MyContacts

Design

Indice

Changelog

- Classe **GestoreOperazioniRubrica**: Abbiamo rimosso questa classe e ora gestiamo le operazioni direttamente all'interno delle classi Rubrica e RubricaPreferiti

- Attributo **Numeri**: Ora utilizziamo un tipo String anziché il tipo Integer

---

Ora analizziamo le modifiche singolarmente per ogni classe:

- Classe **Contatto:**
    - Attributo **numeri**: Modificato da List<String> a Set<String>, adattando di conseguenza i metodi get per restituire un Set invece di una List.
    - Attributo **email**:Modificato da List<String> a Set<String>, adattando di conseguenza i metodi get per restituire un Set invece di una List.
    - Metodi **set**: Rimosso l'implementazione dei metodi set per Nome, Cognome, Numero, ed Email.
    - Rimossi metodi ModificaContatto() e SwitchPreferiti() in quanto non usati

- **Classe Rubrica:**
        - **Attributo elenco**: Modificato da List a ObservableSet, con conseguenti modifiche ai metodi di accesso.
        - **Metodi di gestione contatti**:
    - Rinominato il metodo **addContatto() in aggiungiContatto()**, per uniformità con altre classi.
    - Rinominato il metodo **removeContatto() in rimuoviContatto()**.
- **Nuovi metodi**: Aggiunti i metodi **esportaRubrica()** e **importaRubrica().**

- Classe **RubricaPreferiti:**
    - Attributo **elencoPreferiti**: Modificato da List a ObservableSet, con conseguente aggiornamento del metodo getElencoPreferiti() per restituire un ObservableSet.

- **MainViewController:**
    - Attributo **leftPane**: Modificato da AnchorPane a StackPane.
    - Nuovo **attributo**: Aggiunta la definizione della classe Rubrica con il relativo attributo rubrica.

- **LeftViewController:**
    - **Pulsante resetButton**: Aggiunto un nuovo tasto
    - **Nuovi attributi**: Aggiunti gli attributi rubrica, listaContatti, listaContattiPreferiti e mainViewController.

- **RightView1Controller:**
    - **Nuovi attributi**: Aggiunti gli attributi rubrica, listaContatti, listaContattiPreferiti e mainViewController.
    - **Metodi aggiornati**: Aggiunto il metodo **setContatto()** e rimosso il metodo **switchPreferito()**.

- **RightView2Controller:**
    - **Attributi**: Aggiunti nuovi attributi alla classe.
    - **Metodi aggiornati**: Aggiunti i metodi setContatto(Contatto contatto) e controlloNumero(KeyEvent event)

Introduzione

Il design del sistema “MyContacts” è stato sviluppato con l’obiettivo di creare una struttura solida e facilmente manutenibile, ponendo particolare attenzione al principio di elevata coesione e basso accoppiamento. Questi due principi cardine rappresentano il fondamento per garantire una buona struttura, ove ogni componente sia ben definito e possa operare in modo indipendente.

L'approccio seguito ha permesso di suddividere chiaramente le responsabilità tra i vari moduli, migliorando la leggibilità e semplificando eventuali interventi futuri di manutenzione o estensione del sistema.

Coesione

La coesione è stata massimizzata attraverso una chiara suddivisione dei compiti. Ogni componente del sistema è stato progettato per avere una responsabilità ben definita:

- **Classi CONTROLLER:** si occupano della sola gestione di ogni componente richiamando metodi della classi Contatto , Rubrica e RubricaPreferiti per la gestione delle stesse, tutte le classi controller sono gestite da MainViewController.
- **Classi «Rubrica» e «RubricaPreferiti»:** Si occupano della sola gestione dei contatti all'interno delle collezioni.
- **Classe «Contatto»:** si occupa della sola gestione del formato dei dati del contatto e della loro validità.

Questa suddivisione ad alta coesione permette a ciascun modulo di concentrarsi su un insieme specifico di funzioni, riducendo la complessità interna e facilitando le modifiche.

Accoppiamento

Per minimizzare l'accoppiamento sono state utilizzate tecniche di progettazione basate sull'indipendenza di ogni modulo, nello specifico:

- **Classi CONTROLLER:** le classi RightViewController e LeftViewController sono indipendenti tra di loro e gestiscono componenti di gruppi diversi della GUI.
- **Classi «Rubrica» e «RubricaPreferiti»:** gestiscono le collezioni di contatti garantendone la divisione, allo stesso tempo ignorano il formato di ogni contatto, garantendo l'indipendenza dalla classe stessa.
- **Classe «Contatto»:** è la classe meno dipendente dalle altre in quanto non utilizza metodi o attributi di altre classi.

Questo approccio garantisce che i cambiamenti in un modulo abbiano un impatto minimo sugli altri, semplificando la manutenzione e favorendo il riutilizzo del codice.

Definizione delle Classi

Le classi presenti all'interno del nostro progetto sono:

- Contatto
- Rubrica
- RubricaPreferiti
- MyContacts
- MainViewController
- LeftViewController
- RightView1Controller
- RightView2Controller
- InvalidContactException

Oltre alle classi sono presenti anche dei file FXML per la modifica del layout dell'applicazione:

- MainView
- LeftView
- RightView1
- RightView2

Descrizione funzionale delle classi

Contatto

La classe Contatto rappresenta un'entità all'interno di un sistema di gestione contatti. Ogni oggetto di questa classe conserva informazioni relative a un contatto, inclusi dati anagrafici (nome, cognome) e informazioni di comunicazione (numeri telefonici e indirizzi email). La classe prevede anche un attributo booleano per identificare se un contatto è stato contrassegnato come "preferito".

