

# Analisi dei requisiti

Gruppo 02

December 14, 2024

## 1 Introduzione

Il programma implementa una rubrica per gestire i contatti telefonici e/o e-mail. Il programma consente agli utenti registrati di inserire, eliminare o modificare dei contatti all'interno della rubrica. Permette di effettuare un login con email e password e a nuovi utenti di registrarsi.

## 2 Requisiti

### 2.1 Requisiti funzionali

- **Funzionalità individuali:**

**IF-1** Il sistema deve permettere agli utenti non registrati di registrarsi inserendo un'e-mail e una password.

**IF-2** Il sistema deve permettere agli utenti registrati di accedere tramite e-mail e password. Dopo l'accesso vengono visualizzati i contatti salvati in precedenza.

**IF-3** Il sistema deve permettere agli utenti registrati di inserire un nuovo contatto.

**IF-4** Il sistema notifica con un messaggio se l'utente tenta di aggiungere un contatto duplicato. Due contatti sono uguali se hanno lo stesso nome e cognome.

**IF-5** Il sistema deve permettere agli utenti registrati di eliminare un contatto selezionato.

**IF-6** Il sistema deve permettere agli utenti di modificare le informazioni di un contatto selezionato.

**IF-7** Il sistema deve permettere di ricercare un contatto tramite una sottostringa inserita in una barra di ricerca.

**IF-8** Il sistema consente di aggiungere tag ai contatti.

**IF-9** Il sistema consente di filtrare i contatti tramite la selezione di un tag.

**IF-10** Il sistema deve permettere di ordinare le righe della tabella in base a nome o cognome e con un ordinamento che può essere crescente o decrescente.

**IF-11** Il sistema deve permettere agli utenti registrati di importare i contatti da un file o esportare i contatti in un file.

- **Esigenze di Dati e informazioni:**

**DF-1** Dati del profilo utente:

- e-mail
- password

**DF-2** Dati dei contatti:

- nome e cognome (almeno uno dei due campi deve essere non vuoto)
- da zero a tre numeri di telefono
- da zero a tre indirizzi e-mail
- da zero a tre tag

- **Interfaccia utente:**

**UI-1** Pagina di login e registrazione dove inserire l'e-mail e la password per il login o l'e-mail e la password (e conferma password) per la registrazione.

**UI-2** La rubrica viene visualizzata in forma tabellare. Ogni colonna della tabella corrisponde a un campo del generico contatto. Ogni riga della tabella corrisponde a un singolo contatto.

**UI-3** La selezione di un contatto avviene cliccando la corrispondente riga con il mouse.

**UI-4** Nella pagina principale della rubrica sono presenti diversi bottoni e menu per eseguire le operazioni.

**UI-5** Messaggi di errore: in caso di errore nelle informazioni inserite dall'utente l'interfaccia deve indicare quale informazione è errata ed eventuali vincoli non rispettati.

**UI-6** Se l'utente prova a inserire un contatto già esistente compare un messaggio di notifica.

- **Interfacce con sistemi esterni:**

**IS-1** Database per salvare le informazioni.

### 3 Vincoli di progettazione

1. Il sistema deve essere implementato tramite un'applicazione Java compatibile con Maven.
2. La consegna del progetto è prevista per il 15/12/2024.

## 4 Casi d'uso

### 4.1 Attori

- Utente registrato: può effettuare il login; una volta effettuato il login, visualizza la pagina principale della rubrica contenente i contatti salvati.
- Utente: può registrarsi inserendo i dati richiesti.
- Database: gestisce il servizio di autenticazione e di memorizzazione dei contatti.

### 4.2 Descrizione dei casi d'uso

#### REGISTRAZIONE

**Attori partecipanti:** utente, database

**Precondizioni:**

- L'utente apre l'applicazione

**Postcondizioni:**

- L'utente ha un account e diventa un utente registrato

**Flusso di eventi:**

1. L'utente inserisce e-mail e password (e conferma password)
2. Verifica della correttezza dei dati inseriti
3. I dati vengono inseriti nel database
4. Registrazione completata

**Flusso di eventi alternativo:**

- 3.a L'indirizzo e-mail è già presente nel database. L'utente deve effettuare il login.
- 3.b La password inserita non rispetta le condizioni di sicurezza consigliate; si richiede l'inserimento di una password differente.
- 3.c Le password non corrispondono.

## LOGIN

**Attori partecipanti:** utente registrato, database

**Precondizioni:**

- L'utente deve essere registrato

**Postcondizioni:**

- L'utente ha accesso alla homepage della rubrica

**Flusso di eventi:**

1. L'utente registrato inserisce e-mail e password.
2. Verifica delle credenziali (nel database).
3. Verifica andata a buon fine.

