di archiviazione.
- **Glide**: Libreria esterna per il caricamento e la gestione delle immagini nell'applicazione. Glide si occupa di ottimizzare il caricamento delle immagini da Firebase Storage,

gestendo il caching e il ridimensionamento per garantire prestazioni elevate e un'esperienza utente fluida. È un attore esterno poiché non fa parte direttamente del sistema di prenotazione delle lezioni, ma fornisce un servizio fondamentale per la visualizzazione delle immagini, contribuendo a migliorare l'interfaccia utente e l'efficienza dell'app.

2.4 Casi d'uso

I casi d'uso sono uno strumento fondamentale nella progettazione del sistema in quanto descrivono le interazioni tra gli utenti e il sistema per raggiungere determinati obiettivi. Essi offrono una visione delle funzionalità dell'applicazione dal punto di vista degli utenti, garantendo che tutti i requisiti siano soddisfatti e che il sistema risponda efficacemente alle esigenze degli utenti.

In questo paragrafo esploreremo i principali casi d'uso del sistema, evidenziando le azioni che gli utenti possono intraprendere. Il diagramma dei casi d'uso allegato fornisce una rappresentazione visiva delle interazioni tra gli attori e i vari casi d'uso, facilitando la comprensione delle funzionalità. Di seguito il **Diagramma dei casi d'uso**:

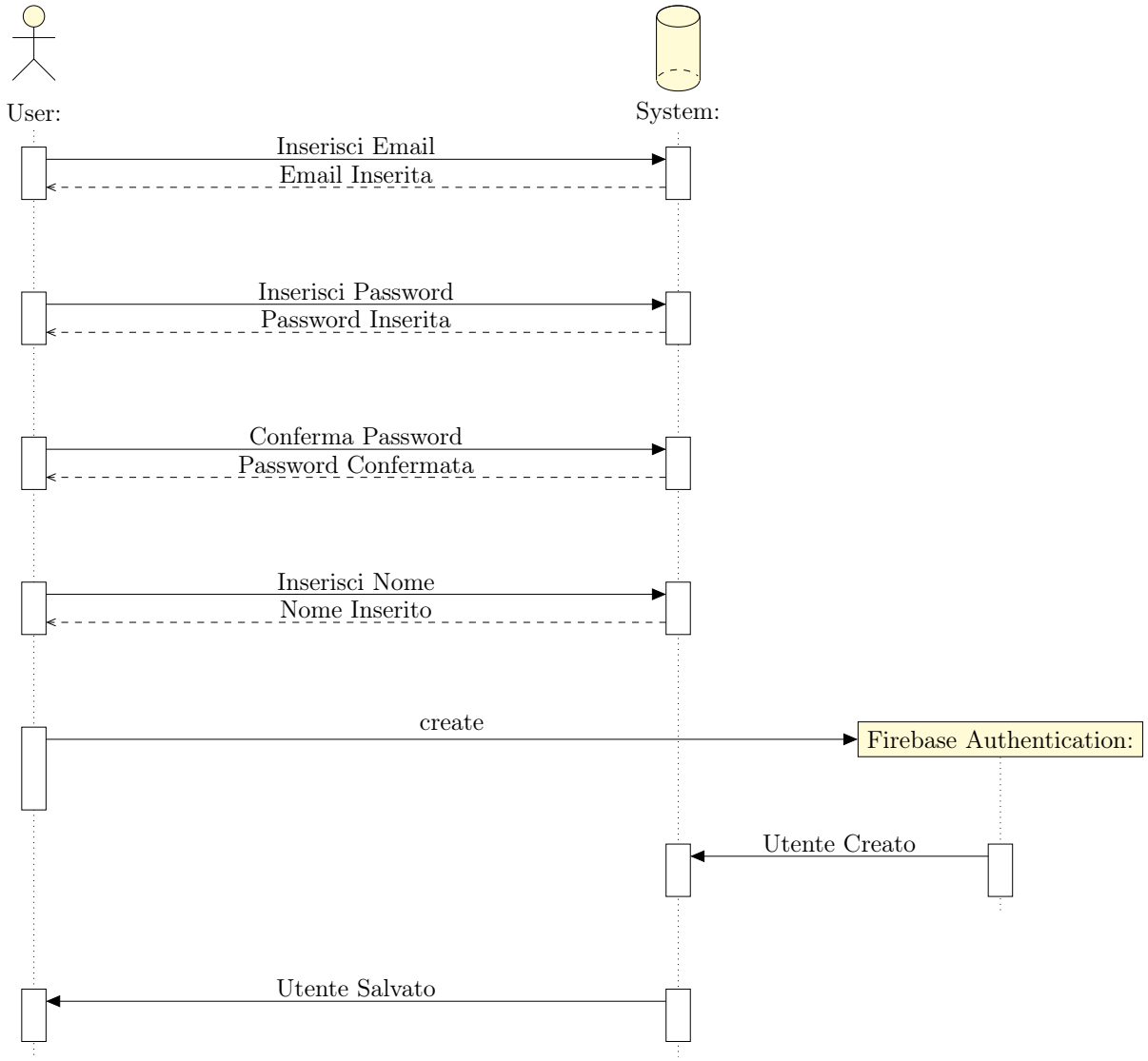


In seguito sono descritti i casi d'uso principali. Per ogni caso d'uso, è incluso un **Diagramma di sequenza** che illustra lo scenario di successo, mostrando in dettaglio le interazioni tra gli attori e il sistema. I diagrammi di sequenza aiutano a comprendere il flusso delle operazioni e visualizzare come le richieste degli utenti vengono gestite dal sistema.

2.4.1 UC01: Registrazione Utente

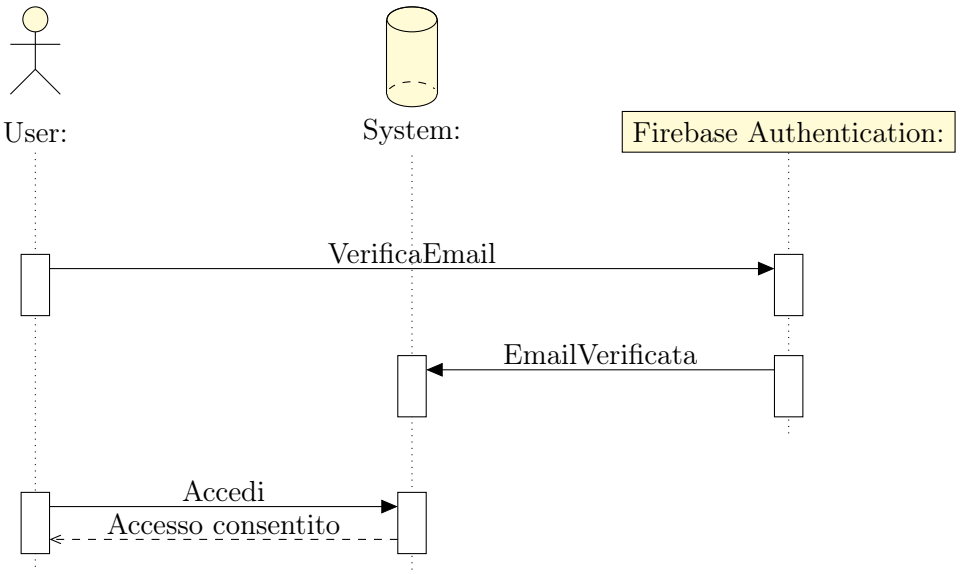
Use Case: Registrazione Utente	
Componente	**Descrizione**
ID	UC01
Attore Primario	Utente Generico
Attore Finale	Utente Generico
Pre Condizioni	L'utente deve registrarsi per accedere all'applicazione. Ha avviato l'applicazione e si trova nella pagina di accesso. Per accedere deve inserire la mail fornita dall'università (dominio: @campus.unimib.it), una password e un nome
Post Condizioni	Si verificano se il caso d'uso viene completato con successo. Dopo aver inserito nel campo di testo l'email, la password e il nome per accedere al servizio, l'utente verrà direzionato nella pagina per verificare la mail
Scenario di Successo	1. L'utente apre l'applicazione per la prima volta 2. L'utente non ha un account, preme il bottone *"Register"* 3. L'utente digita la mail fornita dall'università 4. L'utente digita la password rispettando i requisiti 5. L'utente conferma la password 6. L'utente digita il nome rispettando i requisiti 7. L'utente preme il bottone *"Sign In"* 8. L'utente si trova nella pagina per verificare la mail

Estensioni:	3a. L'utente inserisce una mail non valida. L'applicazione richiede di inserire una mail corretta. 4a. L'utente inserisce una password che non rispetta i criteri di sicurezza. L'applicazione invita a inserire una nuova password. 5a. L'utente inserisce una password e una conferma password che non coincidono. L'applicazione richiede di correggere la conferma della password. 6a. L'utente inserisce un nome non valido. L'applicazione richiede di inserire un nome che rispetti i requisiti. 7a. Firebase Authentication non verifica la mail a causa di un errore del servizio. L'utente deve reinserire la mail per autenticarsi nuovamente.



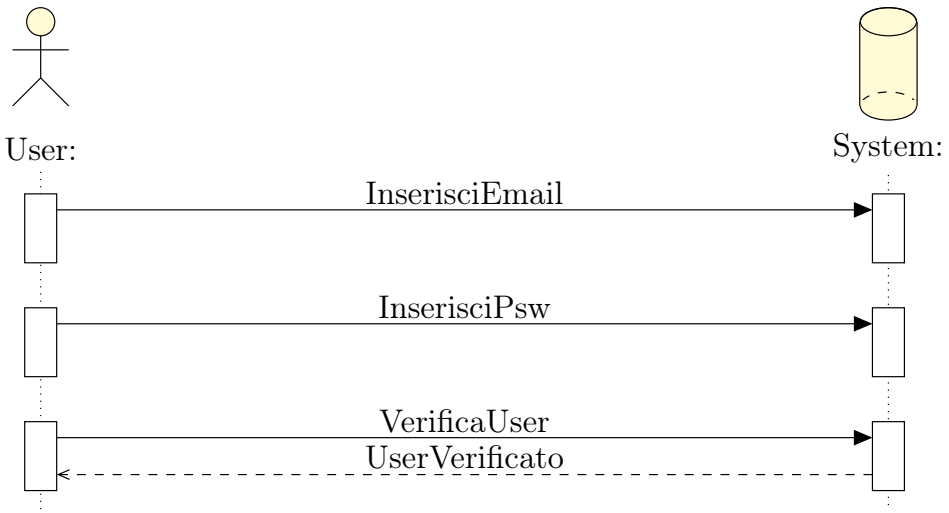
2.4.2 UC02: Verifica Email

Use Case: Verifica Email	
Componente	**Descrizione**
ID	UC02
Attore Primario	Utente Generico
Attore Finale	Utente Generico
Pre Condizioni	L'utente ha effettuato la registrazione all'applicazione mediante l'inserimento di email, password e nome. L'utente si trova ora nella schermata relativa alla verifica dell'email
Post Condizioni	L'utente ha verificato la email mediante l'apposito link inviato da Firebase Authentication. Dopo aver premuto il tasto *"Sign In"* si troverà nella Home Page
Scenario di Successo	1. L'utente deve verificare la propria email 2. Firebase Authentication invia un link per verificare la Email 3. L'utente, tramite il proprio gestore Email, verifica la Email e ritorna nell'applicazione 4. L'utente preme il bottone *"Sign In"* 5. L'utente viene direzionato nella Home Page
Estensioni:	2a. L'utente non riceve l'email di verifica o il link è scaduto. L'utente preme *"Send Email"* e riprova. 4a. L'utente prova a cliccare *"Sign In"* senza aver verificato la mail. L'applicazione notifica di verificare la mail prima di accedere



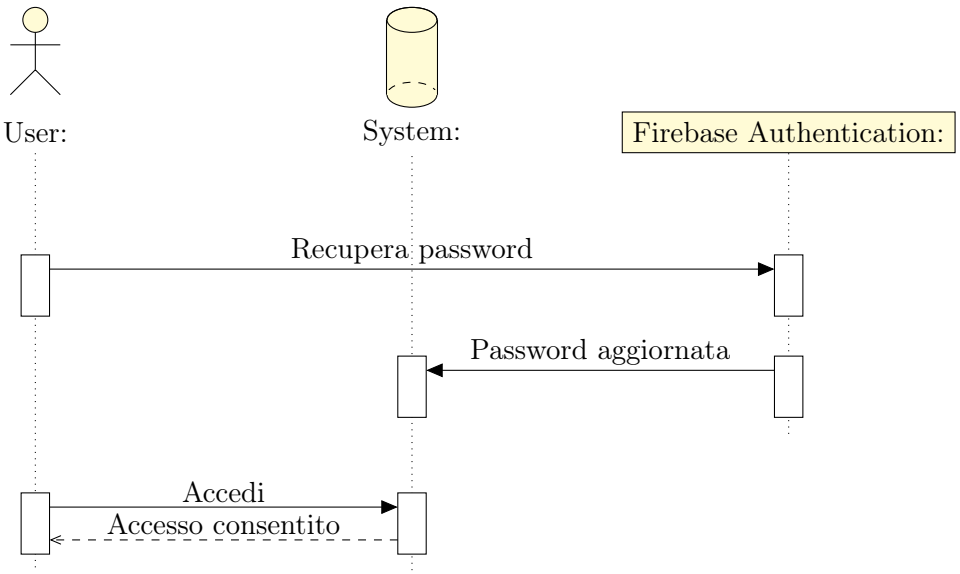
2.4.3 UC03: Autenticazione Utente

Use Case: Autenticazione Utente	
Componente	**Descrizione**
ID	UC03
Attore Primario	Utente Generico
Attore Finale	Utente Generico
Pre Condizioni	L'utente è registrato e si trova nella pagina di Login, nel quale dovrà inserire Email e password per accedere.
Post Condizioni	Dopo che lo studente si è autenticato, se la mail è stata verificata in fase di registrazione, si troverà nella home page
Scenario di Successo	1. L'utente si trova nella pagina Log In 2. L'utente inserisce la email utilizzata in fase di registrazione 3. L'utente inserisce la password 4. L'utente preme il bottone *"Sign In"*, accede alla Home Page
Estensioni:	1a - 3a. L'utente inserisce una email o una password errata. L'applicazione invita a inserire email e password corretti. 1b - 4b. L'utente prova ad accedere senza inserire email e/o password. L'applicazione notifica di inserire email e password per accedere. 1c - 4c L'utente prova ad accedere senza aver verificato la email, viene direzionato alla pagina per verificare l'email



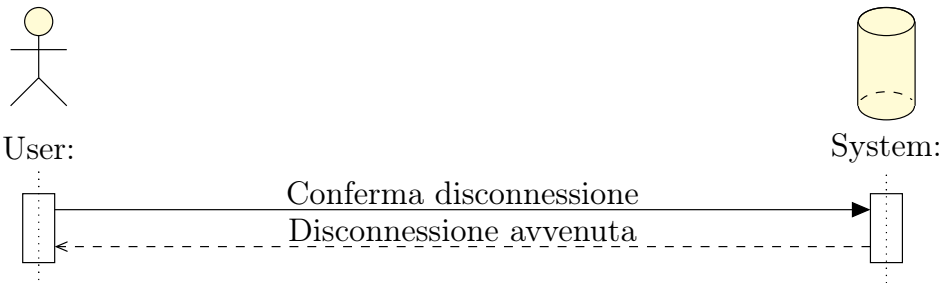
2.4.4 UC04: Recupero Password

Use Case: Recupero Password	
Componente	**Descrizione**
ID	UC04
Attore Primario	Utente Generico
Attore Finale	Utente Generico
Pre Condizioni	L'utente è registrato e si trova nella pagina di Login, ma non ricorda la password.
Post Condizioni	Dopo che l'utente ha recuperato la password, si troverà nuovamente nella pagina di Login, ma potrà accedere
Scenario di Successo	1. L'utente preme il bottone *"Forgot Password"* 2. L'utente inserisce la email utilizzata in fase di registrazione 3. L'utente preme il bottone *"Send Email"* 4. Firebase Authentication invierà all'utente un link per reimpostare la password 5. L'utente, tramite il proprio gestore Email, reimposta la password 6. L'utente ritorna nella pagina di Log In e si autentica
Estensioni:	3a L'utente preme il bottone *"Send Email"* senza aver inserito una email. L'applicazione notifica l'utente di inserire una email. 2a - 3b L'utente inserisce una email non valida e preme il bottone *"Send Email"*. Non verrà inviata nessuna mail e l'utente torna nella pagina di login.



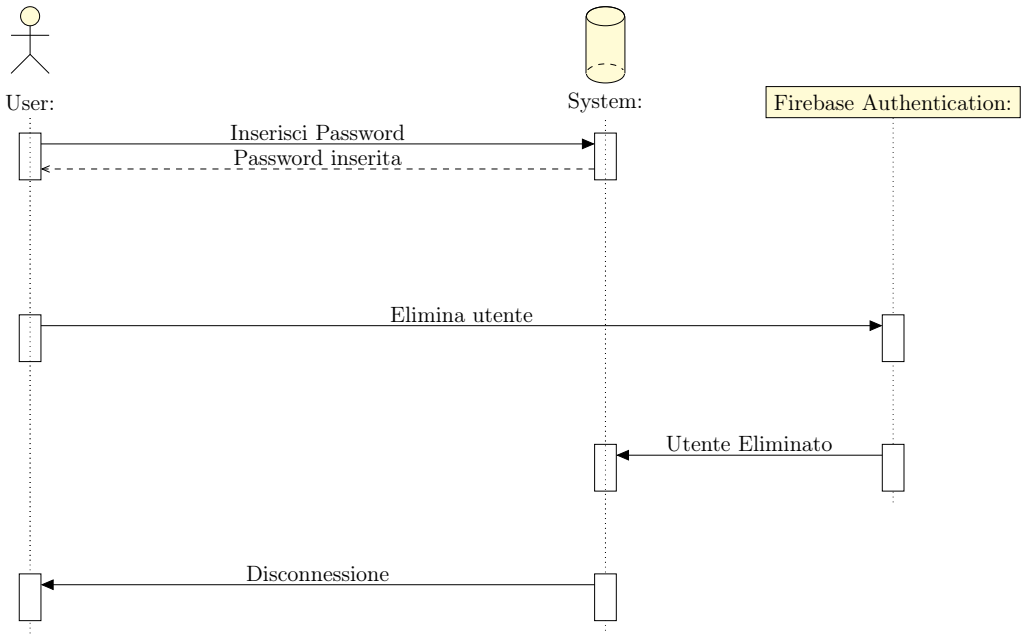
2.4.5 UC05: Log Out Utente

Use Case: Log Out utente	
Componente	**Descrizione**
ID	UC05
Attore Primario	Utente Generico
Attore Finale	Utente Generico
Pre Condizioni	L'utente è registrato e autenticato e si trova nella pagina "Profile" e desidera disconnettersi dall'applicazione
Post Condizioni	Dopo che l'utente ha eseguito la disconnessione dall'applicazione, si ritroverà nella pagina relativa al Log in
Scenario di Successo	1. L'utente si trova nella pagina Profile 2. L'utente preme nel menù la voce *"Sign Out"* per disconnettersi 3. Si aprirà un Dialog che chiederà all'utente se vuole disconnettersi 4. L'utente preme il bottone *"Confirm"* per effettuare la disconnessione 5. L'utente viene direzionato nella pagina Log In
Estensioni:	2a - 3a L'utente preme il bottone *"Cancel"* nel dialog di conferma disconnessione. La disconnessione non viene effettuata e l'utente ritorna nella pagina Profile.