Attributi

- nome (String): Rappresenta il nome del contatto.
- cognome (String): Rappresenta il cognome del contatto.
- numeri (Set<String>): Un Set di numeri telefonici, che consente di memorizzare fino a 3 numeri associati al contatto.
- email (Set<String>): Un Set di indirizzi email, che permette di memorizzare fino a 3 email associate al contatto.
- preferito (boolean): Indica se il contatto è stato aggiunto ai "preferiti" dell'utente (true se preferito, false altrimenti).

Metodi

- **Costruttore**
  - Contatto(String nome, String cognome, Set<String> numeri, Set<String> email, boolean preferito): Crea un contatto che ha come attributi i valori passati come parametri. Se sia il nome che il cognome non vengono inseriti lancia un'eccezione.

- **Metodi di Accesso (Getter)**
  - String getNome(): Ottiene il nome del contatto.
  - String getCognome(): Ottiene il cognome del contatto.
  - Set<Integer> getNumeri(): Ottiene la lista dei numeri di telefono.
  - Set<String> getEmail(): Ottiene la lista degli indirizzi email.

- boolean isPreferito(): Ottiene lo stato "preferito" del contatto.

- **Metodi di Confronto e Ordinamento**
  - int compareTo(Contatto o): Implementa il confronto per ordinare i contatti (prima per cognome e poi per nome).
  - boolean equals(Object obj): Verifica se due contatti sono uguali. Due contatti sono considerati uguali se hanno lo stesso nome, cognome, numeri e email.

MyContacts

La classe MyContacts è progettata per avviare l'interfaccia grafica dell'applicazione. Si occupa di caricare il file FXML per l'interfaccia utente, visualizzarlo in una finestra e gestire il cambiamento dinamico del contenuto della scena.

Attributi

- scena (Scene): Rappresenta la scena attuale dell'applicazione.

Metodi

- **Metodi di Avvio e Gestione della Scena:**
  - void start(Stage stage): Istanzia e visualizza la schermata principale. Il controllore non viene istanziato manualmente, ma viene caricato tramite il loader FXML.
  - Parent loadFXML(String fxml): Carica un file FXML, che definisce l'interfaccia utente della finestra principale. Utilizza FXMLLoader per leggere il file FXML nella directory delle risorse.
  - void main(String[] args): chiama la funzione di sistema launch(), che avvia l'applicazione.

Rubrica

La classe Rubrica gestisce l'elenco principale dei contatti, con operazioni fondamentali per la manipolazione di questi ultimi, come l'aggiunta, la rimozione e la verifica dello stato della rubrica.

- **Attributi**
  - elenco (ObservableSet<Contatto>): Rappresenta l'insieme dei contatti principali, utilizzando un ObservableSet perché notifica automaticamente i cambiamenti, sincronizzando i dati con la UI e garantendo unicità.
  - elencoPreferiti (RubricaPreferiti): Contiene un'istanza della classe RubricaPreferiti, che gestisce separatamente i contatti preferiti.

- **Metodi**

- **Costruttore**
  - Rubrica(): Inizializza l'attributo elenco ed elencoPreferiti come ObservableSet.

- **Metodi di Accesso**
  - ObservableSet<Contatto> getElenco(): Restituisce l'elenco di tutti i contatti presenti nella rubrica.
  - ObservableSet getElencoPreferiti(): Restituisce l'oggetto responsabile della gestione dei contatti preferiti.

- **Metodi di Gestione dei Contatti**
  - void aggiungiContatto(Contatto c): Aggiunge un contatto alla rubrica.
  - void removeContatto(Contatto c): Rimuove un contatto dalla rubrica.

- void resetRubrica(): Svuota completamente la rubrica, rimuovendo tutti i contatti.
- boolean isRubricaVuota(): Verifica se la rubrica è vuota e restituisce un valore booleano.
- void esportaRubrica() : Esporta i contatti della rubrica in un file o formato esterno per backup o condivisione.
- void importaRubrica(): Importa contatti da un file o formato esterno per aggiornare la rubrica.

- **Gestione dei Contatti Preferiti**
  - void aggiungiAPreferiti(Contatto c): Aggiunge un contatto alla lista dei preferiti.
  - void rimuoviDaPreferiti(Contatto c): Rimuove un contatto dalla lista dei preferiti.

RubricaPreferiti

La classe RubricaPreferiti si occupa di gestire i contatti contrassegnati come preferiti. Questi contatti vengono trattati separatamente rispetto agli altri per facilitare un accesso rapido.

- **Attributi**
  - elencoPreferiti (ObservableSet<Contatto>): Contiene l'insieme dei contatti preferiti.

- **Metodi**
  - **Costruttore**
    - RubricaPreferiti(): Inizializza l'attributo elencoPreferiti come un nuovo ObservableSet.

- **Metodi di Gestione dei Contatti Preferiti**
  - void addContattoPreferito(Contatto c): Aggiunge un contatto all'elenco dei preferiti.
  - void removeContattoPreferito(Contatto c): Rimuove un contatto dall'elenco dei preferiti.
  - void resetRubricaPreferiti(): Svuota completamente l'elenco dei contatti preferiti.

- **Metodi di Accesso**
  - ObservableSet<Contatto> getElencoPreferiti(): Restituisce l'elenco dei contatti preferiti come un Set.

MainViewController

La classe MainViewController funge da controller principale. Gestisce il layout principale composto da un pannello sinistro ed un pannello destro, caricando dinamicamente i contenuti della parte destra sulla base delle azioni dell'utente.