**Flusso di eventi alternativo:**

- 3.a E-mail non presente nel database → Registrazione.
- 3.b Credenziali errate

## LOGOUT

**Attori partecipanti:** utente registrato, database

**Precondizioni:**

- L'utente ha effettuato il login

**Postcondizioni:**

- L'utente torna alla schermata iniziale

**Flusso di eventi:**

1. L'utente preme sul bottone per uscire dall'account.
2. Viene chiusa la connessione col database.

## **OPERAZIONI RUBRICA**

**Attori partecipanti:** utente registrato

**Precondizioni:**

- L'utente ha eseguito il login

**Postcondizioni: Flusso di eventi:**

1. L'utente visualizza la pagina principale della rubrica e sceglie quale operazione effettuare

**Flusso di eventi alternativo:**

- 1.a L'utente seleziona un contatto e l'operazione da effettuare su di esso.

## **AGGIUNGERE CONTATTO**

**Attori partecipanti:** utente registrato

**Precondizioni:**

- L'utente preme sul bottone per l'aggiunta di un contatto

**Postcondizioni:**

- Un contatto viene aggiunto alla rubrica

**Flusso di eventi:**

1. L'utente specifica le informazioni da associare al nuovo contatto.
2. Le modifiche vengono salvate.

**Flusso di eventi alternativo:**

- 2.a I campi nome e cognome sono entrambi vuoti; il contatto non viene inserito in rubrica.
- 2.b Contatto duplicato.
- 2.c L'utente annulla l'operazione.

## **RIMUOVERE CONTATTO**

**Attori partecipanti:** utente registrato

**Precondizioni:**

- Nella rubrica è presente almeno un contatto.
- L'utente indica che vuole rimuovere un contatto tramite l'apposito bottone.

**Postcondizioni:**

- Il contatto selezionato è eliminato dalla rubrica

**Flusso di eventi:**

1. Il contatto viene rimosso dalla rubrica.
2. Le modifiche vengono salvate.

## **MODIFICARE INFORMAZIONI PER UN CONTATTO GIÀ ESISTENTE**

**Attori partecipanti:** utente registrato

**Precondizioni:**

- Nella rubrica è presente almeno un contatto
- L'utente indica di voler modificare un contatto tramite l'apposito bottone

**Postcondizioni:**

- Le informazioni del contatto selezionato sono modificate

**Flusso di eventi:**

1. L'utente modifica le informazioni del contatto selezionato.
2. Le modifiche vengono salvate.

**Flusso di eventi alternativo:**

- 2.a Non è possibile salvare la modifica effettuata perché sono stati eliminati sia il nome che il cognome.
- 2.b Contatto duplicato.
- 2.c L'utente annulla l'operazione.

## **RICERCA CONTATTI**

**Attori partecipanti:** utente registrato

**Precondizioni:**

- L'utente seleziona la barra di ricerca o un tag dal menu

**Postcondizioni:**

- L'utente visualizza solo i contatti corrispondenti ai criteri di ricerca.

**Flusso di eventi:**

1. L'utente inserisce la sottostringa da ricercare e/o seleziona il/i tag.

## **IMPORTARE CONTATTI**

**Attori partecipanti:** utente registrato

**Precondizioni:**

- L'utente preme il tasto per importare i contatti.

**Postcondizioni:**

- I contatti da importare sono aggiunti alla rubrica.

**Flusso di eventi:**

1. L'utente seleziona il file contenente i contatti da importare.
2. I contatti vengono aggiunti.

**Flusso di eventi alternativo:**

2.a Errore di importazione: messaggio d'errore.

2.b L'utente annulla l'operazione.

## **ESPORTARE CONTATTI**

**Attori partecipanti:** utente registrato

**Precondizioni:**

- L'utente preme il tasto per esportare i contatti.

**Postcondizioni:**

- I contatti presenti in rubrica vengono esportati nel file indicato.

**Flusso di eventi:**

1. L'utente seleziona il file in cui esportare i contatti.
2. I contatti vengono esportati.

**Flusso di eventi alternativo:**

2.a Errore di esportazione: messaggio d'errore.

2.b L'utente annulla l'operazione.

## **ORDINARE CONTATTI**

**Attori partecipanti:** utente registrato

**Precondizioni:**

- L'utente indica un criterio di ordinamento cliccando su uno dei campi della tabella

**Postcondizioni:**

- I contatti della rubrica sono visualizzati secondo il criterio scelto indicato

**Flusso di eventi:**



1. L'ordinamento è stato effettuato correttamente.

## **CONTATTO DUPLICATO**

**Attori partecipanti:** utente registrato

**Precondizioni:**

- L'utente sta aggiungendo/modificando un contatto con nome e cognome già associati a un altro contatto.

