Implementazione

Indice

[1 Revisione dei class diagram 2](#_Toc211277385)

[2 Manuale utente 6](#_Toc211277386)

[3 Tabella di Tracciabilità 13](#_Toc211277387)

# 1 Revisione dei class diagram

Durante l'implementazione, ho apportato le seguenti modifiche ai class diagram per garantirne la coerenza e la correttezza rispetto al codice finale.

|  |  |
| --- | --- |
| **AppRunner** (*main class/ bootstrap class*): è la classe principale che si occupa di avviare l’applicazione. | |
| METODI | DESCRIZIONE |
| main(): | Svolge due compiti importanti prima di avviare la finestra:   * Imposta l’aspetto grafico per adeguarlo al sistema operativo * Imposta il font “Verdana” per l’intera applicazione |
| Initialize(): void | Entrambi i metodi sono stati **cancellati**. Durante la fase di progettazione è buona pratica prevedere metodi separati per l’inizializzazione e la chiusura. Tuttavia, durante l'implementazione, la logica di avvio si è rivelata molto semplice (creare 2-3 oggetti). |
| shutdownHook(): void |

* 1. Package runner
  2. Package classes

|  |  |
| --- | --- |
| **Contatto** (*entity class/ core class*): rappresenta l’entità principale della rubrica. E’una classe del dominio (*MODEL*), che incapsula i dati di un singolo contatto | |
| ATTRIBUTI | DESCRIZIONE |
| nome: String | Nome del contatto. Può essere vuoto se è presente il cognome |
| cognome: String | Cognome del contatto. Può essere vuoto se è presente il nome. |
| telefoni: List<String> [0…3] | Lista di massimo 3 numeri telefonici. |
| email: List<String> [0…3] | Lista di massimo 3 indirizzi email. |
| METODI | DESCRIZIONE |
| Contatto(nome: String, cognome: String) | Costruttore che inizializza un nuovo contatto. |
| getNome(): String | Restituisce il nome. |
| getCognome(): String | Restituisce il cognome. |
| getTelefoni(): List<String> | Restituisce i numeri telefonici del contatto. |
| getEmail(): List<String> | Restituisce le email del contatto. |
| setNome(nome: String): void | Modifica il nome. |
| setCognome(cognome: String): void | Modifica il cognome. |
| aggiungiTelefono(numero: String): void | Aggiunge un numero alla lista (rispettando il limite di 3). |
| rimuoviTelefono(numero: String): void | Rimuove un numero della lista. |
| aggiungiEmail(indirizzo: String): void | Aggiunge un’email alla lista (rispettando il limite di 3). |
| rimuoviEmail(indirizzo: String): void | Rimuove un un’email della lista. |
| valida(): boolean | Inizialmente assegnavo troppe responsabilità a questo metodo. Quindi **valida()** controlla solo la coerenza interna(nome/cognome). La validazione di regole esterne (come il numero massimo di telefoni) è stata delegata a una classe di servizio apposita, **ValidatoreDati**. |
| equals() | Metodi implementati per confrontare due oggetti Contatto, per capire se sono “lo stesso.” |
| hashCode() |

|  |  |
| --- | --- |
| **Rubrica** (*aggregate entity class*): gestisce l’insieme dei contatti. E’ la classe contenitore che fornisce operazioni di ricerca e gestione. | |
| ATTRIBUTI | DESCRIZIONE |
| contatti: List<Contatto> | Lista di tutti i contatti della rubrica. |
| METODI | DESCRIZIONE |
| aggiungiContatto(contatto: Contatto): void | Aggiunge un contatto. |
| modificaContatto(id: int, nuovoContatto: Contatto): void | Sostituisce un contatto esistente con uno nuovo. |
| eliminaContatto(id: int): void | Rimuove un contatto tramite un identificativo. |
| getContatto(id: int): Contatto | Restituisce il contatto corrispondente all’id. |
| getContattiOrdinati(): List<Contatto> | Restituisce i contatti ordinati (alfabeticamente per nome e cognome). |
| cercaContatti(sottostringa: String): List<Contatto> | Ricerca contatti che contengono la sottostringa. |
| isVuota(): boolean | Restituisce “*true*” se la rubrica è vuota. |
| aggiungiContatti(List<Contatto>) | ll **RubricaController** ora legge la **Rubrica** dal file e poi usa questo metodo per "fondere" i contatti caricati con quelli già presenti in memoria. |
| getIndiceContatto(Contatto) | Permette di passare dall'oggetto **Contatto** al suo indirizzo fisico (l'indice) nel modello dati, rendendo le operazioni di modifica ed eliminazione indipendenti e immuni da qualsiasi ordinamento o filtro applicato dall'utente nell'interfaccia grafica. |

