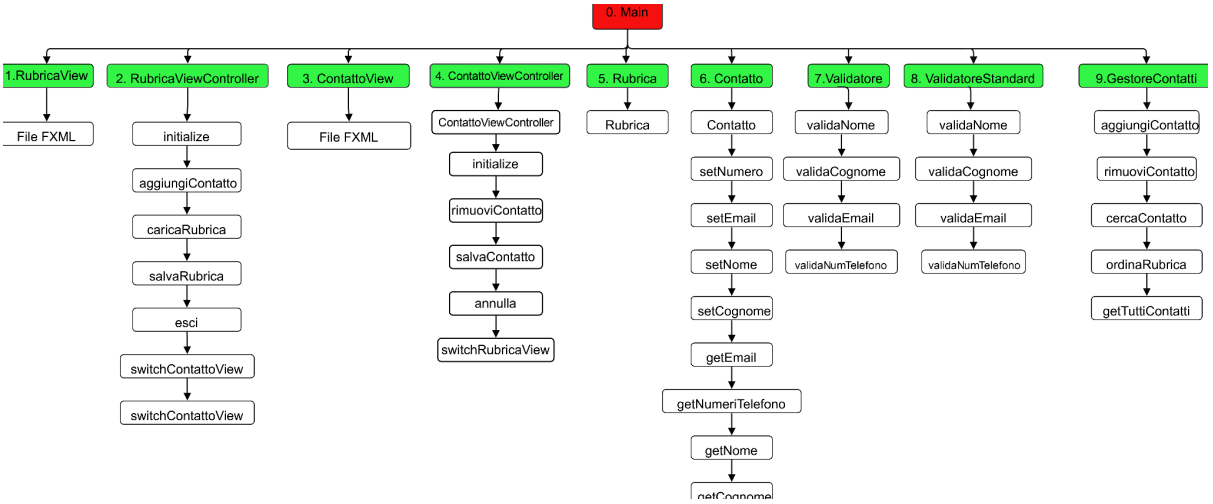
DESIGN

1 Preview

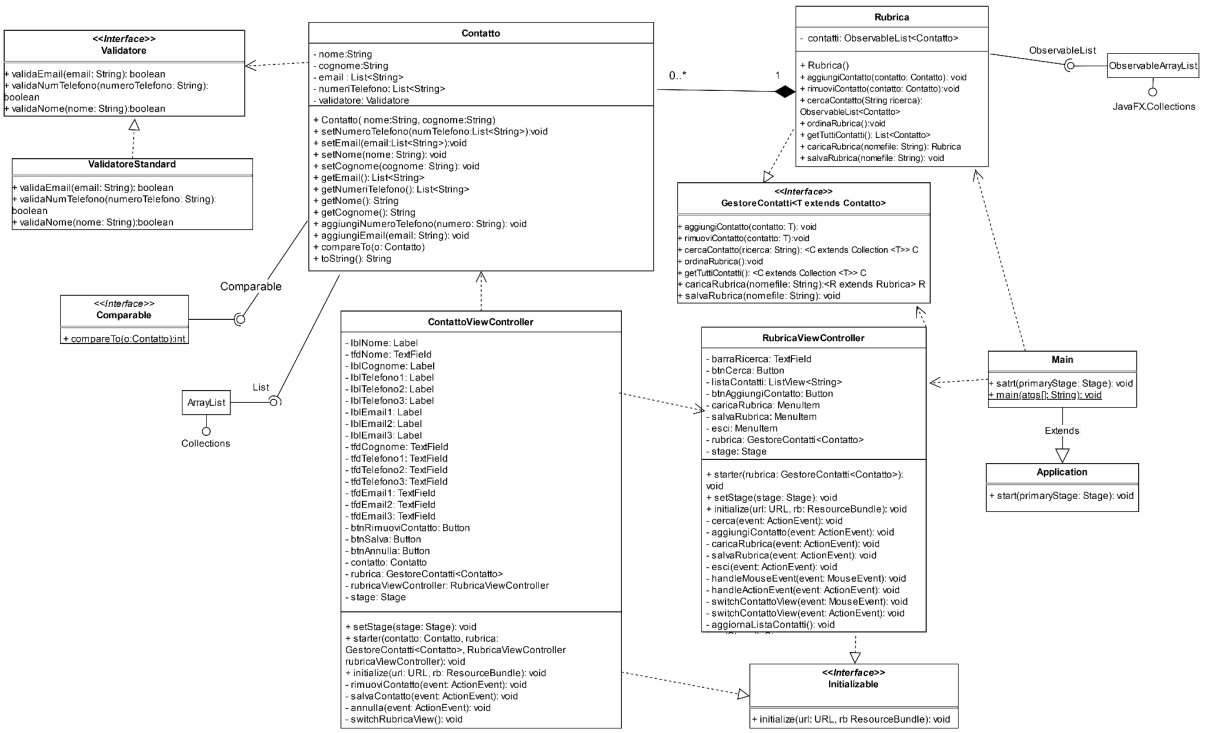
Il documento descrive le scelte progettuali intraprese per la strutturazione della rubrica. Per massimizzare quanto più la comprensione, saranno utilizzati più diagrammi, aggiungendo una sezione di chiarimento per i passaggi meno intuitivi. Il diagramma delle classi sarà utile per comprendere le relazioni tra le classi, evidenziando coesione e accoppiamento. Infine sarà presente un diagramma delle sequenze per presentare il funzionamento effettivo del sistema, attraverso l'interazione tra l'utente ed il sistema in casi d'uso specifici.