Attributi

- leftPane (StackPane): È il pannello sinistro dell'interfaccia utente.
- rightPane (StackPane): È il pannello destro dinamico dell'interfaccia utente.
- rubrica(Rubrica): Riferimento alla rubrica che gestisce i contatti, utilizzata per le operazioni principali sui dati.

Metodi

- **Metodi di Inizializzazione**
  - initialize(URL url, ResourceBundle rb):Carica il contenuto del leftPane da un file FXML chiamato leftView. Collega il controller della vista

sinistra all'istanza del controller principale, abilitando la comunicazione tra i due.

- **Metodi get**
  - getRubrica(): Metodo che restituisce un'istanza della classe Rubrica.

- **Metodi di Gestione degli Eventi**
  - loadView(String fxmlFileName):Carica dinamicamente le diverse view nel pannello rightPane.
  - loadView1(Contatto contatto):Carica la vista rightView1 all'interno di rightPane e passa il riferimento di un contatto. Invoca internamente loadView.
  - loadView2(Contatto contatto):Carica la vista rightView2 all'interno di rightPane e passa il riferimento di un contatto. Invoca internamente loadView .

LeftViewController

La classe LeftViewController è il controller per il pannello sinistro dell'applicazione. Questa classe gestisce la logica relativa agli elementi dell'interfaccia utente situati nel pannello sinistro, che include un menu, un campo di ricerca, un toggle per i preferiti, e una tabella dei contatti.

Attributi

- **Elementi FXML**
  - labelRubrica(Label):Label che indica quale rubrica si sta visualizzando(preferiti o non).
  - addButton (MenuItem): Pulsante per aggiungere un nuovo contatto.
  - importButton (MenuItem): Pulsante per importare una rubrica.
  - exportButton (MenuItem): Pulsante per esportare una rubrica.
  - resetButton(MenuItem): Pulsante per ripristinare una rubrica.

- searchField (TextField): Campo di testo per effettuare ricerche tra i contatti.
- prefToggle (ToggleButton): Pulsante a scorrimento per mostrare/nascondere i contatti preferiti.
- contattiTable (TableView<Contatto>): Tabella che visualizza i contatti.
- nomeColumn (TableColumn< Contatto,String>):Colonna della tabella per il nome
- cognomeColumn (TableColumn<Contatto, String>): Colonna della tabella per il cognome.
- listaContatti (ObservableList<Contatto>): Riferimento alla lista osservabile di tutti i contatti nella rubrica.
- listaContattiPreferiti (ObservableList<Contatto>): Riferimento alla lista osservabile dei contatti preferiti.

- **Dati e Riferimenti**
  - mainViewController (MainViewController): Riferimento al controller principale, utilizzato per coordinare le interazioni tra rubrica e interfaccia utente.
  - rubrica (Rubrica): Riferimento alla rubrica principale per gestire i contatti.

Metodi

- **Metodi di Inizializzazione**
  - initialize(URL url, ResourceBundle rb): Metodo chiamato automaticamente durante l'inizializzazione.
  - setMainViewController(MainViewController mainViewController): Imposta il riferimento al controller principale per abilitare la comunicazione e la gestione centralizzata della logica applicativa.

- **Metodi di Gestione degli Eventi**
  - aggiungiContatto(ActionEvent event):Viene invocato quando l'utente preme il pulsante nella barra di menù “Aggiungi contatto”. Chiama il

metodo loadView2() del controller principale permettendo all'utente di inserire dati per la creazione di un nuovo contatto.
- importaRubrica(ActionEvent event):Viene invocato quando l'utente preme il pulsante nella barra di menù "Importa". Importa in rubrica i contatti presenti in un file scelto dall'utente. Lancia un'eccezione se i contatti sono scritti in un formato errato.
- esportaRubrica(ActionEvent event):Viene invocato quando l'utente preme il pulsante nella barra di menù "Esporta". Esporta i contatti della rubrica in un file scelto dall'utente.
- ricercaContatto(KeyEvent event):implementa la ricerca dei contatti basata sull'input dell'utente nel campo di ricerca.
- mostraPreferiti(ActionEvent event):Viene invocata quando l'utente preme il ToggleButton dei preferiti. Attiva o disattiva la visualizzazione dei contatti preferiti.
- resetRubrica(ActionEvent event): viene invocata quando l'utente preme il pulsante nella barra di menù "Resetta". Ripristina la rubrica eliminando tutti i contatti presenti.

RightView1Controller

La classe RightView1Controller è il controller per il pannello destro dell'applicazione. Questa vista gestisce la visualizzazione e l'interazione con i dettagli di un contatto selezionato, includendo funzioni per modificarlo, eliminarlo e contrassegnarlo come preferito.

Attributi

- **Elementi FXML**
  - modifyButton (MenuItem): Pulsante per modificare il contatto attualmente visualizzato.
  - deleteButton (MenuItem): Pulsante per eliminare il contatto attualmente visualizzato.
  - nomeLabel(Label) Etichetta che mostra il nome del contatto.
  - cognomeLabel (Label): Etichetta che mostra il cognome del contatto.

- numero1Label, numero2Label, numero3Label (Label): Etichette per i numeri di telefono del contatto.
- email1Label, email2Label, email3Label (Label): Etichette per gli indirizzi email del contatto.
- preferitoCheck (CheckBox): Checkbox per indicare se il contatto è contrassegnato come preferito.

