

UNIVERSITÀ DEGLI STUDI DI SALERNO

DIEM - Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica
Applicata



Corso di Laurea Triennale in Ingegneria Informatica

Progetto Ingegneria del Software **Rubrica Telefonica**

Gruppo 07:

Emanuele Tocci - 0612707488

Alessio Leo - 0612707279

Rossella Pale - 0612707284

Claudia Montefusco - 0612707404

ANNO ACCADEMICO 2024/2025

Indice

I	Introduzione	3
1	Richiesta di progetto	3
2	Anteprima interfaccia grafica	3
II	Ingegneria dei requisiti	7
3	Elicitazione dei requisiti	7
4	Analisi e definizione dei requisiti	8
4.1	Categorizzazione dei requisiti	8
4.1.1	Requisiti funzionali	8
4.1.2	Requisiti non funzionali	8
4.2	Categorizzazione di dettaglio	8
4.2.1	Funzionalità individuali	9
4.2.2	Business Flow	9
4.2.3	Esigenze di dati e informazioni	10
4.2.4	Interfacce utente	10
4.2.5	Ulteriori vincoli	10
4.3	Indicizzazione dei requisiti	11
4.4	Definizione delle priorità	11
4.4.1	Business Value	11
4.4.2	Rischio tecnico	12
4.5	Casi d'uso	13
4.5.1	Descrizione dei casi d'uso	13
4.5.2	Diagramma dei casi d'uso	15
III	Design	17
5	Attributi di qualità (QA)	17
5.1	QA esterni	17
5.2	QA interni	17
6	Diagramma dei package	17
7	Diagramma delle classi	18
7.1	Descrizione ed analisi delle classi	21
7.1.1	Classe "Rubrica"	21
7.1.2	Classe "Contatto"	21
7.1.3	Classe "ContattoEsteso"	21
7.1.4	Classe "GenericController"	21
7.1.5	Classe "RubricaController"	22
7.1.6	Classe "CreaContattoController"	22
7.1.7	Classe "Import"	22

INDICE

7.1.8	Classe "Export"	22
8	Diagrammi di sequenza	23

Parte I

Introduzione

1 Richiesta di progetto

L'obiettivo del progetto é la realizzazione di un'applicazione che simuli una **rubrica telefonica** per la gestione di contatti. Ad ogni contatto sono associate diverse informazioni tra cui: nome, cognome, numeri di telefono, indirizzi email, compleanno, residenza, sito web e alcune note personali. La rubrica consente all'utente di interagire mediante una semplice **interfaccia grafica** offrendo diverse diverse funzionalità come l'aggiunta, la cancellazione e la modifica di un contatto.

2 Anteprima interfaccia grafica

Si presenta rispettivamente un mockup dell'interfaccia grafica dell'applicazione e l'interfaccia effettivamente **implementata** in JavaFX.

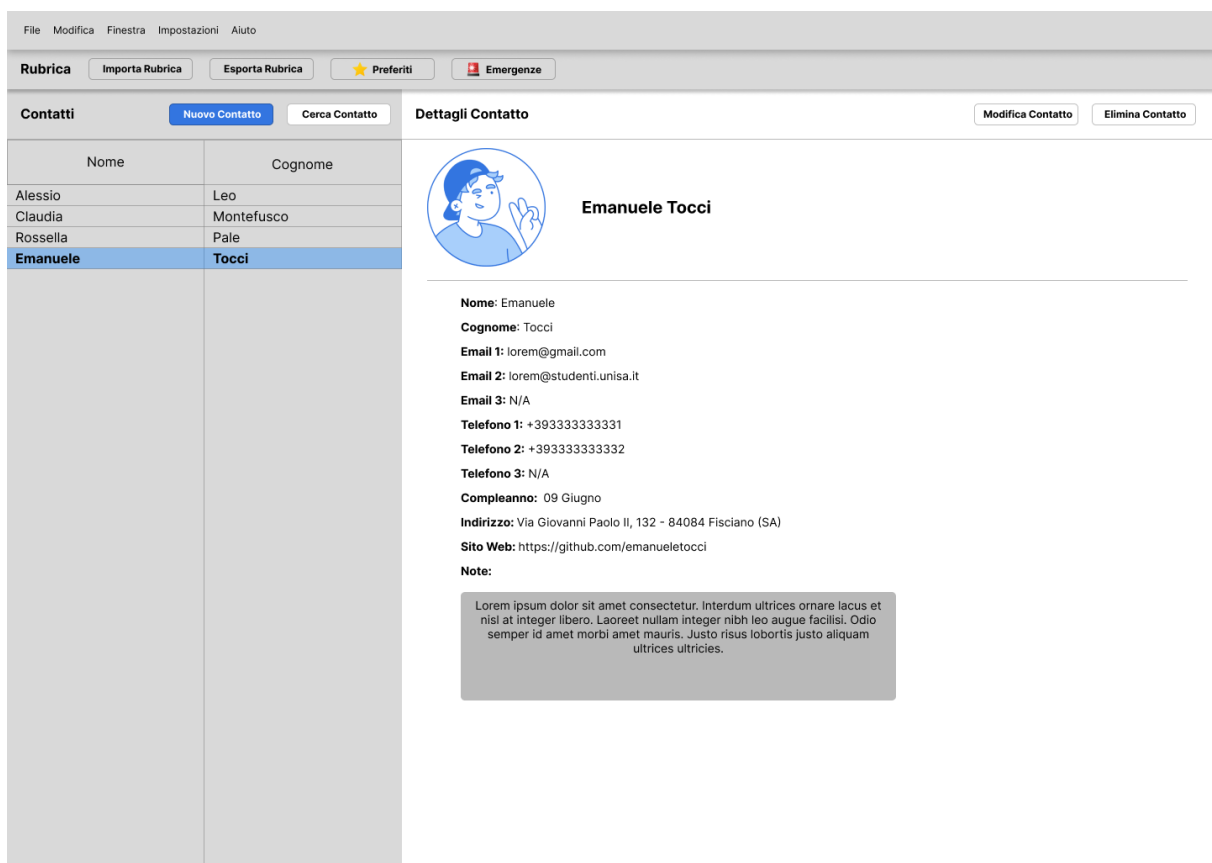



Figura 1: Mockup vista generica (homepage)