|  |  |
| --- | --- |
| **RubricaController** (*controller class*): gestisce la logica dell’applicazione, funge da intermediario tra interfaccia utente (*scinterface*) e modello (*Rubrica*) | |
| ATTRIBUTI | DESCRIZIONE |
| rubrica: Rubrica | Riferimento alla rubrica gestita. |
| validatore: ValidatoreDati | Usato per controllare la correttezza dei dati. |
| gestoreFile: GestoreFile | Usato per leggere/scrivere la rubrica da/su file. |
| METODI | DESCRIZIONE |
| creaContatto(nome, cognome, telefoni, email): void | Crea un nuovo contatto valido e lo aggiunge alla rubrica. |
| rimuoviContatto(indice: int): void | Elimina un contatto dalla rubrica. |
| salvaRubrica(filePath: String): void | Salva la rubrica su file. |
| caricaRubrica(filePath: String): void | Carica la rubrica da file. |
| getContatti(ordinati: boolean): List<Contatto> | Restituisce i contatti ordinati o meno. |
| ricercaContatti(sottostringa: String): List<Contatto> | Ricerca contatti per sottostringa. |
| getIndiceContatto(Contatto) | Permette di passare dall'oggetto **Contatto** al suo indirizzo fisico (l'indice) nel modello dati, rendendo le operazioni di modifica ed eliminazione indipendenti e immuni da qualsiasi ordinamento o filtro applicato dall'utente nell'interfaccia grafica. |

* 1. Package scienterface

|  |  |
| --- | --- |
| **MainView** (*main view class*): rappresenta la componente *View* dell’architettura MVC. Il suo ruolo è unicamente quello di presentare i dati e raccogliere l’input, delegando la logica al *RubricaController*. | |
| ATTRIBUTI | DESCRIZIONE |
| controller: RubricaController | Gestisce la logica applicativa. |
| panelStrumenti: PanelStrumenti | Pannello inferiore con i bottoni di gestione. |
| panelLista: PanelListaContatti | Pannelòp centrale con la lista di contatti. |
| METODI | DESCRIZIONE |
| MainView(controller:RubricaController) | Costruttore che inizializza la view. |
| mostraMessaggio(messaggio: String): void | Mostra i messaggi di sistema o errore. |
| modificaContatto(Contatto c) | Sono nuovi metodi che vengono chiamati direttamente dai pulsanti sulle singole schede. |
| eliminaContatto(Contatto c) |
| visualizzaRubricaOrdinata() |

|  |  |
| --- | --- |
| **PanelListaContatti** (*UI subview*): gestisce la visualizzazione della lista, la ricerca, la selezione e le azioni sul contatto selezionato (Modifica, Elimina). | |
| METODI | DESCRIZIONE |
| aggiornaLista(contatti:List): void | Aggiorna la lista visualizzata. |
| GestiscRicerca(): void | **Metodo cancellato**, perché la barra di ricerca è stata spostata fisicamente e logicamente nella MainView. |
| GestisciSeelezionaContatto(): void | **Metodo cancellato,** perché non si selezionano contatti, ma l’interazione avviene direttamente sui pulsanti di ogni scheda. |
| GestiscieliminaClick(): void | **Metodo cancellato,** perché il pulsante generico “Elimina selezionato” non esiste più, ma l’interazione avviene direttamente sui pulsanti di ogni scheda. |
| createContactCardPanel(Contatto contatto) | Un metodo privato che riceve un oggetto **Contatto** e costruisce da zero l'intero JPanel (la "scheda") che lo rappresenta con il Nome, Cognome, Num. di telefoni, indirizzi email e il proprio pulsante di “Modifica” e “Elimina” |
| createWrappingTextArea(String text | Piccolo metodo di supporto per mantenere il testo in posizione specifiche. |