- **Dati e Riferimenti**
  - rubrica (Rubrica): Riferimento alla rubrica principale per gestire i contatti.
  - Contatto contatto: Riferimento alla classe contatto.
  - mainViewController (MainViewController): Riferimento al controller principale, utilizzato per coordinare le interazioni tra rubrica e interfaccia utente.

Metodi

- **Metodi di Inizializzazione**
  - initialize(URL url, ResourceBundle rb):Metodo chiamato automaticamente durante l'inizializzazione della vista.
  - setMainViewController(MainViewController mainViewController): Imposta il riferimento al controller principale per consentire la gestione centralizzata delle azioni e delle transizioni di vista.

- **Metodi di Gestione degli Eventi**
  - setContatto(Contatto contatto):Imposta un riferimento al contatto specificato, aggiornando le informazioni nel contesto corrente.
  - modificaContatto(ActionEvent event):Viene invocato quando l'utente preme il pulsante nella barra di menù “Modifica”. Chiama il metodo loadView2() del controller principale permettendo all’utente di inserire dati per la modifica del contatto.
  - eliminaContatto(ActionEvent event):Viene invocato quando l'utente preme il pulsante nella barra di menù “Elimina”. Rimuove dalla rubrica il contatto selezionato.

RightView2Controller

La classe RightView2Controller è il controller per il pannello destro dell'applicazione. Questa vista gestisce l'inserimento o la modifica di un contatto, fornendo campi di input per i dettagli del contatto e opzioni per confermare o annullare le modifiche.

Attributi

- **Elementi FXML**
  - stopButton (MenuItem): Pulsante per annullare l'operazione corrente.
  - confirmButton (MenuItem): Pulsante per confermare e salvare i dettagli del contatto.
  - nomeField, cognomeField(TextField): Campi per l'inserimento del nome e del cognome del contatto.
  - numero1Field, numero2Field, numero3Field (TextField): Campi per l'inserimento di uno, due o tre numeri di telefono.
  - email1Field, email2Field, email3Field: Campi per l'inserimento di uno, due o tre indirizzi email.
  - preferitoCheck (CheckBox): Casella di controllo per contrassegnare il contatto come preferito.

- **Dati e Riferimenti**
  - mainViewController (MainViewController): Riferimento al controller principale per abilitare la comunicazione e la gestione delle transizioni tra le diverse viste.
  - Contatto contatto: Riferimento alla classe contatto
  - rubrica (Rubrica): Riferimento alla rubrica principale per gestire i contatti.

Metodi

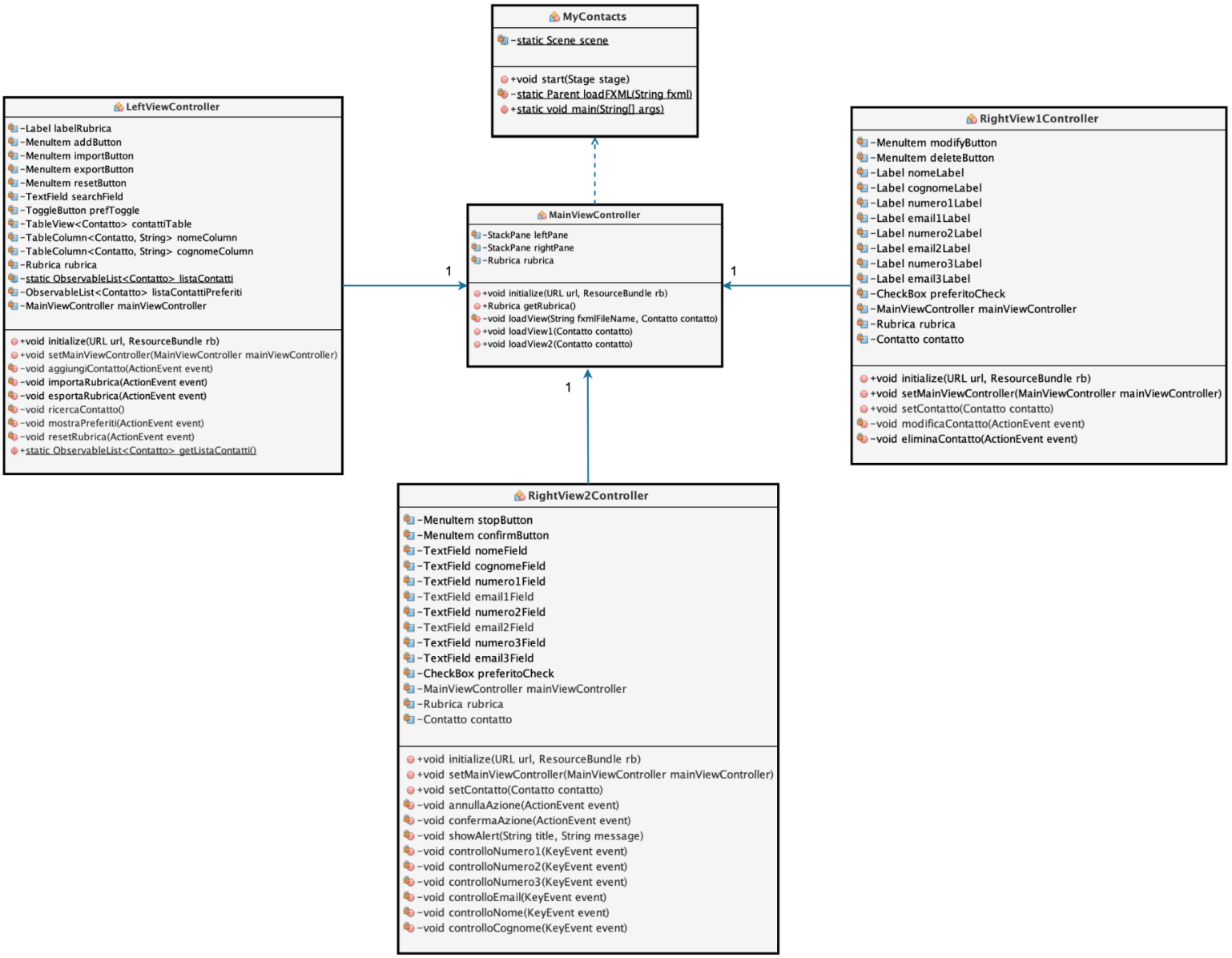
- **Metodi di Inizializzazione**
  - initialize(URL url, ResourceBundle rb): Metodo chiamato automaticamente durante l'inizializzazione della vista.
  - setMainViewController(MainViewController mainViewController): Imposta il riferimento al controller principale per consentire la gestione centralizzata delle azioni e delle transizioni di vista.