2 Diagramma delle classi

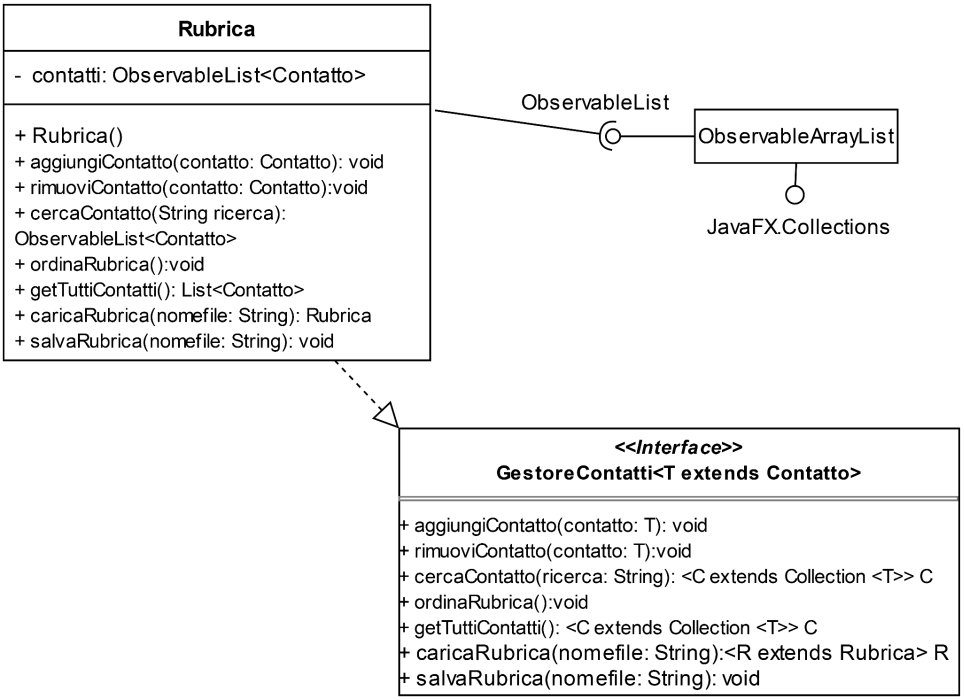
Per facilitare la visione, è stato deciso di frammentare il diagramma totale in sotto-diagrammi descrittivi delle più significative relazioni tra classi. Per maggiori informazioni sulla singola classe, i loro metodi e attributi, si rimanda il lettore al seguente link per visionare la documentazione doxygen del progetto.