|  |  |
| --- | --- |
| **PanelStrumenti** (*UI subview*): è un pannello che gestisce la sezione dell’interfaccia per le azioni principali: File I/O e Creazione/Modifica. | |
| DA METODO A ISTRUZIONE | DESCRIZIONE |
| salvaBtn.addActionListener(e -> mainView.salvaRubrica()); | I metodi precedenti (GestisciSalvaClick(), GestisciCaricaClick(), GestisciCreaClick()) sono stati sostituiti da un’istrizione più diretta che elimina codice superfluo. |
| caricaBtn.addActionListener(e -> mainView.caricaRubrica()); |
| aggiungiBtn.addActionListener(e -> mainView.creaNuovoContatto()); |
| METODO | DESCRIZIONE |
| GestisciModificaClick(): void | **Metodo cancellato,** perché il pulsante “Modifica” non si trova più su questo pannello. |

# Manuale utente

Questo manuale guiderà l’utente attraverso le funzionalità principali e l’operatività del sistema di gestione rubrica.

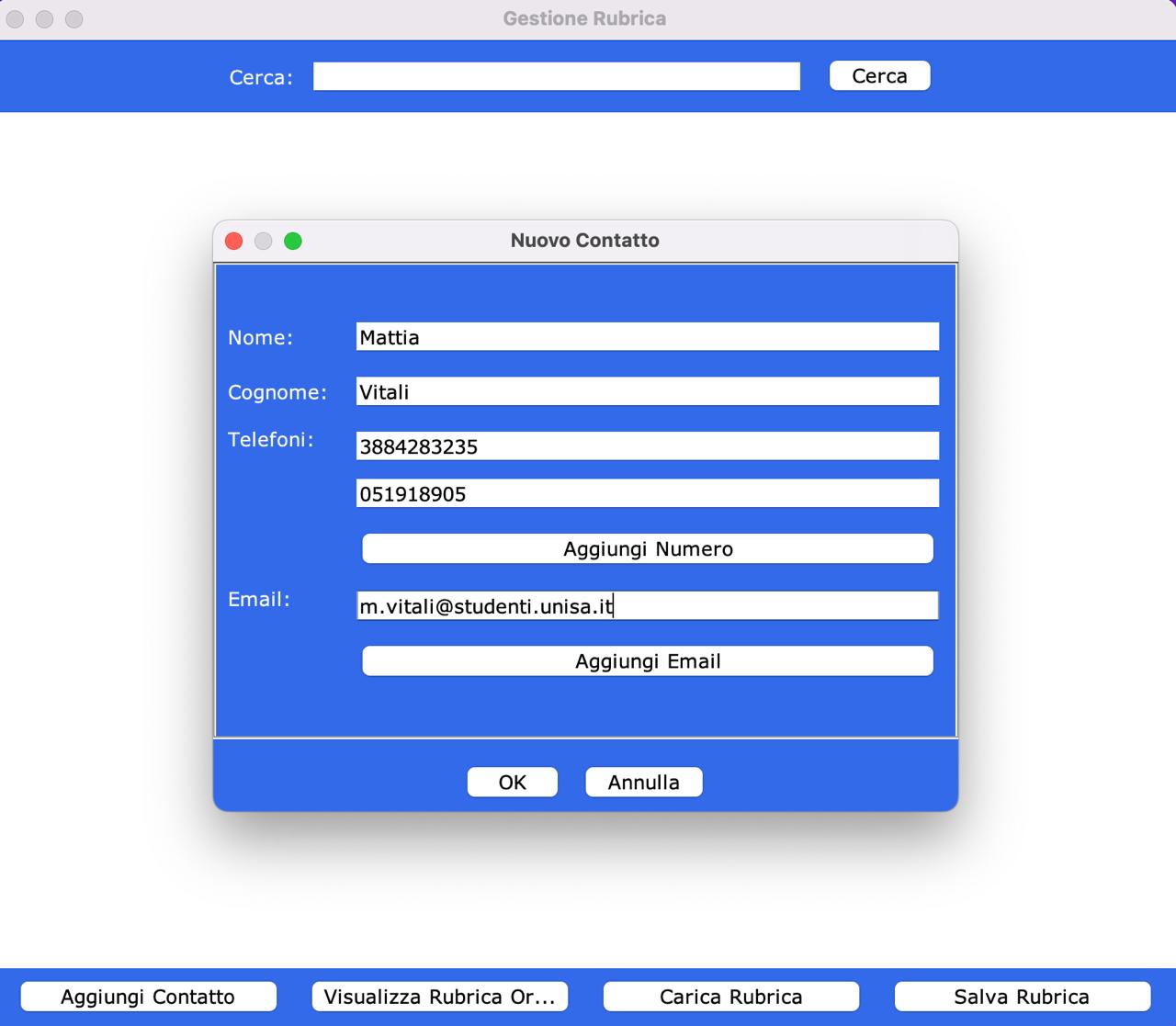
La rubrica è progetta per poter gestire i propri contatti personali, come meglio si crede.