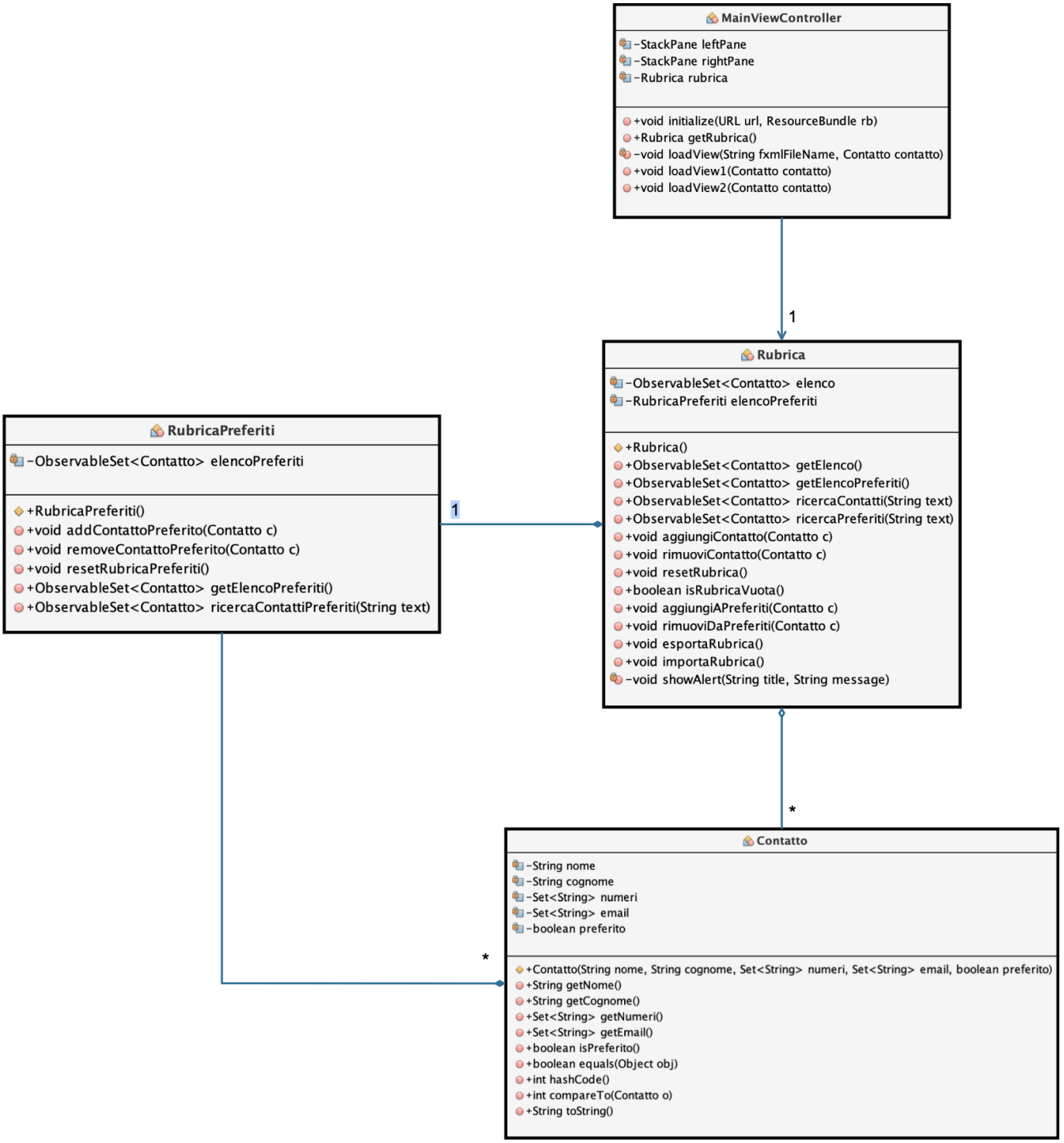
- **Metodi di gestione dei contatti**
  - setContatto(Contatto contatto): Imposta un riferimento al contatto specificato, aggiornando le informazioni nel contesto corrente.
  - void controlloNumero(KeyEvent event): Controlla in tempo reale l'input del numero di telefono per verificarne la validità.