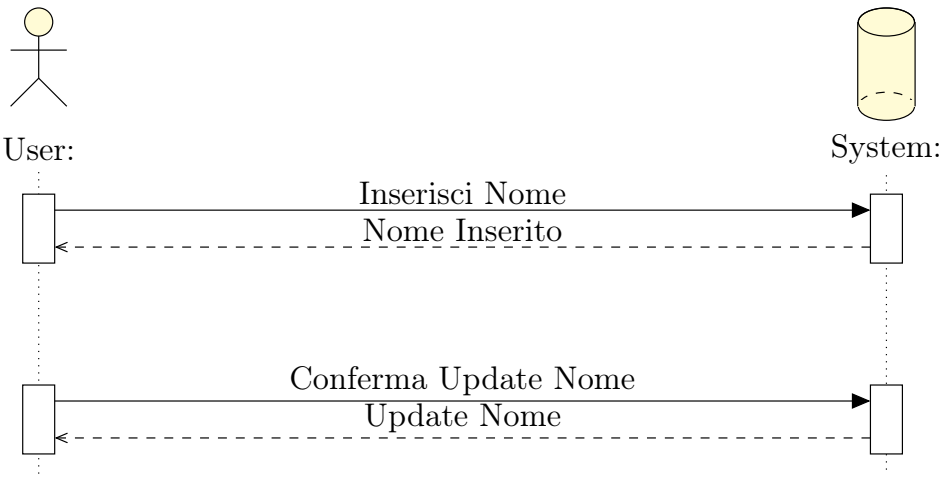
2.4.6 UC06: Eliminazione Utente

Use Case: Eliminazione Utente	
Componente	**Descrizione**
ID	UC06
Attore Primario	Utente Generico
Attore Finale	Utente Generico
Pre Condizioni	L'utente è registrato e autenticato e si trova nella pagina Profile, desidera eliminare il proprio profilo
Post Condizioni	L'utente si trova nella pagina Log in, alla quale potrà accedere solo dopo aver registrato un nuovo account.
Scenario di Successo	1. L'utente si trova nella pagina Profile, seleziona la voce *"Delete Account"* 2. Si aprirà un Dialog che chiederà all'utente la conferma dell'eliminazione dell' account 3. l'utente dovrà inserire un'ultima volta la propria password, per questioni di sicurezza 4. L'utente Preme sul bottone *"Delete Account"* 5. L'utente viene ridirezionato alla pagina Log in
Estensioni:	4a L'utente preme il bottone *"Delete Account"* senza aver inserito la password. L'applicazione richiede di inserire la password. 3a. L'utente inserisce una password errata. L'applicazione notifica che la password è errata e richiede di inserire la password corretta.



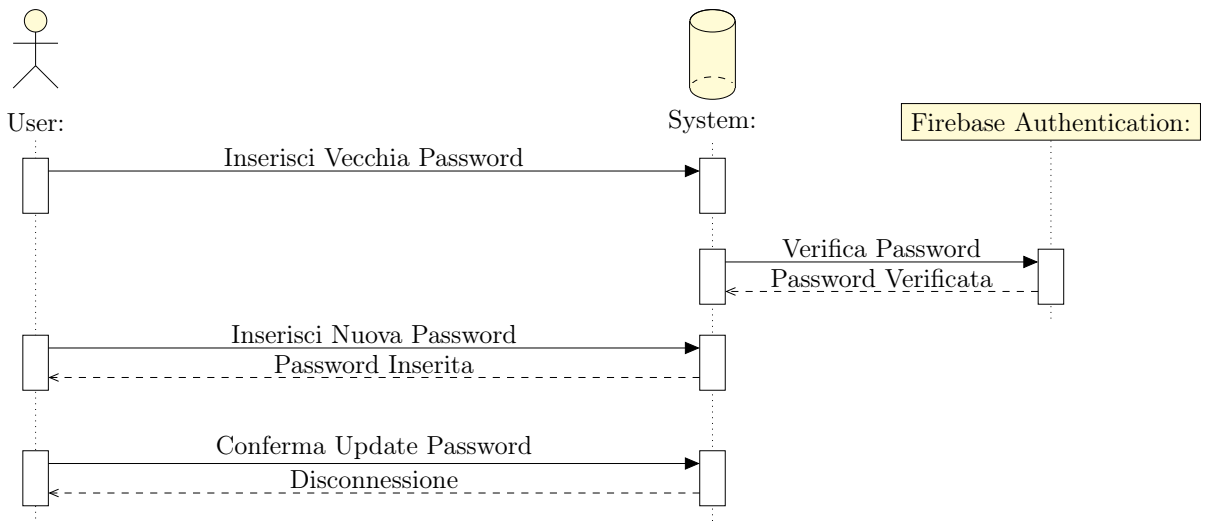
2.4.7 UC07 Update Nome Utente

Use Case: Update Nome Utente	
Componente	**Descrizione**
ID	UC07
Attore Primario	Utente Generico
Attore Finale	Utente Generico
Pre Condizioni	L'utente è registrato e autenticato e si trova nella pagina Profile, desidera modificare il proprio nome utente
Post Condizioni	L'utente si trova nella pagina Profile dopo aver modificato il nome utente. La modifica verrà visualizzata a schermo nella pagina Profile ed influenzerà l'utente Studente ed eventualmente l'utente Tutor
Scenario di Successo	1. L'utente si trova nella pagina Profile, seleziona la voce *"Name"* 2. Si aprirà un Dialog che chiederà all'utente di modificare il nome utente 3. L'utente inserisce il proprio nome, rispettando i criteri nell'apposito campo di testo 4. L'utente Preme sul bottone *"Confirm"* 5. L'utente visualizzerà a schermo le modifiche apportate
Estensioni:	4a L'utente prova a modificare il nome senza inserire nulla. L'applicazione richiede di inserire un nome valido e rimane nel dialog. 3a. L'utente inserisce un nome non valido. L'applicazione invita a inserire un nome valido e rimane nel dialog. 5a. L'utente inserisce un nome valido / lascia il campo di testo vuoto, ma preme *"Cancel"*. L'utente viene riportato alla pagina Profile senza modifiche.



2.4.8 UC08: Update Password Utente

Use Case: Update Password Utente	
Componente	**Descrizione**
ID	UC08
Attore Primario	Utente Generico
Attore Finale	Utente Generico
Pre Condizioni	L'utente è registrato e autenticato e si trova nella pagina Profile, desidera modificare la propria password
Post Condizioni	L'utente si trova nella pagina Login dopo aver modificato la password. L'utente per accedere nuovamente all'applicazione dovrà utilizzare le nuove credenziali
Scenario di Successo	1. L'utente si trova nella pagina Profile e sceglie nel menù la voce *"Password"* 2. Si aprirà un Dialog che chiederà all'utente di modificare la password 3. L'utente inserisce la vecchia password nel campo di testo *"Insert old password"* 4. L'utente inserisce la nuova password nel campo di testo *"Insert new password"* 5. L'utente preme il tasto *"Confirm"* e viene disconnesso
Estensioni:	4a L'utente inserisce la vecchia password anche nel campo della nuova password. L'applicazione richiede una nuova password diversa dalla vecchia. 4b - 5a. L'utente lascia vuoto il campo della vecchia password e inserisce una nuova password. L'applicazione richiede di inserire la vecchia password.



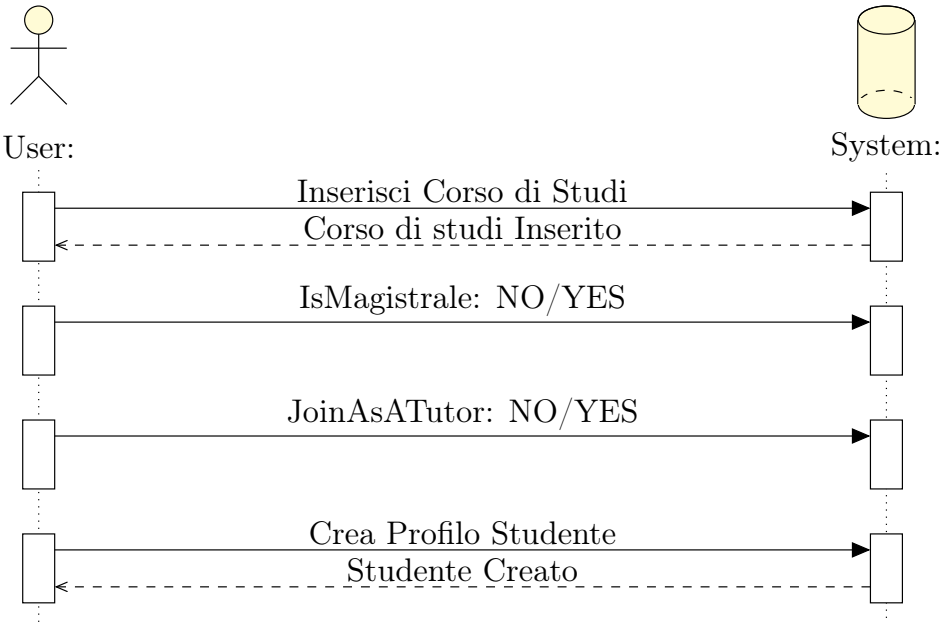
2.4.9 UC09: Update Foto Utente

Use Case: Update Foto Utente	
Componente	**Descrizione**
ID	UC09
Attore Primario	Utente Generico
Attore Finale	Utente Generico
Pre Condizioni	L'utente è registrato e autenticato e si trova nella pagina Profile, desidera modificare la propria foto profilo
Post Condizioni	L'utente si trova nella pagina Profile dopo aver modificato la propria foto: i risultati dell'aggiornamento verranno visualizzati a schermo, verrà mostrata la foto aggiornata
Scenario di Successo	1. L'utente si trova nella pagina Profile e preme sull'icona Profilo sopra il nome utente 2. Si aprirà il gestore foto del dispositivo 3. L'utente sceglierà la foto da associare al proprio profilo 4. L'utente inserisce la nuova password nel campo di testo 5. La modifica della foto verrà visualizzata a schermo: al posto dell'icona profilo sarà presente la foto utente
Estensioni:	Nessuna

2.4.10 UC10: Completa Profilo Studente

Use Case: Completa Profilo Studente	
Componente	**Descrizione**
ID	UC10
Attore Primario	Utente Generico
Attore Finale	Utente Generico / Studente
Pre Condizioni	L'utente è registrato e autenticato e si trova nella pagina Profile, desidera creare il proprio profilo studente
Post Condizioni	Dopo aver completato la creazione del profilo studente e se l'utente ha risposto *"NO"* alla domanda *"Join as a Tutor"*, l'utente verrà indirizzato alla pagina del profilo studente, dove vedrà l'opzione *"Update Student Profile"* al posto di *"Complete Student Profile"*. Se l'utente ha invece risposto *"YES"* alla domanda *"Join as a Tutor"*, sarà reindirizzato alla pagina *"Join as a Tutor"* per completare la creazione del profilo Tutor, ma solo al momento della prima creazione.
Scenario di Successo	1. L'utente si trova nella pagina Profile e sceglie nel menù la voce *"Complete Student Profile"* 2. L'utente inserisce il proprio corso di studi nel campo di testo *"Study Program"* 3. L'utente spunta la casella *"Master's degree"* se sta frequentando un corso di laurea magistrale. 4. L'utente seleziona una risposta alla domanda "Join as a Tutor": • L'utente sceglie NO (a) L'utente preme il tasto *"Create Student Profile"* (b) L'utente viene direzionato alla pagina Profile, dove vedrà a schermo *"Update Student Profile"*, a significare che la creazione del profilo studente è avvenuta con successo. • L'utente sceglie YES (a) L'utente preme il tasto *"Create Student Profile"* (b) L'utente viene direzionato alla pagina per la creazione del profilo Tutor.

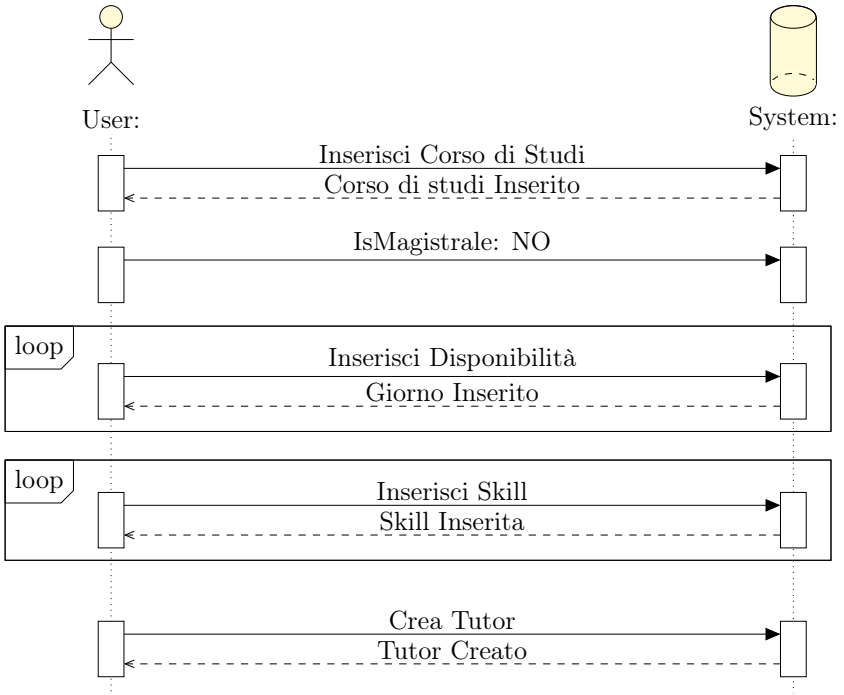
Estensioni:	1a L'utente non inserisce nessun corso di studi nel campo di testo *"Study Program"*. L'applicazione notifica l'utente di inserire un corso di studi valido. 1b L'utente inserisce un corso di studi errato. L'applicazione notifica l'utente di inserire un corso di studi valido. 4a. L'utente non seleziona una risposta alla domanda *"Join as a Tutor"*. L'applicazione notifica l'utente di fare una scelta obbligatoria. 6a. L'utente preme il tasto *"Back to Profile"* durante la creazione del profilo. Il profilo studente non viene creato e l'utente torna alla pagina Profile con *"Complete Student Profile"* nel menù.



2.4.11 UC11: Completa Profilo Tutor

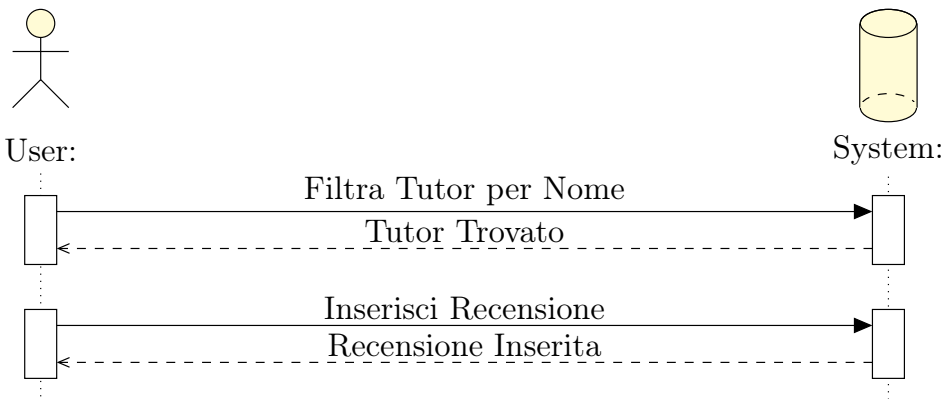
Use Case: Completa Profilo Tutor	
Componente	**Descrizione**
ID	UC11
Attore Primario	Utente Generico
Attore Finale	Utente Generico / Tutor
Pre Condizioni	L'utente è registrato e autenticato e si trova nella pagina Profile, desidera creare il proprio profilo tutor. Il profilo Tutor può essere creato solo se è stato precedentemente creato il profilo Studente
Post Condizioni	Dopo aver completato la creazione del profilo tutor, l'utente si troverà nella pagina Profile. La creazione del profilo Tutor non verrà notificata a schermo come per la creazione del profilo Studente

Scenario di Successo	1. L'utente si trova nella pagina Profile e sceglie nel menù la voce *"Join as a tutor"* 2. L'utente potrà modificare il corso di studi salvato durante la fase di completamento profilo studente 3. L'utente deve scegliere almeno un giorno della settimana per fornire lezioni di tutoring. 4. L'utente nel campo di testo *"Input your top skills"* potrà inserire gli argomenti su cui è più preparato. Gli argomenti verranno salvati tramite il pulsante *"Add"* (questa operazione non è obbligatoria) 5. L'utente preme il tasto *"Create Tutor Profile"* e ritorna nella pagina Profile
Estensioni:	3a L'utente non seleziona nessun giorno della settimana per fornire lezioni di tutoring. L'applicazione notifica l'utente di selezionare almeno un giorno. 2a L'utente inserisce un corso di studi errato nel campo di testo *"Study Program"*. L'applicazione notifica l'utente di inserire un corso di studi valido. 4a. L'utente preme il tasto *"Add"* senza inserire alcuna skill nel campo di testo *"Input your top skills"*. L'applicazione notifica l'utente di inserire un argomento per aggiungere una skill. 5a. L'utente preme il tasto *"Back to Profile"* durante la creazione del profilo tutor. Il profilo Tutor non viene creato e l'utente torna alla pagina Profile