**Postcondizioni:**

- La rubrica viene aggiornata.

**Flusso di eventi:**

1. Compare un messaggio di scelta.
2. Viene aggiunto/modificato un contatto.

**Flusso di eventi alternativo:**

- 2.a.1 L'utente torna a modificare le informazioni del contatto.
- 2.a.2 L'utente salva le modifiche.
- 2.b L'utente annulla l'operazione.

## **SALVATAGGIO**

**Attori partecipanti:** utente registrato, database

**Precondizioni:**

- L'utente indica di voler salvare una modifica.

**Postcondizioni:**

- Le modifiche vengono salvate.

**Flusso di eventi:**

1. La rubrica aggiornata viene salvata.

## 5 Descrizione dei requisiti

ID	Stato del requisito	Priorità	Rischio Tecnico
IF-1	Accettato per questa release	Alta	Medio
IF-2	Accettato per questa release	Alta	Basso
IF-3	Accettato per questa release	Alta	Basso
IF-4	Accettato per questa release	Bassa	Basso
IF-5	Accettato per questa release	Alta	Basso
IF-6	Accettato per questa release	Media	Basso
IF-7	Accettato per questa release	Media	Basso
IF-8	Accettato per questa release	Bassa	Basso
IF-9	Accettato per questa release	Bassa	Basso
IF-10	Accettato per questa release	Media	Basso
IF-11	Accettato per questa release	Medio	Medio
DF-1	Accettato per questa release	Alta	Basso
DF-2	Accettato per questa release	Alta	Basso
UI-1	Accettato per questa release	Alta	Basso
UI-2	Accettato per questa release	Alta	Basso
UI-3	Accettato per questa release	Alta	Basso
UI-4	Accettato per questa release	Alta	Basso
UI-5	Accettato per questa release	Media	Basso
UI-6	Accettato per questa release	Bassa	Basso
IS-1	Accettato per questa release	Alta	Alto

