Documentazione progettazione di dettaglio

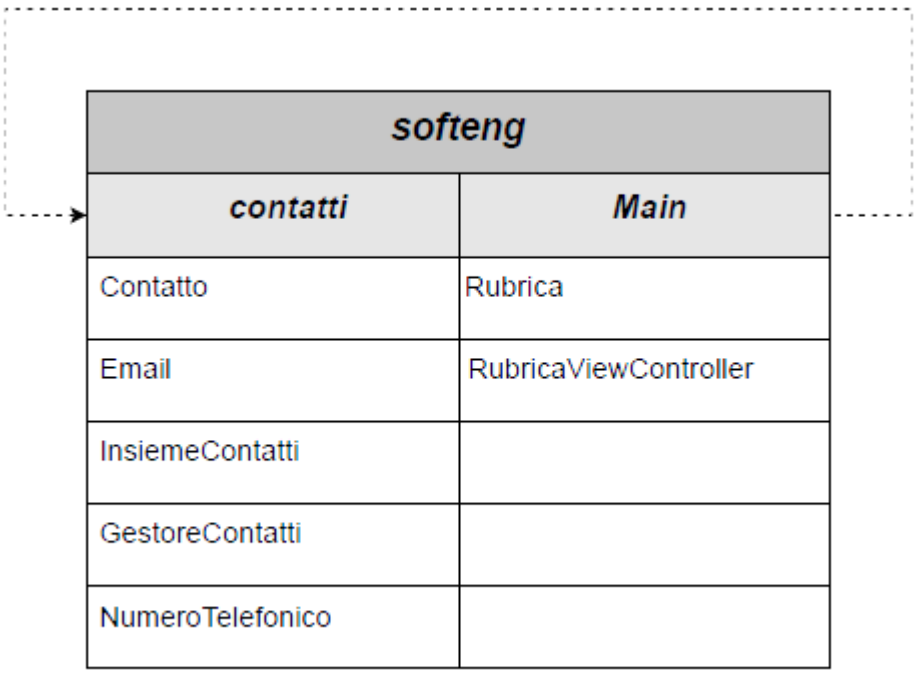
Indice:

1. Diagramma dei package

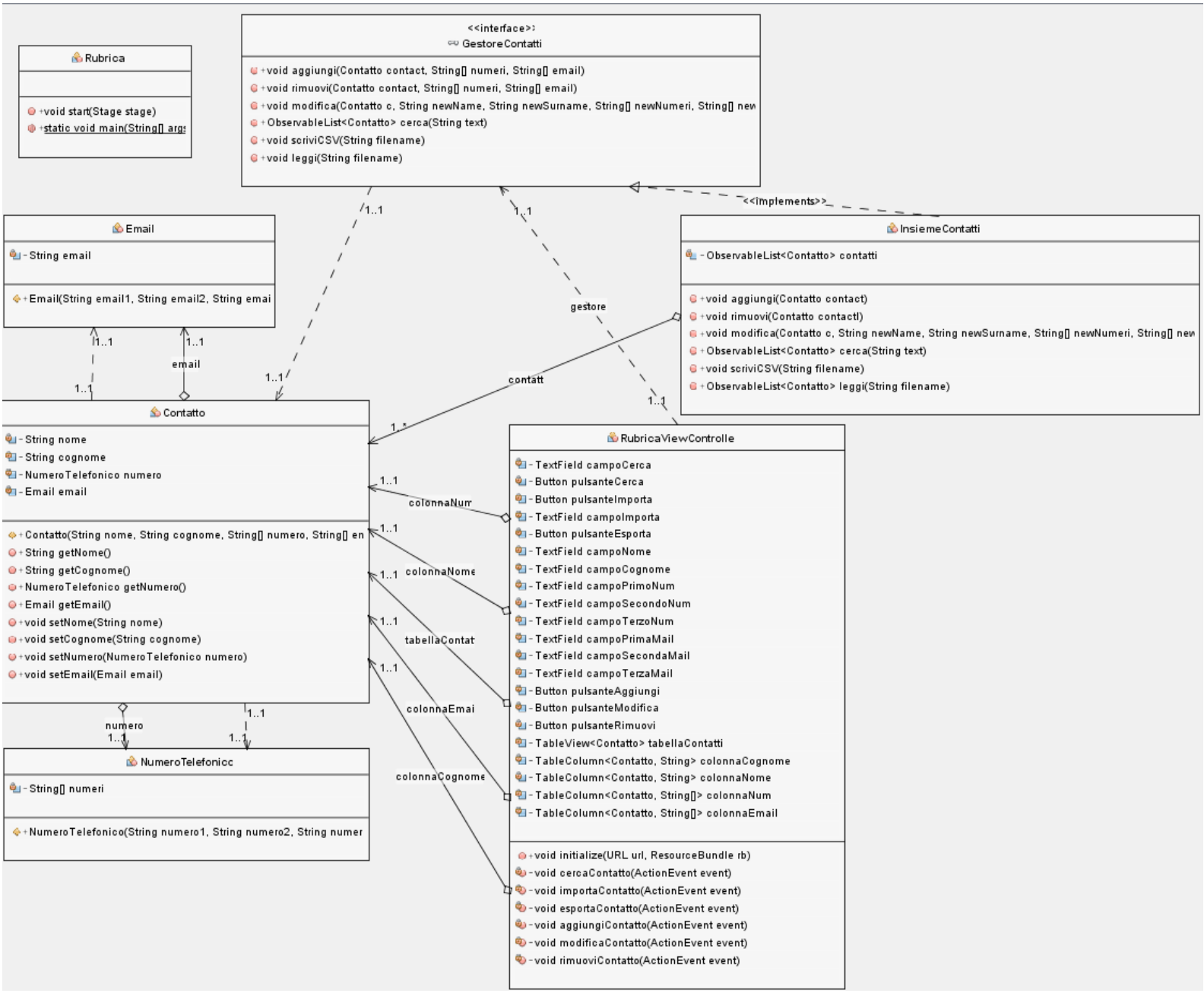
VISTA GENERALE

VISTA IN DETTAGLIO

con dipendenze



2. Diagramma delle classi



Le classi implementate sono:

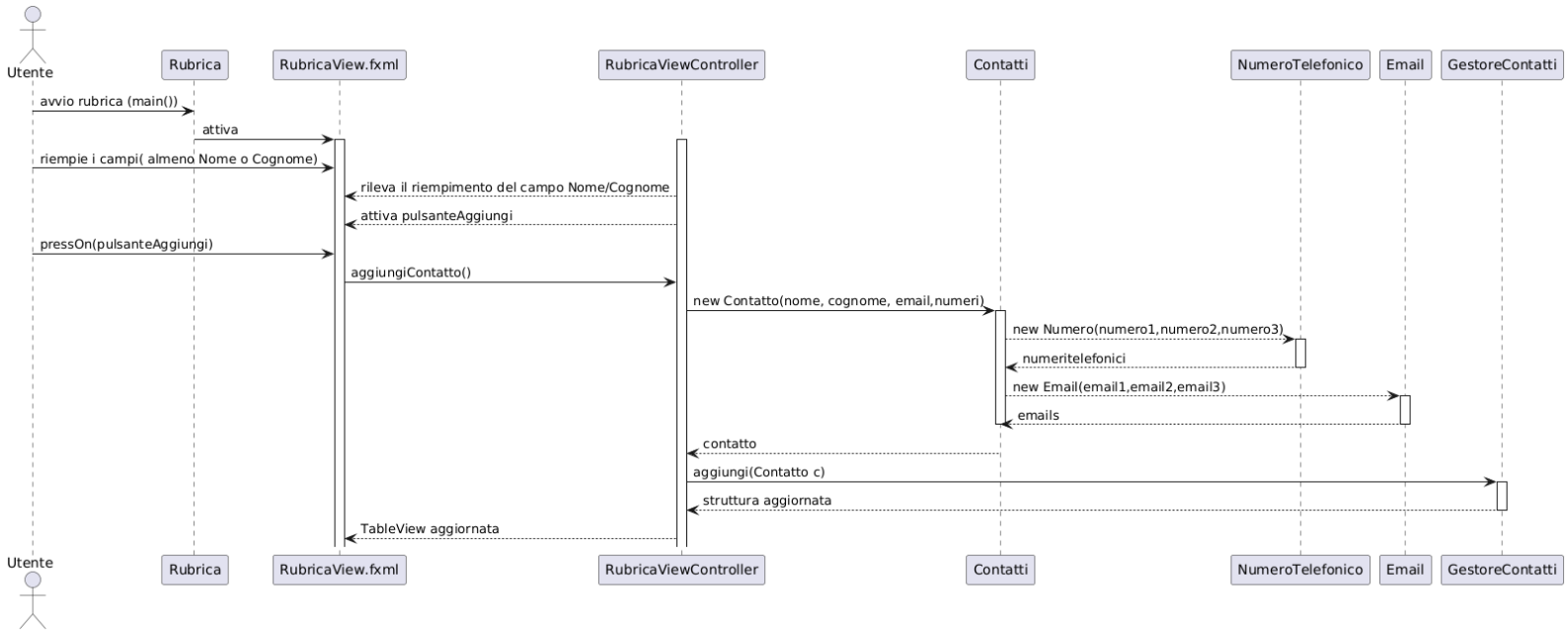
- **Classe contato:** Rappresenta un'entità definita *(il contatto)*. I metodi implementati e gli attributi sono legati alla gestione del contatto. L'utilizzo di altre entità come **Email** e **NumeroTelefonico** permette di delegare la gestione di altre strutture ad altre classi, aumentando la coesione. La classe contatto comunicherà con le classi NumeroTelefonico ed Email solo mediante i loro metodi pubblici *(accoppiamento per dati)*. La classe si occupa di aggregare gli elementi contenuti in NumeroTelefonico e Email ma lascia la gestione degli elementi alle classi.
- **Classe NumeroTelefonico:** La classe è responsabile della gestione di un insieme di numeri telefonici *(max 3)* il fatto di essere una classe separata da contatto permette di tenere la