- **Metodi di Gestione degli Eventi**
  - annullaAzione(ActionEvent event): Invocato quando l'utente preme il pulsante “Annulla”. Chiama il metodo loadView1() del controller principale per tornare alla visualizzazione precedente senza salvare le modifiche o l’aggiunta di un contatto.
  - confermaAzione(ActionEvent event): Invocato quando l'utente preme il pulsante “Conferma”. Salva le modifiche al contatto corrente o l’aggiunta di un contatto e ritorna alla visualizzazione principale chiamando il metodo loadView1() del controller principale.
  - switchPreferito(ActionEvent event): Permette di contrassegnare o rimuovere il contatto come preferito, in base allo stato della checkbox.
  - void showAlert(String title,String message): è un'utility per mostrare un messaggio di errore tramite

Diagramma delle classi

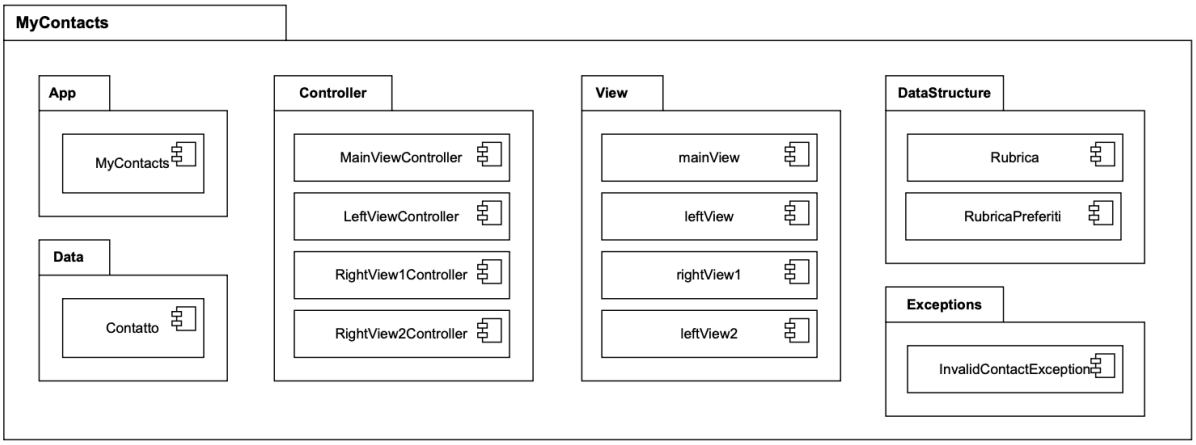
In questo paragrafo è presente il Class Diagram completo del progetto MyContacts. Grazie a questo diagramma, possiamo risalire a tutte le interazioni e alle relazioni tra le classi che abbiamo deciso di implementare, così come ai metodi e agli attributi necessari per il loro corretto funzionamento. La struttura del diagramma ci consente di comprendere come ogni classe interagisce con le altre, evidenziando la separazione delle responsabilità.




MainViewController
-StackPane leftPane
-StackPane rightPane
-Rubrica rubrica
+void initialize(URL url, ResourceBundle rb)
+Rubrica getRubrica()
-void loadView(String fxmlFileName, Contatto contatto)
+void loadView1(Contatto contatto)
+void loadView2(Contatto contatto)
1
Rubrica
-ObservableSet<Contatto> elenco
-RubricaPreferiti elencoPreferiti
+Rubrica()
+ObservableSet<Contatto> getElenco()
+ObservableSet<Contatto> getElencoPreferiti()
+ObservableSet<Contatto> ricercaContatti(String text)
+ObservableSet<Contatto> ricercaPreferiti(String text)
+void aggiungiContatto(Contatto c)
+void rimuoviContatto(Contatto c)
+void resetRubrica()
+boolean isRubricaVuota()
+void aggiungiAPreferiti(Contatto c)
+void rimuoviDaPreferiti(Contatto c)
+void esportaRubrica()
+void importaRubrica()
-void showAlert(String title, String message)
RubricaPreferiti
-ObservableSet<Contatto> elencoPreferiti
+RubricaPreferiti()
+void addContattoPreferito(Contatto c)
+void removeContattoPreferito(Contatto c)
+void resetRubricaPreferiti()
+ObservableSet<Contatto> getElencoPreferiti()
+ObservableSet<Contatto> ricercaContattiPreferiti(String text)
1
*
*
Contatto
-String nome
-String cognome
-Set<String> numeri
-Set<String> email
-boolean preferito
+Contatto(String nome, String cognome, Set<String> numeri, Set<String> email, boolean preferito)
+String getNome()
+String getCognome()
+Set<String> getNumeri()
+Set<String> getEmail()
+boolean isPreferito()
+boolean equals(Object obj)
+int hashCode()
+int compareTo(Contatto o)
+String toString()