Aggiungi foto

Nome

Cognome

Telefono 1

+

Email 1

+

Compleanno

Indirizzo

Sito Web

Note

Crea/Modifica Contatto

Figura 2: Mockup vista che consente l'aggiunta e la modifica di un contatto

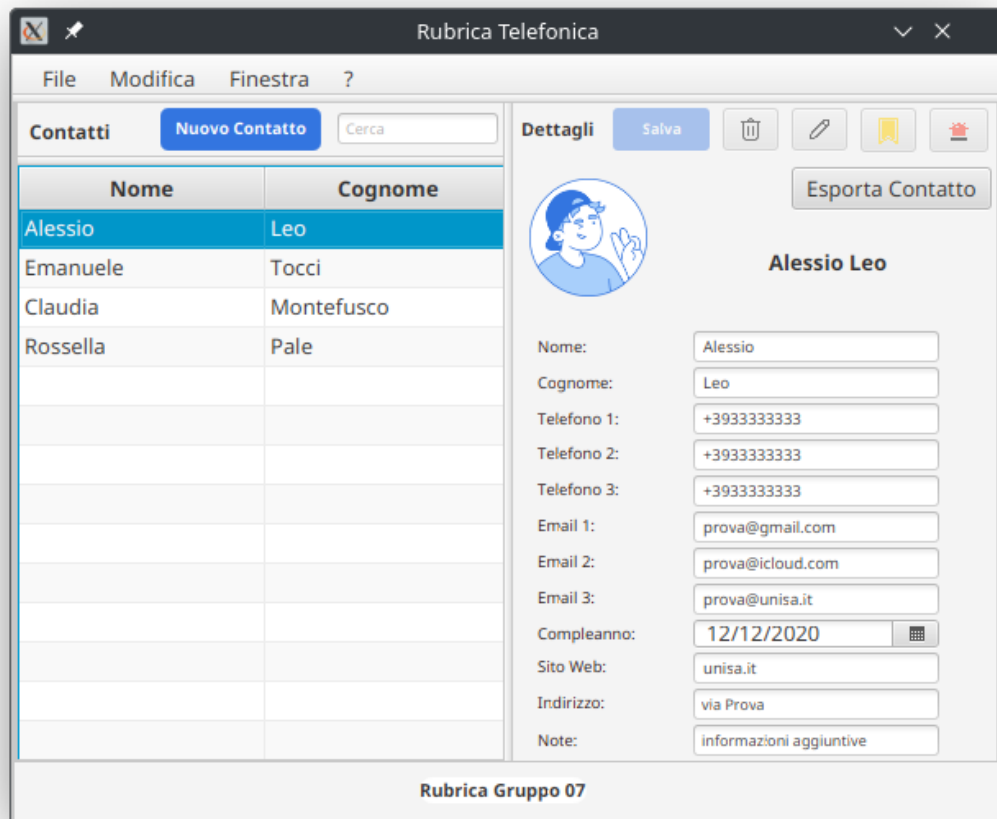
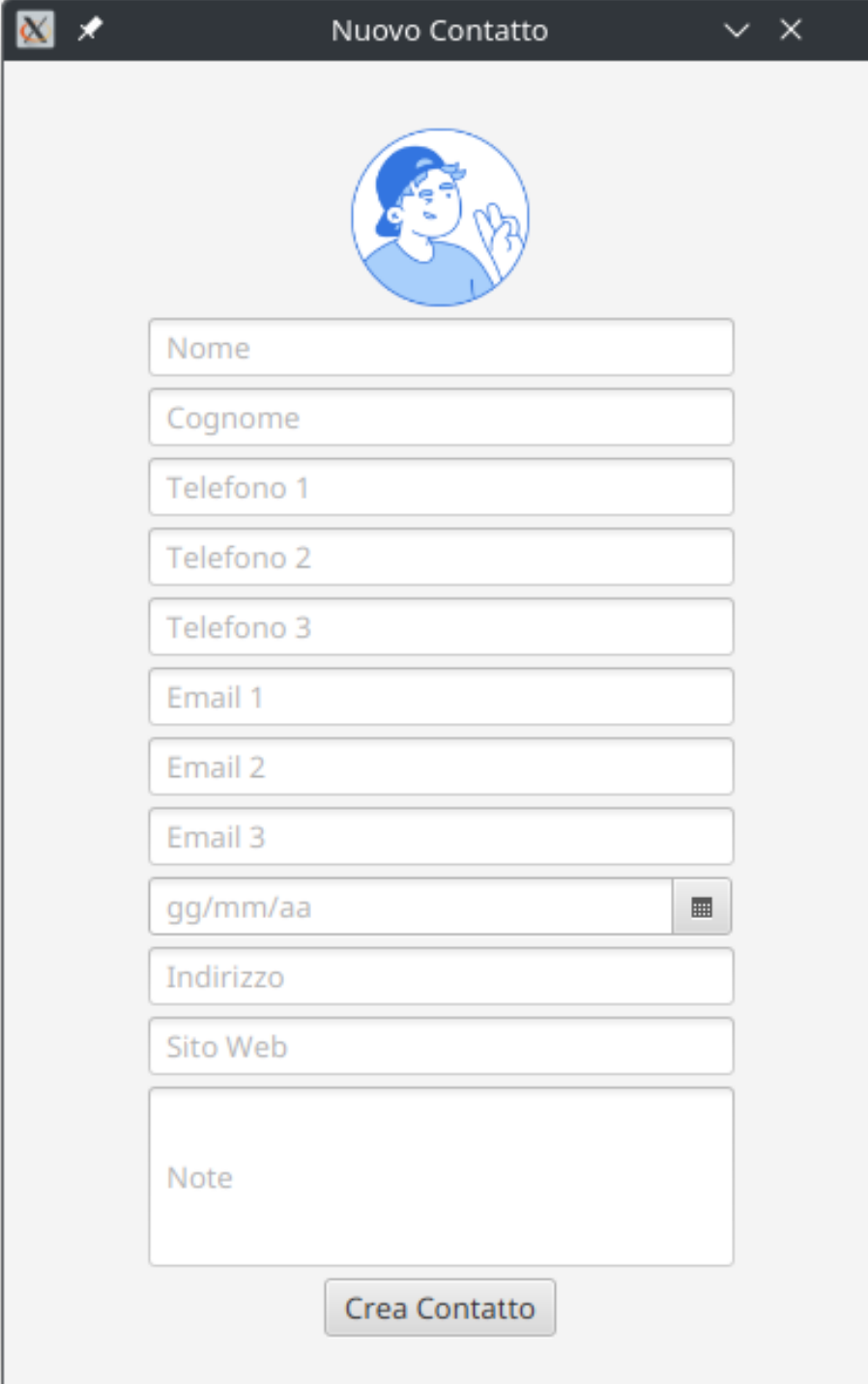


Figura 3: Vista generica (javaFX)



The image shows a JavaFX window titled "Nuovo Contatto" (New Contact). The window has a dark title bar with standard OS controls (minimize, maximize, close) on the right. Inside the window, there is a light gray background. At the top center is a circular profile picture placeholder containing a blue cartoon illustration of a person wearing a cap and making a hand gesture. Below the profile picture is a vertical stack of text input fields: "Nome", "Cognome", "Telefono 1", "Telefono 2", "Telefono 3", "Email 1", "Email 2", "Email 3", a date field labeled "gg/mm/aa" with a calendar icon, "Indirizzo", "Sito Web", and a larger "Note" field. At the bottom center is a button labeled "Crea Contatto".

Figura 4: Vista che consente l'aggiunta di un nuovo contatto (javaFX)

Parte II

Ingegneria dei requisiti

3 Elicitazione dei requisiti

1. Creare un contatto
2. Modificare un contatto
3. Eliminare un contatto
4. Visualizzare un contatto
5. Ricercare un contatto
6. Ordinare alfabeticamente i contatti
7. Catalogare i contatti in "preferiti" ed "emergenza"
8. Ogni contatto deve avere almeno nome o cognome
9. Ogni contatto deve avere da 0 a 3 numeri di telefono
10. Ogni contatto deve avere da 0 a 3 indirizzi email
11. Ogni contatto può avere informazioni aggiuntive opzionali (eg. compleanno, indirizzo di residenza, sito web, note)
12. Interfaccia grafica (GUI) minimale
13. Importare rubrica esistente
14. Importare singolo contatto
15. Esportare rubrica esistente
16. Esportare singolo contatto
17. Utilizzare il formato vCard per Import/Export
18. Applicazione cross-platform: Windows, macOS, Linux
19. Scadenza: 15/12/24
20. Utilizzo di Maven

4 Analisi e definizione dei requisiti

4.1 Categorizzazione dei requisiti

4.1.1 Requisiti funzionali

1. Gestire contatti: creazione, lettura, modifica ed eliminazione
2. Ricercare contatto
3. Ordinare alfabeticamente i contatti
4. Catalogare i contatti in "preferiti" ed "emergenza"
5. Ogni contatto può avere informazioni aggiuntive opzionali
6. Interfaccia grafica (GUI)
7. Import/Export

4.1.2 Requisiti non funzionali

1. Ogni contatto deve avere da 0 a 3 numeri di telefono
2. Ogni contatto deve avere da 0 a 3 indirizzi email
3. Ogni contatto deve avere almeno nome o cognome
4. Standard vCard per Import/Export
5. Da consegnare entro il 15 dicembre
6. Applicazione cross-platform: Windows, macOS, Linux
7. Utilizzo di Maven

4.2 Categorizzazione di dettaglio

I requisiti vengono categorizzati secondo le "6 dimensioni dei requisiti" e mediante un **prefisso** univoco rappresentante la relativa dimensione. Al prefisso é stato associato un **numero** del tipo [X.Y], composto da due cifre ed un punto. Il primo valore [X] rappresenta un'**area di interesse** e raggruppa pertanto i requisiti comuni; il secondo valore [Y] specializza il requisito. Pertanto requisiti che hanno il primo valore [X] uguale sono **correlati**.

I requisiti "BF1.1 , BF1.2 , BF1.3 , BF1.4" per esempio, riguardano tutti la gestione dei contatti.

4.2.1 Funzionalità individuali

- **IF1.1: Catalogazione dei contatti in “preferiti” e/o “emergenza”**
 - Il sistema offre la possibilità di marcare i contatti come “preferiti” e/o “di emergenza”
- **IF2.1: Ricerca contatti**
 - Il sistema deve supportare la ricerca di un contatto tramite la sottostringa (parte) del nome o cognome.
 - Il sistema deve evidenziare i risultati nella lista e mostrare il relativo contatto all’utente
- **IF3.1: Ordinamento alfabetico**
 - Il sistema offre la possibilità di ordinare i contatti per nome o cognome

4.2.2 Business Flow

Interazioni tra uno o più utenti e il sistema per la realizzazione di uno specifico processo aziendale

- **BF1.1: Creazione contatto**
 - Il sistema consente la creazione di un nuovo contatto e verifica automaticamente che i requisiti necessari alla creazione vengano rispettati
- **BF1.2: Modifica contatto**
 - Il sistema consente di modificare un contatto già presente in rubrica
- **BF1.3: Eliminazione contatto**
 - Il sistema consente di eliminare un contatto già presente in rubrica
- **BF1.4: Visualizzazione/Lettura contatto**
 - Il sistema consente di visualizzare i contatti presenti in rubrica
- **BF2.1: Import/Export (vCard)**
 - Salvataggio della rubrica in formato .vcf
 - Salvataggio di un singolo contatto in formato .vcf
 - Caricamento di una rubrica da file .vcf
 - Caricamento di un singolo contatto da file .vcf

4.2.3 Esigenze di dati e informazioni

- **DF1.1: Informazioni essenziali sul contatto**
 - Nome, cognome, numeri telefonici, email
 - Il sistema deve controllare automaticamente i campi (es. email valida) e restituire un errore nel caso in cui il campo non sia valido
- **DF1.2: Informazioni opzionali sul contatto**
 - Il sistema offre la possibilità di aggiungere dei dettagli aggiuntivi ai contatti: data di nascita, indirizzo fisico, sito web, note.