*Tabella dei requisiti*

## 6 Diagramma dei casi d'uso

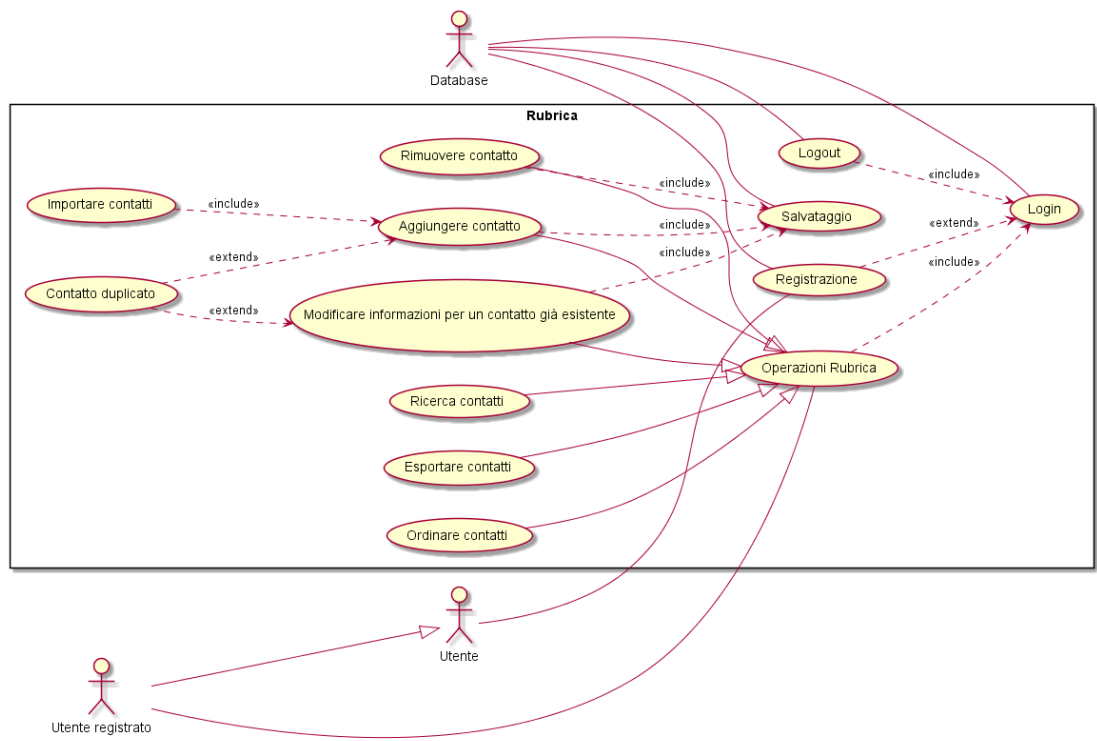


Figure 1: Diagramma UML

## 7 Attributi di qualità

### 7.1 Esterni

- **Sicurezza:** Il sistema nell'inserimento dei dati sul database effettua una verifica che rende difficile l'attacco tramite session hijacking. Le password degli utenti salvate su database sono crittografate.
- **Scalabilità:** Il sistema è stato progettato per supportare l'espansione senza degradare significativamente le prestazioni; infatti, l' utilizzo del database permette la restituzione di più record in tempi ottimali.  
Nota: Attualmente, il prototipo è ospitato su un server con capacità limitate. Tuttavia, la progettazione del sistema prevede la possibilità di migrare a server ad alte prestazioni e distribuiti in futuro, garantendo così una gestione efficiente dell'aumento del carico di lavoro.)
- **Usabilità:** Il programma dispone di un'interfaccia grafica che consente agli utenti di accedere facilmente a tutte le funzionalità. I contatti possono essere classificati in tre categorie: job, university e home. Tramite un apposito menù, è possibile visualizzare solo i contatti appartenenti ai tag selezionati. Inoltre, una barra di ricerca permette di filtrare i contatti, mostrando solo quelli che contengono la parola inserita in uno qualsiasi dei campi.
- **Disponibilità:** l'utente per accedere ai dati della rubrica può collegarsi alla rete Internet, avviare l' applicazione su qualsiasi PC, o laptop, effettuare l'accesso e consultare i propri contatti.

### 7.2 Interni

- **Manutenibilità:** Il sistema organizza i dati in un database facilitando l'implementazione di future ed eventuali nuove funzioni (Ad esempio la creazione di una web app che permetterà agli utenti di collegarsi tramite browser). Il sistema è facilmente manutenibile grazie ai livelli di coesione, che sono quasi tutti funzionali ad eccezione di uno e ai livelli di accoppiamento, che sono alternativamente per dati o per comunicazione.
- **Portabilità:** L'applicazione è stata sviluppata in Java, garantendo la compatibilità con tutti i principali sistemi operativi (Windows, macOS, Linux, ecc.). Grazie alla Java Virtual Machine (JVM), il codice può essere eseguito senza modifiche su qualsiasi piattaforma che la supporti.

## 8 Diagrammi delle classi

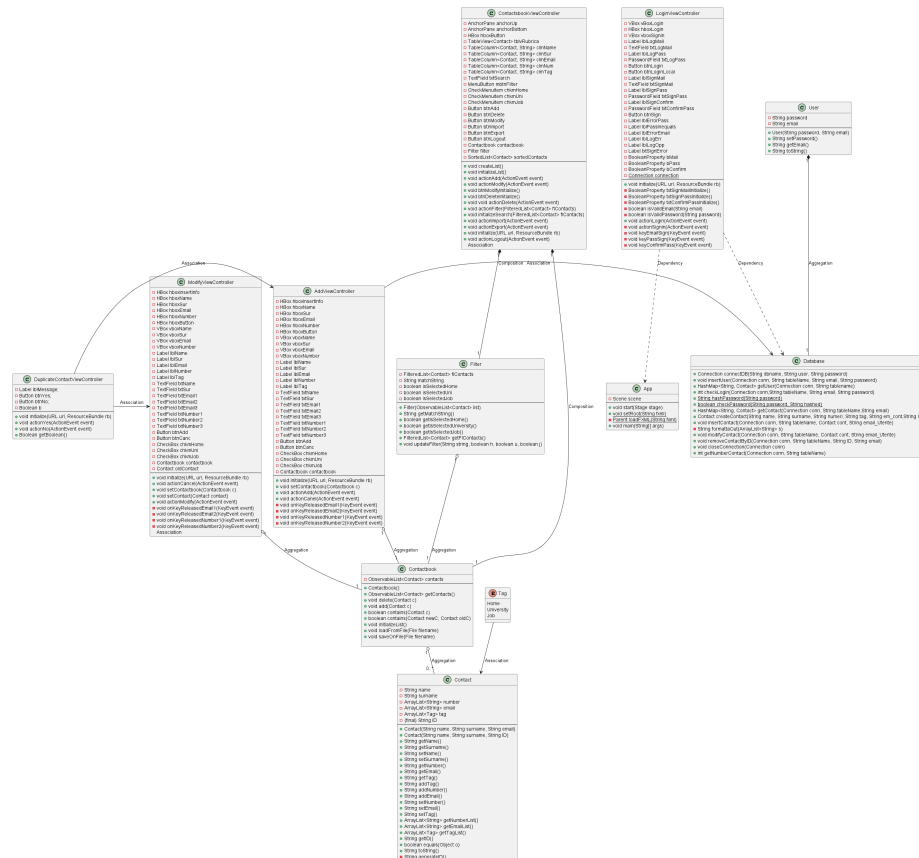


Figure 2: Diagramma delle classi

**Nota:** Il campo `getUser` nella classe `Database` è di tipo `HashMap<String,String>`, mentre il campo `getContact` è di tipo `HashMap<String,Contact>`.  
I campi `TableColumn` in `ContactsbookViewController` sono di tipo `TableColumn<Contact,String>` ad eccezione di `clmTag` che è di tipo `TableColumn<Contact,Tag>`.