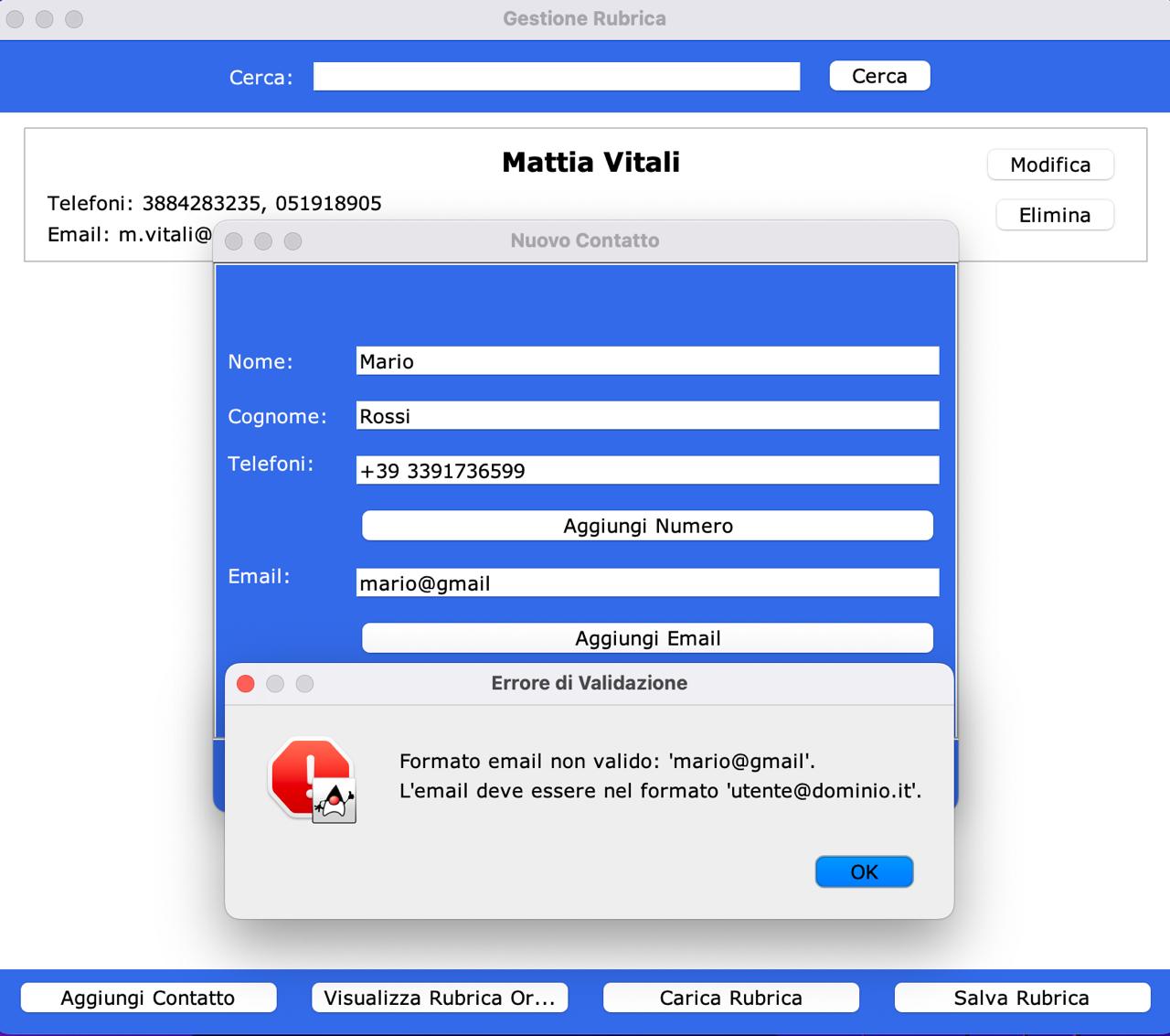
L’utente può:

* **Creare un nuovo contatto**: premendo il pulsante relativo apparirà una finestra di dialogo in cui potrà inserire tutti i dati relativi al contatto che si desidera creare, potendo inserire un nome e/o un cognome, fino a 3 numeri telefonici e indirizzi email.