4.2.4 Interfacce utente

- **UI1.1: GUI (Interfaccia Grafica)**
 - Il sistema deve avere una GUI intuitiva e user-friendly
 - La GUI deve consentire l'accesso rapido alle principali funzionalità del sistema (es. ricerca, aggiunta, modifica, eliminazione)

4.2.5 Ulteriori vincoli

- **FC1.1: Numero min/max di numeri telefonici ed email registrabili per ogni contatto**
 - Ogni contatto può avere da 0 a 3 numeri di telefono e da 0 a 3 indirizzi email
- **FC1.2: Vincoli sulla creazione di un contatto**
 - Il sistema consente la creazione del contatto solamente se l'utente inserisce almeno nome o cognome (una delle due informazioni può essere vuota)
- **FC2.1: Standard vCard**
 - Standard vCard per Import/Export
- **FC3.1: Deadline**
 - Da consegnare entro il 15/12/24
- **FC4.1: Cross-Platform**
 - Applicazione cross-platform compatibile con Windows, macOS e Linux
- **FC5.1: Utilizzo di Maven**
 - Il progetto deve essere compilabile tramite Maven

4.3 Indicizzazione dei requisiti

Area dei requisiti	Prefisso	Numerazione delle dichiarazioni dei requisiti
Funzionalità individuale	IF	IF1.1, IF2.1, IF3.1
Business flow	BF	BF1.1, BF1.2, BF1.3, BF1.4, BF2.1
Esigenze di dati ed informazioni	DF	DF1.1, DF1.2
Interfaccia utente	UI	UI1.1
Ulteriori vincoli	FC	FC1.1, FC1.2, FC2.1, FC3.1, FC4.1, FC5.1

Tabella 1: Indicizzazione dei requisiti

4.4 Definizione delle priorità

4.4.1 Business Value

- **Alto** (must have): se il requisito non è soddisfatto, il sistema non è utilizzabile
- **Medio** (should have): importante, ma non comporta un fallimento se omesso
- **Basso** (nice to have): da soddisfare solo se non richiede grandi sforzi

Priorità	Requisito
Must-have	<ul style="list-style-type: none">• Ogni contatto deve avere almeno nome o cognome• Interfaccia grafica (GUI)• Gestione contatti: creazione, modifica ed eliminazione• Ricerca contatti
Should-have	<ul style="list-style-type: none">• Import/Export
Nice-to-have	<ul style="list-style-type: none">• Ogni contatto può avere informazioni aggiuntive opzionali• Catalogare i contatti in "preferiti" ed "emergenza"• Standard vCard

Tabella 2: Definizione delle priorità in base al "business value"

4.4.2 Rischio tecnico

- **Alto:** è difficile prevedere se si riuscirà a realizzare il requisito
- **Medio:** si ritiene di poter realizzare il requisito, ma ci sono alcuni fattori di rischio che non possono essere controllati
- **Basso:** si ha piena fiducia nella propria capacità di realizzarlo

Rischio tecnico	Requisito
Alto	<ul style="list-style-type: none">• Import/Export (vCard)
Medio	<ul style="list-style-type: none">• Interfaccia grafica (GUI)• Catalogare i contatti in "preferiti" ed "emergenza"
Basso	<ul style="list-style-type: none">• Ogni contatto deve avere almeno nome o cognome• Gestione contatti: creazione, modifica ed eliminazione contatto• Ricerca contatti• Lista ordinata alfabeticamente• Ogni contatto deve avere da 0 a 3 numeri di telefono• Ogni contatto deve avere da 0 a 3 indirizzi email• Visualizzare i contatti in ordine alfabetico (per nome o cognome)

Tabella 3: Definizione delle priorità in base al "rischio tecnico"

4.5 Casi d'uso

1. Creazione di un contatto
2. Modifica di un contatto
3. Eliminazione di un contatto
4. Ricerca di un contatto
5. Aggiunta di un contatto fra i "contatti di emergenza"
6. Rimozione di un contatto dai "contatti di emergenza"
7. Aggiunta di un contatto fra i "contatti preferiti"
8. Rimozione di un contatto dai "contatti preferiti"
9. Import fa file .vcf
10. Export su file .vcf

4.5.1 Descrizione dei casi d'uso

1. Creazione di un contatto

- **Attore:** Utente
- **Pre-condizioni:** l'utente è in possesso di Nome, cognome, indirizzo e-mail e numero telefonico valido
- **Post-condizioni:** il contatto viene creato e salvato in rubrica
- **Flusso standard di eventi:**
 - (a) L'utente inserisce i dati negli appositi campi di testo
 - (b) L'utente conferma l'operazione cliccando il pulsante "Salva"
 - (c) Il sistema controlla la validità dei campi inseriti
 - (d) Il sistema salva e aggiunge il contatto in rubrica
- **Flusso alternativo di eventi:**
 - (a) L'utente inserisce i dati negli appositi campi di testo
 - (b) L'utente decide di lasciare invariato il contatto che aveva intenzione di modificare
 - (c) L'utente NON clicca il pulsante "Salva"

2. Eliminazione di un contatto

- **Attore:** Utente
- **Pre-condizioni:** il contatto è presente in rubrica
- **Post-condizioni:** il contatto è rimosso dalla rubrica
- **Flusso standard di eventi:**
 - (a) L'utente cerca nella rubrica il contatto
 - (b) L'utente seleziona il contatto da eliminare

- (c) L'utente clicca il tasto "elimina contatto"

3. Ricerca di un contatto

- **Attore:** Utente
- **Pre-condizioni:** l'utente conosce nome o cognome del contatto che desidera cercare
- **Post-condizioni:** l'utente ha trovato il contatto
- **Flusso standard di eventi:**
 - (a) L'utente seleziona la barra di ricerca;
 - (b) L'utente inserisce nome o cognome del contatto
 - (c) Il sistema mostra i risultati della ricerca secondo il criterio di ordinamento
- **Flusso alternativo di eventi:**
 - (a) L'utente seleziona la barra di ricerca
 - (b) L'utente inserisce nome o cognome
 - (c) Il contatto non è presente nella rubrica
 - (d) Il sistema aggiorna la vista mostrando la tabella vuota

4. Export della rubrica


- **Attore:** Utente
- **Pre-condizioni:** La rubrica è creata ed è presente almeno 1 contatto
- **Post-condizioni:** Il programma restituisce un file .vcf
- **Flusso standard di eventi:**
 - (a) L'utente seleziona dalla barra degli strumenti la voce "File->Export"
 - (b) L'utente sceglie il percorso in cui salvare il file sul disco fisso
 - (c) Il sistema genera il file .vcf della rubrica e lo salva nel percorso specificato dall'utente

5. Aggiunta di un contatto nei "preferiti"


- **Attore:** Utente
- **Pre-condizioni:** Il contatto interessato è presente in rubrica
- **Post-condizioni:** Il contatto selezionato è aggiunto alla sotto-rubrica "Preferiti"
- **Flusso standard di eventi:**
 - (a) L'utente seleziona il contatto da aggiungere ai preferiti
 - (b) L'utente, nella scheda del contatto, clicca l'apposita icona ★
 - (c) Il sistema aggiunge il contatto dalla sotto-rubrica "Preferiti"

6. Rimozione di un contatto dai "preferiti"


- **Attore:** Utente
- **Pre-condizioni:** Il contatto da rimuovere deve essere nei "Preferiti"

- **Post-condizioni:** Il contatto selezionato è rimosso alla sotto-rubrica “Preferiti”
- **Flusso standard di eventi:**
 - (a) L’utente seleziona il contatto da rimuovere dai preferiti
 - (b) L’utente, nella scheda del contatto clicca l’apposita icona 
 - (c) Il sistema rimuove il contatto dalla sotto-rubrica “Preferiti”

7. Aggiunta di un contatto fra i “contatti di emergenza”

- **Attore:** Utente
- **Pre-condizioni:** Il contatto interessato è presente in rubrica
- **Post-condizioni:** Il contatto selezionato è aggiunto alla sotto-rubrica “Emergenza”
- **Flusso standard di eventi:**
 - (a) L’utente seleziona il contatto da aggiungere ai contatti d’emergenza
 - (b) L’utente, della scheda del contatto, clicca l’apposita icona “”
 - (c) Il sistema aggiunge il contatto alla sotto-rubrica “Emergenza”

8. Rimozione di un contatto da i “contatti di emergenza”

- **Attore:** Utente
- **Pre-condizioni:** Il contatto da rimuovere deve essere nei “Contatti di Emergenza”
- **Post-condizioni:** Il contatto selezionato è rimosso dalla sotto-rubrica “Emergenza”
- **Flusso standard di eventi:**
 - (a) L’utente seleziona il contatto da rimuovere dai contatti d’emergenza
 - (b) L’utente, della scheda del contatto, clicca l’apposita icona “”
 - (c) il sistema rimuove il contatto dalla sotto-rubrica “Emergenza”