2.4.12 UC12: Inserisci Recensione

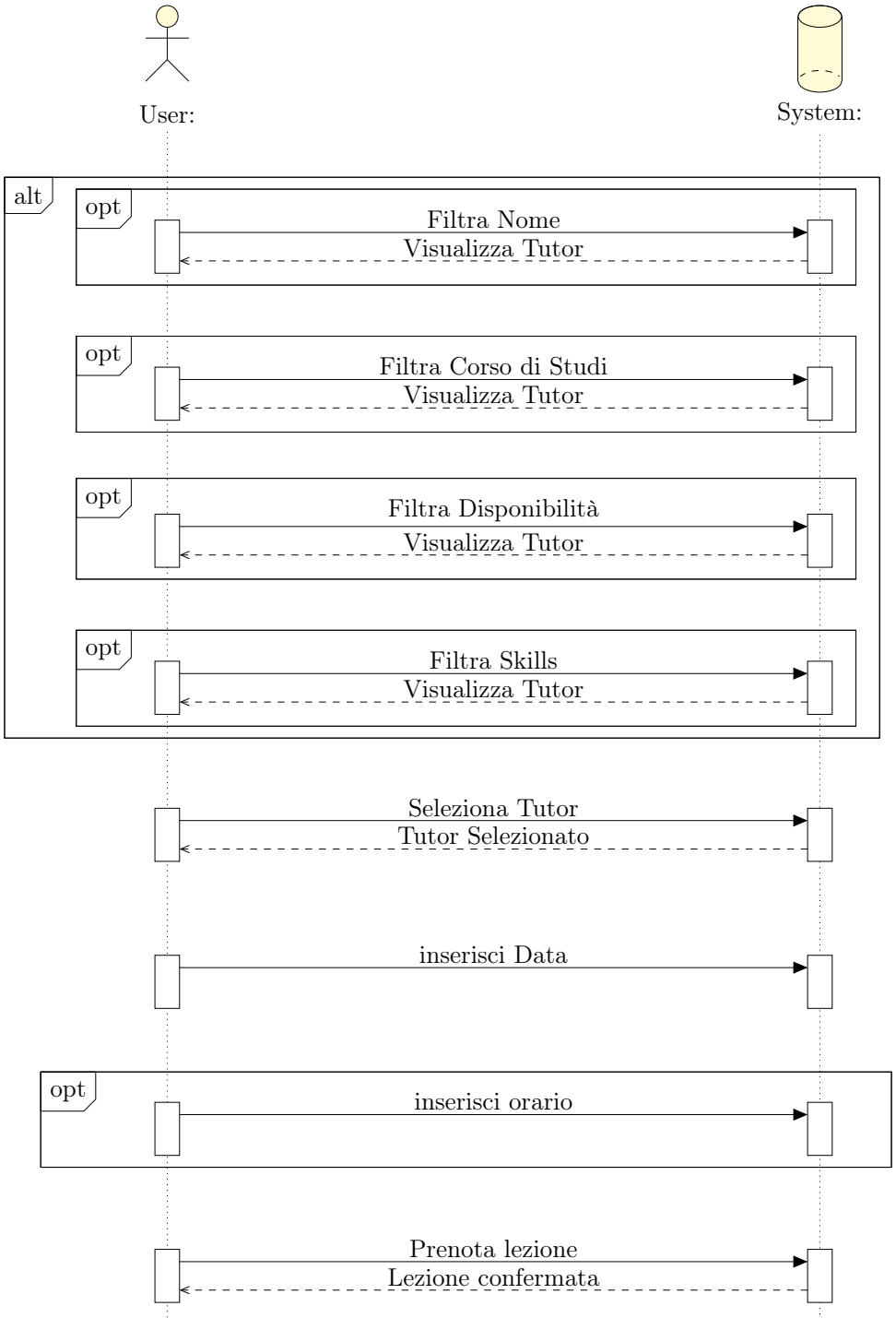
Use Case: Inserisci Recensione	
Componente	**Descrizione**
ID	UC12
Attore Primario	Utente Studente
Attore Finale	Utente Studente
Pre Condizioni	L'utente è registrato e autenticato ha completato il profilo studente e si trova nella pagina Profile, vuole recensire un tutor
Post Condizioni	L'utente si trova nella pagina *"Review your Tutor"* dopo aver inserito una recensione. L'utente potrà continuare a inserire quante recensioni vuole, può però inserire una sola recensione per Tutor
Scenario di Successo	1. L'utente si trova nella pagina Profile e sceglie nel menù la voce *"Review"* 2. L'utente inserisce nel campo di testo *"Name"* un nome valido di un tutor 3. L'utente seleziona quante stelle vuole assegnare al tutor (da 0 a 5) 4. L'utente preme sul bottone *"Make a review"* 5. L'applicazione mostrerà a schermo la recensione aggiunta, mostrando nome del tutor e valutazione assegnata
Estensioni:	2a L'utente non inserisce nessun nome o inserisce un nome errato di un tutor. L'applicazione notifica l'utente di inserire un nome valido. 5a L'utente preme il bottone *"Back to profile"* invece di completare la recensione. L'utente viene direzionato alla pagina Profile e la recensione non viene aggiunta.



2.4.13 UC13: Prenota Lezione

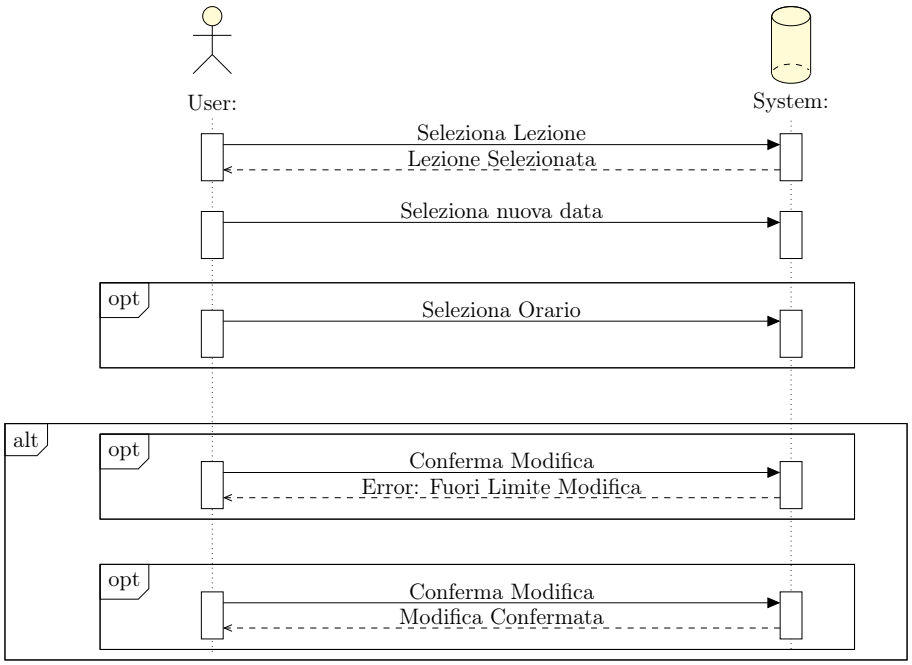
Use Case: Prenota Lezione	
Componente	**Descrizione**
ID	UC13
Attore Primario	Utente Studente
Attore Finale	Utente Studente
Pre Condizioni	Uno studente deve essersi autenticato all'interno dell'applicazione, completato il proprio profilo e aver scelto l'insegnante con il quale prenotare una lezione di tutoring. La lezione con l'insegnante scelto potrà essere prenotata solo se ci sarà disponibilità nelle date e orari scelti dallo studente
Post Condizioni	Dopo che lo studente ha prenotato la lezione le informazioni sulla prenotazione verranno registrate nel sistema, includendo dettagli come: • Data • Ora • Nome Lezione • Nome Tutor E tutte le informazioni necessarie. Lo studente potrà vedere la lezione prenotata nella home page o nella sezione *"Your lesson"* nella pagina Profile
Scenario di Successo	1. L'utente vuole prenotare una lezione di tutoring, si sposta nella pagina dell'applicazione dedicata alla ricerca dell'insegnante 2. Lo studente potrà filtrare la ricerca inserendo nel campo *"search"* il nome (di default) oppure inserendo il corso di studi, la disponibilità o le skill selezionando l'apposito bottone 3. L'utente seleziona il tutor 4. L'utente seleziona una data dal calendario, premendo il bottone *"Select your day"* 5. L'utente selezionerà un orario disponibile per quel determinato giorno 6. L'utente preme il bottone *"Book your class"*, verrà direzionato nella Home Page, dove visualizzerà il riepilogo della lezione prenotata

Estensioni:	5a L'utente non seleziona un orario dopo aver selezionato la data. L'applicazione avvisa l'utente di selezionare uno slot orario 4a L'utente non seleziona una data prima di prenotare la lezione. L'applicazione avvisa l'utente di selezionare sia la data che l'orario per prenotare una lezione



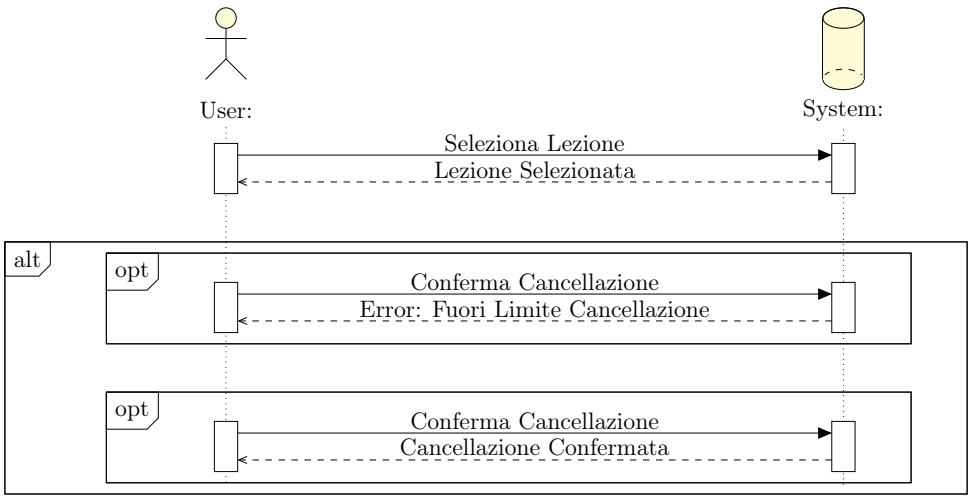
2.4.14 UC14: Modifica Lezione

Use Case: Modifica Lezione	
Componente	**Descrizione**
ID	UC14
Attore Primario	Utente Studente
Attore Finale	Utente Studente
Pre Condizioni	Affinché una lezione possa essere modificata, l'utente studente deve averla prenotata. L'utente studente può cambiare a suo piacimento data, orario, argomenti... sempre rispettando le disponibilità dell'insegnante e effettuando la modifica entro 1 giorno dalla lezione.
Post Condizioni	L'utente dopo aver modificato la lezione visualizzerà i dati aggiornati della lezione nella Home Page
Scenario di Successo	1. L'utente seleziona nella home page la Lezione da modificare 2. L'utente preme il bottone *"Select your day"* per selezionare il nuovo giorno 3. L'utente seleziona un orario tra quelli disponibili per il giorno selezionato 4. L'utente preme il bottone *"Reschedule"* e modifica la lezione
Estensioni:	3a L'utente non seleziona un orario disponibile. L'applicazione informa l'utente che deve selezionare un orario per completare la riprogrammazione. 2a L'utente non seleziona un nuovo giorno. L'applicazione informa l'utente di selezionare sia una data che un orario per riprogrammare la lezione.



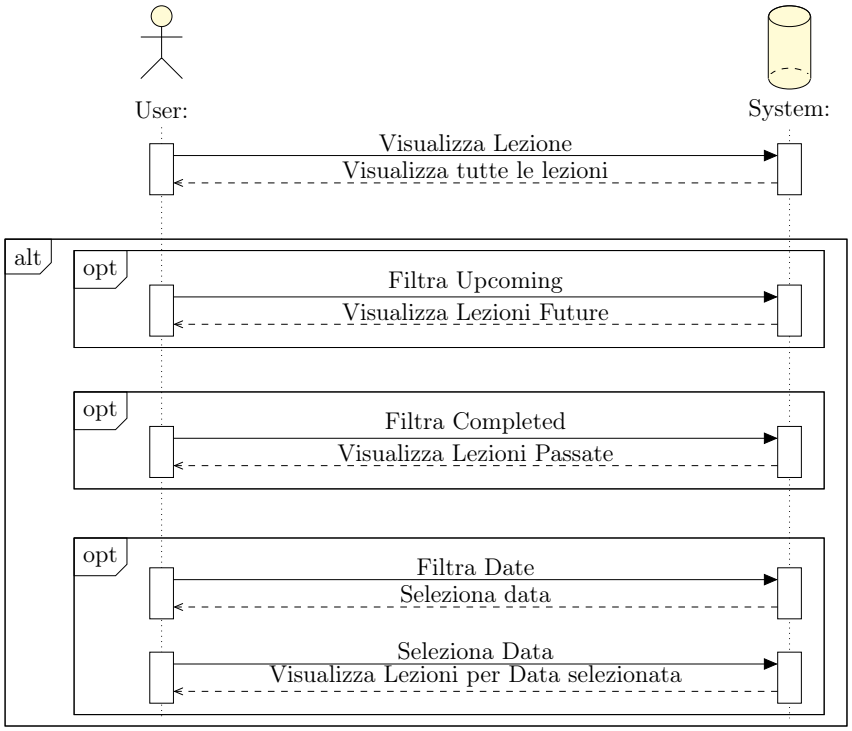
2.4.15 UC15: Cancella Lezione

Use Case: Cancella Lezione	
Componente	**Descrizione**
ID	UC15
Attore Primario	Utente Studente
Attore Finale	Utente Studente
Pre Condizioni	L'utente che vuole effettuare la cancellazione deve essere autenticato e deve esistere una lezione prenotata. L'utente deve selezionare la lezione nella home page e cancellarla. La cancellazione della lezione deve avvenire entro il periodo consentito dalle politiche dell'applicazione (1 giorno prima)
Post Condizioni	Una volta che l'utente ha cancellato la lezione, non verrà più mostrata nella home Page e nella sezione *"Your Lessons"* della pagina Profile
Scenario di Successo	1. L'utente si trova nella Home Page 2. L'utente seleziona la lezione che vuole cancellare, tra quelle prenotate 3. L'utente preme il bottone *"Cancel"* e cancella la lezione (se rispetta i limiti imposti dall'applicazione) 4. Il sistema aggiorna ed elimina la lezione
Estensioni:	3a L'utente tenta di cancellare la prenotazione oltre il limite di tempo consentito (1 giorno prima della lezione). L'applicazione avverte l'utente che non è possibile effettuare la cancellazione.



2.4.16 UC16: Filtra Lezione

Use Case: Filtra Lezione	
Componente	**Descrizione**
ID	UC16
Attore Primario	Utente Studente
Attore Finale	Utente Studente
Pre Condizioni	L'utente che vuole visualizzare una lista delle lezioni future e passate si trova nella pagina Profile, le visualizzerà selezionando nel menù la voce *"Your Lessons"*.
Post Condizioni	Una volta che l'utente ha selezionato la voce *"Your Lessons"* potrà visualizzare la lista di tutte le lezioni, filtrandole se necessario
Scenario di Successo	1. Lo studente si trova nella pagina Profile e seleziona la voce *"Your Lessons"* 2. Nella sezione *"Your Lessons"* vedrà di default la lista di tutte le lezioni, partendo dalla data più lontana 3. Inserendo il nome del tutor nella casella di testo e premendo il bottone *"Filter"*, può filtrare la lista per il nome tutor 4. L'utente può filtrare per Data, lezioni future e passate, premendo l'apposito bottone
Estensioni:	3a L'utente tenta di filtrare la lista senza inserire alcun nome nella casella di testo. Il sistema resetta i filtri e torna alla visualizzazione di default.



3 Implementazione

Il capitolo seguente descrive l'implementazione dell'applicazione BicoccaHelp, con particolare attenzione all'architettura software adottata e alle tecnologie integrate per realizzare le principali funzionalità (*Architettura*). Verranno esaminati nel dettaglio anche gli aspetti legati all'interfaccia utente(*UI*), con l'obiettivo di evidenziare come sia stata progettata per offrire agli studenti una UI intuitiva, garantendo facilità d'uso ed efficienza.

L'architettura adottata per lo sviluppo è **Model - View - View Model** (*MVVM*), scelta per separare i componenti dell' app e facilitare la gestione dei dati, della logica di business e delle interfacce utente.

Le tecnologie principali utilizzate includono:

- **Firebase Authentication:** Gestione dell'Autenticazione
- **Firestore Database:** Gestione dei Dati
- **Firebase Storage:** Gestione risorse multimediali
- **Glide:** Gestione risorse multimediali

Queste tecnologie verranno descritte più approfonditamente nel capitolo

3.1 Architettura

Il pattern architetturale **MVVM** facilita una separazione della logica di business dell'applicazione dall'interfaccia utente. Questa netta separazione consente di gestire lo sviluppo, semplifica la fase di testing e l'evoluzione dell'applicazione. Un altro vantaggio dell'architettura MVVM è la possibilità di modifica e riutilizzo del codice nel futuro.

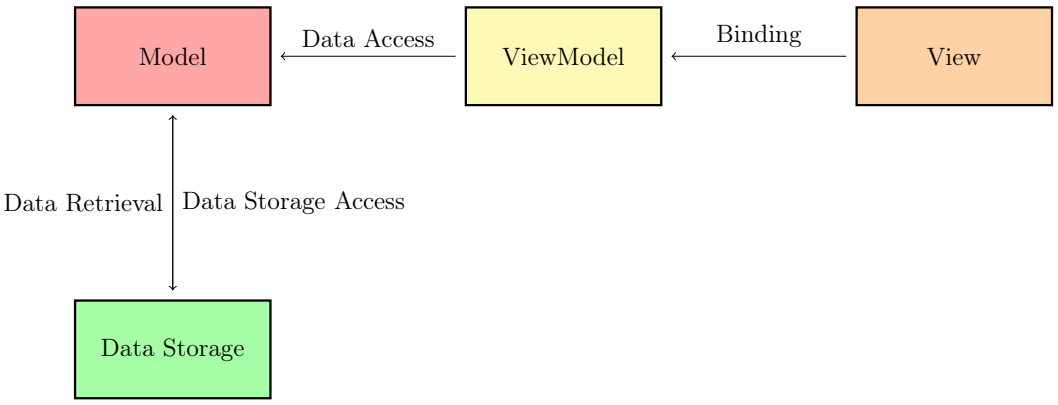


Figura 1: MVVM Architecture

I tre componenti hanno uno scopo ben preciso e distinto:

- **View:** Si occupa della gestione della UI; condivide il suo stato attraverso un *Lifecycle* e osserva i *LiveData* dal View Model per aggiornare l'interfaccia utente
- **View Model:** Mantiene i dati che popolano l'interfaccia utente e gestisce la logica. Si interfaccia con il Model per l'accesso ai dati. È il mezzo di comunicazione fra Fragment.
- **Model:** composto da *Repository* e *RemoteDataSource*, si occupa di recuperare i dati

È importante comprendere come i componenti interagiscono: la View *"conosce"* il View Model, il quale *"conosce"* il Model. Il Model non è a conoscenza del View Model e il View Model non è a conoscenza della View.