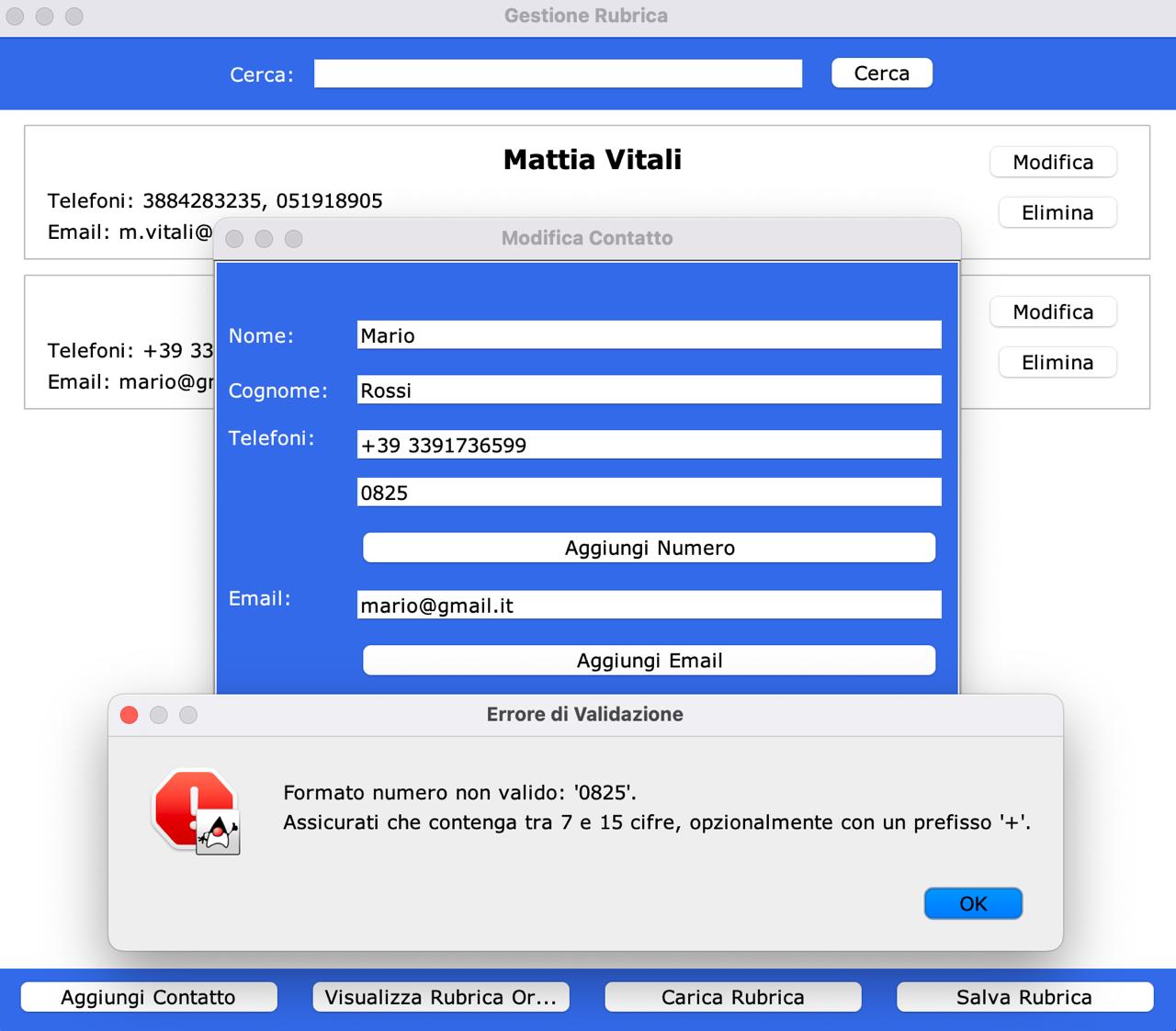


*Figura 1*

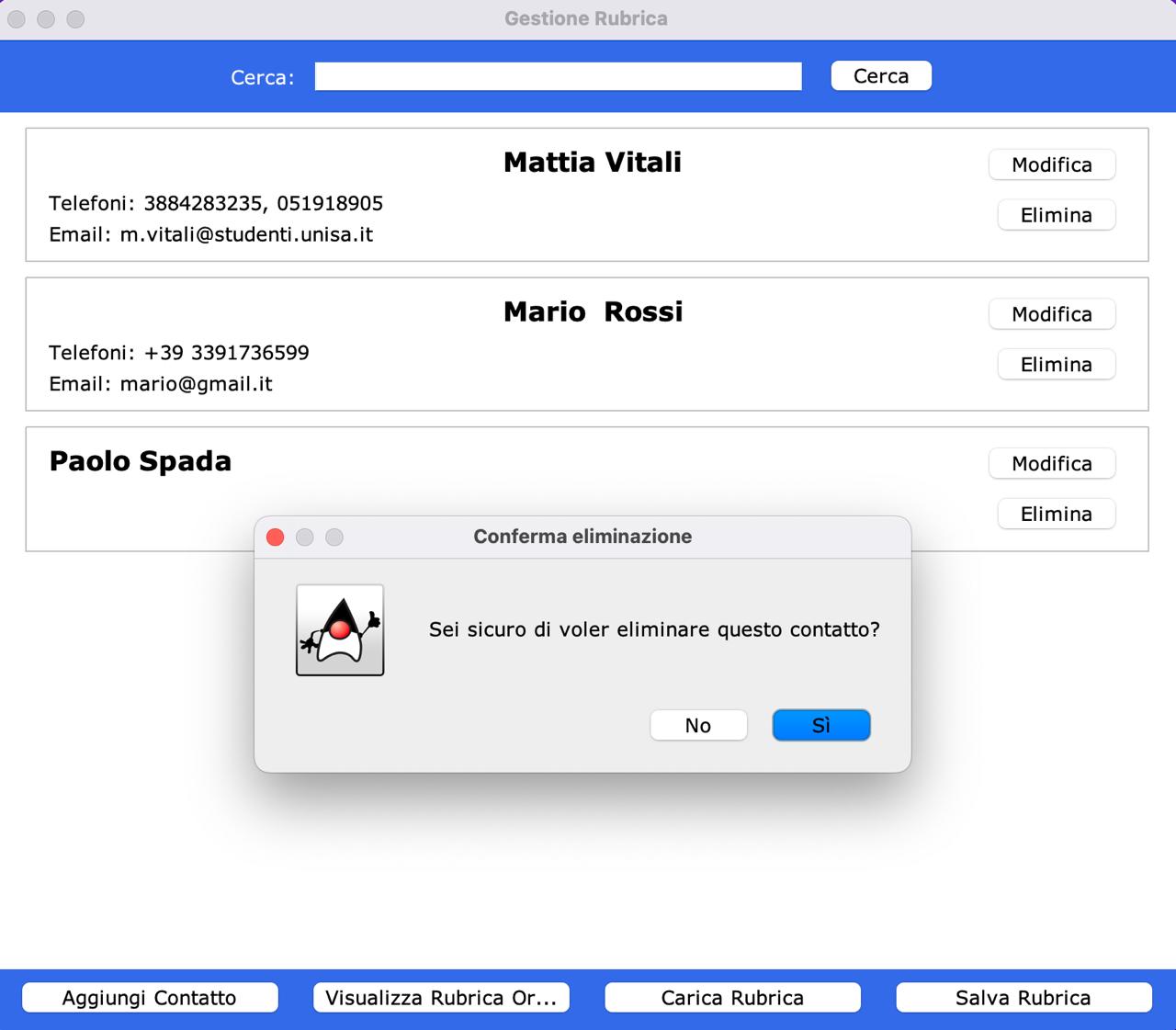
  
 *Figura 2*

  
 *Figura 3*

* **Modificare un contatto**: premendo il pulsante relativo apparirà una finestra di dialogo in cui potrà modificare i dati relativi ad un contatto precedentemente salvato.