4.5.2 Diagramma dei casi d’uso

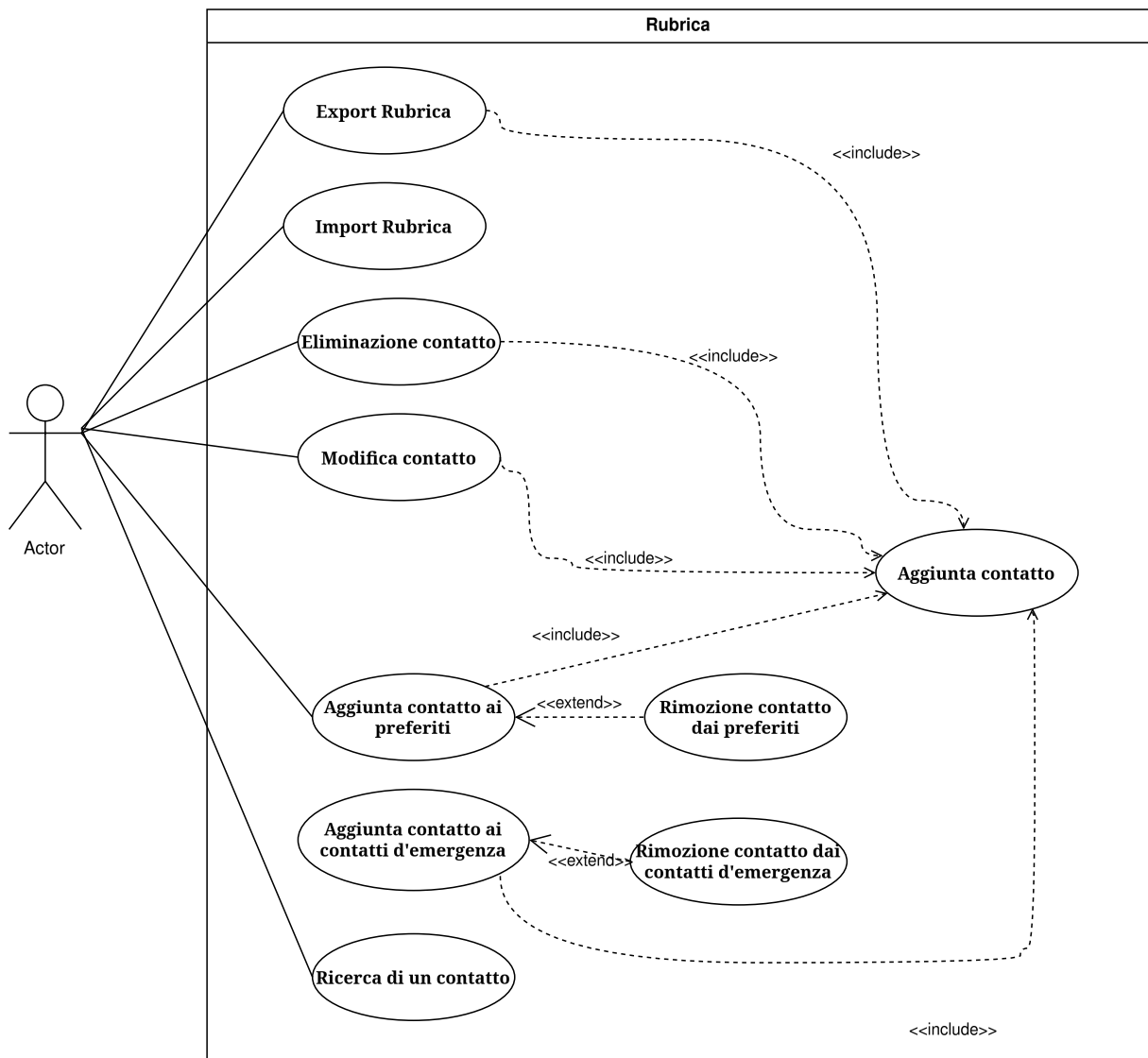


Figura 5: Diagramma dei casi d'uso

Parte III

Design

5 Attributi di qualità (QA)

5.1 QA esterni

- **Scalabilità:** il sistema è stato progettato e pensato per supportare eventuali espansioni in futuro, con modifiche minimali al codice sorgente.
- **Usabilità:** il sistema mette a disposizione un'interfaccia grafica semplice e chiara.

5.2 QA interni

- **Manutenibilità:** il sistema è stato progettato con un alto livello di coesione e basso livello di accoppiamento al fine di garantire una semplice manutenzione del codice.
- **Riusabilità e Modularità:** la suddivisione in moduli (principalmente secondo il pattern MVC), garantisce la separazione delle responsabilità. Tali moduli possono essere tranquillamente utilizzati in altri progetti.
- **Portabilità:** il sistema è stato progettato per funzionare su qualunque dispositivo Desktop che supporti la macchina virtuale java - JVM (Java Virtual Machine).

6 Diagramma dei package

Diagramma dei package

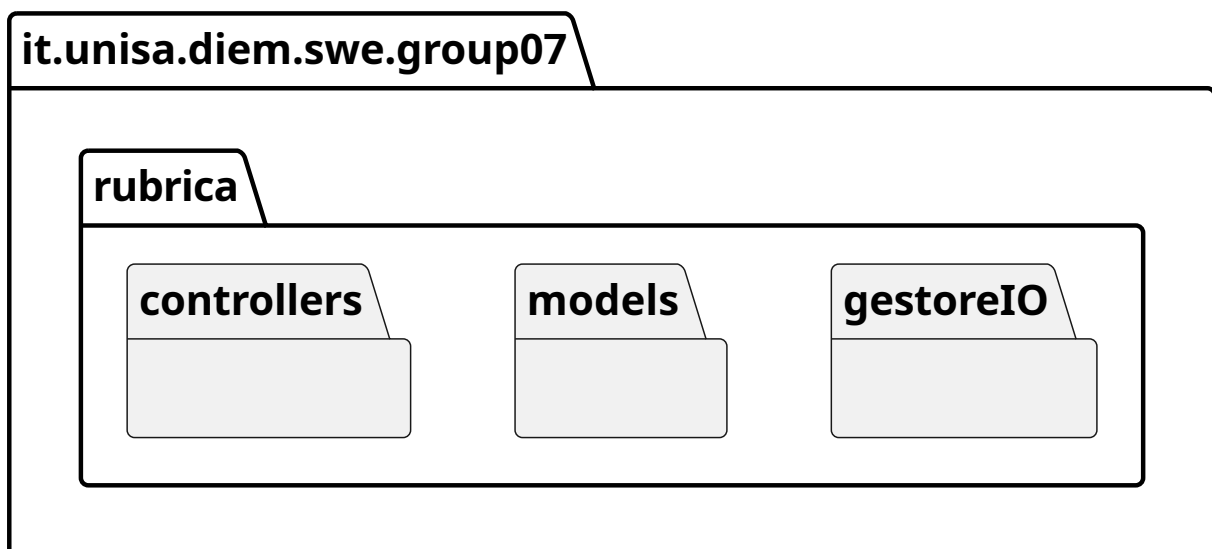


Figura 6: Diagramma dei package

Il diagramma fig. 6 rappresenta l'organizzazione del codice in package differenti, al fine di separare al meglio funzionalità e responsabilità. La suddivisione segue il pattern MVC:

1. **gestoreIO**: si occupa della gestione dell'input/output e del formato vCard
2. **models**: contiene le classi che rappresentano le entità principali del dominio applicativo (modelli)
3. **controllers**: package dedicato alla logica applicativa e all'interazione tra modelli e viste

7 Diagramma delle classi

Per quanto riguarda la struttura delle classi, si è deciso di presentare **2 diagrammi** delle classi differenti: rispettivamente una versione **"lite"** fig. 7 in cui non compaiono gli elementi FXML e i metodi setter/getter, ed una versione completa fig. 8.