coesione alta e consente, in futuro, di poter anche implementare metodi per future logiche di validazione. La classe non dipende da alcuna altra classe *(nessun accoppiamento)*.

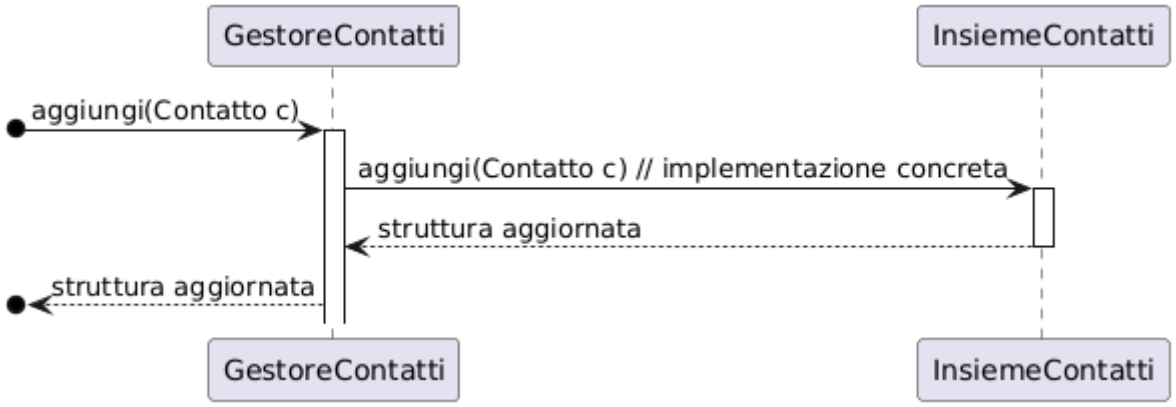
- **Classe Email:** Analogamente a come avviene per i numeri, questa classe si occupa della gestione di un insieme di Email *(max 3)*. Le stesse considerazioni sulla coesione e sulle future logiche di validazione implementabili valgono anche per questa classe. Analogamente a come avviene per la classe NumeroTelefonico, anche questa classe non dipende da nessuno *(accoppiamento per dati)*.
- **Interfaccia GestoreContatti:** L'interfaccia permette di eliminare la rigida dipendenza che normalmente esisterebbe tra la classe **InsiemeContatti** e **RubricaViewController**, utilizzando la tecnica dell'inversione della dipendenza; in questo modo sarà possibile sostituire in futuro la collezione di contatti con, ad esempio, un database o un'altra struttura che implementi questa interfaccia. I metodi presenti in questa interfaccia hanno lo scopo di definire funzionalità di base che un qualsiasi "contenitore" di Contatti deve avere.
- **Classe InsiemeContatti:** La classe rappresenta un'implementazione dell'interfaccia **GestoreContatti**, funge da "contenitore" per il generico oggetto Contatto. Implementa i metodi dell'interfaccia permettendo alla struttura di comunicare con il controller.
  La classe conterrà diversi Contatti e richiamerà i soli metodi pubblici per interagire con essi *(accoppiamento per dati)*.
- **Classe RubricaViewController:** La classe rappresenta il mezzo attraverso il quale le implementazioni della struttura sono in grado di comunicare con gli elementi grafici, gestendo anche le azioni degli utenti *(come il clic sui pulsanti)*.
  Questa classe comunicherà con l'interfaccia **GestoreContatti** *(la classe Controller non gestirà direttamente i contatti, ma delega questa responsabilità)*, richiamando i metodi associati a ciascun pulsante inserito nel file. fxml.
  Le interazioni con la classe Contatti saranno dettate dai soli metodi pubblici *(accoppiamento per dati)*, anche l'interazione con il GestoreContatti sarà guidata dai metodi messi a disposizione.
- **Classe Rubrica:** La classe rappresenta il Mian, permette di caricare la View inserita in un file fxml e di avviare la GUI. Risulta totalmente indipendente da tutte le classi elencate in precedenza.