Package Diagram

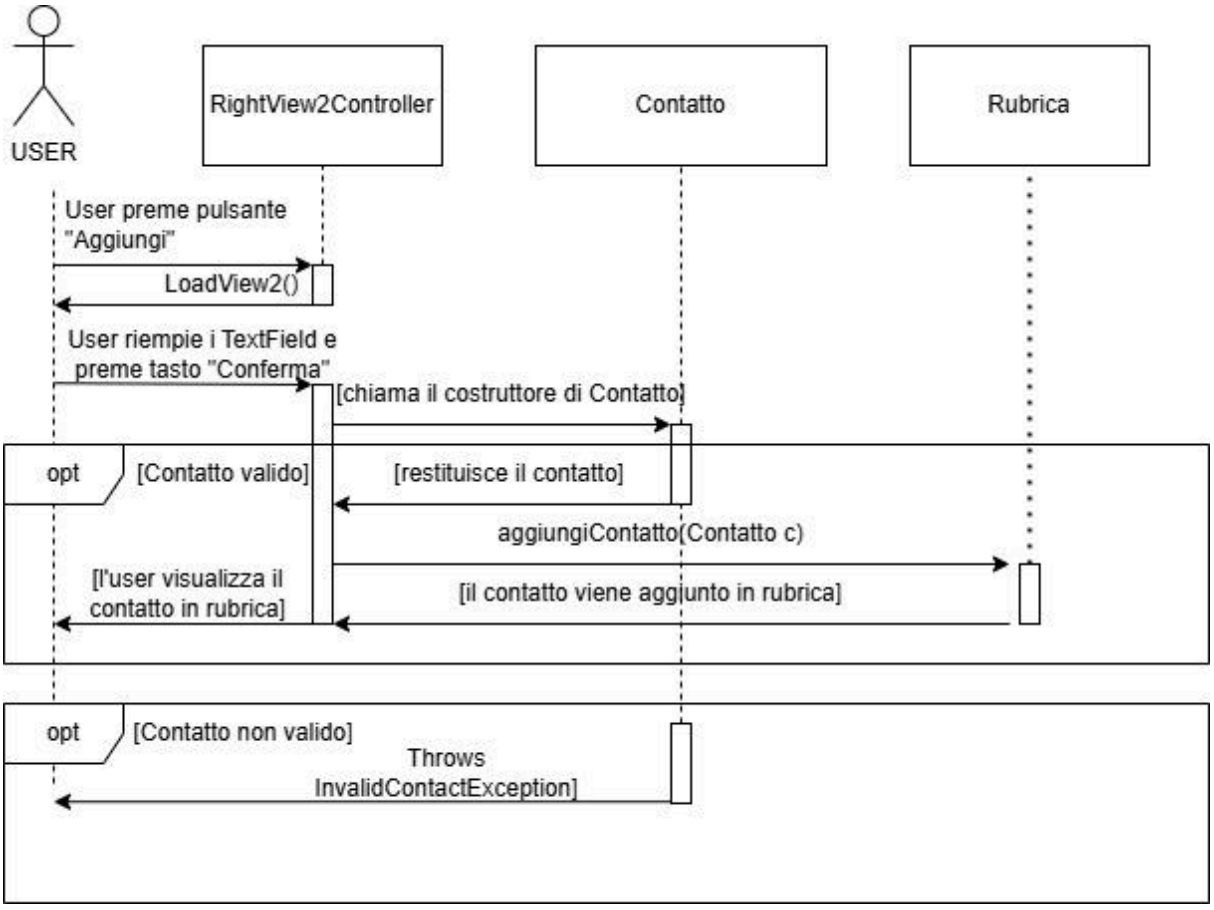
Il seguente Package Diagram descrive come abbiamo diviso a livello logico e strutturale i Package del nostro progetto, questo anche per avere un ulteriore organizzazione all'interno di esso. Il diagramma rappresenta l'architettura del sistema organizzata in pacchetti secondo una chiara separazione tra i dati (data, datastructure), la logica applicativa (controller) e la visualizzazione (view).



Sequence Diagram

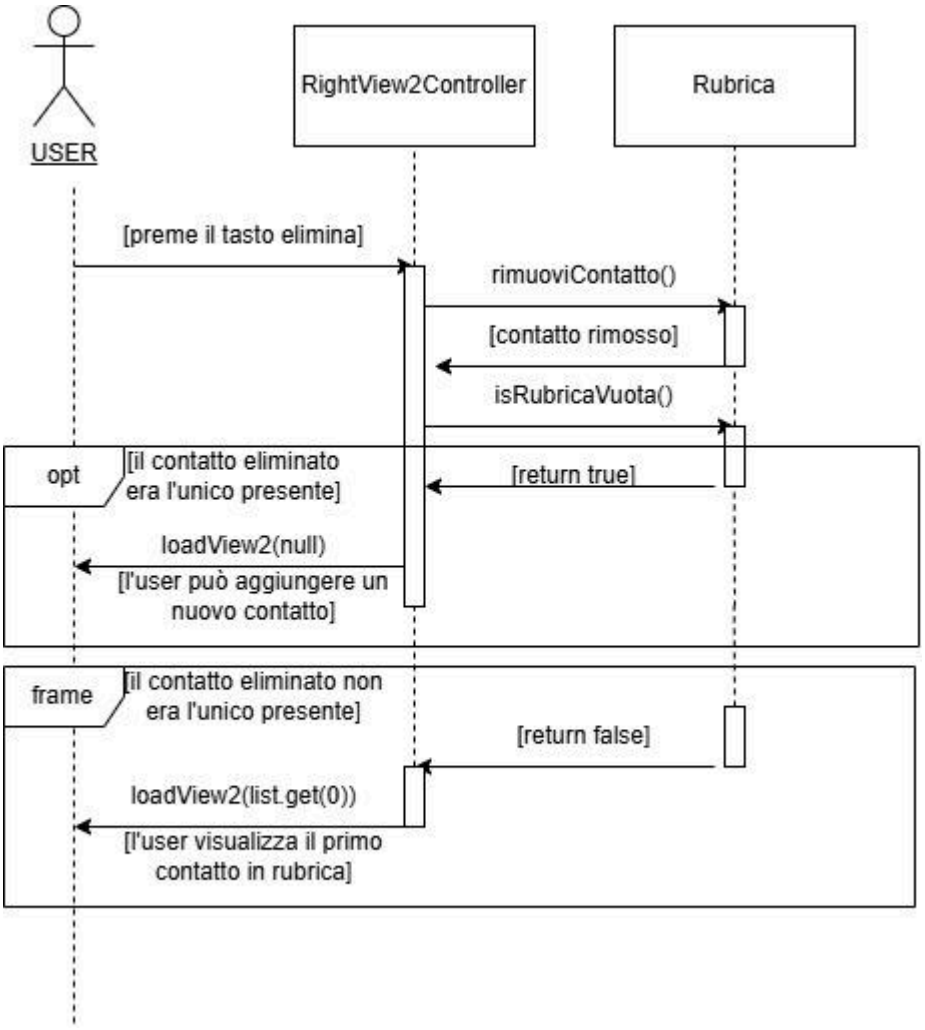
Aggiunta contatto

Il seguente diagramma mostra l'interazione tra le classi durante il processo di aggiunta di un contatto alla rubrica.