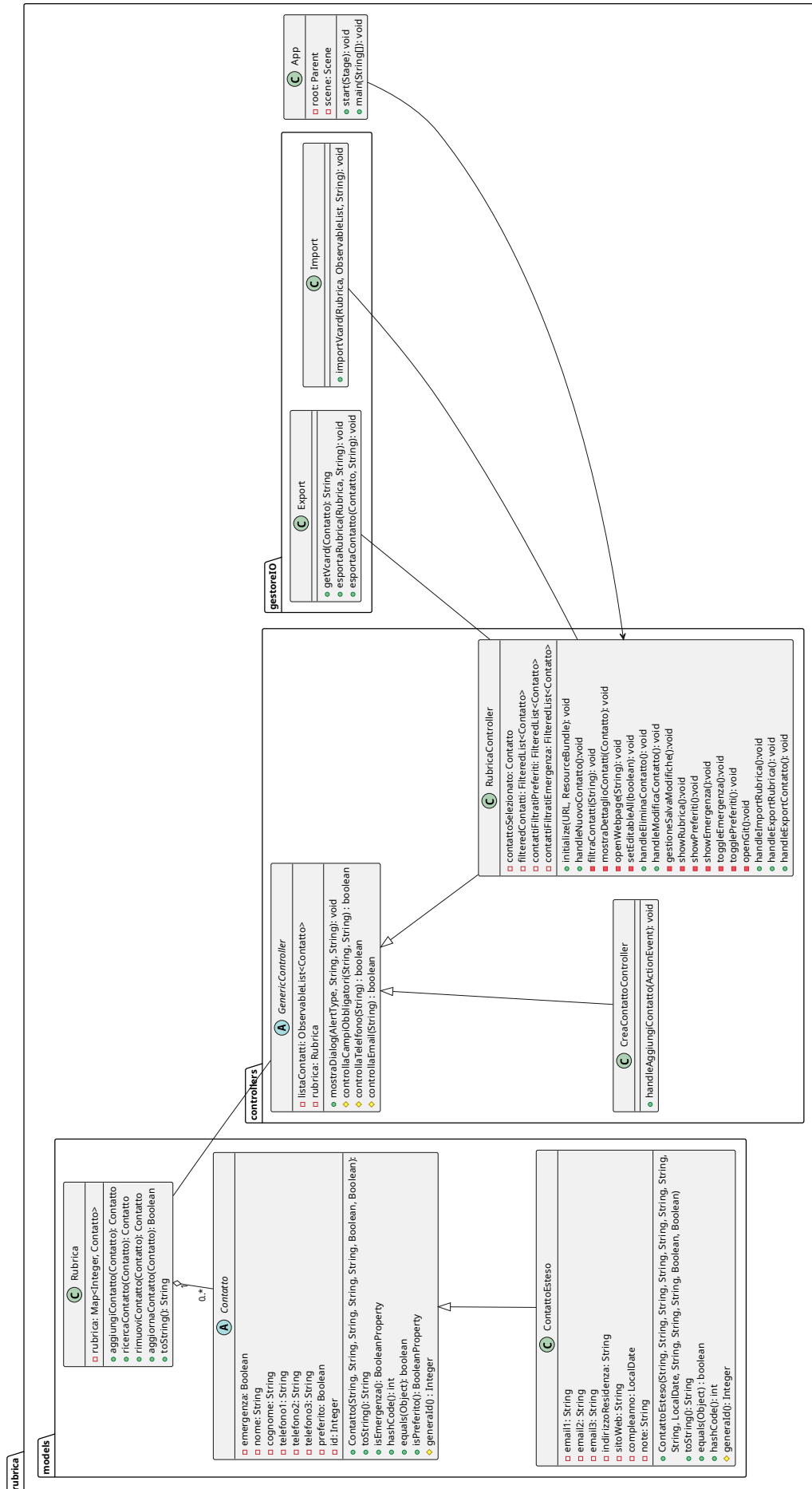


Figura 7: Diagramma delle classi compatto

7 DIAGRAMMA DELLE CLASSI

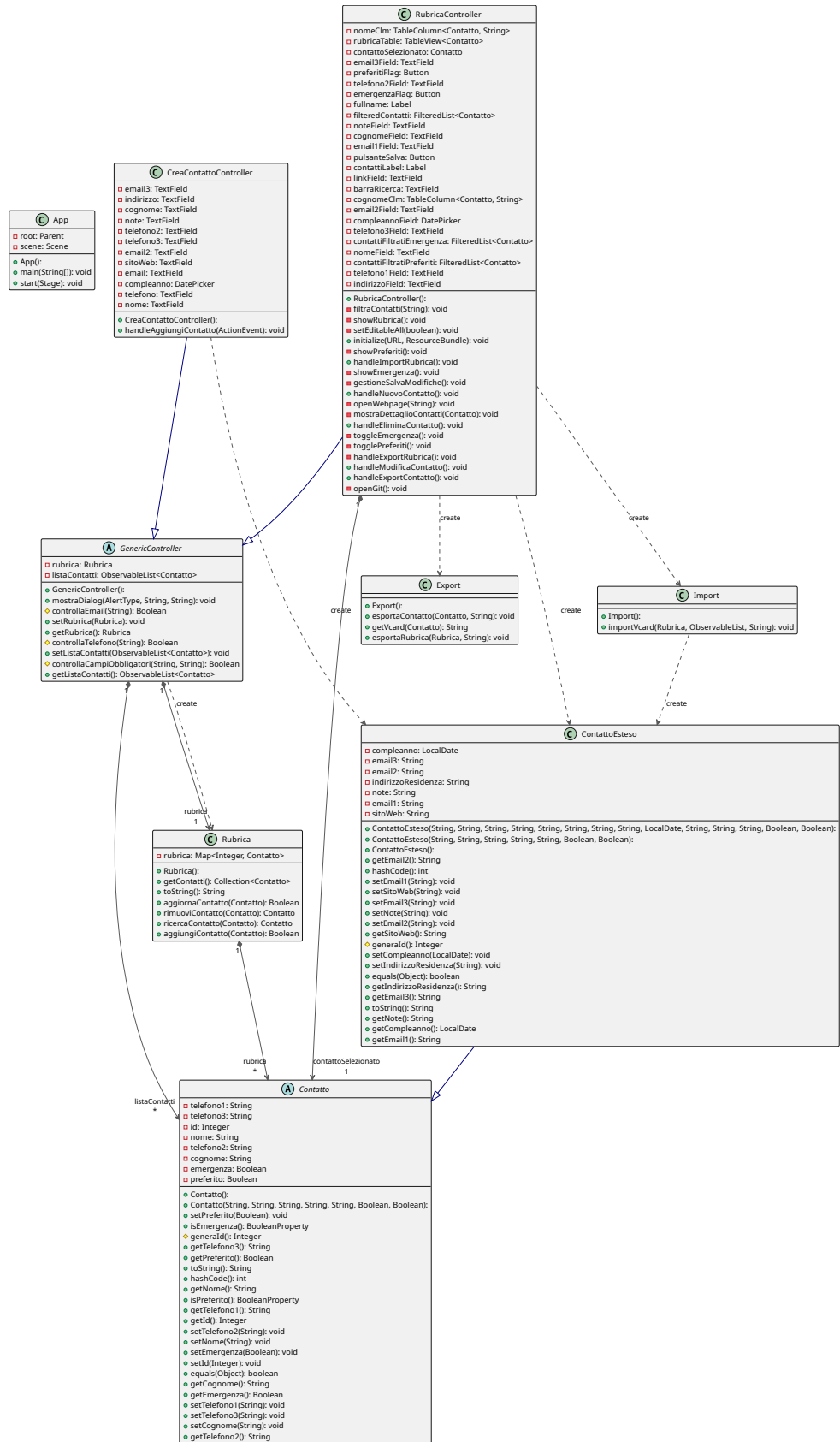


Figura 8: Diagramma delle classi esteso

7.1 Descrizione ed analisi delle classi

Si analizzano le varie classi implementate in termini di coesione e accoppiamento. E' bene ricordare che per ottenere una buona decomposizione è buona norma ottenere **alta coesione** e **basso accoppiamento**.

7.1.1 Classe "Rubrica"

La classe rubrica si occupa della definizione e gestione della rubrica, offrendo operazioni di aggiunta, rimozione, modifica e ricerca di contatti. Ogni metodo implementato contribuisce all'obiettivo generale della gestione della rubrica e non sono presenti inter-dipendenze per cui, il livello di **coesione** é **funzionale**.

La classe rubrica è accoppiata (e dipendente) alle classi "Contatto" e "ContattoEsteso" mediante la struttura dati "Map<Integer, Contatto>", la quale usa come valori, degli oggetti di tipo "Contatto". Si ha in tal senso un **accoppiamento per dati**.

Si ha un accoppiamento anche con i "controllers", in particolar modo con "GenericController" che mantiene il riferimento alla rubrica condivisa istanziandola.

7.1.2 Classe "Contatto"

La classe "Contatto" rappresenta un semplice "contatto telefonico" dotato di un set ristretto di attributi: nome, cognome, 3 numeri telefonici, preferito ed emergenza. La classe è stata pensata appositamente **astratta** (e quindi non istanziabile) per una questione di semplicità, soprattutto in ottica di aggiornamenti futuri e nel rispetto dei principi **SOLID** (principio di sostituzione di Liskov). Rappresentando l'entità basilare del dominio applicativo, la classe "Contatto" è quella che dovrebbe essere estesa qualora si volesse aggiungere una funzionalità aggiuntiva (come é stato fatto con la classe "ContattoEsteso") al sistema.

7.1.3 Classe "ContattoEsteso"

La classe "ContattoEsteso" al momento, rappresenta l'unico oggetto istanziabile aggiungibile all'interno della rubrica. Estende (accoppiamento per dati) la classe "Contatto", da cui eredita attributi e metodi pubblici e si occupa di gestire le **informazioni aggiuntive** di un generico contatto. In particolar modo la classe definisce i seguenti attributi accessori: 3 email, compleanno, indirizzo di residenza, sito web, note.

7.1.4 Classe "GenericController"

La classe "GenericController" astrae il concetto di un generico **controller** (MVC) responsabile dell'interazione della vista con i modelli (view-model). La classe mantiene un riferimento (ed istanzia) alla **rubrica** (Map), alla lista osservabile e mette a disposizione alcuni metodi getter/setter ed alcuni metodi di utilità generici (come per esempio mostraDialog() o i controlli sulle email o i numeri telefonici). Ciò garantisce il minor spreco possibile, in quanto evita che ogni controller debba creare ogni volta un nuovo oggetto "Rubrica". Si soddisfa in questo modo il principio **DRY** - "Don't Repeat Yourself".

7.1.5 Classe "RubricaController"

La classe si occupa di gestire le interazioni dell'utente con l'**interfaccia grafica principale** (fig. 3). Estende la classe "GenericController" ereditandone gli attributi, in particolare il riferimento alla rubrica.

7.1.6 Classe "CreaContattoController"

La classe si occupa di gestire le **interazioni** dell'utente con l'interfaccia grafica secondaria (fig. 4), responsabile della creazione di un nuovo contatto. Estende la classe "GenericController" ereditandone gli attributi, in particolare il riferimento alla rubrica.

7.1.7 Classe "Import"

La classe si occupa unicamente della **logica di importazione** da file .vcf. Il metodo implementato consente di leggere un file e salvare i dati nell'apposita struttura dati, garantendo in questo modo alta coesione. La classe è direttamente accoppiata con la **rubrica** (passata come parametro al metodo "importVcard()") e pertanto al generico controller "GenericController" e al controller principale del sistema "RubricaController".

7.1.8 Classe "Export"

La classe si occupa unicamente della **logica di esportazione** su file .vcf. I metodi implementati consentono di prelevare dalla struttura dati e scrivere su un file .vcf rispettivamente un singolo contatto o l'intera rubrica. La classe è pertanto direttamente accoppiata con la **rubrica** (passata come parametro al metodo "importVcard()") e quindi al generico controller "GenericController" e al controller principale del sistema "RubricaController".

8 Diagrammi di sequenza

Per non appesantire eccessivamente i diagrammi si è deciso di accorpare le entità "View" e "Controller" sotto un unico attore del tipo **"*ViewController"** e di mostrare solamente le interazioni principali tra gli attori partecipanti, escludendo per esempio, alcuni valori di ritorno scontati o gli aggiornamenti espliciti della TableView.

L'attore **"Rubrica"**, che compare nei vari diagrammi, è una rappresentazione **ideale** delle collezioni dati utilizzate... in particolar modo rappresenta una Mappa, una lista osservabile e una lista filtrabile.

Le prime due strutture sono simmetriche: ad un aggiornamento della mappa corrisponde un aggiornamento della lista. Per questo motivo abbiamo deciso di rappresentare la "Rubrica" come un **unico attore** partecipante.