I vantaggi dell'uso del pattern architetturale MVVM sono molteplici:

- Il View Model ha la funzione di adattatore per le classi del Model, impedendo di apportare modifiche importanti al codice del modello
- l'UI dell'applicazione può essere riprogettata senza modificare il codice del Model
- È possibile lavorare in modo indipendente e simultaneo sui componenti

Per utilizzare efficacemente il pattern MVVM, è fondamentale capire come organizzare il codice nelle classi appropriate e definire chiaramente le interazioni tra di esse. [4]

L'applicazione che ho sviluppato è basata sul pattern MVVM ed è organizzata in due macrostrutture:

- **Data:** Questa sezione gestisce l'interazione con i dati dell'applicazione, occupandosi del recupero, della manipolazione e della memorizzazione delle informazioni necessarie al funzionamento delle funzionalità principali.
- **UI:** La parte UI raggruppa le principali aree dell'interfaccia utente, suddividendo le responsabilità tra la View, che gestisce la presentazione dei dati, e il ViewModel, che contiene la logica dell'applicazione necessaria per interagire con i dati e aggiorna la View di conseguenza.

3.1.1 Data

Nell'architettura dell'applicazione, la gestione dei dati è effettuata tramite Repository e RemoteDataSource. Questi componenti interagiscono direttamente con Firestore DB per operazioni di lettura, scrittura e aggiornamento dei dati in tempo reale. I Repository gestiscono l'accesso ai dati e le operazioni sui modelli, mentre i RemoteDataSource si occupano della comunicazione con Firestore DB. Questo approccio consente di mantenere una struttura chiara e modulare, assicurando che le informazioni siano sempre aggiornate e coerenti con l'interfaccia utente.
Il package Data è suddiviso in vari sotto-package per organizzare e gestire in modo efficiente le diverse funzionalità e i dati dell'applicazione. Questa suddivisione consente di isolare

e separare le responsabilità legate alla gestione dei dati, facilitando la manutenzione e l'estensibilità del codice. Ogni *package* si occupa di specifiche entità dell'applicazione, come l'autenticazione, i corsi di studio, le lezioni e le recensioni, assicurando che le operazioni di accesso e manipolazione dei dati siano gestite in modo modulare e coerente.

- **auth:** Gestisce l'autenticazione degli utenti. Include `AuthRemoteDataSource`, che interagisce con `FirebaseAuth` per l'autenticazione, e `AuthRepository`, che funge da intermediario tra l'app e `AuthRemoteDataSource`.
- **corsoDiStudi:** Gestisce i corsi di studio. Comprende `CorsoDiStudiModel`, un modello dati per i corsi di studio, `CreateCorsoDiStudiRequest`, una richiesta per creare un corso di studio, `CorsoDiStudiRemoteDataSource`, che interagisce con Firestore per i corsi di studio, e `CorsoDiStudiRepository`, un intermediario tra l'app e `CorsoDiStudiRemoteDataSource`.
- **date:** Gestisce date e orari dei tutor. Include `CreateDateRequest`, una richiesta per creare una data, `DateModel`, un modello per le date esistenti, `DateRemoteDataSource`, che interagisce con Firestore per le date e gli orari, e `DateRepository`, un intermediario tra l'app e `DateRemoteDataSource`.
- **lesson:** Gestisce le lezioni. Comprende `CreateLessonRequest`, una richiesta per creare una lezione, `LessonModel`, un modello per le lezioni esistenti, `LessonRemoteDataSource`, che interagisce con Firestore per le lezioni, e `LessonRepository`, un intermediario tra l'app e `LessonRemoteDataSource`.
- **review:** Gestisce le recensioni dei tutor. Include `CreateReviewRequest`, una richiesta per creare una recensione, `ReviewModel`, un modello per le recensioni, `ReviewRemoteDataSource`, che interagisce con Firestore per le recensioni e calcola la recensione media, e `ReviewRepository`, un intermediario tra l'app e `ReviewRemoteDataSource`.
- **User:** Gestisce le informazioni degli utenti. Comprende `UserModel`, un modello dati per un utente, `UserRemoteDataSource`, che interagisce con `FirebaseAuth` per operazioni come recuperare l'utente corrente e aggiornare nome e foto, `UserAssetsRemoteDataSource`, che gestisce l'upload delle risorse come le foto del profilo utilizzando `FirebaseStorage`, e `UserRepository`, un intermediario tra l'app e `UserRemoteDataSource` e `UserAssetsRemoteDataSource`.
- **Tutor:** Gestisce le informazioni sui tutor. Include `CreateTutorRequest`, una richiesta per creare un tutor, `TutorModel`, un modello dati per un tutor, `TutorRemoteDataSource`, che interagisce con Firestore per i tutor, e `TutorRepository`, un intermediario tra l'app e `TutorRemoteDataSource`.
- **Student:** Gestisce le informazioni sugli studenti. Comprende `CreateStudentRequest`, una richiesta per la creazione di uno studente, `StudentModel`, un modello dati per uno studente, `StudentRemoteDataSource`, che interagisce con Firestore per gli studenti, e `StudentRepository`, un intermediario tra l'app e `StudentRemoteDataSource`.

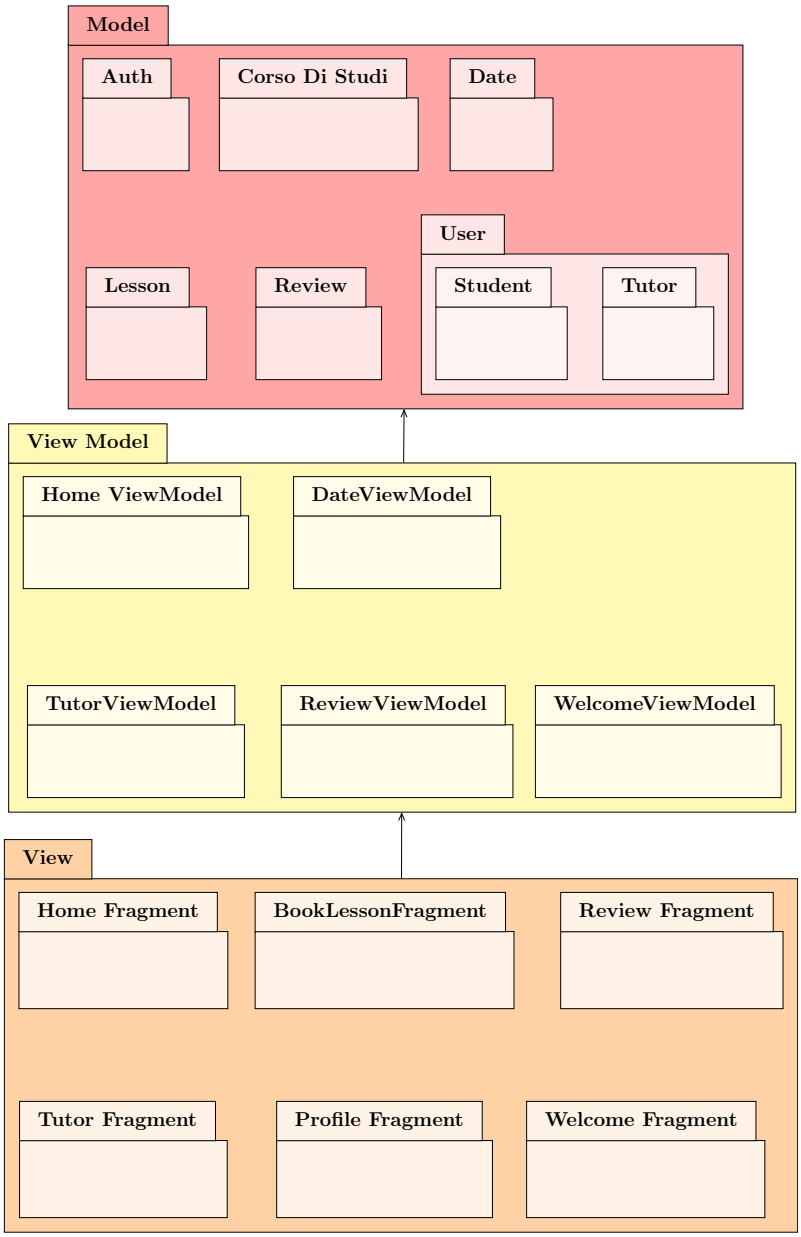
3.1.2 UI

Il package `UI` gestisce l'interfaccia utente dell'applicazione e la logica di business associata. Questo package si occupa della visualizzazione dei dati attraverso i `Fragment` e della gestione della logica applicativa tramite i `ViewModel`. È suddiviso in sotto-package che corrispondono alle diverse pagine dell'applicazione: `home`, `lessonBooking`, `profile`, `review` e `welcome`.

Il package `UI` si occupa della gestione dell'interfaccia utente e della logica di business attraverso l'uso dei ViewModel e dei Fragment. Il package è suddiviso in diverse aree:

- **Home:** Gestisce l'interfaccia della pagina principale dell'app. Include i fragment per la visualizzazione (tra cui `HomeFragment`) e i View Model per la logica di business, `HomeViewModel`.
- **LessonBooking:** Si occupa della prenotazione delle lezioni. Comprende fragment per l'interfaccia utente (tra cui: `BookLessonFragment` e `TutorFragment`)e view model per gestire le interazioni con i dati e le chiamate ai repository: `DateViewModel` e `TutorViewModel`
- **Profile:** Gestisce le informazioni del profilo utente. Include fragment per visualizzare e aggiornare le informazioni dell'utente (tra cui: `DeleteUserDialogFragment`, `LogoutDialogFragment`, `ProfileFragment`, `ProfilePhotoFragment`, `UpdateNameDialogFragment` e `UpdatePasswordDialogFragment`), i fragment per completare il profilo student e utente (`CompleteStudentProfileFragment` e `CompleteTutorProfileFragment`) e il fragment `LessonFragment` per visualizzare lo storico lezioni. Ogni fragment è associato al rispettivo View Model per la logica di interazione con i dati: `CompleteProfileViewModel`, `LessonViewModel` e `ProfileViewModel`.
- **Review:** Si occupa della gestione delle recensioni. Comprende fragment per la visualizzazione delle recensioni (`ReviewFragment`) e View Model per gestire le operazioni relative alle recensioni (`ReviewViewModel`)
- **Welcome:** Gestisce le schermate di benvenuto e introduzione all'app. Include fragment per le schermate di benvenuto (`RegistrationFragment`, `LoginFragment`, `ForgotPasswordFragment`, `VerifyEmailFragment`) e View Model per l'inizializzazione e configurazione dell'app (`WelcomeViewModel`).

Di seguito è presentato il modello architetturale dell'applicazione, suddiviso nei *package*, Fragment e View Model appena descritti:



3.2 Gestione dell'Autenticazione

In questa sezione viene descritta l'implementazione del sistema di autenticazione dell'applicazione, che consente agli utenti di registrarsi, effettuare il login, recuperare la password ed effettuare il logout. L'integrazione è stata realizzata tramite il servizio Firebase Authentication, che consente di gestire gli utenti in modo semplice e sicuro.

3.2.1 Firebase Authentication

Firebase Authentication fornisce un sistema di gestione degli utenti che consente l'autenticazione tramite email e password, ma anche tramite provider esterni come Google,

Facebook, e altri. Tuttavia, nell'applicazione BicoccaHelp è stata abilitata l'autenticazione solo tramite email e password, rendendo il processo di registrazione più selettivo attraverso l'uso esclusivo di email con dominio `@campus.unimib.it`. Questo garantisce che solo gli studenti dell'Università di Milano-Bicocca possano accedere al servizio.

L'utilizzo dell'API di Firebase semplifica la gestione del ciclo di vita dell'utente, dal momento della registrazione fino al logout. Firebase si occupa in modo sicuro della gestione delle credenziali, e grazie ai metodi forniti, operazioni come il login, la creazione di nuovi utenti, il reset della password e il logout vengono gestite con poche righe di codice. La gestione dell'autenticazione utente e l'interazione con Firebase è stata gestita nei Remote Data Source `AuthRemoteDataSource` e `UserRemoteDataSource`, che a loro volta comunicano con le Repository `AuthRepository` e `UserRepository`.

I seguenti metodi sono responsabili della registrazione utente e del Login utente. Consentono di autenticare e registrare gli utenti tramite email e password, interagendo direttamente con Firebase attraverso l'API di FirebaseAuth. Questi metodi sono stati inclusi come esempi per illustrare come l'applicazione interagisce con Firebase per le operazioni di autenticazione

```
public void register(String email, String password, Callback<Void> callback){
    auth.createUserWithEmailAndPassword(email, password)
        .addOnSuccessListener(authResult -> callback.onSucces(null))
        .addOnFailureListener(callback::onFailure);
}
```

La chiamata a `createUserWithEmailAndPassword` di Firebase, invia una richiesta ai server per creare un nuovo account con le credenziali fornite. Se l'operazione va a buon fine, Firebase ritorna un oggetto `Authresult` che rappresenta l'utente autenticato. Non essendo richiesto un risultato, in caso di successo viene invocato il metodo `addOnSuccessListener` che esegue il codice `authResult -> callback.onSucces(null)`. Il fallimento viene gestito tramite il metodo `addOnFailureListener` che intercetta l'errore e chiama `callback::onFailure`.

Il seguente metodo gestisce l'autenticazione dell'utente già registrato:

```
public void login(String email, String password, Callback<Void> callback) {
    auth.signInWithEmailAndPassword(email, password)
        .addOnSuccessListener(authResult -> callback.onSucces(null))
        .addOnFailureListener(callback::onFailure);
}
```

Viene chiamato il metodo `signInWithEmailAndPassword`, che invia una richiesta a Firebase per autenticare l'utente con le credenziali fornite. In caso di autenticazione riuscita, viene eseguita `authResult -> callback.onSucces(null)`, che chiama il metodo `onSucces` del callback, segnalando che il login è andato a buon fine. Se le credenziali invece non sono corrette o non esistono account associati alla email fornita, l'errore viene catturato da `addOnFailureListener` e gestito da `callback::onFailure`.

3.3 Gestione dei Dati e dei File

3.3.1 Panoramica DB

Per l'applicazione BicoccaHelp è stato scelto Firestore DB, un database NoSQL basato su documenti. I dati sono organizzati in **documenti**, che a loro volta sono raccolti in **collezioni**. Ogni documento, identificato da un ID univoco, è costituito da coppie chiave-valore e può includere sottocollezioni o oggetti annidati. Anche se Firestore è privo di uno schema fisso, nell'applicazione si è deciso di mantenere una struttura uniforme dei campi all'interno di ogni documento di una collezione, così da semplificare l'esecuzione delle query. Firestore gestisce automaticamente la creazione di documenti e collezioni: quando si assegna un nuovo dato, la piattaforma crea le relative entità se non esistono già.

Ho scelto Firestore DB per la sua flessibilità e scalabilità e perché offre un' integrazione nativa con Firebase Authentication e altre soluzioni Firebase già utilizzate nell'app, semplificando la gestione dei dati utente. [5]

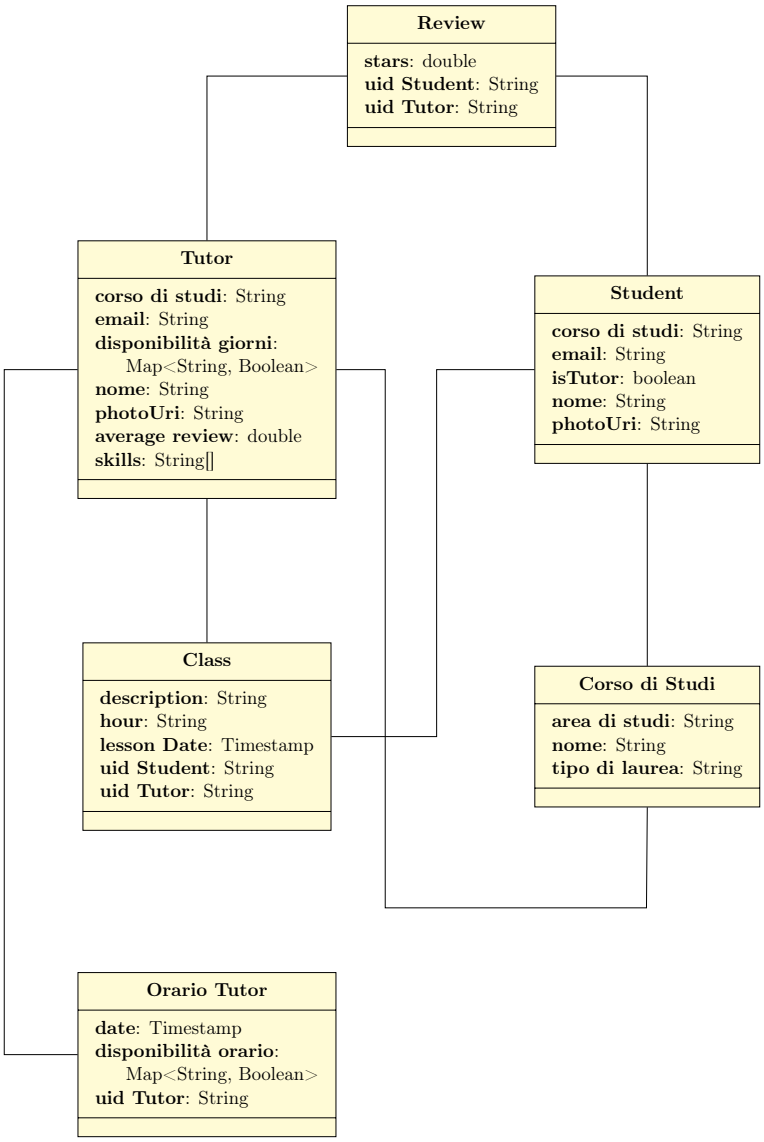
3.3.2 Struttura dei dati

Il database **Firestore** utilizzato per l'applicazione BicoccaHelp è organizzato in collezioni che rappresentano le principali entità dell'applicazione: lezioni, recensioni, studenti, tutor, corsi di studi e disponibilità oraria dei tutor. Ciascuna collezione contiene documenti che rappresentano i singoli record di queste entità, ciascuno caratterizzato da un insieme di campi chiave-valore.