2.0 Diagramma complessivo



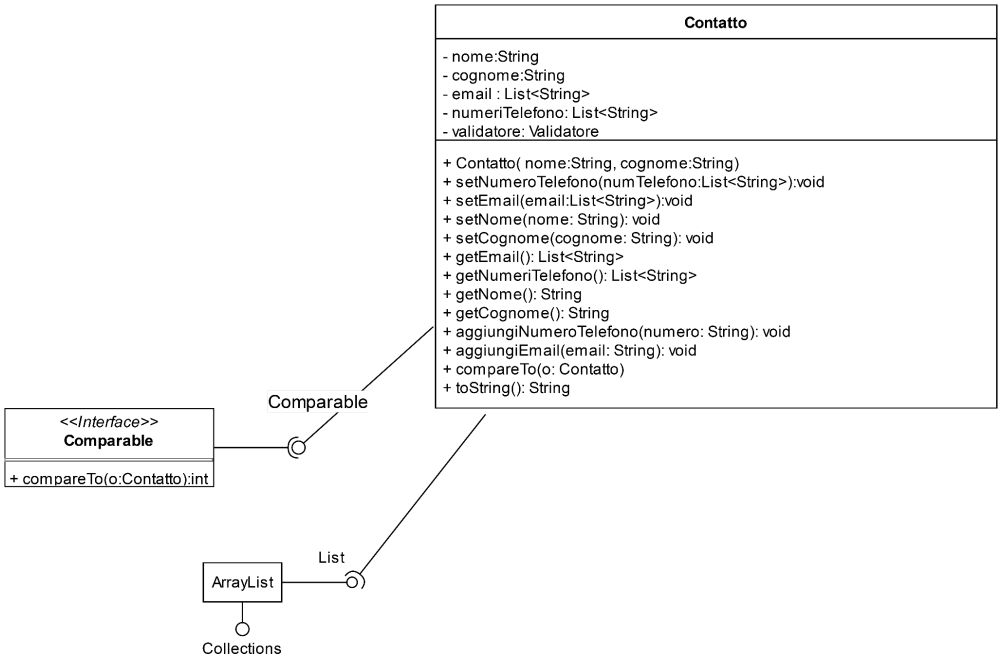
2.1 Diagramma della Rubrica

Il seguente diagramma mostra il funzionamento della classe Rubrica e di come questa implementi l'interfaccia funzionale FileIO, per salvare la rubrica su file o caricarla da questo.