Il diagramma completo delle classi è molto complesso e contiene informazioni sulla struttura dell'interfaccia grafica (HBox, VBox, ...) che ne rendono più difficile la lettura, senza apportare informazioni rilevanti. Di seguito è quindi proposta una versione snellita del diagramma delle classi dove non sono

presenti tali attributi.

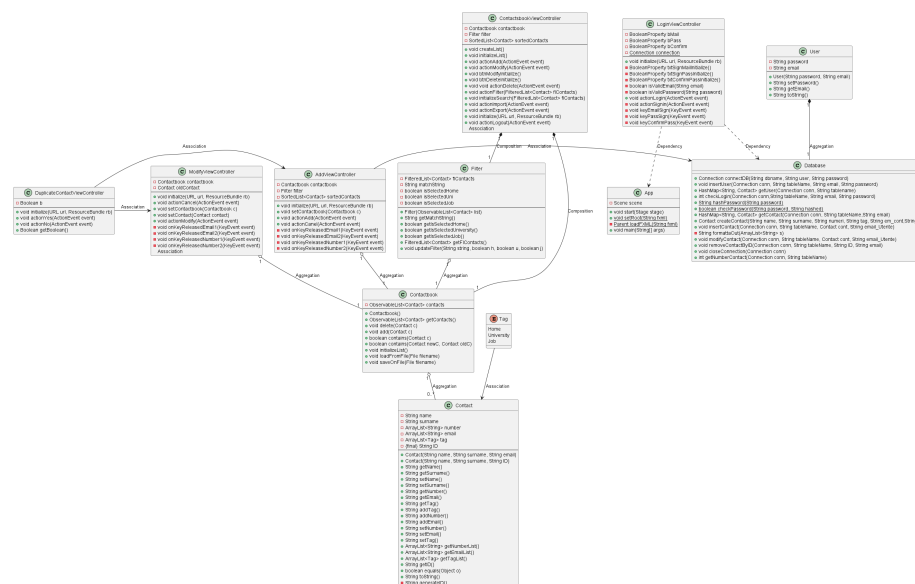


Figure 3: Diagramma delle classi in versione alternativa

## 9 Diagrammi delle sequenze

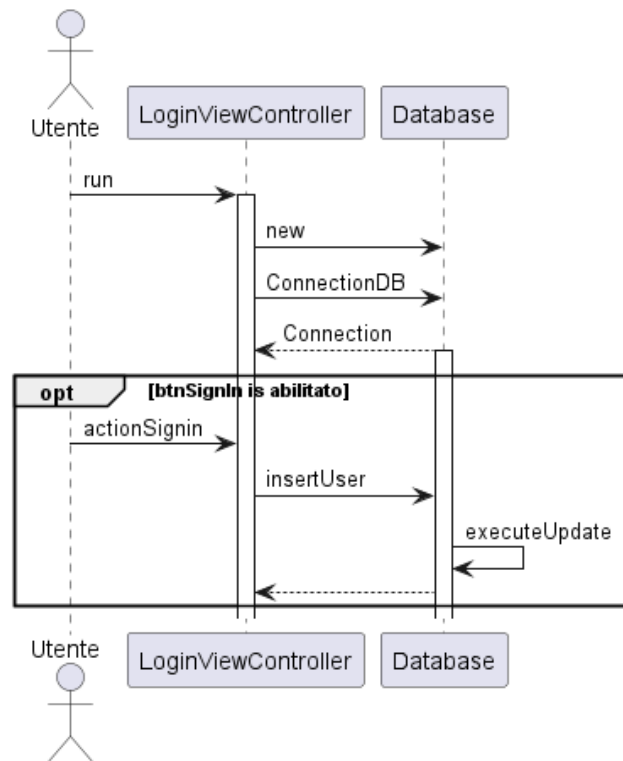


Figure 4: Diagramma delle sequenze relativo alla registrazione di un nuovo utente.