- **Class**: memorizza le informazioni relative alle lezioni prenotate. Ogni documento contiene campi per la descrizione della lezione (`description`), l'orario di inizio (`hour`), la data della lezione (`lesson` come timestamp), e gli identificativi univoci dello studente (`uid_Student`) e del tutor (`uid_Tutor`).
- **Corso di Studi**: contiene le informazioni sui corsi di studi degli studenti e dei tutor, con campi come l'area disciplinare (`area di studi`), il nome del corso (`nome`), e il tipo di laurea (`tipo di laurea`).
- **Orario Tutor**: registra la disponibilità settimanale di ciascun tutor. Ogni documento include una data specifica `date` (come timestamp) e una mappa che indica le fasce orarie (`disponibilità orario`) in cui il tutor è disponibile o meno, insieme all'identificativo del tutor (`uid_Tutor`).
- **Review**: raccoglie le recensioni assegnate dagli studenti ai tutor. Ogni recensione contiene una valutazione in stelle (`stars`), l'identificativo dello studente che ha lasciato la recensione (`uid_Student`) e l'identificativo del tutor recensito (`uid_Tutor`).
- **Student**: memorizza i dati degli studenti, tra cui l'identificativo del loro corso di studi (`corso di studi`), l'indirizzo email (`email`), il nome completo (`nome`), un flag per indicare se lo studente è anche tutor (`isTutor`), e l'URI della foto profilo (`photoUri`).

- **Tutor**: archivia le informazioni relative ai tutor, incluse le valutazioni medie ricevute (`average review`), il corso di studi di appartenenza (`corso di studi`), la disponibilità giornaliera (`disponibilità giorni`) come mappa di giorni e disponibilità, l'email (`email`), il nome (`nome`), l'URI della foto profilo (`photoUri`) e le competenze (`skills`).

Questa struttura permette di gestire facilmente le informazioni essenziali per le funzionalità dell'applicazione, consentendo query efficienti per l'accesso e la manipolazione dei dati. Di seguito il modello dei dati:



3.3.3 Gestione dei dati

In questa sezione verrà illustrato come l'applicazione BicoccaHelp gestisce le operazioni sui dati utilizzando Firestore. La gestione dei dati è una componente cruciale per garantire la funzionalità e l'efficienza dell'applicazione. In questa sezione, verranno analizzati i seguenti aspetti:

- **Operazioni CRUD:** le operazioni principali per creare, leggere, aggiornare e cancellare i dati su Firestore.

- **Query e Indici:** fondamentali per ottimizzare le prestazioni del database e garantire che le operazioni di lettura dei dati siano rapide ed efficienti

CREATE

Nel sistema BicoccaHelp, l'operazione di creazione di documenti nel database Firestore è un'operazione fondamentale che implica diverse fasi: connessione al database, preparazione dei dati, inserimento nel database e gestione del risultato. Di seguito viene descritto in dettaglio il processo per la creazione di una lezione, utilizzando il metodo `createLesson`.

```
private final FirebaseFirestore db = FirebaseFirestore.getInstance();
private final CollectionReference lesson = db.collection("Class");

public void createLesson(CreateLessonRequest request, Callback<LessonModel>
 ↪  callback) {

    Map<String, Object> data = new HashMap<>();

    data.put(UID_STUDENT, request.getUid_Student());
    data.put(UID_TUTOR, request.getUid_tutor());
    data.put(DESCRIPTION, request.getDescription());
    data.put(LESSON_DATE, request.getData());
    data.put(ORA, request.getOra());

    lesson.add(data)
            .addOnSuccessListener(documentReference -> {

                LessonModel lessonModel = new LessonModel(
                        documentReference.getId(), request.getUid_Student(),
                        ↪  request.getUid_tutor(),
                        request.getData(),
                        request.getOra(),
                        request.getDescription());

                callback.onSuccess(lessonModel);

            })
            .addOnFailureListener(callback::onFailure);
}
```

1. **Connessione al Database e Selezione della Collezione:** Il metodo `createLesson` inizia stabilendo una connessione con il database Firestore e selezionando la collezione `Class`. Questo è realizzato con le seguenti righe:

```
private final FirebaseFirestore db = FirebaseFirestore.getInstance();
private final CollectionReference lesson = db.collection("Class");
```

   - `FirebaseFirestore.getInstance()` : Ottiene un'istanza del database Firestore.

- `db.collection("Class")`: Seleziona la collezione `Class` all'interno del database Firestore dove i documenti relativi alle lezioni verranno salvati. Se la collezione non esiste, verrà creata da Firestore DB

2. **Preparazione dei Dati:** I dati per la lezione vengono raccolti in una mappa (`Map<String, Object>`) che rappresenta il documento da inserire. Questa mappa include i seguenti campi:

   - `UID_STUDENT`: ID dello studente che prenota la lezione.
   - `UID_TUTOR`: ID del tutor che tiene la lezione.
   - `DESCRIPTION`: Descrizione della lezione.
   - `LESSON_DATE`: Data della lezione.
   - `ORA`: Orario della lezione.

   Questi dati vengono estratti dall'oggetto `CreateLessonRequest`, che viene passato al metodo `createLesson`. Questo oggetto è progettato per garantire che tutti i campi necessari siano presenti e non nulli.

3. **Inserimento nel Database:** La mappa dei dati viene quindi inserita nel database Firestore utilizzando il metodo `add(data)`:

   - `lesson.add(data)`: Aggiunge un nuovo documento alla collezione `Class` con i dati specificati. Firestore genera automaticamente un ID univoco per il nuovo documento.

4. **Gestione del Risultato:** Dopo l'inserimento, il metodo gestisce il risultato dell'operazione:

   - **Successo:** Se l'inserimento è riuscito, viene creato un oggetto `LessonModel` utilizzando l'ID del documento restituito (`documentReference.getId()`) e i dati della lezione. Questo oggetto viene passato al callback di successo:

```
LessonModel lessonModel = new LessonModel(
    documentReference.getId(), request.getUid_Student(),
    ↪  request.getUid_tutor(),
    request.getData(),
    request.getOra(),
    request.getDescription());

callback.onSuccess(lessonModel);
```

   - **Errore:** Se l'inserimento fallisce, il metodo gestisce il fallimento chiamando `callback.onFailure(exception)`, dove `exception` rappresenta l'errore riscontrato.

READ

Nel sistema BicoccaHelp, l'operazione di Read è essenziale per recuperare documenti dal database Firestore in base a criteri specifici. Un esempio pratico di questa operazione è il recupero delle lezioni prenotate da uno studente, ordinate per data in ordine decrescente. Di seguito viene descritto il processo per il recupero delle lezioni, utilizzando il metodo `listLessonsByStudentDES`. Il metodo `listLessonsByStudentDES` è utilizzato per ottenere una lista di lezioni associate a uno specifico studente, ordinate per data in ordine decrescente. La sua implementazione è la seguente:

```
public void listLessonsByStudentDES(String uidStudent, Long limit,
↪  Callback<List<LessonModel>> callback) {
    Query query = lesson.whereEqualTo(UID_STUDENT, uidStudent)
            .orderBy(LESSON_DATE, Query.Direction.DESCENDING)
            .limit(limit);

    query.get().addOnCompleteListener(task -> {
        if (task.isSuccessful()) {
            List<LessonModel> lessons = new ArrayList<>();
            QuerySnapshot snapshot = task.getResult();
            if (snapshot != null && !snapshot.isEmpty()) {
                for (QueryDocumentSnapshot document : snapshot) {
                    LessonModel lessonModel = new LessonModel(
                            document.getId(),
                            document.getString(UID_STUDENT),
                            document.getString(UID_TUTOR),
                            document.getTimestamp(LESSON_DATE),
                            document.getString(ORA),
                            document.getString(DESCRIPTION)
                    );
                    lessons.add(lessonModel);
                }
                callback.onSuccess(lessons);
            }
        } else {
            callback.onFailure(task.getException());
        }
    });
}
```

Il metodo accetta tre parametri:

- `uidStudent`: l'identificativo univoco dello studente per il quale recuperare le lezioni.
- `limit`: il numero massimo di lezioni da restituire.
- `callback`: un'istanza di `Callback<List<LessonModel>>` che gestisce i risultati dell'operazione di lettura.

Il metodo crea una query per recuperare i documenti dalla collezione delle lezioni filtrati in base all'UID dello studente. Successivamente, ordina i documenti per data della lezione

in ordine decrescente, limitandone il numero in base al parametro specificato. Infine, gestisce il risultato dell'operazione di lettura:

- **Successo:** Se la lettura è riuscita, viene creata una lista di oggetti `LessonModel` a partire dai documenti recuperati. La lista viene passata al callback di successo: `callback.onSuccess(lessons);`
- **Errore:** Se la lettura fallisce, il metodo gestisce il fallimento chiamando `callback.onFailure(task.getException());`, dove `exception` rappresenta l'errore riscontrato.

UPDATE

Nel sistema BicoccaHelp, l'operazione di aggiornamento dei documenti nel database Firestore permette di modificare uno o più campi di un documento esistente. Di seguito viene illustrato il processo di aggiornamento degli orari di disponibilità di un tutor utilizzando il metodo `updateOrario`.

```
public void updateOrario(String uidTutor, Date date, String timeSlot, boolean
↪  availability, Callback<Void> callback) {
    CollectionReference orarioTutor = db.collection("Orario Tutor");

    Map<String, Object> updates = new HashMap<>();
    updates.put("disponibilità orario." + timeSlot, availability);

    orarioTutor
        .whereEqualTo("uid Tutor", uidTutor)
        .whereEqualTo("date", date)
        .get()
        .addOnSuccessListener(queryDocumentSnapshots -> {
            if (!queryDocumentSnapshots.isEmpty()) {
                for (DocumentSnapshot document : queryDocumentSnapshots) {
                    document.getReference().update(updates)
                        .addOnSuccessListener(aVoid ->
                        ↪  callback.onSuccess(null))
                        .addOnFailureListener(callback::onFailure);
                }
            } else {
                callback.onFailure(new Exception("Documento non trovato"));
            }
        })
        .addOnFailureListener(callback::onFailure);
}
```

1. **Preparazione dell'Aggiornamento:** I dati da aggiornare vengono organizzati in una mappa (`Map<String, Object>`) che specifica il campo e il nuovo valore. In questo caso, il campo aggiornato è una sottochiave all'interno della mappa `disponibilità orario`:

- `updates.put("disponibilità orario." + timeSlot, availability)`: Specifica l'aggiornamento per un determinato intervallo orario, utilizzando `timeSlot` come chiave e `availability` (boolean) come nuovo valore.

2. **Ricerca del Documento:** Il metodo esegue una query per trovare il documento corretto in base all'ID del tutor e alla data:

   - `orarioTutor.whereEqualTo("uid Tutor", uidTutor).whereEqualTo("date", date)`

     Esegue una query per trovare il documento corrispondente all'ID del tutor e alla data specificati.

   Se viene trovato un documento, viene aggiornato con i nuovi dati. In caso contrario, il metodo gestisce l'errore, notificando che il documento non è stato trovato.

3. **Aggiornamento del Documento:** Una volta trovato il documento corretto, l'aggiornamento viene effettuato tramite il metodo `update()`:

   - `document.getReference().update(updates)`: Aggiorna il documento con i dati presenti nella mappa `updates`.

4. **Gestione del Risultato:** Il metodo gestisce sia il successo che il fallimento dell'aggiornamento:

   - **Successo:** Se l'aggiornamento è completato con successo, viene chiamato il callback di successo.
   - **Errore:** Se l'aggiornamento fallisce o il documento non viene trovato, viene chiamato il callback di fallimento.

DELETE

L'operazione di delete in Firestore consente di rimuovere documenti specifici da una collezione. Di seguito viene descritto il processo per l'eliminazione di una lezione dalla collezione delle lezioni, utilizzando il metodo `deleteLesson`. La sua implementazione è la seguente:

```
public void deleteLesson(String lessonUid, Callback<Void> callback){
    lesson.document(lessonUid)
            .delete()
            .addOnSuccessListener(callback::onSuccess)
            .addOnFailureListener(callback::onFailure);
}
```

Il metodo accetta due parametri:

- `lessonUid`: l'identificativo univoco della lezione da eliminare.
- `callback`: un'istanza di `Callback<Void>` che gestisce i risultati dell'operazione di eliminazione.

1. **Accesso al Documento:** Il metodo utilizza l'UID della lezione per accedere al documento specifico nella collezione delle lezioni: `lesson.document(lessonUid)`
2. **Eliminazione del Documento:** Il metodo `delete()` viene chiamato sul documento per avviare l'eliminazione.
3. **Gestione del Risultato:** Dopo l'eliminazione, il metodo gestisce il risultato dell'operazione:
   - **Successo:** Se l'eliminazione è riuscita, viene invocato il metodo `.addOnSuccessListener(callback::onSuccess)`
   - **Errore:** Se l'eliminazione fallisce, il metodo gestisce il fallimento chiamando `.addOnFailureListener(callback::onFailure)`

Questo metodo consente di rimuovere in modo sicuro e affidabile i documenti relativi alle lezioni dalla collezione.

Indici in Firestore

Per eseguire query con ordinamento e filtri complessi in Firestore, è necessario utilizzare indici. Nel caso della query relativa al metodo `listLessonsByStudentDES`, gli indici corrispondenti sono configurati come segue:

- `uid Student` → Ordinamento Crescente
- `uid Tutor` → Ordinamento Crescente
- `lesson Date` → Ordinamento Decrescente
- `name` → Ordinamento Decrescente

Firestore richiede indici composti quando si ordinano i risultati su più campi e quando si applicano filtri. Gli indici ottimizzano le query e garantiscono che i risultati vengano restituiti in modo rapido ed efficiente. Se una query richiede un indice che non è stato creato, Firestore fornirà un errore con un link per creare automaticamente l'indice richiesto.

3.3.4 Sicurezza e Autorizzazioni

Per garantire la sicurezza dei dati e controllare l'accesso alle informazioni sensibili, sono state implementate delle regole di sicurezza nel database Firestore dell'applicazione BicoccaHelp. Queste regole determinano chi può leggere e scrivere nei documenti del database, basandosi sull'autenticazione dell'utente e sull'identità dell'utente stesso. Di seguito sono descritte le regole di sicurezza configurate:

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if request.auth != null;
    }

    match /Utenti/{userId} {
      allow read, write: if request.auth != null && request.auth.
          uid == userId;
    }

    match /Corsi/{courseId} {
      allow read: if request.auth != null;
      allow write: if request.auth != null;
    }

    match /Lezioni/{lessonId} {
      allow read: if request.auth != null;
      allow write: if request.auth != null;
    }
  }
}
```

Le regole generali applicate a tutte le collezioni nel database stabiliscono che solo gli utenti autenticati hanno accesso ai dati. Questo significa che ogni operazione di lettura e scrittura è consentita solo se l'utente ha effettuato l'accesso.

Per la collezione Utenti, le regole specificano che un utente può leggere e scrivere solo i propri dati. Questa restrizione è implementata verificando che l'ID dell'utente autenticato (`request.auth.uid`) corrisponda all'ID del documento richiesto (`userId`). In questo modo, gli utenti non possono accedere o modificare i dati di altri utenti.

Per la collezione Corsi, le regole stabiliscono che tutti gli utenti autenticati possono leggere i corsi e scrivere nei documenti della collezione.

Le regole per la collezione Lezioni sono simili a quelle della collezione Corsi. Gli utenti autenticati possono leggere le lezioni e anche scrivere nei documenti della collezione. Questa configurazione assicura che le informazioni relative alle lezioni siano visibili a tutti gli utenti autenticati e modificabili solo da utenti che hanno effettuato l'accesso.

3.3.5 Gestione dell'immagine utente con Firebase Storage e Glide

Nell'applicazione BicoccaHelp, Firebase Storage viene utilizzato per gestire il caricamento delle foto profilo degli utenti, mentre Glide si occupa della visualizzazione all'interno dell'interfaccia utente (UI).

- **Firebase Storage:** Servizio di archiviazione cloud
- **Glide:** libreria di gestione delle immagini per Android

Quando un utente desidera aggiornare la propria foto profilo, viene attivato un *Intent* per selezionare un'immagine dalla galleria utilizzando `ActivityResultLauncher`:

```
ActivityResultLauncher<PickVisualMediaRequest> pickMedia =
    registerForActivityResult(new ActivityResultContracts.PickVisualMedia(),
    ↪  uri -> {
        if (uri != null) {
            updatePhoto(uri);
        }
    });
```

Una volta che l'utente ha selezionato una foto, l'applicazione carica l'immagine in Firebase Storage e aggiorna l'URI della foto nel documento utente nel database Firestore. Il percorso in cui viene salvata l'immagine è definito dal metodo `updatePhoto` all'interno del repository, che assegna un percorso specifico basato sull'UID dell'utente.