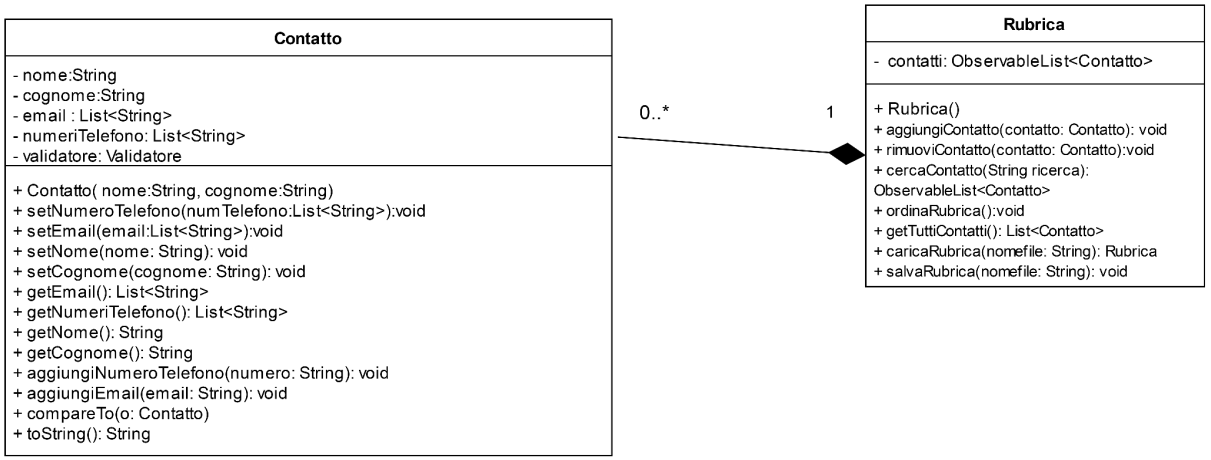


2.2 Diagramma del Contatto

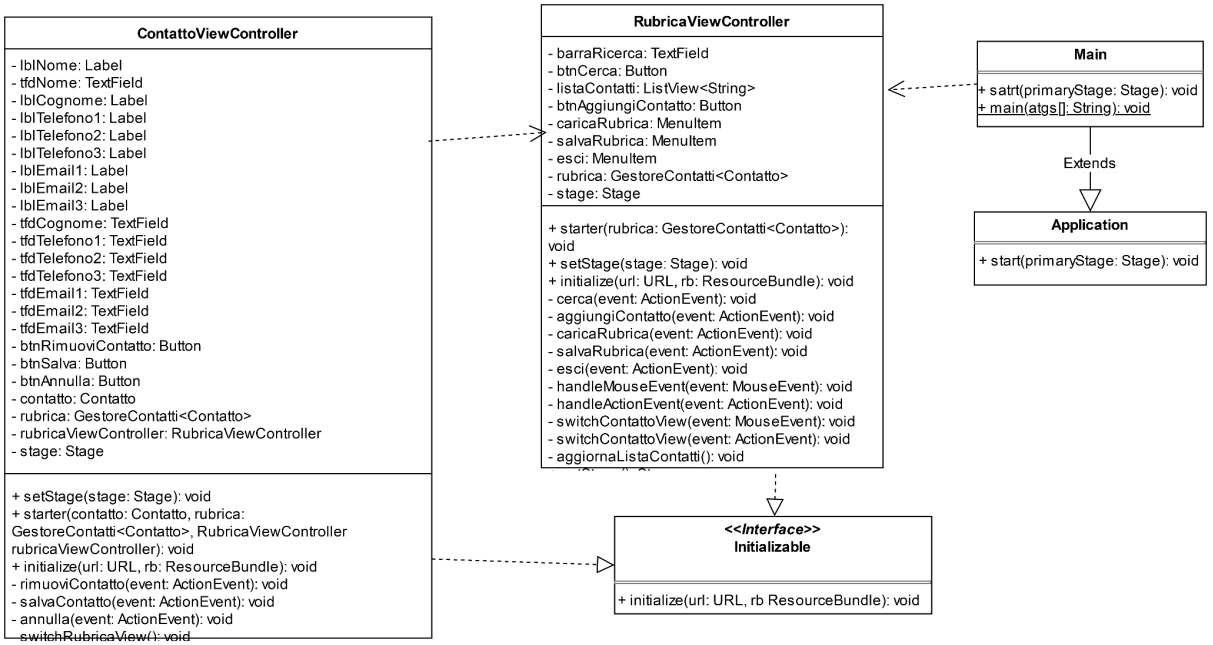
Il seguente diagramma mostra il funzionamento della classe Contatto, che implementa le interfacce List, Comparable e Validatore.