3. Diagrammi di sequenza

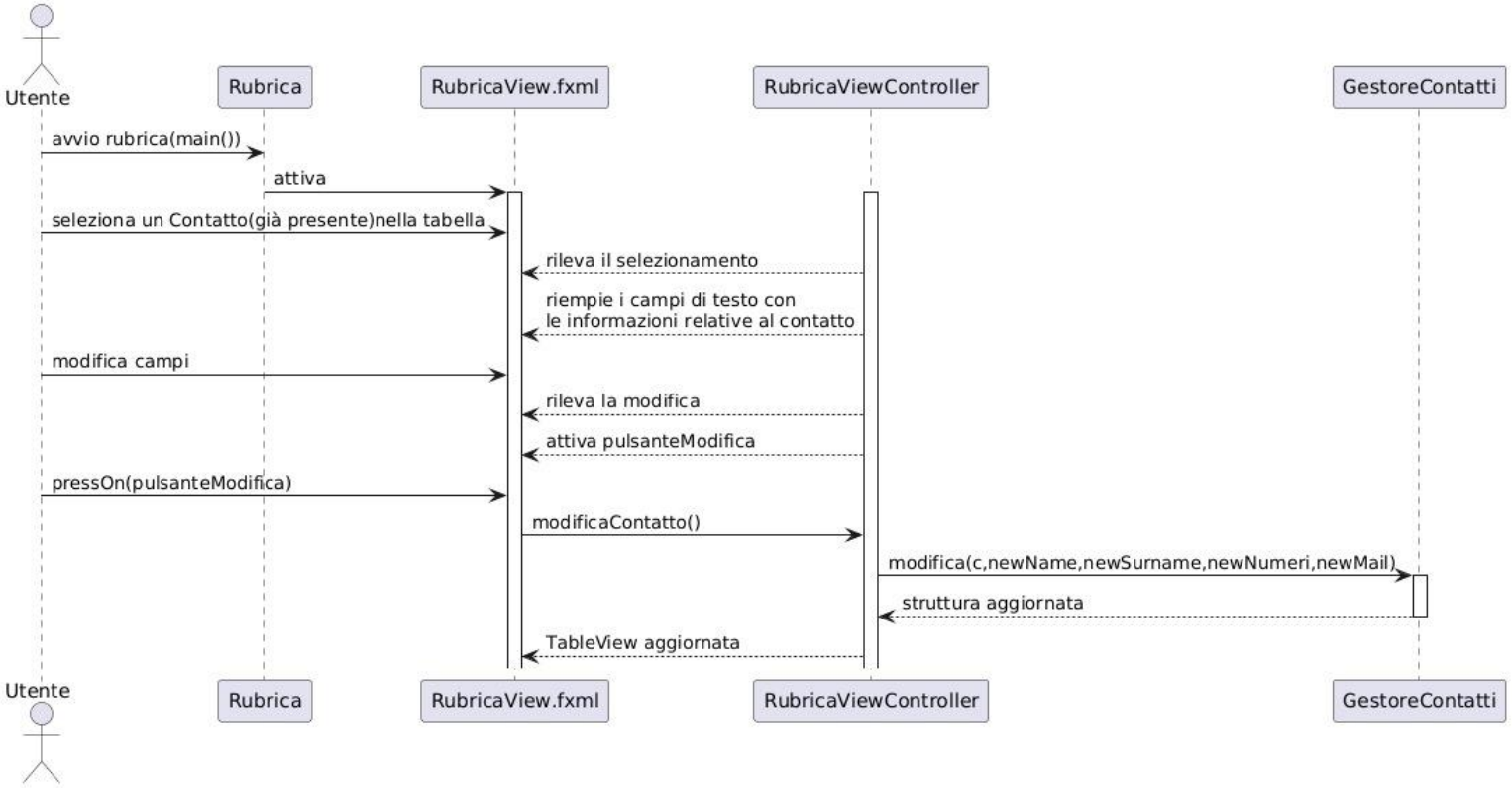
a. AGGIUNGI



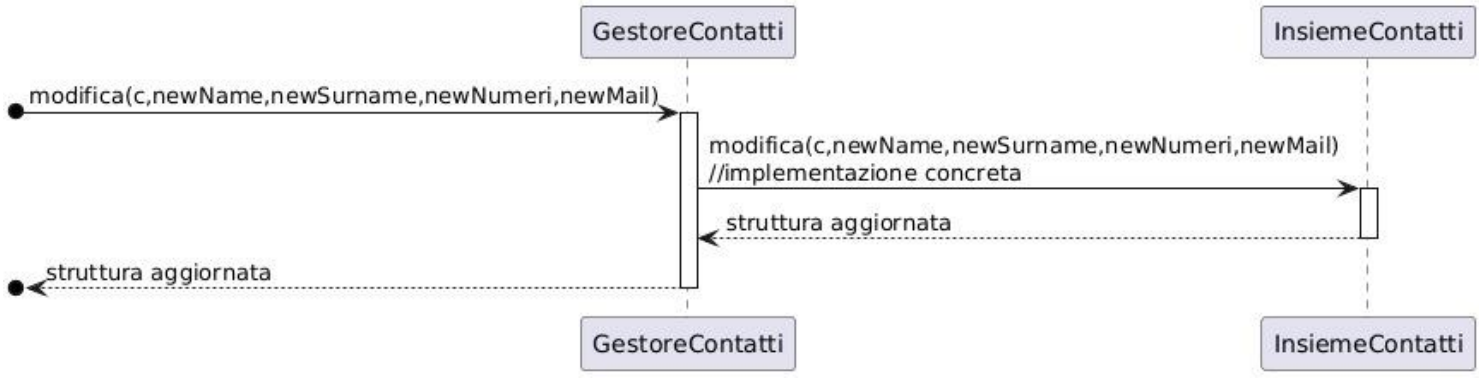
a.1 COMUNICAZIONE TRA IL GESTORE E LA SUA IMPLEMENTAZIONE