```
String userPhotoPath = "user/" + user.getUid() + "/profile_photo.jpeg";
userAssetsRemoteDataSource.upload(userPhotoPath, photoUri, new
↪  Callback<String>() {
    @Override
    public void onSucces(String photoPath) {
        userRemoteDataSource.updatePhoto(Uri.parse(photoPath), callback);
    }
});
```

La foto viene salvata all'interno di Firebase Storage nella directory `user/` seguita dall'UID univoco dell'utente, e il file viene denominato `profile_photo.jpeg`. Questo garantisce che ogni utente abbia il proprio percorso dedicato per la foto profilo.

Dopo che l'immagine è stata caricata correttamente su Firebase Storage, viene utilizzato Glide per caricare e visualizzare la nuova foto profilo all'interno dell'app. Glide permette di gestire il caricamento delle immagini direttamente dalla rete o dalla cache locale, riducendo il tempo di caricamento e ottimizzando l'esperienza utente. Il frammento di codice seguente mostra come avviene il caricamento dell'immagine con Glide:

```
public void loadPhoto(Uri photoUri, boolean skipCache) {
    if (photoUri.getScheme() != null &&
    ↪  photoUri.getScheme().equals(getString(R.string.https))) {
        Glide.with(requireActivity().getApplicationContext())
                .load(photoUri)
                .into(binding.userPhoto); // Visualizza la foto profilo
        return;
    }
    Glide.with(requireActivity().getApplicationContext())
            .load(GlideLoadModel.get(photoUri.toString()))
            .skipMemoryCache(skipCache)
            .diskCacheStrategy(skipCache ? DiskCacheStrategy.NONE :
            ↪  DiskCacheStrategy.AUTOMATIC)
            .into(binding.userPhoto);
}
```

Il metodo `loadPhoto()` gestisce il caricamento dell'immagine, permettendo di saltare la cache se necessario per garantire che l'immagine aggiornata venga visualizzata.

3.4 Interfaccia Utente

L'interfaccia utente (UI) di BicoccaHelp è stata progettata per offrire un'esperienza intuitiva e accessibile agli studenti che utilizzano l'applicazione per prenotare lezioni di supporto accademico. Un'interfaccia ben progettata è cruciale per facilitare l'interazione degli utenti con le funzionalità dell'applicazione, rendendo l'esperienza complessiva più piacevole ed efficiente.

Per la progettazione della UI, ho scelto di adottare un approccio ibrido. Alcune sezioni seguono le linee guida del **Material Design**, garantendo coerenza visiva, un'esperienza utente ottimizzata e accessibilità su diverse piattaforme. Tuttavia, altre parti dell'app sono state sviluppate con maggiore flessibilità, in modo da adattarsi meglio alle esigenze specifiche dell'applicazione e del target di utenti.

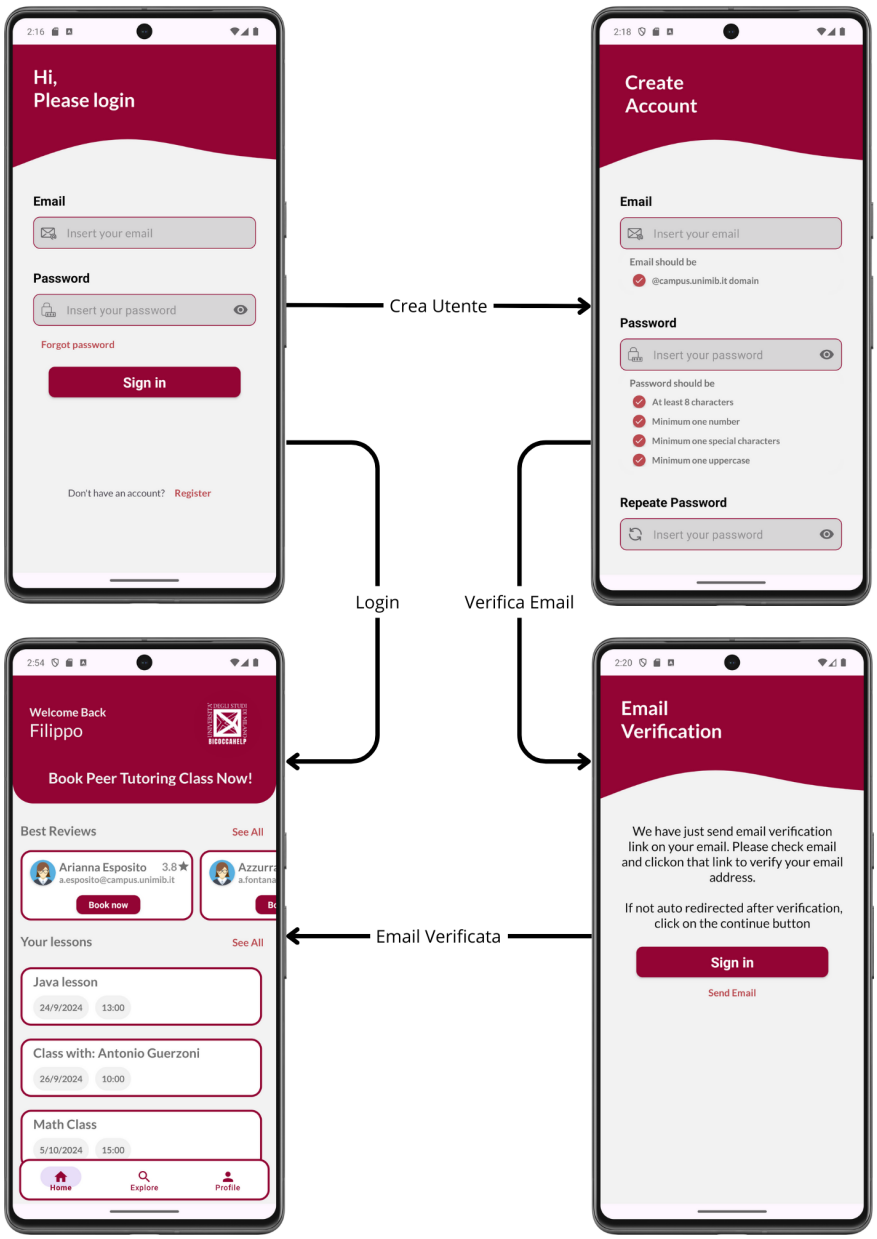
Nel corso di questo capitolo verranno presentati i principali componenti dell'interfaccia, le scelte stilistiche adottate e i compromessi che sono stati fatti per bilanciare l'usabilità con le esigenze di sviluppo.

Color Gradation

Nella progettazione dell'interfaccia utente di BicoccaHelp, la scelta della scala colori si è ispirata alla palette istituzionale dell'Università di Milano-Bicocca. L'obiettivo era creare un'interfaccia riconoscibile e in linea con l'identità dell'ateneo.

I colori principali selezionati includono diverse sfumature di rosso, tipiche del logo dell'Università e toni di grigio che bilanciano il contrasto cromatico, garantendo una buona leggibilità. I colori includono sfumature come *"Dark Red"* (`#930434`) e *"Deep Red"* (`#8d1c1c`), che richiamano il colore dell'università, mentre i grigi come *"Misty Gray"* (`#D2D2D2`) e *"Dark Gray"* (`#707070`) forniscono un equilibrio al layout.

3.4.1 Welcome



Il flusso di navigazione presentato accompagna l'utente attraverso il processo di registrazione e autenticazione necessario per accedere all'applicazione. Al primo avvio, l'utente visualizza la schermata iniziale rappresentata dalla **WelcomeActivity**. Da qui, l'utente può scegliere tra due opzioni principali:

1. **Autenticazione per utenti già registrati:**
   Gli utenti che hanno già creato un account possono accedere inserendo l'email e la password negli appositi campi di input, implementati come **TextInputEditText**, seguendo le linee guida del Material Design.

   - *"Email"* → `login_email_editText`
   - *"Password"* → `login_password_editText`

Dopo aver compilato questi campi, l'utente può premere sul pulsante *"Sign in"* per effettuare il login e accedere all'applicazione. La transizione dalla schermata iniziale `activity_welcome` alla pagina principale dell'applicazione `activity_main` è gestita dall'**Action** `action_from_login_to_main` nel `welcome_navigation_graph`.

2. **Creazione di un nuovo account per utenti non registrati:**
   Nel caso in cui l'utente non abbia ancora un account, può premere il pulsante *"Register"* (`login_button_register`) che lo porterà alla schermata di registrazione, tramite l'**Action** `action_from_login_to_registration` tra `activity_welcome` e `fragment_registration`.

   Nella pagina dedicata alla creazione dell'account, l'utente dovrà compilare i seguenti campi:

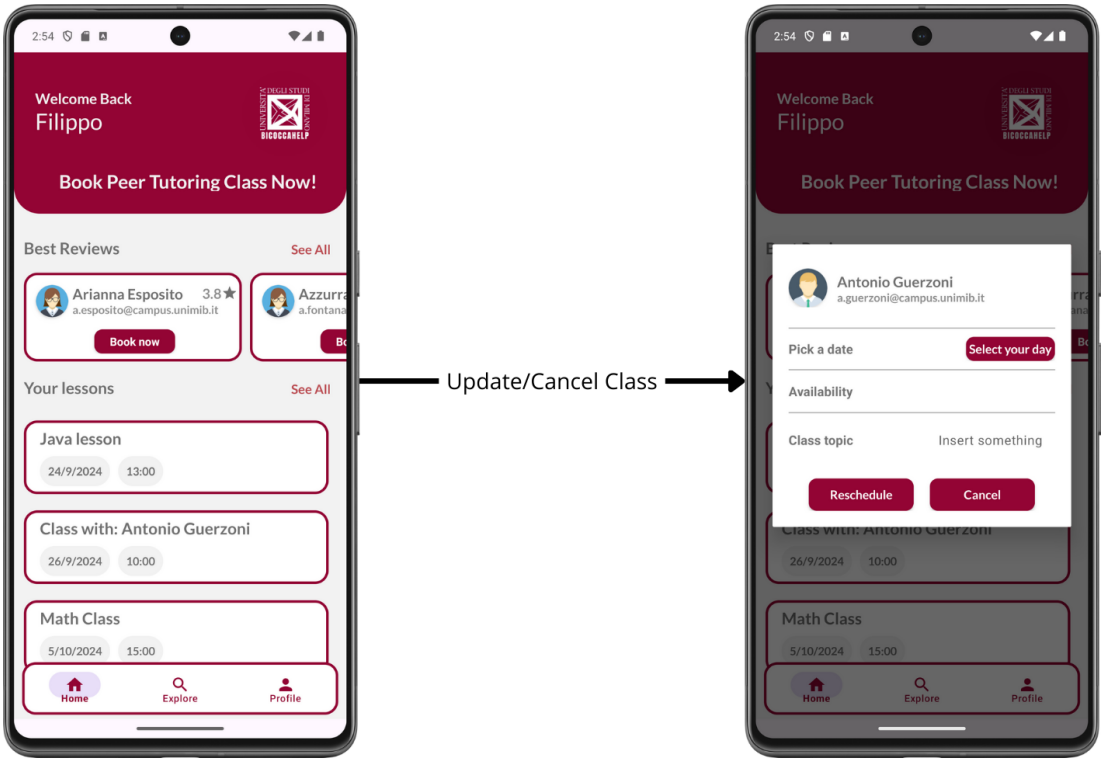
   - *"Email"* → `create_account_email_editText`
   - *"Password"* → `create_account_password_editText`
   - *"Ripeti Password"* → `create_account_repeat_password_editText`
   - *"Nome"* → `create_account_name_editText`

   Dopo aver inserito tutte le informazioni richieste, l'utente potrà completare la registrazione, passando al fragment `fragment_verify_email` mediante la **Action** `action_from_registration_to_verify_email` .

   Una volta registrato, verrà inviata un'email di verifica che dovrà essere confermata per attivare l'account e consentire l'accesso all'applicazione. Dopo che l'email è stata verificata con successo, l'utente potrà semplicemente premere sul pulsante *"Sign In"* (definito da `continue_button`) per completare l'accesso all'app tramite l'**Action** `action_from_verify_email_to_main` tra `fragment_verify_email` e `activity_main`.

   Se l'utente non ricevesse l'email di verifica o desiderasse riceverne una nuova, potrà richiederla nuovamente premendo il pulsante *"Send Email"* (`resend_button`), che provvederà a rinviarla.

3.4.2 Home



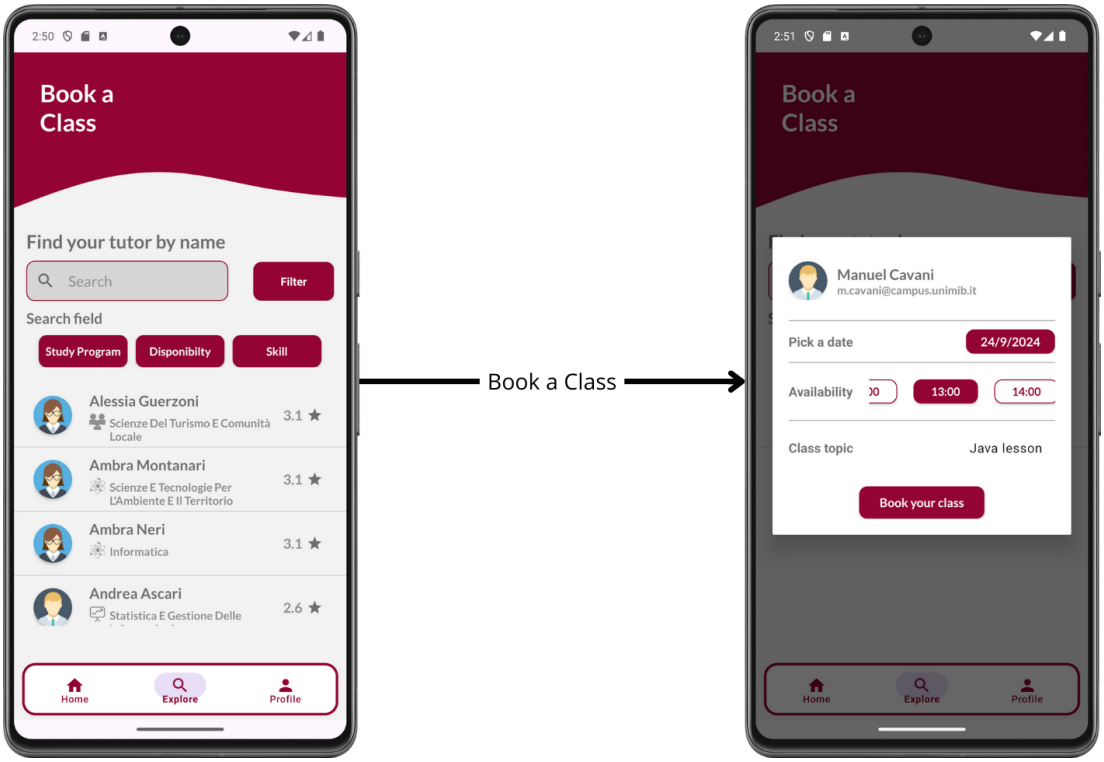
Il flusso di navigazione per la modifica o la cancellazione di una lezione inizia con la selezione di una lezione dall'elenco presente nel **RecyclerView** definito come `LessonRecycleView`, situato nella sezione *"Your lessons"*.

Quando l'utente clicca su una lezione, si apre un **Dialog** basato sul layout chiamato `lesson_card`, che visualizza i dettagli della lezione stessa. L'**Action** tra `home_fragment` e `lesson_card` è detta `actionToLessonCardFromHome` definita nel `main_navigation_graph`.

Per modificare la data della lezione, l'utente può premere il pulsante *"Select your day"* (`select_day_button`), che farà apparire un nuovo **RecyclerView** nella sezione *"Availability"*, chiamato `HourRecyclerView`. In questa lista, ogni orario disponibile è rappresentato da un **ToggleButton** denominato `hourRadio`, permettendo così all'utente di selezionare l'orario preferito.

Una volta effettuate le modifiche, l'utente può confermare premendo il pulsante *"Reschedule"* (`UpdateLessonButton`), oppure annullare la lezione utilizzando il pulsante *"Cancel"* (`deleteLessonButton`). Questo flusso interattivo rende semplice e veloce la gestione delle lezioni direttamente dalla schermata principale dell'applicazione, garantendo un'esperienza utente fluida.

3.4.3 Ricerca Tutor e Prenota Lezione



Il flusso di navigazione per la ricerca e la prenotazione di una lezione inizia dal **Fragment** chiamato `fragment_tutor`, dove l'utente ha la possibilità di cercare un tutor in diversi modi. Se si conosce già il nome del tutor, è possibile inserirlo direttamente nel campo di testo **TextInputEditText** chiamato `search_tutor_editText` e premere il pulsante *"Filter"* (`filter_name_button`) per visualizzare i risultati corrispondenti.

Se invece si desidera filtrare i risultati in base al corso di studi, alla disponibilità o alle competenze del tutor, l'utente può selezionare uno dei tre **Toggle Button** nella sezione *Search Field*:

- *"Study Program"*: il cui ID nel layout è `corsoToggle`
- *"Disponibility"*: il cui ID nel layout è `dispoToggle`
- *"Skill"*: il cui ID nel layout è `skillsToggle`

Dopo aver selezionato il filtro desiderato, l'utente può inserire i termini di ricerca nel **TextInputEditText** e premere nuovamente il pulsante *"Filter"* per raffinare la ricerca.

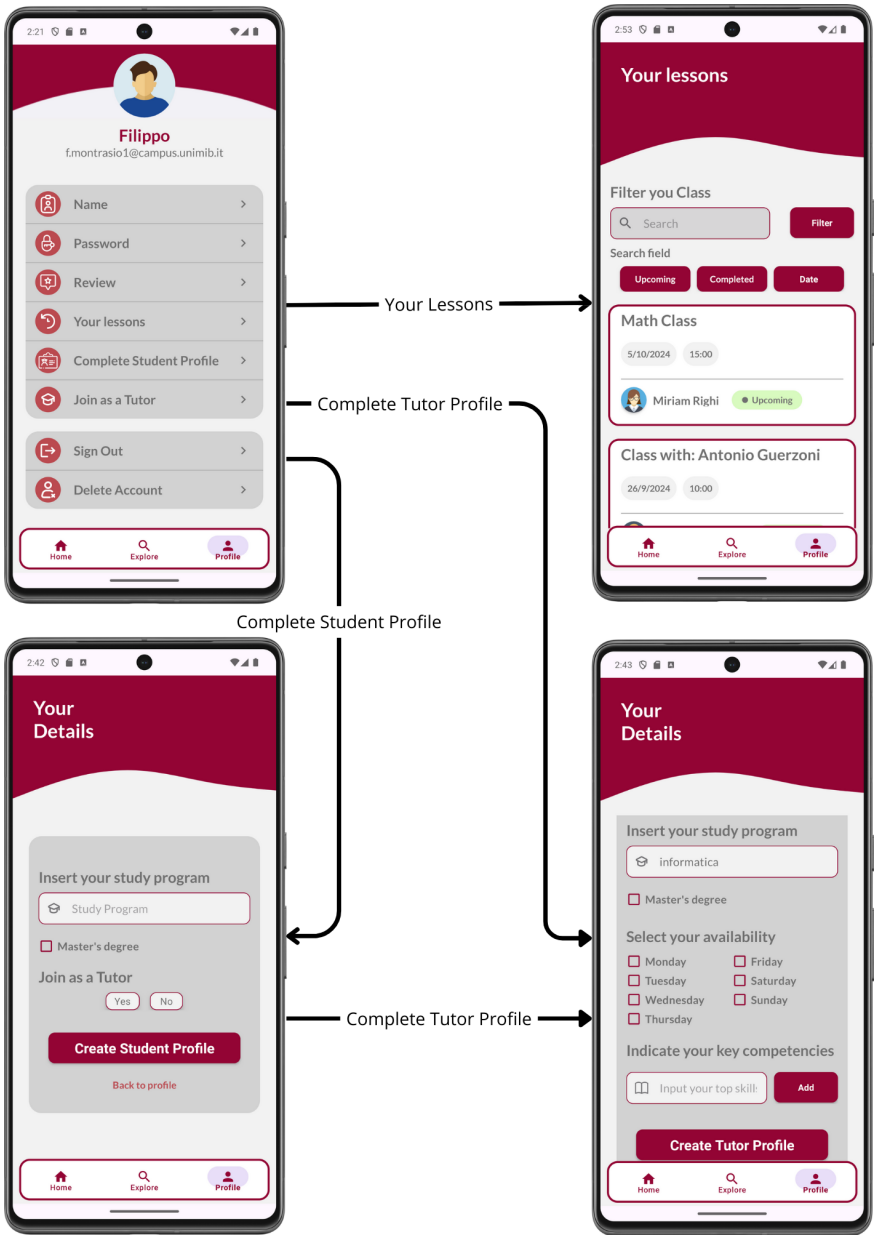
I tutor filtrati vengono visualizzati nel **RecyclerView** chiamato `tutorRecyclerView`, dove ogni elemento è rappresentato dal layout `tutor_list_item`. Ogni elemento della lista include informazioni principali come l'immagine del profilo, il nome del tutor e il corso di studi. Quando l'utente clicca su uno di questi elementi, si apre un **Dialog**, tramite l'**Action** `actionToLessonCard` che mostra i dettagli del tutor e permette la prenotazione della lezione.

All'interno del **Dialog**, l'utente può scegliere una data premendo il pulsante *"Select your day"* (`select_day_button`), che apre un calendario definito da un **DatePickerDialog** di Material Design, consentendo di selezionare la data. Dopo che l'utente ha selezionato una data, si apre lista di orari disponibili nel **RecyclerView** `HourRecyclerView`. Ogni orario è rappresentato da un **Toggle Button** chiamato `hourRadio`, e l'utente può selezionare l'orario preferito.

Nel campo **TextInputEditText** denominato `textInputEditTextDescription`, l'utente può inserire il nome della lezione che desidera prenotare. Una volta inserite tutte le informazioni necessarie, è possibile confermare la prenotazione premendo il pulsante *"Book your class"* (`bookLessonButton`).

Questo flusso consente all'utente di trovare e prenotare facilmente una lezione con il tutor desiderato, garantendo un'interazione fluida attraverso l'uso di filtri e una gestione intuitiva delle prenotazioni.

3.4.4 Profilo



Your Lessons

Cliccando sulla voce **"Your lessons"** (definita dall'ID `youLessons_item`), l'utente viene indirizzato alla schermata delle lezioni tramite l'**Action** `action_from_profile_to_your_lesson`. Una volta all'interno del `Lesson Fragment`, è possibile filtrare le lezioni in vari modi.

L'utente può cercare un tutor specifico inserendo il nome nel campo di testo **TextInputEditText** `search_tutor_editText` e premere il pulsante *"Filter"* (`filter_name_button`). Inoltre, ci sono tre opzioni di filtro aggiuntive:

- *"Upcoming"* (`upcomingToggle`)
- *"Completed"* (`TerminatedToggle`)
- *"Date"* (`date_button`)

Cliccando su *"Upcoming"* o *"Completed"*, l'utente può filtrare le lezioni in base al loro stato, premendo poi nuovamente *"Filter"*. Se invece si seleziona *"Date"*, si apre un **DatePickerDialog** di Material Design per la selezione della data, e le lezioni verranno filtrate in base al giorno selezionato dopo aver premuto il tasto di filtro.

Complete Student Profile

Cliccando su `"Complete Student Profile"` (definito dall'ID `complete_student_item`), l'utente naviga verso il `fragment_complete_student` tramite l'**Action** `action_from_profile_to_complete_student_fragment` . All'interno di questo fragment, è possibile completare il profilo dello studente inserendo le informazioni nel campo **TextInputEditText** `create_student_editText` , dove l'utente può inserire il corso di studi frequentato. Inoltre, c'è la possibilità di selezionare il tipo di laurea tramite il **CheckBox** `create_student_check_box` , che consente di scegliere tra laurea triennale o magistrale.

Il **Radio Group** `join_tutor_radio` permette all'utente di indicare se desidera diventare anche tutor. Se viene selezionato *"Yes"* (`yes_radio_button`) e si preme il pulsante *"Create Student Profile"* (`create_student_button`), si viene indirizzati al `fragment_complete_tutor` tramite l'azione `action_from_complete_student_to_complete_tutor` . Se invece viene selezionato *"No"* (`no_radio_button`), l'azione `action_from_complete_profile_to_profile_fragment` porta al `Profile Fragment`.

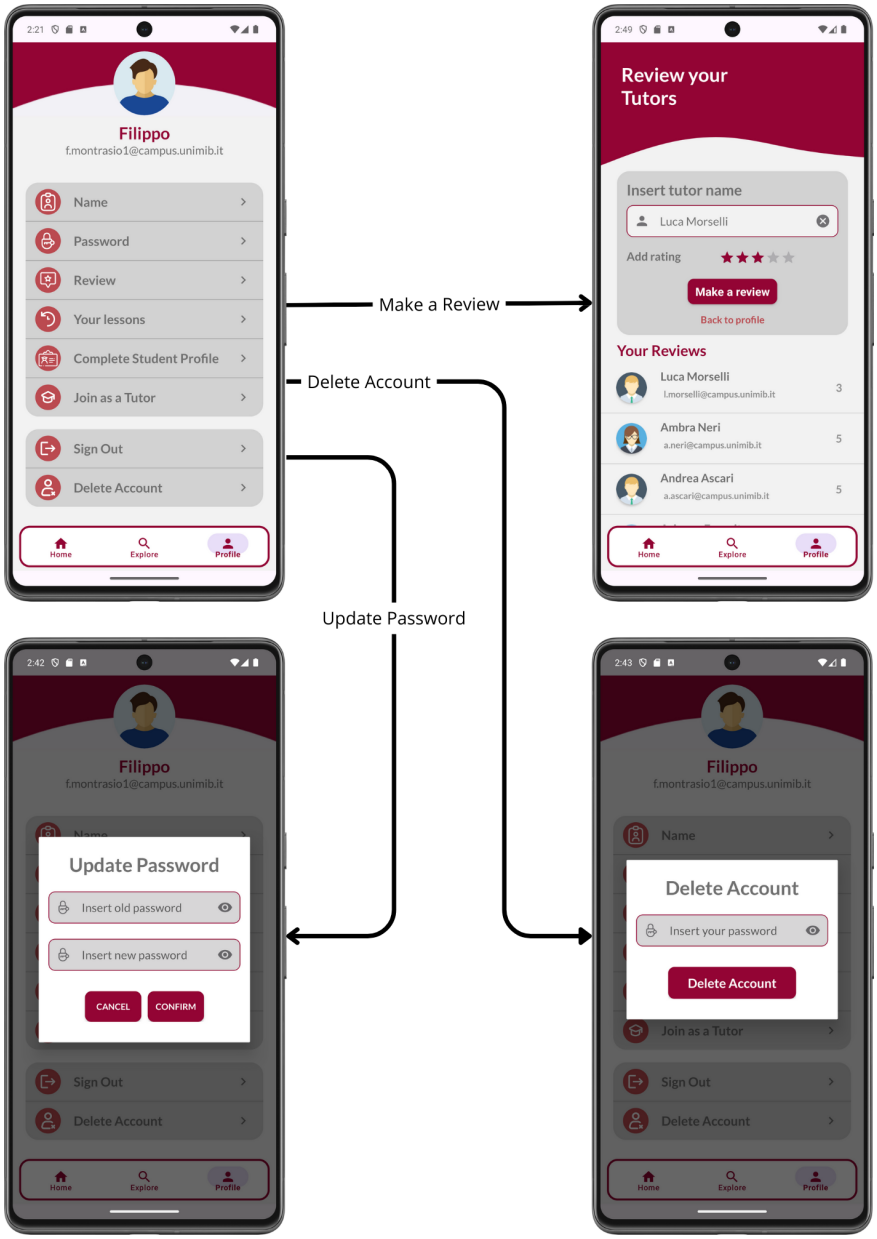
Join as a Tutor

Nel `Profile Fragment`, cliccando sulla voce *"Join as a tutor"* (`become_a_tutor_item`), l'utente viene reindirizzato al `fragment_complete_tutor` tramite l'azione `action_from_profile_to_complete_tutor_fragment` .

Nel `fragment_complete_tutor`, l'utente può aggiornare le informazioni relative al proprio corso di studi e selezionare la disponibilità settimanale. La sezione *"Select your availability"* include una serie di **CheckBox** che consente di selezionare i giorni della settimana in cui il tutor è disponibile.

Nella sezione dedicata alle competenze, l'utente può inserire le proprie materie di specializzazione nel campo **TextInputEditText** `best_subject_editText` e aggiungerle alla lista premendo il pulsante *"Add"* (`add_subject_button`).