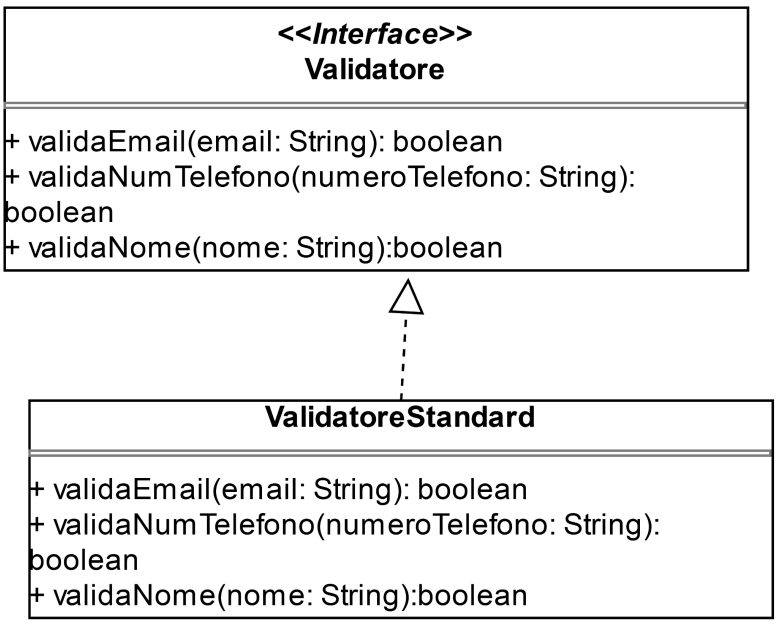
2.3 Diagramma Rubrica-Contatto



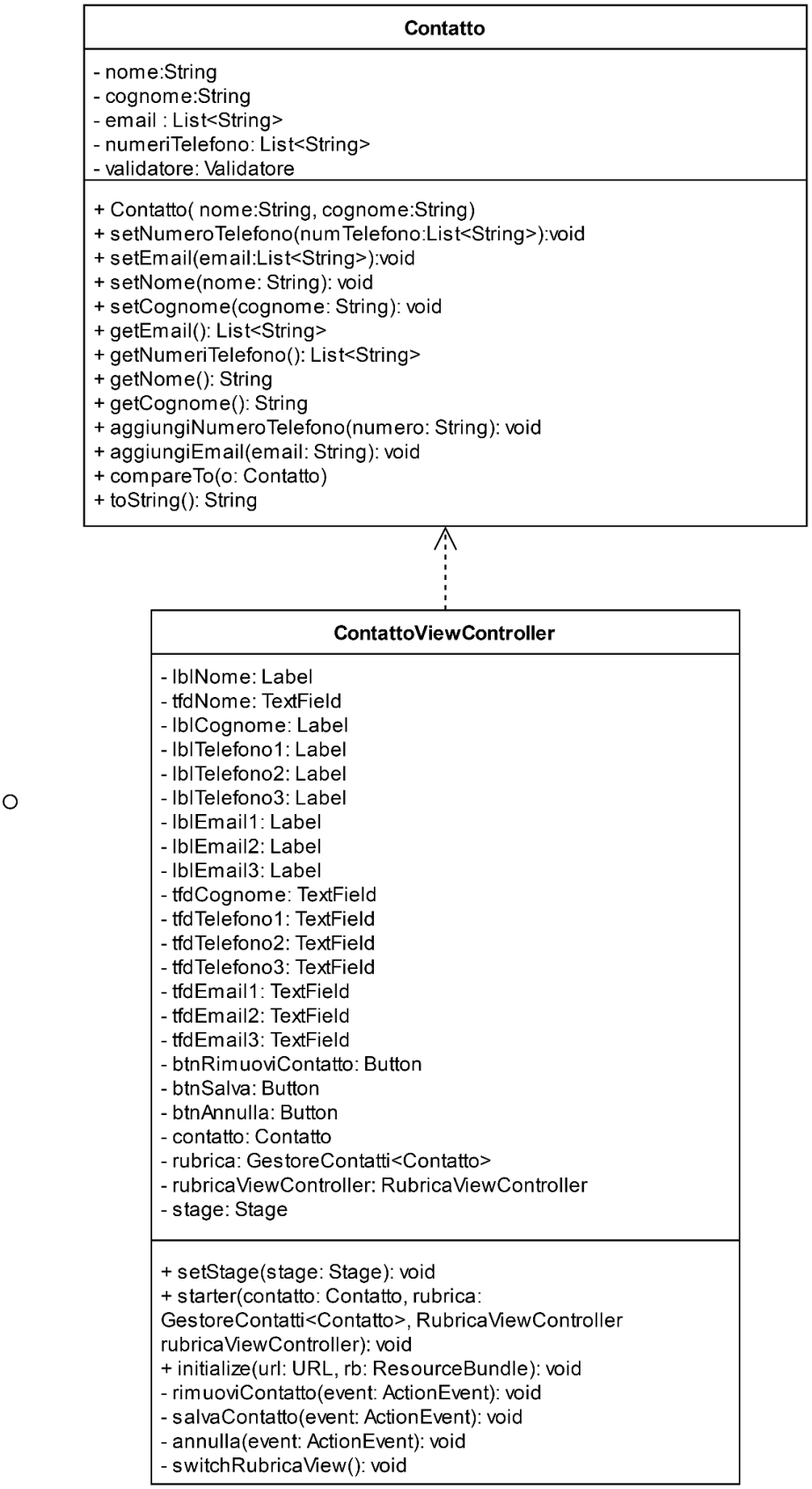
2.4 Diagramma Interfaccia Grafica e Main