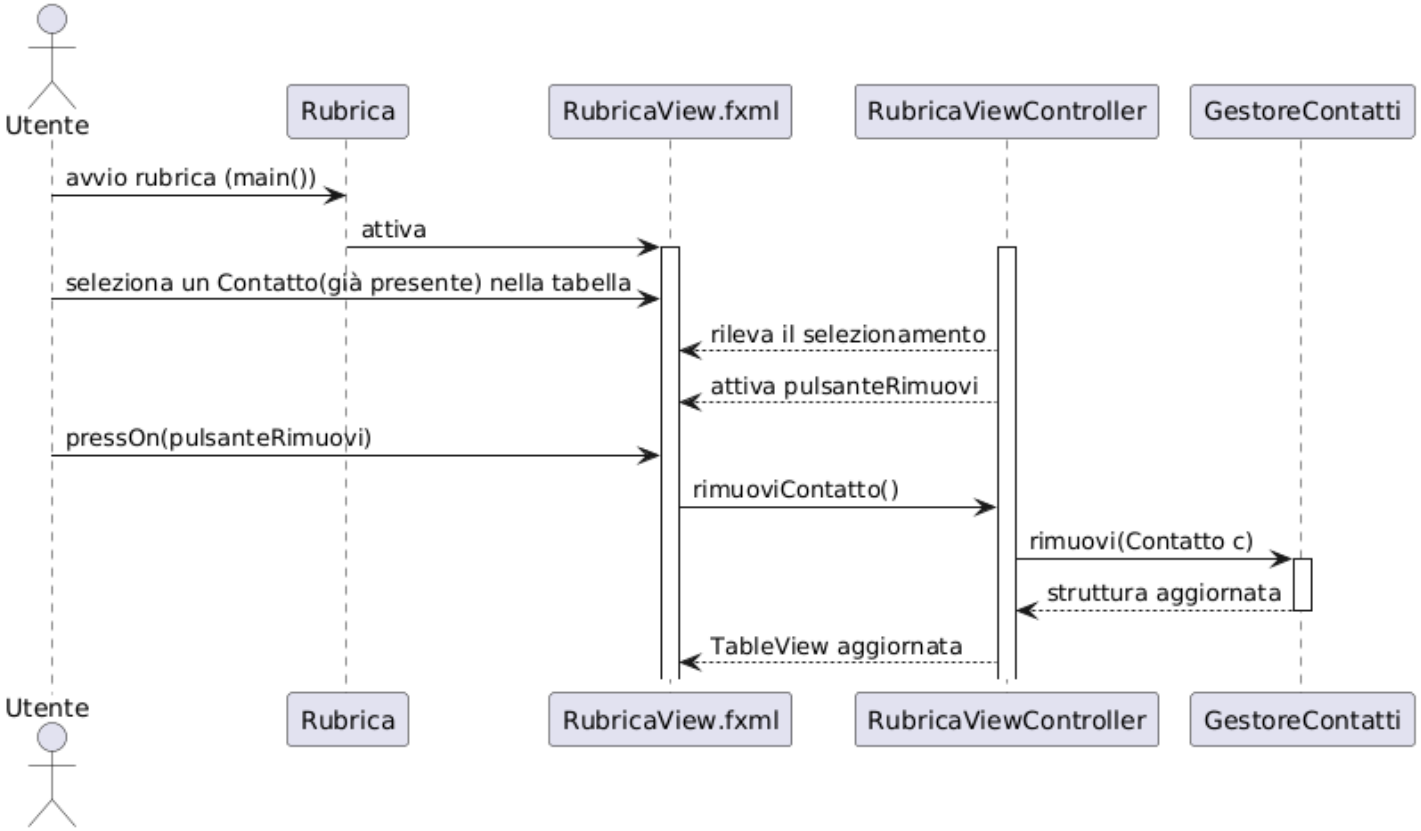
b. MODIFICA



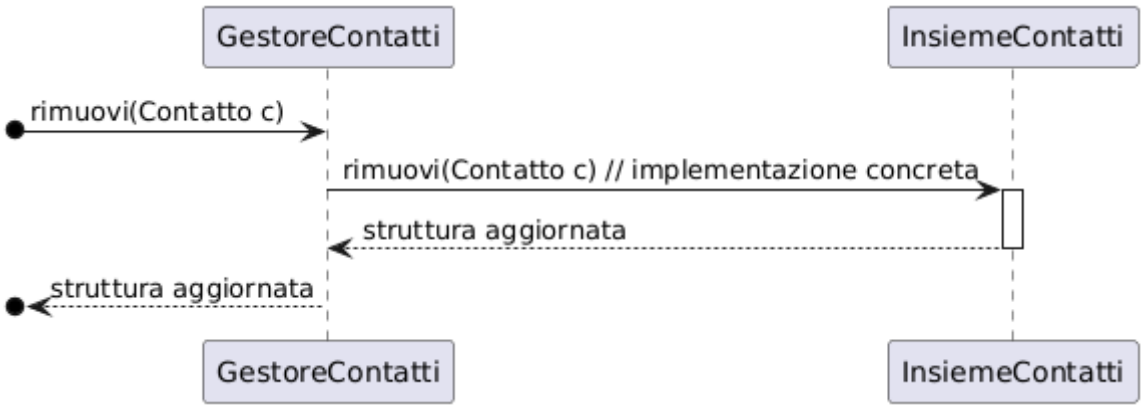
b.1. COMUNICAZIONE TRA IL GESTORE E LA SUA IMPLEMENTAZIONE