Per completare il profilo da tutor, l'utente deve premere il pulsante *"Create Tutor Profile"* (`create_tutor_button`), che lo riporta al `Profile Fragment` tramite l'azione `action_from_complete_tutor_to_profile_fragment` .



Review

Nel `profile_fragment`, selezionando la voce *"Review"* definita da `review_item`, l'utente viene indirizzato al `review_fragment` tramite l'azione `action_from_profile_to_review_fragment`.

All'interno del `review_fragment`, l'utente può recensire i tutor (una sola volta per tutor), inserendo il nome del tutor nel campo **TextInputEditText** `review_student_editText` e selezionando il numero di stelle tramite la **RatingBar** `rating_bar`. Per completare la recensione, l'utente deve premere il pulsante *"Make a review"* (`review_button`), dopo di che vedrà la recensione aggiunta all'interno del **RecyclerView** `reviewRecyclerView`.

Update Password

Se l'utente clicca sulla voce *"Password"* (`update_password_item`) nel menù del `profile_fragment`, si apre il **Dialog** `fragment_update_password_dialog` tramite l'azione `action_from_profile_to_update_password_dialog` . Questo dialog consente di aggiornare la password.

Nel campo **TextInputEditText** *"Insert old password"* (`update_password_old_EditText`), l'utente deve inserire la vecchia password (per la re-autenticazione). Successivamente, la nuova password viene inserita nel **TextInputEditText** *"Insert new password"* (`update_password_new_EditText`). Per confermare il cambiamento, l'utente deve cliccare il pulsante *"Confirm"* (`update_password_button_confirm`). Dopo l'aggiornamento, l'utente verrà reindirizzato alla `activity_welcome` per effettuare nuovamente l'accesso tramite l'azione `action_from_update_password_dialog_to_welcome_Activity` .

Delete Account

Se l'utente seleziona la voce *"Delete Account"* (`delete_profile_item`)nel menù del `profile_fragment`, viene visualizzato il **Dialog** `fragment_delete_user_dialog` tramite l'azione `action_from_profile_to_delete_dialog` . Questo dialogo permette all'utente di eliminare il proprio profilo utente (sia studente che eventualmente tutor).

L'utente deve inserire la propria password nel campo **TextInputEditText** *"Insert your password"* (`delete_account_password_editText`) e confermare cliccando il pulsante *"Delete Account"* (`delete_account_button`). Una volta confermata la cancellazione, l'utente verrà disconnesso e riportato alla `activity_welcome` tramite l'azione `action_from_delete_dialog_to_welcome_activity` .

3.5 Conclusioni e Prospettive Future

Nel corso di questo capitolo, abbiamo esaminato l'implementazione di BicoccaHelp, focalizzandoci su vari aspetti tecnici e funzionali. L'applicazione è stata costruita seguendo il pattern **Model-View-ViewModel** (MVVM), che ha garantito una separazione efficace tra logica di business e interfaccia utente. Questo approccio ha semplificato lo sviluppo, la manutenibilità e la possibilità di testare e scalare l'applicazione nel tempo.

L'integrazione con **Firebase** ha rappresentato una scelta chiave per la gestione delle operazioni principali: dall'autenticazione degli utenti alla memorizzazione e sincronizzazione dei dati in tempo reale tramite Firestore, fino alla gestione delle risorse multimediali tramite Firebase Storage. L'uso di **Glide** ha ulteriormente migliorato l'esperienza utente, rendendo fluido e rapido il caricamento delle immagini, anche su dispositivi con risorse limitate.

L'interfaccia utente è stata progettata seguendo i principi del **Material Design** [6], con l'obiettivo di garantire una navigazione intuitiva e coerente. La scelta dei colori e la disposizione degli elementi visivi sono stati studiati per offrire un'esperienza piacevole e accessibile agli studenti dell'Università di Milano-Bicocca. Funzionalità come la ricerca dei tutor, la prenotazione delle lezioni e la gestione del profilo utente sono state

implementate per ridurre al minimo il numero di interazioni necessarie, migliorando la semplicità d'uso.

3.5.1 ChatGPT

Durante lo sviluppo di BicoccaHelp, ho utilizzato ChatGPT come supporto per vari aspetti della progettazione e implementazione dell'applicazione. L'AI si è rivelata uno strumento prezioso per risolvere problemi tecnici e ottenere suggerimenti su come migliorare l'efficienza del codice. ChatGPT mi ha fornito idee su come strutturare al meglio l'architettura MVVM e mi ha guidato nella gestione delle chiamate a Firebase, migliorando il flusso di autenticazione e la gestione dei dati in tempo reale.

In sintesi, l'uso di ChatGPT ha contribuito non solo a risolvere dubbi tecnici e generare codice, ma anche a velocizzare il processo di implementazione, fornendo assistenza continua e rendendo più efficiente l'intero processo di creazione dell'applicazione. Le valutazioni sull'uso di ChatGPT saranno approfondite nel capitolo 4.

3.5.2 Prospettive Future

Sebbene l'applicazione offra già una base solida, ci sono diverse opportunità per ampliarne le funzionalità e migliorare ulteriormente l'esperienza utente.

- **Prenotazioni di gruppo**: L'espansione delle funzionalità di prenotazione potrebbe includere sessioni di tutoraggio di gruppo, fornendo agli studenti una maggiore flessibilità nella scelta delle lezioni. Inoltre, si potrebbe integrare un sistema di notifiche push per ricordare agli utenti le lezioni imminenti.
- **Gamification e coinvolgimento**: Introdurre elementi di **gamification** come badge o classifiche per i tutor potrebbe migliorare il coinvolgimento degli utenti, incentivando una partecipazione più attiva e costruendo una community dinamica.
- **Layout lato tutor**: Sarà possibile introdurre una nuova interfaccia dedicata ai tutor, consentendo loro di gestire meglio le prenotazioni, modificarle e cancellarle. Inoltre, i tutor potranno migliorare il proprio profilo aggiungendo ulteriori informazioni, e raccogliere punti da utilizzare in università dopo aver completato le lezioni.

4 Integrazione e Utilizzo di ChatGPT nel Processo di Sviluppo

In questo capitolo verrà descritto l'importante ruolo che ChatGPT ha ricoperto durante le diverse fasi dello sviluppo dell'applicazione, a partire dall'analisi dei requisiti fino al testing. L'utilizzo di un modello LLM ha offerto un supporto concreto non solo per affrontare le complessità tecniche, ma anche per migliorare l'efficienza e la qualità complessiva del progetto.

In aggiunta all'uso di ChatGPT, ho esplorato anche altri modelli LLM, come Bing Copilot. Ho riscontrato che le risposte fornite da questi modelli erano molto simili a quelle ottenute da ChatGPT, suggerendo una convergenza nelle capacità di generazione e supporto. Questa esperienza ha ampliato la mia comprensione delle potenzialità offerte da diverse intelligenze artificiali nel processo di sviluppo.

Nel primo sottocapitolo, *Analisi dei Requisiti*, verrà esplorato come ChatGPT ha facilitato il processo creativo e la stesura delle specifiche tecniche, semplificando la formalizzazione dei casi d'uso e la progettazione iniziale dell'architettura del database.

Successivamente, nel sottocapitolo *Implementazione*, verranno approfonditi i contributi offerti dall'intelligenza artificiale durante la creazione dei layout, la definizione del database Firestore e l'implementazione della logica applicativa, con un focus su come l'AI ha semplificato lo sviluppo del codice e la risoluzione di bug.

Infine, nel terzo sottocapitolo *Testing*, si discuterà l'approccio di ChatGPT nel supportare la creazione dei test automatizzati e le sfide incontrate durante questa fase, evidenziando sia i vantaggi sia i limiti di un tale strumento.

Attraverso l'analisi delle varie fasi, verrà evidenziato l'impatto dell'integrazione di ChatGPT nel processo di sviluppo, mostrando come è stato utile per massimizzare la produttività, accelerare i tempi di sviluppo e ridurre al minimo gli errori.

4.1 Analisi Dei Requisiti

Nella fase di analisi dei requisiti, ho fatto affidamento su ChatGPT fin dalle prime fasi del progetto. Inizialmente, l'ho utilizzato per esplorare idee su quale potesse essere l'obiettivo principale dell'applicazione e per la scelta del nome. Ad esempio, ho chiesto a ChatGPT di suggerire nomi che riflettessero il tema dell'applicazione, e le risposte mi hanno dato spunti che non avevo considerato. Successivamente, mi ha supportato nella definizione delle funzionalità principali da implementare e degli elementi da includere nei layout delle varie schermate.

Il contributo di ChatGPT si è rivelato particolarmente prezioso in una fase per me critica: quella creativa. Spesso ho difficoltà a trovare idee iniziali, e il supporto di un modello come ChatGPT mi ha aiutato a superare questa sfida. Ho anche imparato a formulare domande più efficaci, il che ha reso le interazioni con ChatGPT più produttive. Per questo motivo, l'uso di un LLM durante la fase di analisi dei requisiti è stato essenziale per colmare queste difficoltà.

Oltre a ciò, ChatGPT è stato utile nella stesura formale dei casi d'uso, nell'identificazione degli attori e nella definizione dei requisiti. Ha migliorato il lessico e la struttura delle bozze che avevo preparato, rendendole più coerenti e precise, semplificando così il mio lavoro di documentazione. Un aspetto particolarmente utile è stato il supporto nella creazione e nella strutturazione dei diagrammi: ChatGPT mi ha aiutato a definire sia i diagrammi dei casi d'uso specifici, sia il diagramma generale che li raccoglie tutti, fornendomi indicazioni su come rappresentarli correttamente all'interno della documentazione.

Per quanto riguarda la parte tecnica, ho utilizzato ChatGPT anche per la definizione delle collezioni e degli attributi dei documenti nel database, un compito che rientrava anch'esso nella fase di analisi dei requisiti. In questo contesto, mi ha aiutato a strutturare i dati in modo preciso e a progettare un database Firestore ottimizzato per l'applicazione. Grazie al suo supporto, sono riuscito a comprendere meglio il funzionamento di Firestore, un database NoSQL con cui avevo poca familiarità, essendo abituato a lavorare con database relazionali.

Tuttavia, non tutte le risposte ottenute da ChatGPT si sono dimostrate immediatamente utili. Spesso è stato necessario riformulare le domande o chiedere chiarimenti per ottenere risposte più in linea con le mie esigenze, poiché alcune risposte iniziali si discostavano dalle mie aspettative. Ciò mi ha portato a sviluppare una maggiore consapevolezza sull'importanza di porre domande precise e ben strutturate, specialmente durante una fase critica come quella dell'analisi dei requisiti.

In conclusione, l'uso di ChatGPT durante la fase di analisi dei requisiti si è rivelato uno strumento estremamente utile per strutturare e organizzare le idee prima di avviare l'implementazione. La possibilità di avere un supporto immediato per chiarire dubbi e raffinare concetti mi ha permesso di risparmiare tempo e di ottenere una maggiore chiarezza su diversi aspetti del progetto. ChatGPT ha facilitato non solo la parte creativa, ma anche la formalizzazione delle specifiche tecniche e l'organizzazione dei dati.

L'interazione con ChatGPT ha evidenziato l'importanza di porre domande chiare e dettagliate per massimizzare l'efficacia delle risposte. Ho imparato che domande vaghe o mal formulate possono portare a risposte meno utili, sottolineando così la necessità di un'interazione accurata, soprattutto in una fase cruciale come quella dello sviluppo di concetti, dove il focus è più sulla creatività che sulla parte tecnica. Questo processo mi ha aiutato a affinare le mie capacità di comunicazione, un aspetto fondamentale che risulta utile anche al di fuori dell'interazione con un modello linguistico.

In sintesi, ChatGPT si è dimostrato un valido alleato durante la fase di analisi dei requisiti, fornendo supporto non solo nella generazione di idee e nella stesura di documentazione, ma anche nella comprensione delle specifiche tecniche. Quando utilizzato in modo strategico, questo strumento può significativamente migliorare l'efficienza e la qualità del lavoro, rendendo l'intero processo di analisi dei requisiti più fluido e produttivo.

4.2 Implementazione

L'integrazione di ChatGPT nel processo di sviluppo della mia applicazione ha rappresentato un supporto fondamentale in tutte le fasi chiave del progetto, dalla progettazione dell'interfaccia grafica alla gestione del database, fino alla risoluzione dei bug. Grazie alla capacità dell'AI di generare codice, fornire suggerimenti e rispondere a domande specifiche, sono riuscito a ottimizzare il flusso di lavoro, riducendo significativamente i tempi di sviluppo e migliorando la qualità complessiva del prodotto.

Durante la fase di implementazione, l'uso di ChatGPT si è rivelato cruciale per migliorare l'efficienza e l'efficacia del processo di sviluppo software. In particolare, il supporto fornito dal modello ha coperto numerosi aspetti chiave, permettendomi di affrontare la complessità dello sviluppo in modo più organizzato e produttivo.