2.5 Diagramma dei validatori

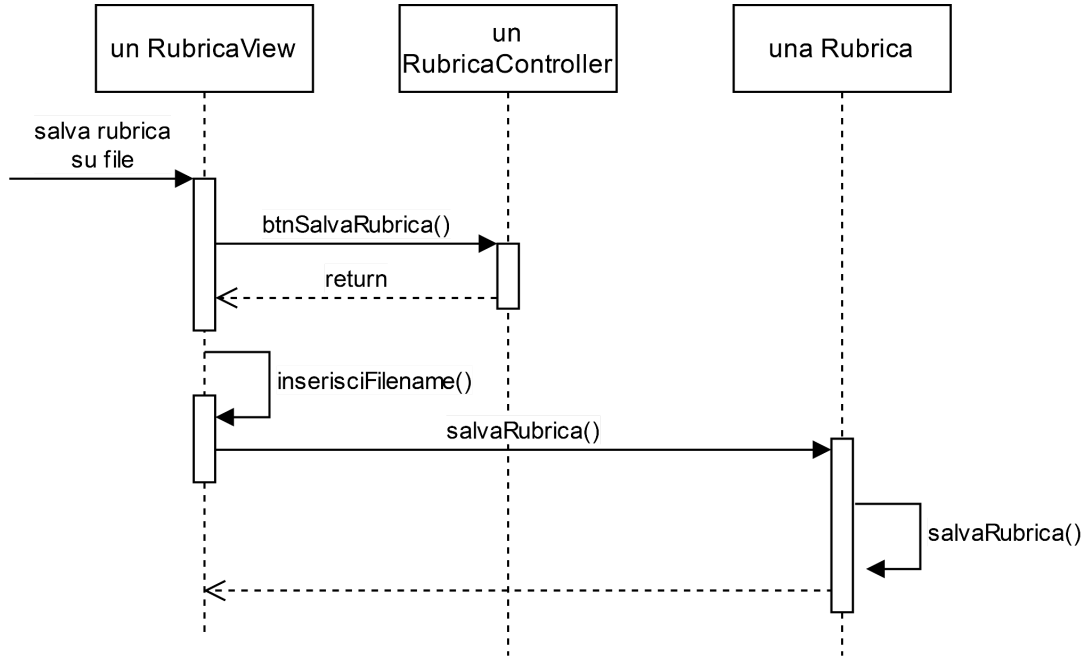


2.6 Diagramma Contatto-ContattoViewController

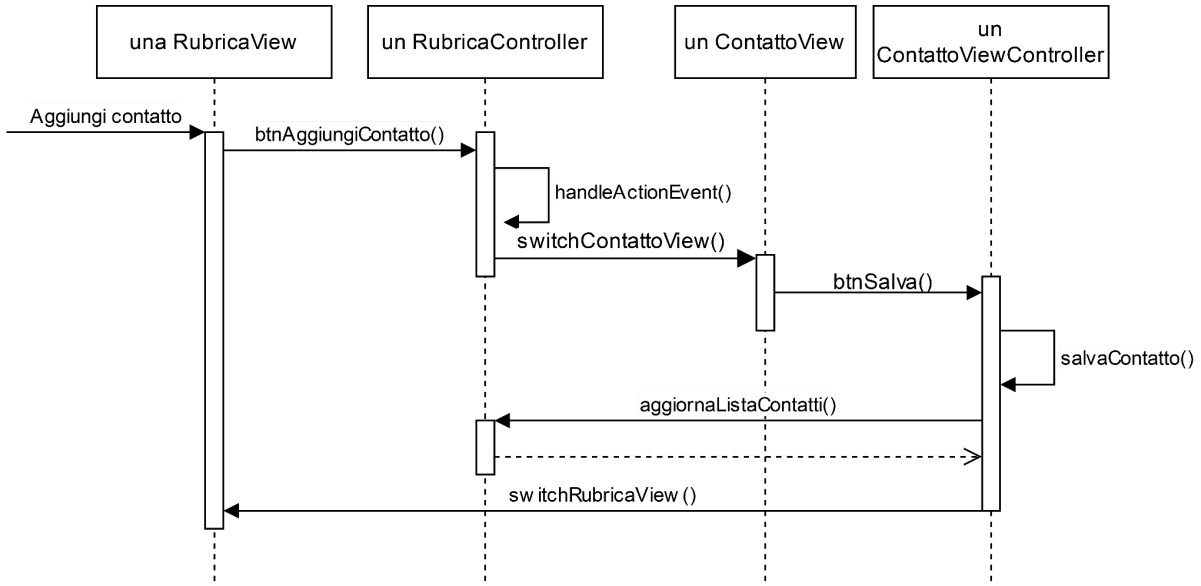


3. Diagrammi delle sequenze

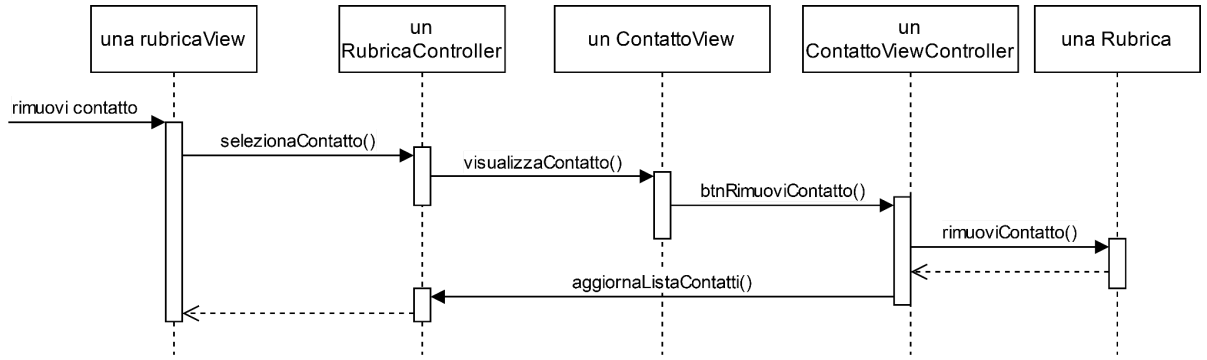
3.1 Salva rubrica su file



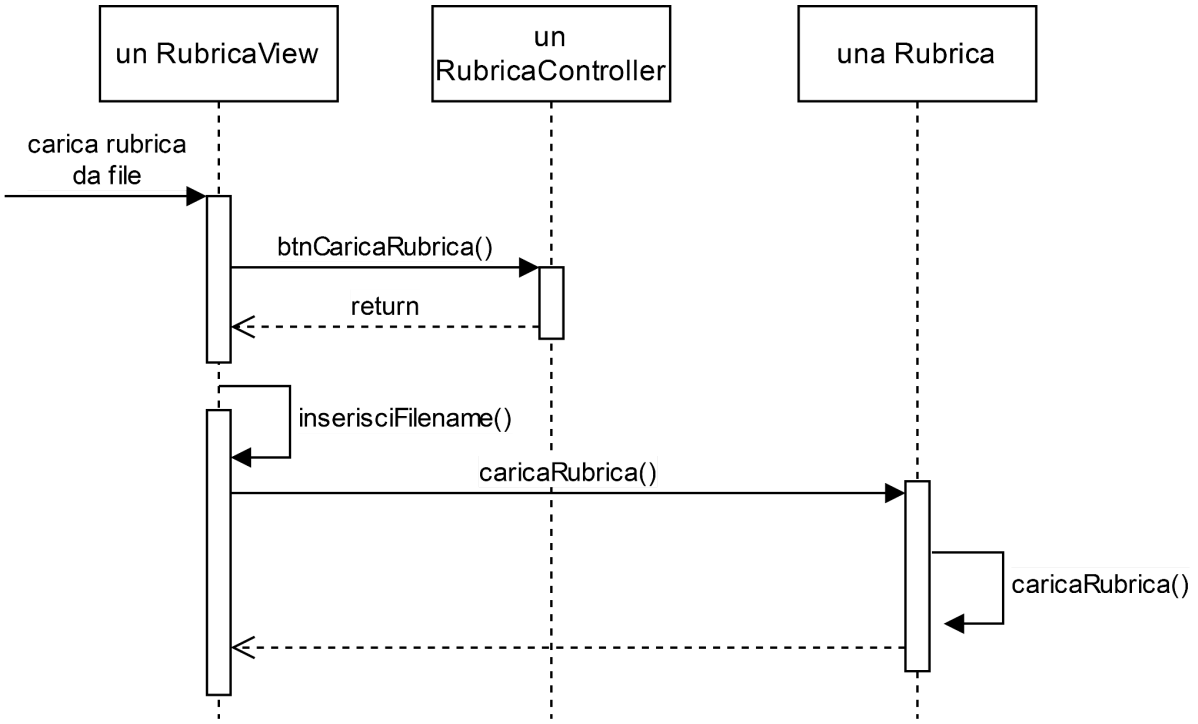
3.2 Aggiungi contatto



3.3 Rimuovi contatto



3.4 Carica rubrica da file



4. Considerazioni su coesione, accoppiamento e princìpi di buona progettazione

4.1 Coesione

La classe Contatto si occupa di rappresentare un contatto, includendo un nome, un cognome, più numeri di telefono ed email. Ogni metodo contribuisce direttamente a tale mansione, perciò la coesione è funzionale.

La classe **ValidatoreStandard** si occupa esclusivamente della validazione dei dati di un contatto, implementando l'interfaccia **Validatore**. Questo porta la classe ad avere anch'essa coesione **funzionale**.

Rubrica è la classe che si occupa della gestione di una lista di contatti. Le operazioni di aggiungiContatto, rimuoviContatto, modificaContatto, cercaContatto, ordinaRubrica hanno una stretta correlazione, in quanto deputate alla gestione di tale lista. Quindi, il livello di coesione è **funzionale**. La scelta di utilizzare una lista osservabile (Observable List) è giustificata dalla necessità di rappresentare graficamente la rubrica, favorendo la semplicità d'uso.