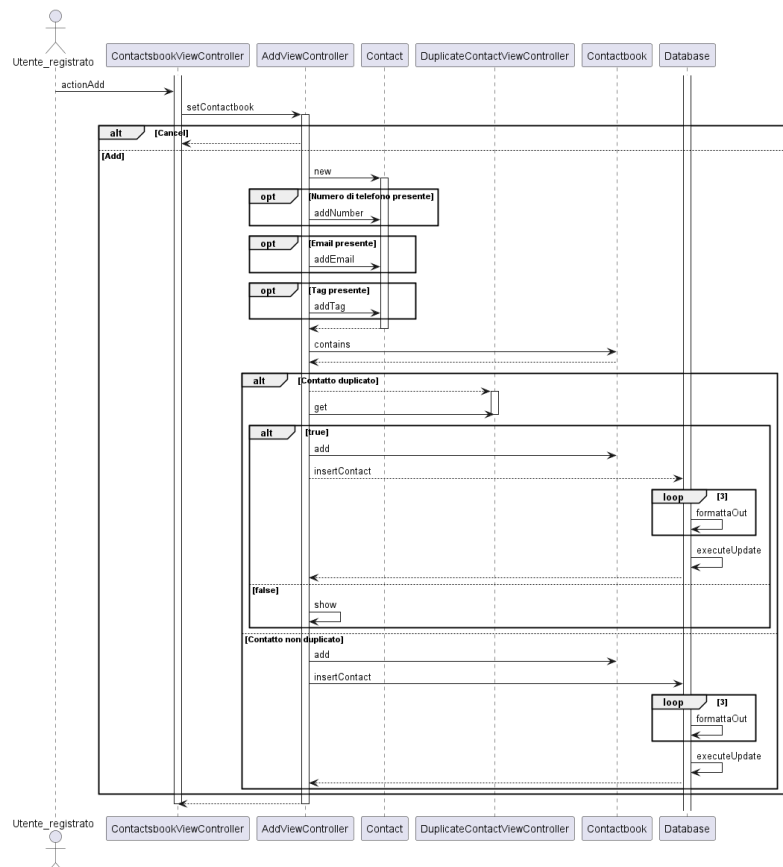


Figure 5: Diagramma delle sequenze relativo al login.



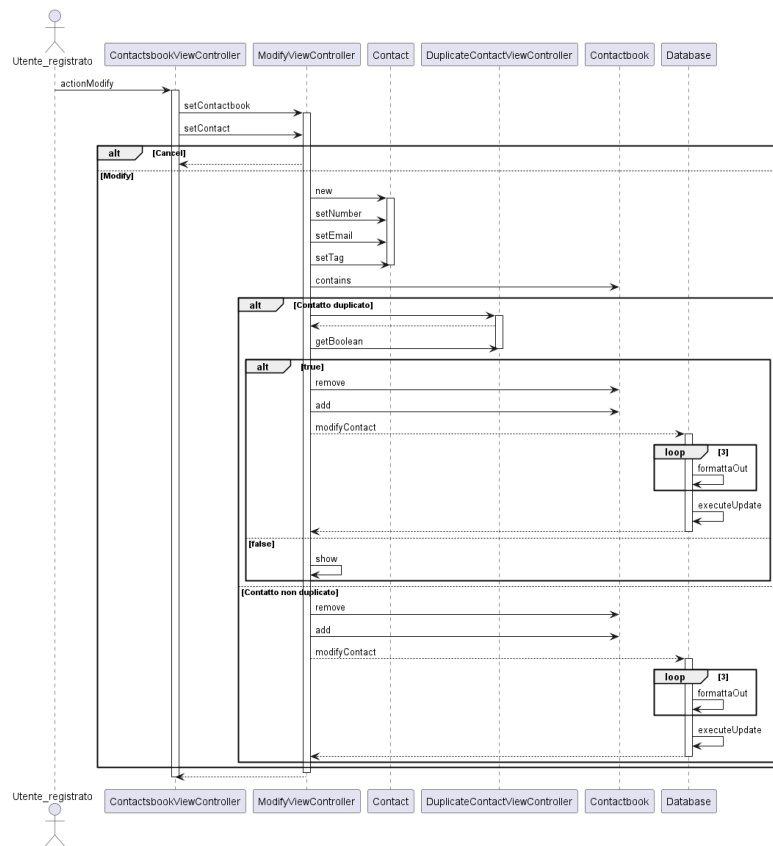


Figure 6: Diagramma delle sequenze relativo alla modifica di un contatto già esistente.