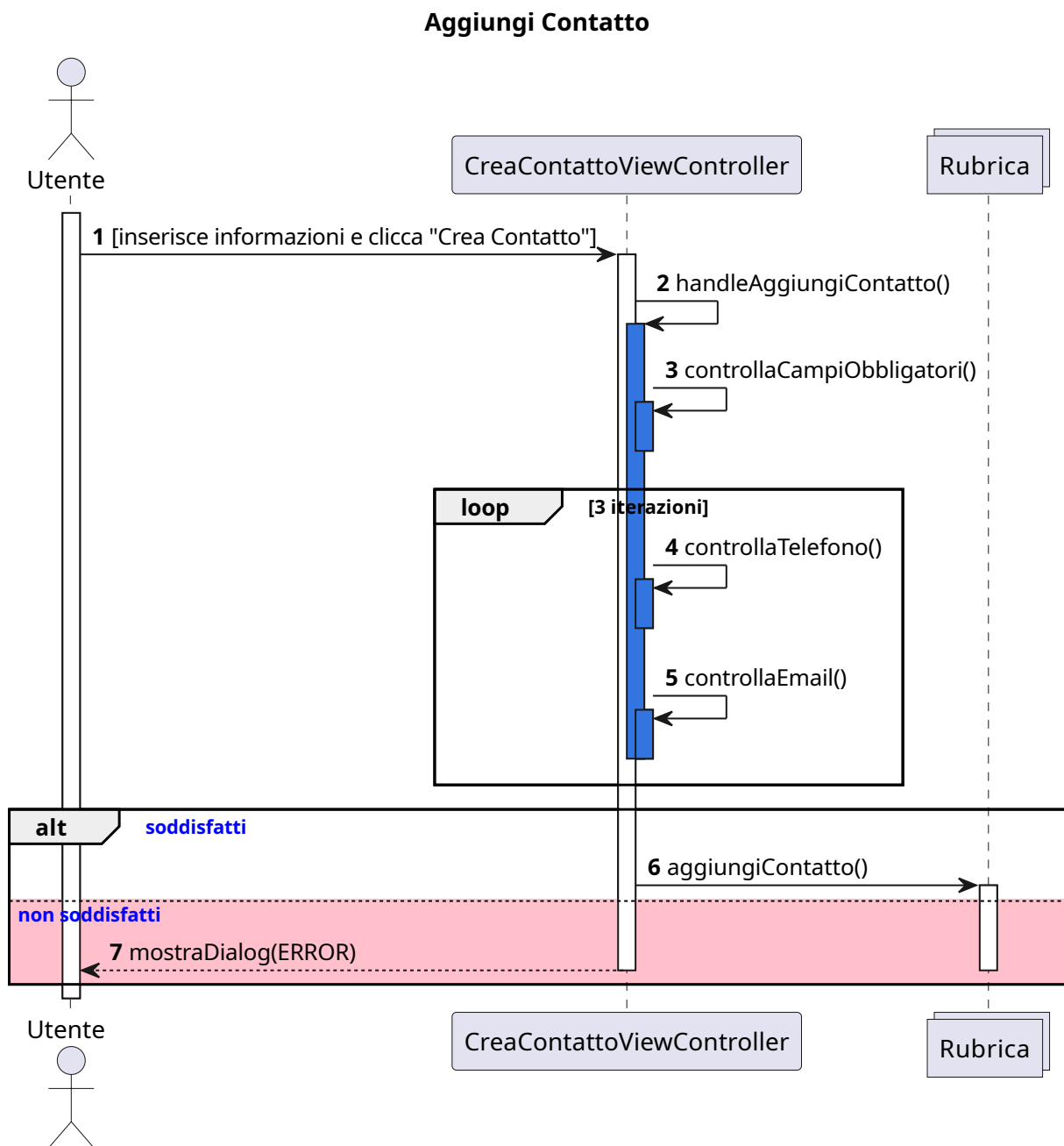


Figura 9: D. Sequenza - Aggiungi Contatto

Il diagramma in fig. 9” mostra lo scenario di aggiunta di un nuovo contatto alla rubrica. Si presuppone, in tal caso, che l’utente abbia già interagito con la vista principale della rubrica e cliccato il pulsante **”Nuovo Contatto”**. L’utente quindi si trova davanti alla vista secondaria (??) che consente l’inserimento delle informazioni sul contatto da aggiungere in rubrica.

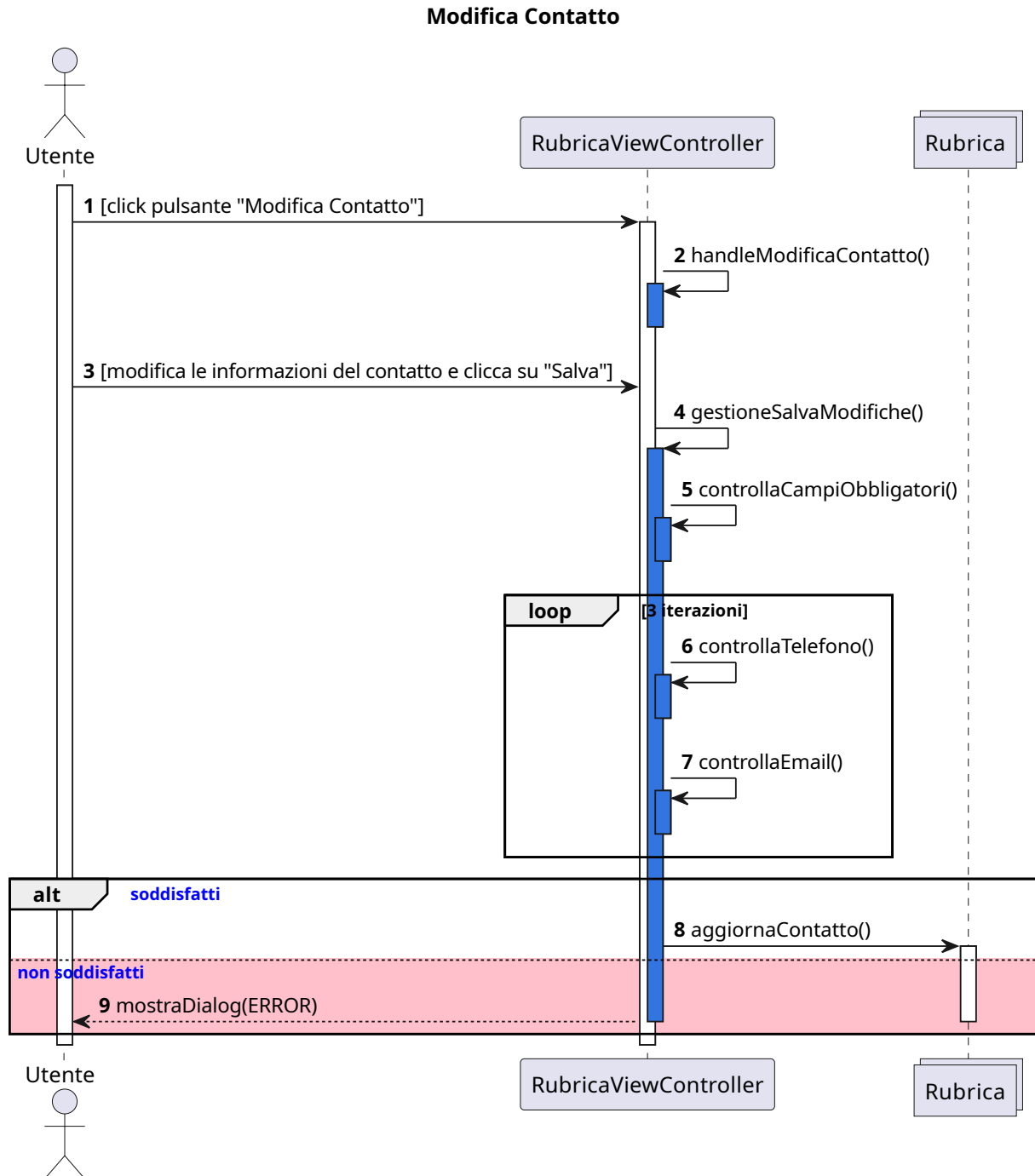


Figura 10: D. Sequenza - Aggiungi Contatto

Il diagramma in fig. 10 mostra lo scenario di modifica di un contatto presente in rubrica. Si presuppone che l’utente abbia già scelto un contatto da modificare selezionandolo dalla TableView.