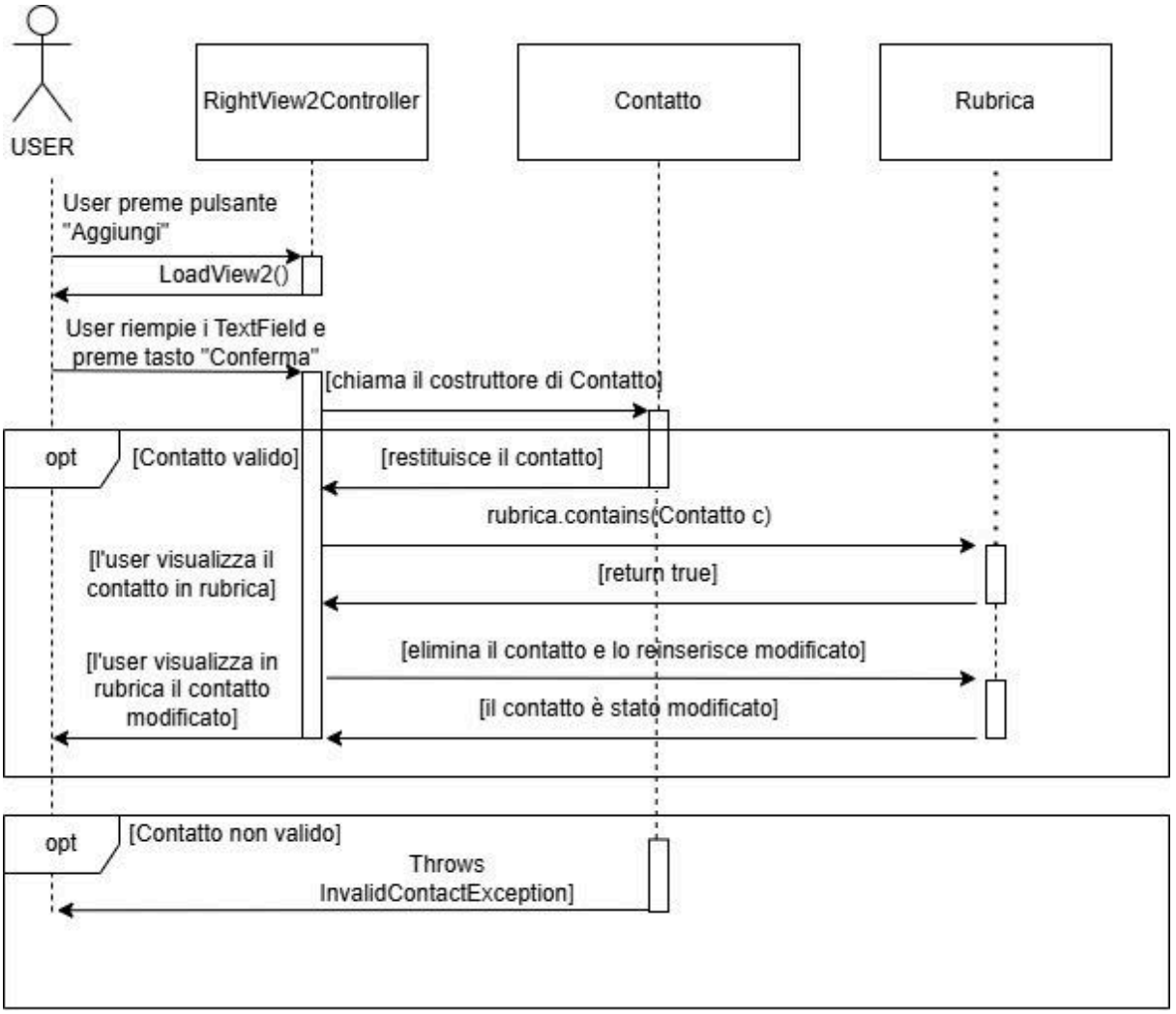
Eliminazione Contatto

Il seguente diagramma mostra l'interazione tra le classi durante il processo di eliminazione di un contatto alla rubrica.



Modifica Contatto

Il seguente diagramma mostra l'interazione tra le classi durante il processo di modifica di un contatto alla rubrica.



Matrice di Tracciabilità

Requisito	Design	Codice	Test	Requisiti correlati
R-1	LeftViewController, RightView1Controller			R-2, R-5, R-6
R-2	LeftViewController, RightView2Controller			R-1, R-3, R-6
R-3	RightView1Controller, RightView2Controller			R-2, R-4
R-4	RightView2Controller			R2, R-3
R-5	RightView1Controller, RightView2Controller			R-1, R-3, R-6
R-6	LeftViewController			R-1, R-6
R-7	LeftViewController			R-1, R-3

Partecipanti

NOME	COGNOME	MATRICOLA
Gabriele	Lupo	0612707641
Michele	Naddeo	0612708622
Andrea	Sangineto	0612707424
Gianmarco	Vitolo	0612708376