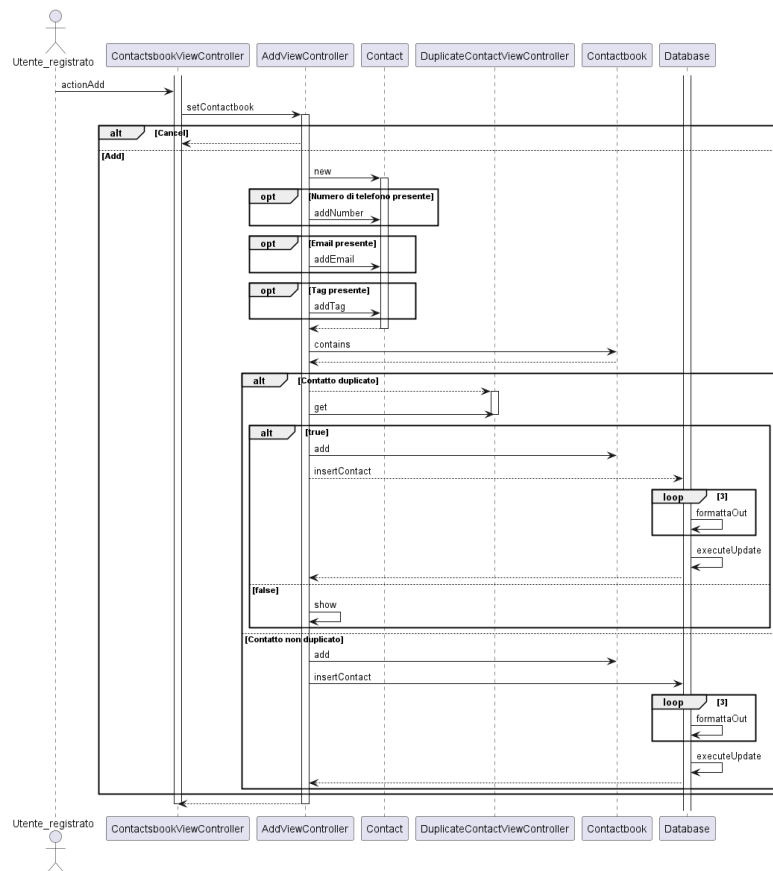


Figure 7: Diagramma delle sequenze relativo all'aggiunta di un nuovo contatto.

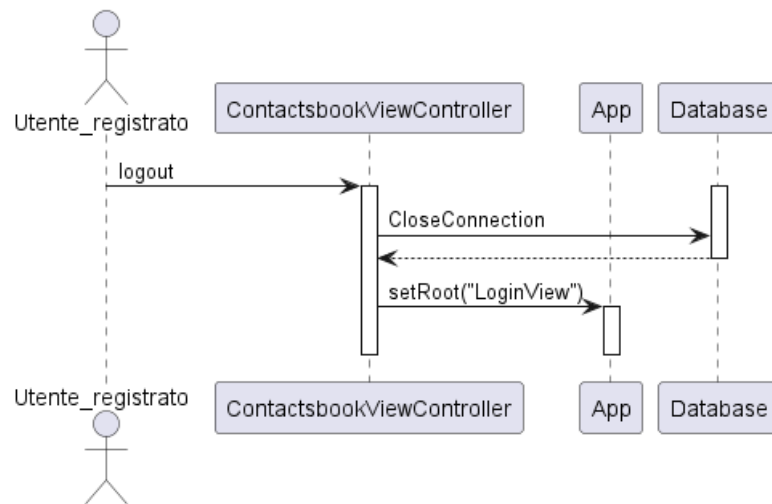


Figure 8: Diagramma delle sequenze relativo al logout.

## 10 Analisi di coesione e accoppiamento

Di seguito sono presenti le tabelle che analizzano coesione e accoppiamento di varie classi.

CLASSE	COESIONE
AddViewController	Funzionale
<b>MOTIVAZIONE</b>	
La classe è responsabile di un singolo compito, ovvero gestire le interazioni dell'utente nella schermata per aggiungere un nuovo contatto. Tutti i metodi e i campi sono strettamente correlati a questo obiettivo: <ul style="list-style-type: none"> <li>I metodi come actionAdd e actionCancel implementano le azioni dell'utente.</li> <li>Gli elementi FXML definiti (ad esempio, txtName, btnAdd, btnCanc) sono usati per interagire con i dati immessi dall'utente.</li> </ul>	

CLASSE	TIPO DI COESIONE
LoginViewController	Funzionale
<b>GIUSTIFICAZIONE</b>	
La classe è responsabile di un singolo compito, ovvero gestire la logica relativa alla pagina di login e registrazione, incluse validazione dei campi, interazioni con il database e gestione dell'interfaccia utente. Ogni metodo è strettamente correlato a questo obiettivo: <ul style="list-style-type: none"> <li>Initialize: configura la schermata e i binding per la disabilitazione dei pulsanti</li> <li>txtSignMailInitialize, txtSignPassInitialize, txtConfirmPassInitialize: gestiscono la validazione e gli aggiornamenti dei campi visivi</li> <li>actionLogin, actionSignIn: implementano operazioni specifiche per login e registrazione</li> </ul>	