Per l'interfaccia grafica è stato deciso di suddividere la visualizzazione per **Rubrica** e **Contatto**. Inoltre, avendo utilizzato SceneBuilder (che fornisce in output un file FXML come classe Viewer) per curare la veste grafica, la coesione di tali interfacce ha uno stretto legame con le operazioni svolte da ognuno dei suoi blocchi. Risulta, dunque, una coesione **procedurale** per tali file FXML descrittivi delle rispettive classi (RubricaView e ContattoView).

Per le classi **RubricaViewController** e **ContattoViewController,** la coesione risulta **procedurale**, in quanto le operazioni svolte all'interno di ognuna sono volte al conseguimento della stessa operazione: l'interfaccia grafica. Lavorando su dati definiti rispettivamente da **Rubrica** e **Contatto**, tali classi lavorano in simbiosi con le rispettive interfacce grafiche (descritte dai rispettivi file FXML). Ciò riduce notevolmente la coesione, ma aumenta la semplicità d'uso.

4.2 Accoppiamento

Grazie all'opportuno uso delle interfacce **GestoreContatti** e **Validatore**, l'accoppiamento risulta debole: sono presenti unicamente accoppiamenti per **dati**.
Per esempio la classe **ValidatoreStandard** accede ai dati della classe **Contatto**, è indipendente dalla sua implementazione. Lo stesso dicasi per **RubricaViewController** e **Rubrica**: sebbene il Controller abbia accesso alla lista di contatti, questo non dipende dai dettagli implementativi di **Rubrica**, abbassando l'accoppiamento ad uno di tipo **per dati**.

4.3 Principi di buona progettazione

I principi di progettazione fondanti utilizzati nella progettazione sono stati il **KISS** (Keep It Simple, Stupid!) ed il principio di **programmazione ad oggetti** per lavorare con la massima astrazione possibile ed elevare al massimo la semplicità d'uso . Il principio **SINE** (Simple Is Not Easy) è stato rispettato per non escludere meticolosità nella progettazione. Si è scelto di basarsi su tali principi anche per evitare di incorporare funzionalità considerate superflue (You Aren't Going to Need It, **YAGNI**) o che avrebbero ecceduto le abilità in ambito di programmazione di ogni membro del gruppo (**Ortogonalità**). Un esempio

dell'applicazione di più di uno di questi principi sono le classi Rubrica e Contatto. In queste classi sono state inserite esclusivamente funzionalità di base (**KISS**), separando accuratamente le responsabilità e decidendo di delegare la validazione dei contatti ad un Validatore e la gestione di tali contatti alla Rubrica(principio **SRP**, Single Responsibility Principle). Sono stati rispettati anche i principi di **DRY** (Don't Repeat Yourself) evitando ridondanze all'interno del codice, infatti non sono presenti metodi che svolgono mansioni equivalenti a metodi di classi altrui.

5. Matrice di tracciabilità

Nome requisito	ID requisito	Design	Codice	Test	Requisiti correlati
Dati contatto	1	Modulo 6 Modulo 8			2,3,4,5,6,8
Creazione contatto	2	Modulo 3 Modulo 4 Modulo 5 Modulo 6 Modulo 7 Modulo 8 Modulo 9			1,8
Modifica contatto	3	Modulo 3 Modulo 4 Modulo 5 Modulo 6 Modulo 7 Modulo 8 Modulo 9			1,8
Eliminazione contatto	4	Modulo 3 Modulo 4 Modulo 5 Modulo 6 Modulo 9			8
Salvataggio rubrica su file	5	Modulo 1 Modulo 2 Modulo 5 Modulo 9			1
Carica rubrica da file	6	Modulo 1 Modulo 2 Modulo 5 Modulo 9			1
Cerca contatto	7	Modulo 1 Modulo 2 Modulo 5			8,9

Nome requisito	ID requisito	Design	Codice	Test	Requisiti correlati
		Modulo 6			
Interfaccia utente	8	Modulo 1 Modulo 2 Modulo 3 Modulo 4			1,2,3,4,5,6,7,9, 10
Ordine rubrica	9	Modulo 1 Modulo 2 Modulo 5			7,8
Semplicità d'uso	10	Modulo 1 Modulo 3			8