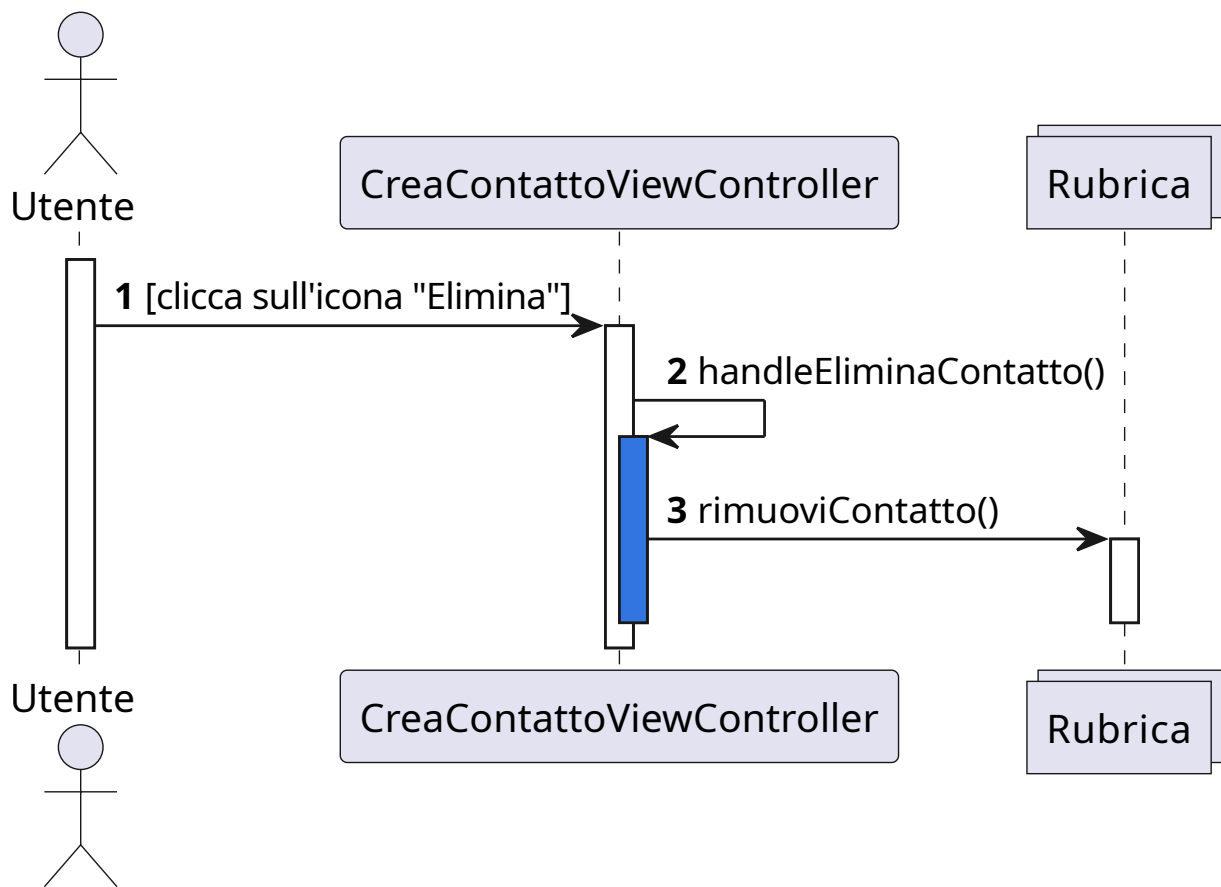
Rimuovi Contatto

Figura 11: D. Sequenza - Aggiungi Contatto

Il diagramma in fig. 11” mostra lo scenario di cancellazione di un contatto presente in rubrica. Si presuppone che l’utente abbia già scelto un contatto da eliminare selezionandolo dalla TableView.

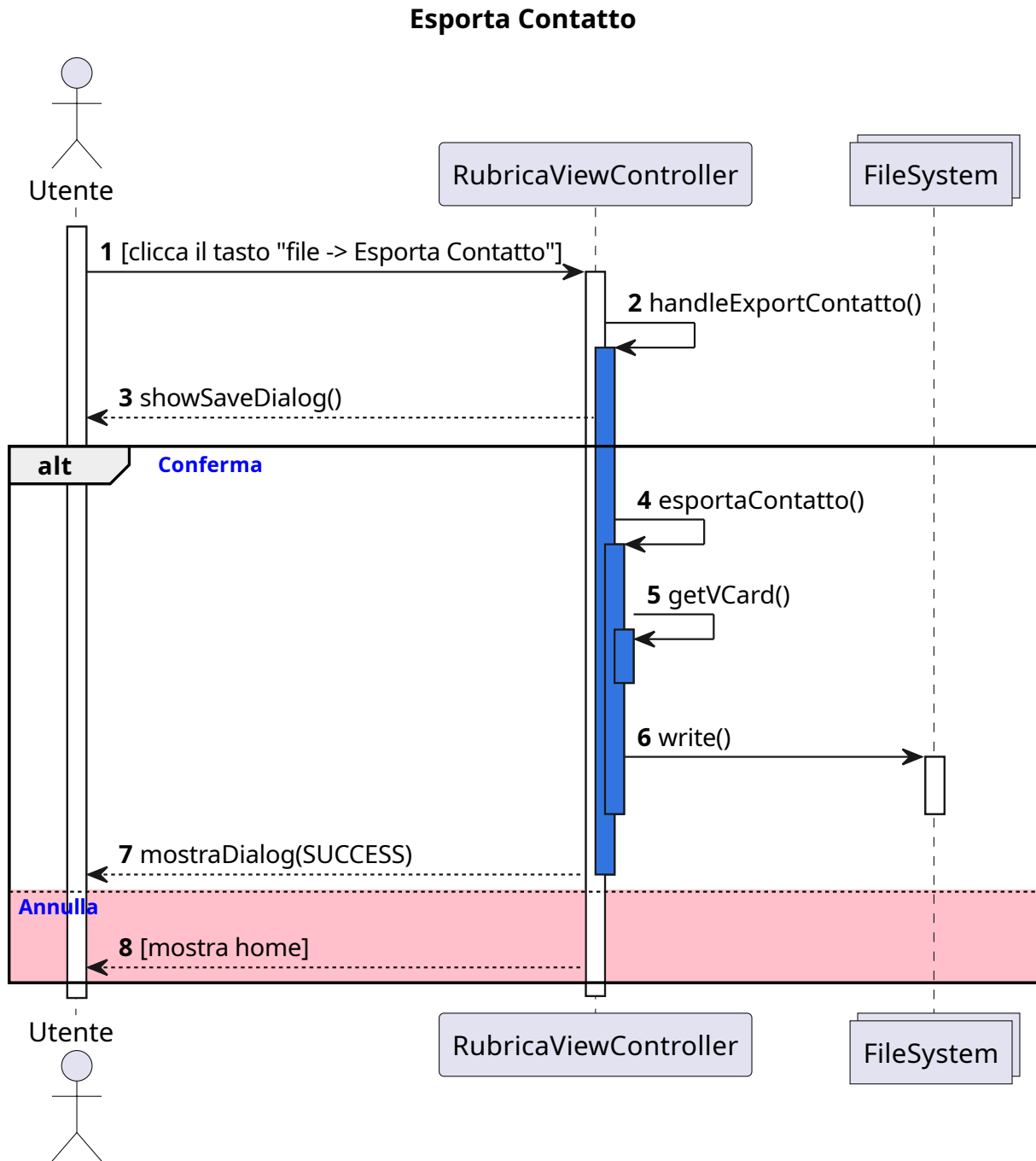


Figura 12: D. Sequenza - Esporta Contatto

Il diagramma in fig. 12" mostra lo scenario di esportazione di un singolo contatto presente in Rubrica. Si presuppone che l'utente abbia già scelto un contatto da esportare selezionandolo dalla TableView. Si é deciso di mostrare l'attore **"FileSystem"** per mostrare esplicitamente il comportamento del **fileChooser** implementato nonché l'interazione dell'utente con il disco rigido. Per non appesantire eccessivamente il diagramma é stata rimossa la Rubrica come partecipante.

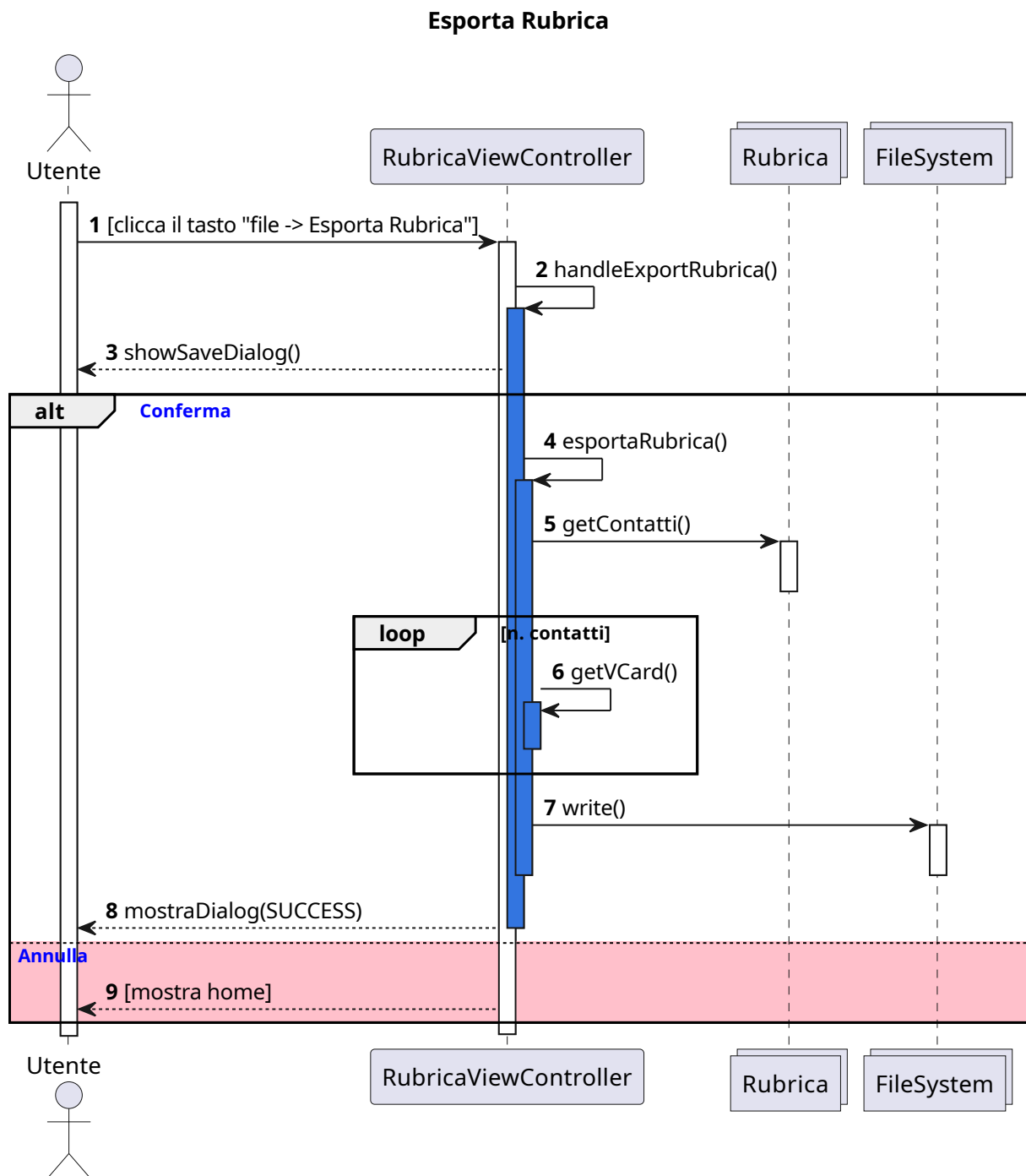


Figura 13: D. Sequenza - Esporta Rubrica

Il diagramma in fig. 12" mostra lo scenario di esportazione dell'intera rubrica. Si é deciso di mostrare l'attore "**FileSystem**" per mostrare esplicitamente il comportamento del **fileChooser** implementato nonché l'interazione dell'utente con il disco rigido.