CLASSE	TIPO DI COESIONE
DuplicateContactViewController	Funzionale
<b>GIUSTIFICAZIONE</b>	
La classe ha il compito specifico di mostrare all'utente un'interfaccia per decidere come gestire un contatto duplicato (aggiungere o modificare) e agire in base alla scelta I metodi implementati rispecchiano questo obiettivo: <ul style="list-style-type: none"> <li>actionYes: Gestisce il caso in cui l'utente sceglie di aggiungere il contatto.</li> <li>actionNo: Gestisce il caso in cui l'utente sceglie di non aggiungere il contatto, ma tornare a modificare le informazioni associate.</li> <li>getBoolean: Restituisce il booleano che indica la scelta dell'utente.</li> </ul>	

CLASSE	TIPO DI COESIONE
ModifyViewController	Funzionale
<b>GIUSTIFICAZIONE</b>	
La classe è responsabile di gestire la schermata di modifica di un contatto. Nello specifico: <ul style="list-style-type: none"> <li>Modifica delle informazioni del contatto.</li> <li>Gestione dei contatti duplicati.</li> </ul>	

CLASSE	TIPO DI COESIONE
ContactsbookViewController	Comunicazionale
<b>GIUSTIFICAZIONE</b>	
La classe gestisce la logica dell'interfaccia utente della rubrica, coordinando operazioni come l'aggiunta, la modifica, l'eliminazione e la visualizzazione dei contatti, la gestione della ricerca e dei filtri, e l'importazione/esportazione dei dati.	

CLASSE	TIPO DI COESIONE
Contact	Funzionale
<b>GIUSTIFICAZIONE</b>	
La classe Contact è focalizzata sul modellare un contatto con nome, cognome, numeri di telefono, email, tag e un identificatore univoco (ID). Le operazioni che svolge sono tutte correlate a questa funzione centrale, come l'aggiunta di numeri di telefono, email e tag.	

CLASSE	TIPO DI COESIONE
Contactbook	Comunicazionale
<b>GIUSTIFICAZIONE</b>	
La classe si occupa di gestire una lista di contatti e le operazioni da eseguire su essa. Le funzioni lavorano sulla stessa lista.	

CLASSE	TIPO DI COESIONE
User	Funzionale
<b>GIUSTIFICAZIONE</b>	
La classe User gestisce le informazioni di un utente, con un focus principale sulla password e sull'email. Le operazioni fornite sono coerenti con questo scopo, come setPassword per modificare la password (per release future) e getEmail per ottenere l'email.	

CLASSE	TIPO DI COESIONE
Filter	Funzionale
<b>GIUSTIFICAZIONE</b>	
La classe Filter si occupa solo di definire la logica del filtro.	

CLASSE	TIPO DI COESIONE
App	Funzionale
<b>GIUSTIFICAZIONE</b>	
La classe si occupa di configurare la scena principale, di caricare il file FXML per la vista di login, e di impostare il titolo e l'icona dell'applicazione. Queste azioni sono tutte necessarie per l'avvio dell'applicazione, quindi sono coerenti con il suo ruolo di gestione dell'interfaccia utente.	

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
LoginViewController, ContactsbookViewController	Nessun accoppiamento
<b>GIUSTIFICAZIONE</b>	
I due controller comunicano solo attraverso il database per stabilire quale utente ha effettuato l'accesso, quindi non dipendono direttamente l'uno dall'altro.	

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
ContactsbookViewController, AddViewController e ModifyViewController	Dati
<b>GIUSTIFICAZIONE</b>	
La classe ContactsbookViewController passa alle classi AddViewController e ModifyViewController la struttura su cui queste ultime devono effettuare delle operazioni.	

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
DuplicateContactViewController, AddViewController, ModifyViewController	Controllo
<b>GIUSTIFICAZIONE</b>	
DuplicateContactViewController restituisce un booleano in base al quale cambia il comportamento di AddViewController o ModifyViewController	

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
ContactsbookViewController, Contactbook, Filter	Dati
<b>GIUSTIFICAZIONE</b>	
Le tre classi condividono le liste contenenti i contatti su cui operare.	

CLASSI INTERESSATE	TIPO DI ACCOPPIAMENTO
ContactsbookViewController, LoginViewController, AddViewController, ModifyViewController, Database	Dati
<b>GIUSTIFICAZIONE</b>	
I vari controller ricevono dati dal database (es: i contatti associati a un utente) e inviano nuove informazioni/modificano quelle già presenti.	