Nel corso dei paragrafi seguenti verranno esplorate nel dettaglio le modalità con cui ChatGPT ha facilitato la realizzazione di ogni fase dell'implementazione, evidenziando come abbia contribuito a risolvere problemi complessi e a semplificare attività ripetitive e laboriose.

4.2.1 Definizione dei layout

Fin dall'inizio, ChatGPT mi ha assistito nella progettazione dei layout dell'applicazione. Durante questa fase, l'AI mi ha suggerito i tag XML più adatti per dare forma alle schermate, offrendo soluzioni ottimali per rispettare i vincoli imposti dall'analisi dei requisiti. Oltre alla scelta dei tag, ChatGPT ha fornito suggerimenti riguardanti le dimensioni degli elementi, gli spazi interni ed esterni, e persino i codici colore da utilizzare per mantenere un'interfaccia coerente e visivamente piacevole.

Questo tipo di supporto è stato particolarmente utile nel garantire che i layout fossero ordinati, ben bilanciati e user-friendly. La possibilità di ricevere consigli mirati su come strutturare le interfacce mi ha permesso di risparmiare tempo prezioso, evitando errori comuni e mantenendo una coerenza stilistica in tutte le schermate dell'applicazione.

4.2.2 Gestione del database

Uno dei compiti più impegnativi è stato progettare e implementare il database, specialmente considerando la mia esperienza limitata con Firestore, un database NoSQL. ChatGPT è stato determinante nella definizione delle collezioni e dei documenti necessari per la gestione dei dati dell'applicazione. Mi ha aiutato nella creazione dei model per i documenti, suggerendo attributi adeguati e i tipi di dati più appropriati per ciascun attributo.

Inoltre, ChatGPT ha fornito un prezioso supporto nella definizione delle operazioni CRUD (Create, Read, Update, Delete) che avrei dovuto utilizzare per manipolare i dati all'interno del database. Ha anche contribuito alla generazione di query per filtrare i dati in modo efficiente, permettendomi di ottenere esattamente le informazioni richieste, riducendo al minimo il rischio di errori e ottimizzando le performance delle query stesse.

4.2.3 Generazione della logica dell'applicazione

Per quanto riguarda lo sviluppo della logica applicativa, ChatGPT si è rivelato un valido strumento per la generazione di classi ben strutturate e pronte per essere integrate nel progetto. Mi ha fornito esempi di implementazioni complesse, che poi adattavo in base alle esigenze specifiche della mia applicazione. Questo ha accelerato notevolmente il processo di sviluppo, poiché gran parte del lavoro iniziale veniva automatizzata grazie all'AI.

Tuttavia, ho appreso che l'efficacia di ChatGPT dipendeva fortemente dalla chiarezza e precisione delle domande poste. Le risposte erano accurate solo se le richieste erano formulate con tutte le informazioni necessarie. In diverse occasioni, domande vaghe o incomprensibili hanno portato a codice errato o incompleto, costringendomi a riformulare la mia richiesta. Questo mi ha insegnato l'importanza di comunicare in modo preciso e specifico, soprattutto quando si lavora con gli LLM.

4.2.4 Risoluzione dei bug

Un altro aspetto in cui ChatGPT ha avuto un impatto significativo è stato il debugging. Durante lo sviluppo, è inevitabile incontrare errori e bug che richiedono attenzione immediata. In questi casi, era sufficiente il problema, fornendo il messaggio di errore e la porzione di codice rilevante, e ChatGPT mi suggeriva soluzioni per risolverlo. Nella maggior parte dei casi, i suggerimenti forniti dall'AI erano corretti e risolvevano il problema in modo efficace, permettendomi di continuare lo sviluppo senza lunghe interruzioni.

Questo tipo di assistenza ha ridotto drasticamente il tempo che normalmente avrei impiegato a cercare soluzioni su forum o documentazioni tecniche. La velocità con cui ChatGPT forniva risposte dettagliate e pertinenti si è rivelata una risorsa preziosa per mantenere un flusso di lavoro regolare e senza intoppi.

ChatGPT ha anche facilitato l'automazione di numerose operazioni ripetitive, come il riempimento del database con dati di test, la corretta indentazione del codice e l'organizzazione delle classi all'interno di directory e package nel progetto. Queste operazioni, se svolte manualmente, avrebbero richiesto molto più tempo e attenzione, ma con l'aiuto dell'AI sono riuscito a completarle in modo rapido e senza errori. La possibilità di ottenere suggerimenti su come strutturare al meglio il progetto mi ha permesso di mantenere una buona organizzazione del codice e delle risorse, contribuendo alla leggibilità e alla manutenibilità del progetto nel lungo periodo.

4.2.5 Conclusioni sull'implementazione

In conclusione, il contributo di ChatGPT durante la fase di implementazione si è rivelato significativo. La sua capacità di generare codice, suggerire soluzioni a bug complessi e automatizzare attività ripetitive ha notevolmente velocizzato il processo di sviluppo. Tuttavia, ho imparato che l'efficacia dell'AI dipende direttamente dalla qualità delle domande poste: quanto più specifici e dettagliati sono stati i miei input, tanto più pertinenti e utili sono state le risposte.

Avere un supporto come ChatGPT durante lo sviluppo non solo ha ridotto i tempi di produzione, ma mi ha permesso di concentrarmi su aspetti più complessi del progetto,

lasciando che l'automazione si occupasse delle attività più meccaniche. Questo approccio ha reso il processo più fluido, consentendomi di ottenere risultati migliori in tempi ridotti, pur mantenendo un alto livello di qualità nel codice prodotto.

4.3 Testing

Durante la fase di testing dell'applicazione, mi sono affidato completamente a ChatGPT, poiché non avevo esperienza pregressa con la creazione di test e questa rappresentava per me un aspetto totalmente nuovo dello sviluppo software. Ho utilizzato ChatGPT per esplorare i vari tipi di test che avrei potuto implementare, tra cui unit test, integration test e instrumented test.

L'idea era di ottenere una panoramica delle metodologie disponibili e delle best practices da seguire. Successivamente, ho impiegato il modello per definire le classi di test.

Dopo aver acquisito una comprensione di base, ho richiesto a ChatGPT di aiutarmi a definire le classi di test necessarie. Tuttavia, ho rapidamente notato che le classi generate non sempre producevano risultati accurati o soddisfacenti. In molti casi, le implementazioni suggerite richiedevano modifiche alle classi della mia applicazione, suggerendo cambiamenti che non erano giustificati, poiché le classi originali erano già corrette. In aggiunta, ho riscontrato che ChatGPT confondeva frequentemente i tipi di test richiesti. Spesso, quando chiedevo test instrumented, il modello restituiva unit test e viceversa, mescolando anche le librerie da utilizzare. Questo ha complicato ulteriormente il lavoro. È stato chiaro che, sebbene ChatGPT fosse uno strumento utile per la generazione di idee e suggerimenti, non poteva sostituire la necessaria competenza umana nella comprensione e nell'implementazione di test efficaci.

Questo processo mi ha insegnato l'importanza di avere una solida conoscenza dei principi di testing e delle strategie appropriate per l'applicazione. Mi sono reso conto che, mentre strumenti come ChatGPT possono generare codice e offrire spunti creativi, la supervisione umana è fondamentale per garantire che le implementazioni siano corrette e pertinenti.

In conclusione, sebbene ChatGPT abbia fornito un valido supporto nell'avviare la fase di testing, la sua efficacia è stata limitata dalla mancanza di precisione nelle risposte e dalla confusione sui tipi di test. Questo ha reso la fase di testing più impegnativa del previsto, spingendomi a dedicare tempo extra per esaminare e correggere i suggerimenti ricevuti. Tuttavia, questa esperienza ha anche evidenziato l'importanza della preparazione e della competenza nel testing software. In futuro, ritengo che sia fondamentale combinare l'uso di strumenti come ChatGPT con una solida formazione e pratica nel testing, per ottimizzare l'efficacia del processo di sviluppo e garantire la qualità dell'applicazione.

4.4 Conclusioni e statistiche

Durante lo sviluppo di BicoccaHelp, ChatGPT ha svolto un ruolo centrale in tutte le fasi del ciclo di vita del software, dalla progettazione alla risoluzione dei bug. Le informazioni riportate in questa sezione sono state raccolte tramite un diario dettagliato, in cui ho annotato ogni interazione con ChatGPT. Complessivamente, ho registrato più di 50 sessioni di chat. Le interazioni sono state così distribuite: 12 richieste per la fase di **Requisiti**, 5 per il **Design**, 40 per l'**Implementazione** e 23 per il **Testing e Fixing**. Questo evidenzia che circa il 50% delle richieste ha riguardato l'implementazione, mentre il 28% è stato dedicato al testing e alla risoluzione dei problemi. Le interazioni relative alla progettazione dell'interfaccia grafica rappresentano circa il 6%, mentre le restanti richieste per i requisiti costituiscono il 15%.

L'utilizzo di ChatGPT si è rivelato particolarmente utile nelle fasi di implementazione e debugging, contribuendo a ridurre significativamente i tempi di sviluppo. Tuttavia, ho riscontrato alcune limitazioni nella fase di testing, dove è stato necessario correggere manualmente la maggiorparte dei test automatizzati suggeriti. Il diario dettagliato, che segna ogni passo delle chat avute con ChatGPT, mi ha permesso di tenere traccia dei risultati ottenuti e di valutare con precisione l'impatto del suo utilizzo sul progetto.

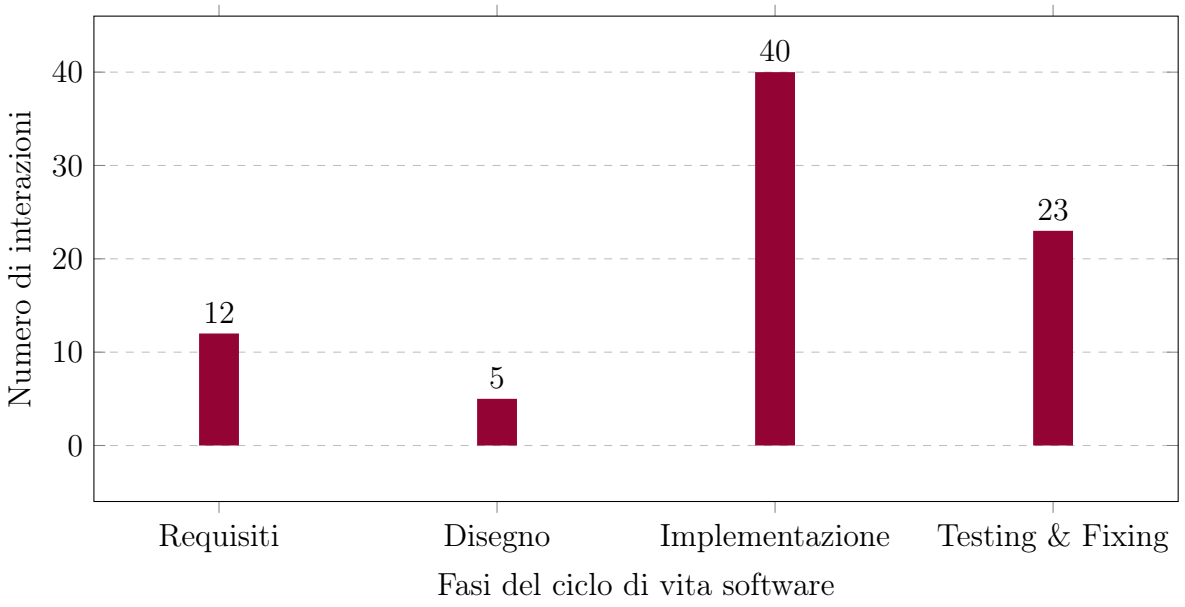


Figura 2: Interazioni con ChatGPT durante le fasi di sviluppo software

5 Conclusioni

L'utilizzo di ChatGPT nello sviluppo dell'applicazione BicoccaHelp ha permesso di valutare concretamente come l'intelligenza artificiale può supportare e semplificare il processo di sviluppo software. In questo contesto, uno degli esiti più rilevanti è stata la capacità di ChatGPT di accelerare le fasi di implementazione, offrendo suggerimenti di codice, risolvendo bug e contribuendo alla stesura della documentazione. Tuttavia, questa esperienza ha anche messo in luce alcuni limiti, soprattutto nella generazione di test accurati e nel contesto del debugging più complesso, dove l'intervento umano rimane essenziale.

Uno dei problemi emersi durante l'utilizzo di ChatGPT è legato alla formulazione delle domande. Per ottenere risposte utili e precise, è stato necessario porre le domande in modo corretto e fornire informazioni sufficienti e chiare. In alcuni casi, un'informazione mancante o una domanda ambigua ha condotto a risposte meno efficaci, costringendo a riformulare il quesito più volte fino a raggiungere una soluzione ottimale. Questo ha evidenziato l'importanza della chiarezza e della precisione nella comunicazione con un modello di intelligenza artificiale.

Il contributo principale di questa tesi consiste nella valutazione dell'efficacia di ChatGPT come strumento di supporto nello sviluppo di un'app mobile. Ho affrontato problematiche concrete tipiche di un progetto software reale. Invece di limitarsi ad una valutazione teorica, l'app BicoccaHelp ha rappresentato un caso di studio pratico, dimostrando come gli sviluppatori potrebbero integrare un modello di intelligenza artificiale nel flusso di lavoro quotidiano. Rispetto alle metodologie tradizionali di sviluppo, l'utilizzo di ChatGPT ha offerto vantaggi tangibili, in particolare per quanto riguarda l'efficienza nella scrittura del codice e nella risoluzione dei problemi riscontrati.

Nel complesso, l'integrazione di ChatGPT si è rivelata un valido alleato, con evidenti risultati nella riduzione dei tempi di sviluppo e nella gestione delle complessità legate alla progettazione dell'interfaccia utente e all'organizzazione dei dati. Le funzionalità sviluppate nell'applicazione, come la gestione delle prenotazioni e dei profili, dimostrano che strumenti come ChatGPT rappresentano un supporto fondamentale per semplificare la realizzazione di progetti complessi.

In conclusione, il presente progetto ha dimostrato che ChatGPT, pur con le sue attuali limitazioni, si configura come uno strumento di grande valore per il futuro dello sviluppo software. Con continui miglioramenti, potrebbe evolvere da strumento di supporto a parte integrante dei processi di sviluppo, capace di gestire anche fasi critiche come il testing e la correzione del codice. BicoccaHelp, nel suo stato attuale, costituisce un buon esempio di come l'intelligenza artificiale possa essere integrata con successo in progetti reali, ponendo le basi per futuri sviluppi e ampliamenti.

Riferimenti bibliografici

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5998–6008, 2017.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. *arXiv preprint arXiv:2005.14165*, 2020.

[3] OpenAI. How ChatGPT and Our Language Models Are Developed, 2024.

[4] Microsoft. Model-View-ViewModel (MVVM), 2024.

[5] Google. Firestore Documentation, 2024.

[6] Google. Material Design Guidelines, 2024.