Imposta Contatto Emergenza

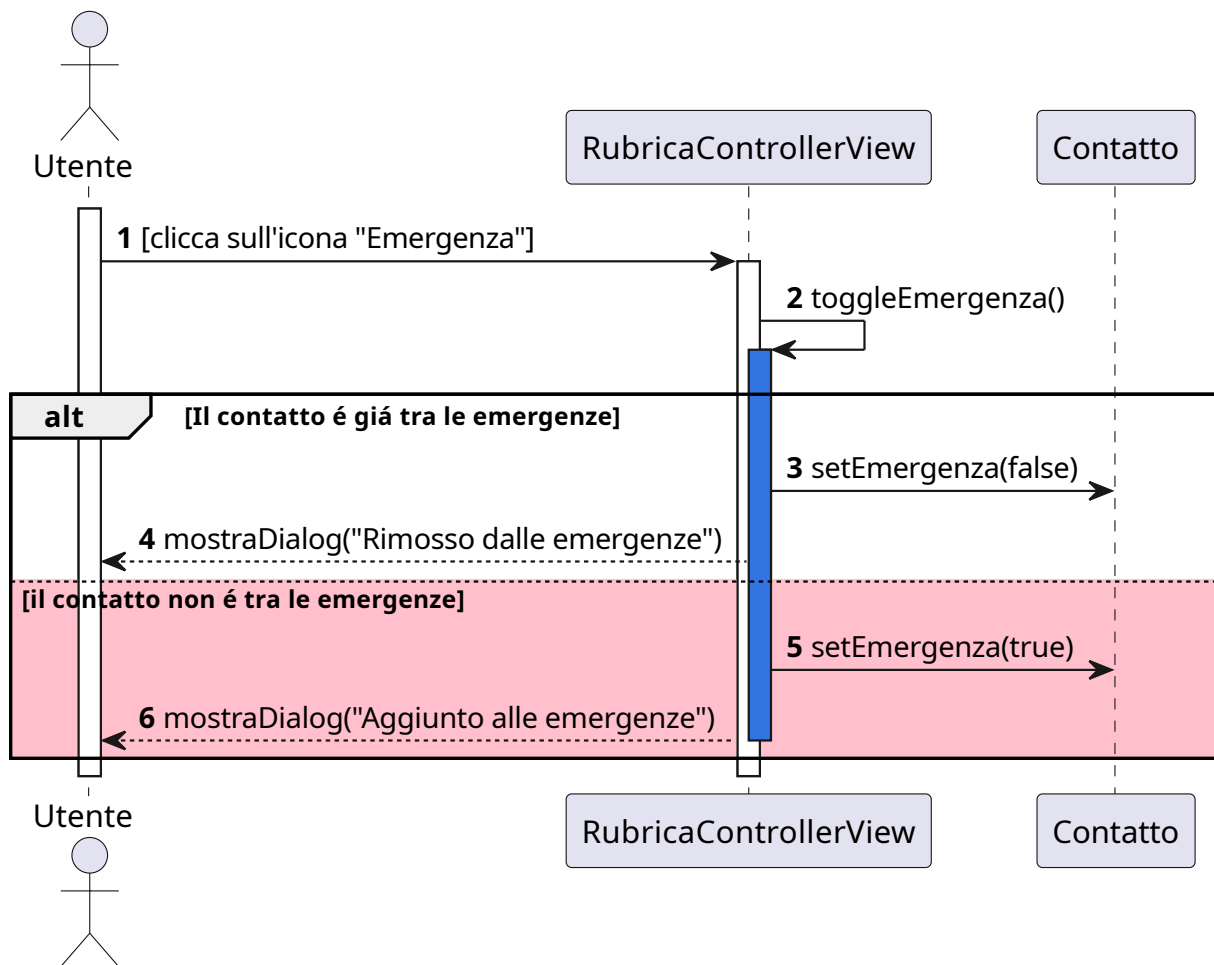


Figura 15: D. Sequenza - Toggle Emergenza

Il diagramma in fig. 15” mostra lo scenario di **aggiunta e rimozione** (toggle) di un contatto dalle "Emergenze". Dal momento che l’aggiunta e la rimozione da tale lista è determinata da un attributo privato del contatto, si è deciso di non mostrare esplicitamente la Rubrica, bensì l’ **attore "Contatto"**.

Imposta Contatto Preferito

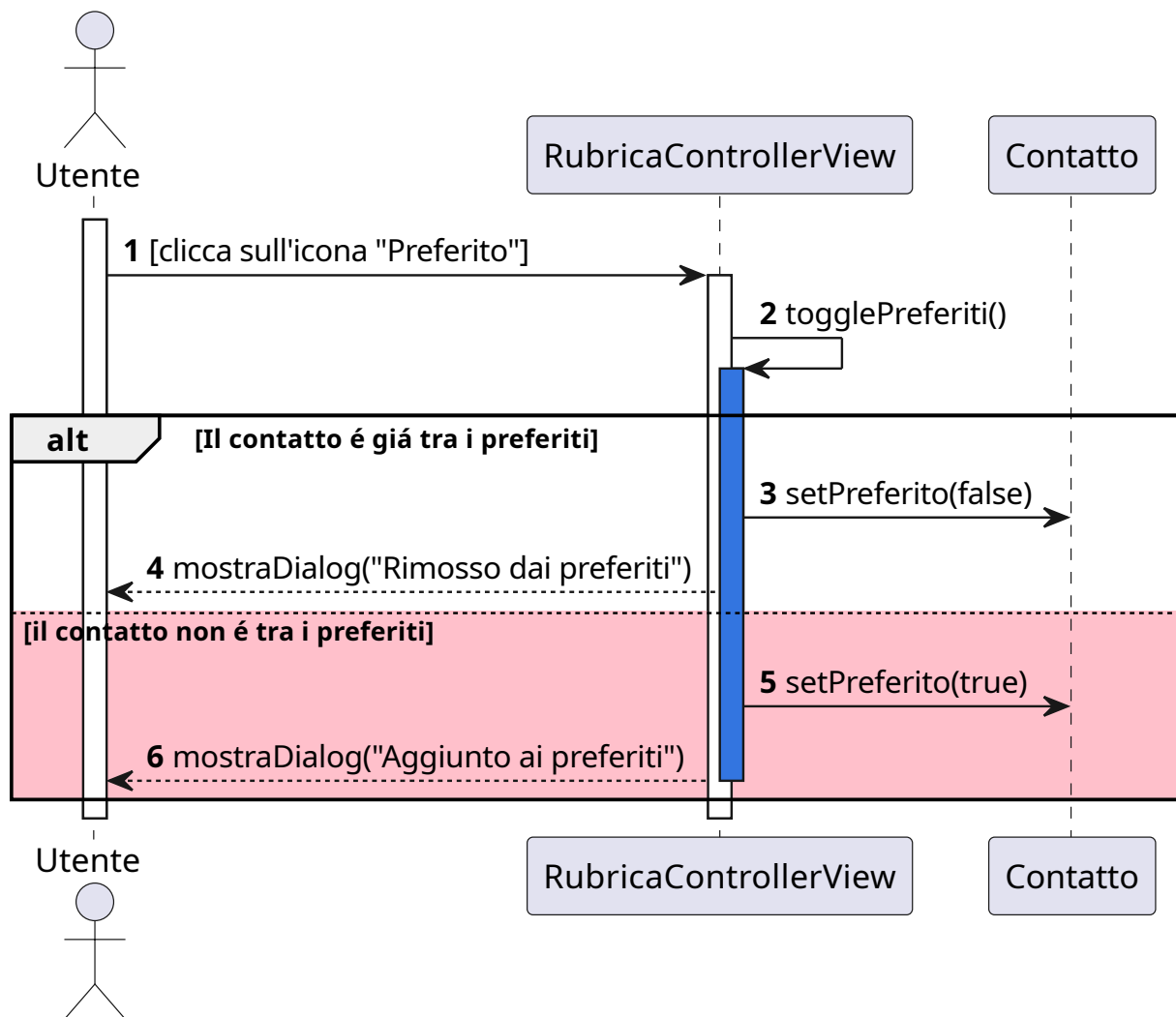


Figura 16: D. Sequenza - Toggle Preferito

Il diagramma in fig. 15" mostra lo scenario di **aggiunta e rimozione** (toggle) di un contatto dai "Preferiti". Dal momento che l'aggiunta e la rimozione da tale lista è determinata da un attributo privato del contatto, si è deciso di non mostrare esplicitamente la Rubrica, bensì l' **attore "Contatto"**.