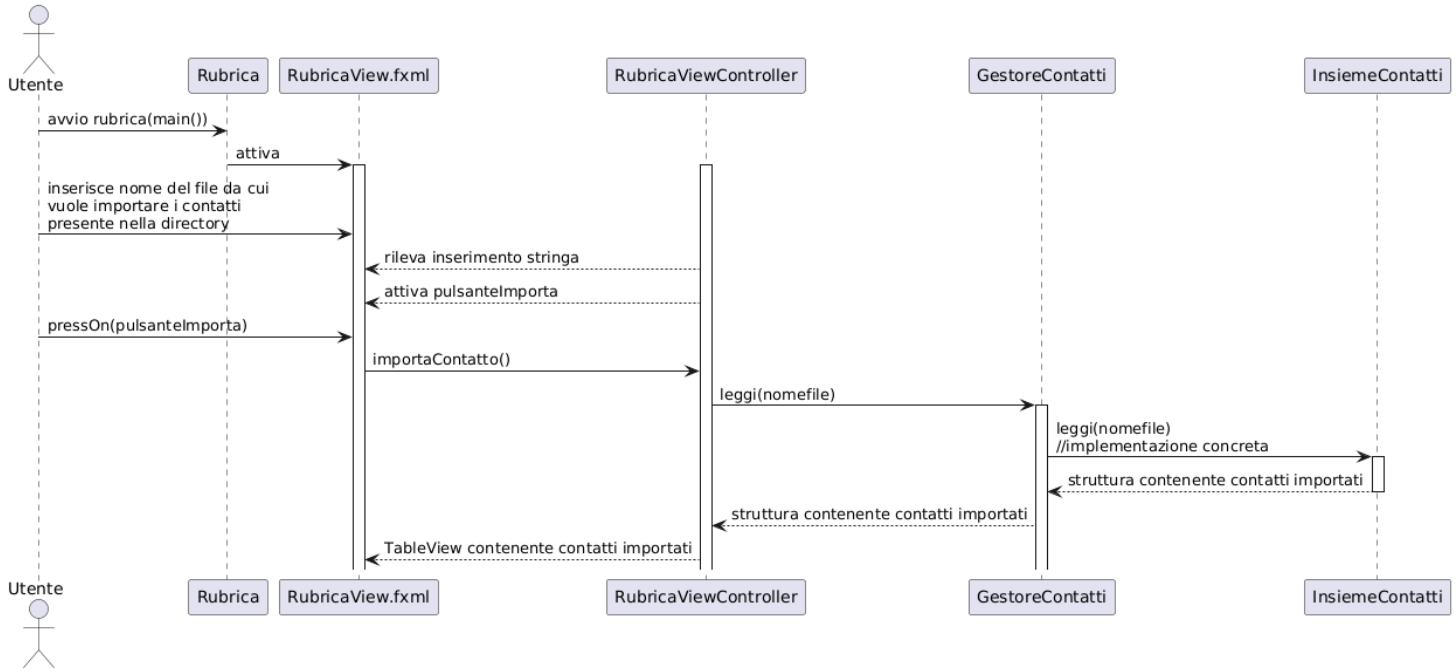
c. RIMUOVI



c.1 COMUNICAZIONE TRA IL GESTORE E LA SUA IMPLEMENTAZIONE



d. IMPORTA



e. ESPORTA