  
 *Figura 4*

* **Eliminare un contatto**: premendo il pulsante relativo apparirà una finestra PopUp che chiederà conferma all’utente di eliminare un contatto precedentemente salvato.

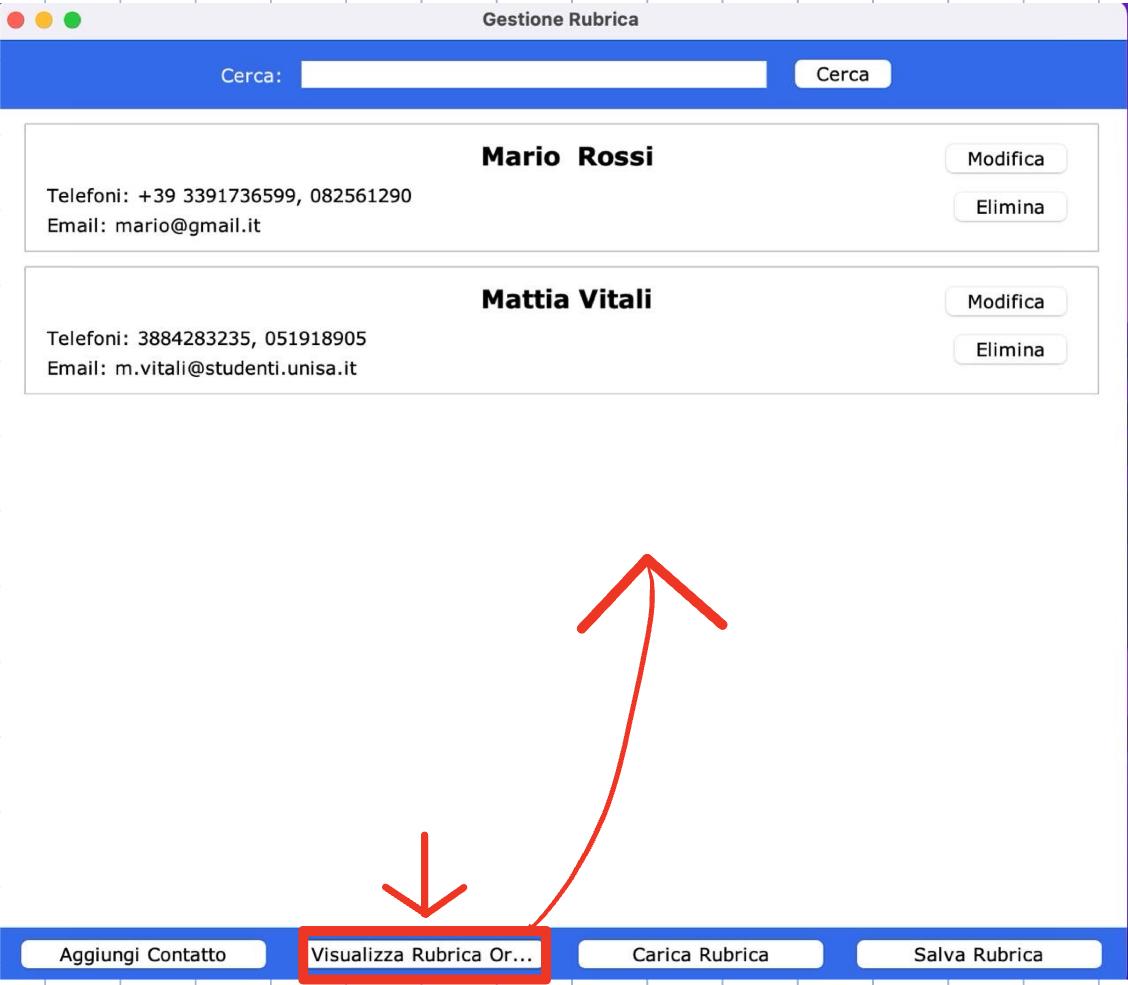
  
 *Figura 5*

* **Ricercare un contatto**: digitando una stringa nella barra di ricerca e, successivamente, premendo il pulsante “Cerca”, si potrà visualizzare tutti i contatti relativi alla ricerca effettuata.

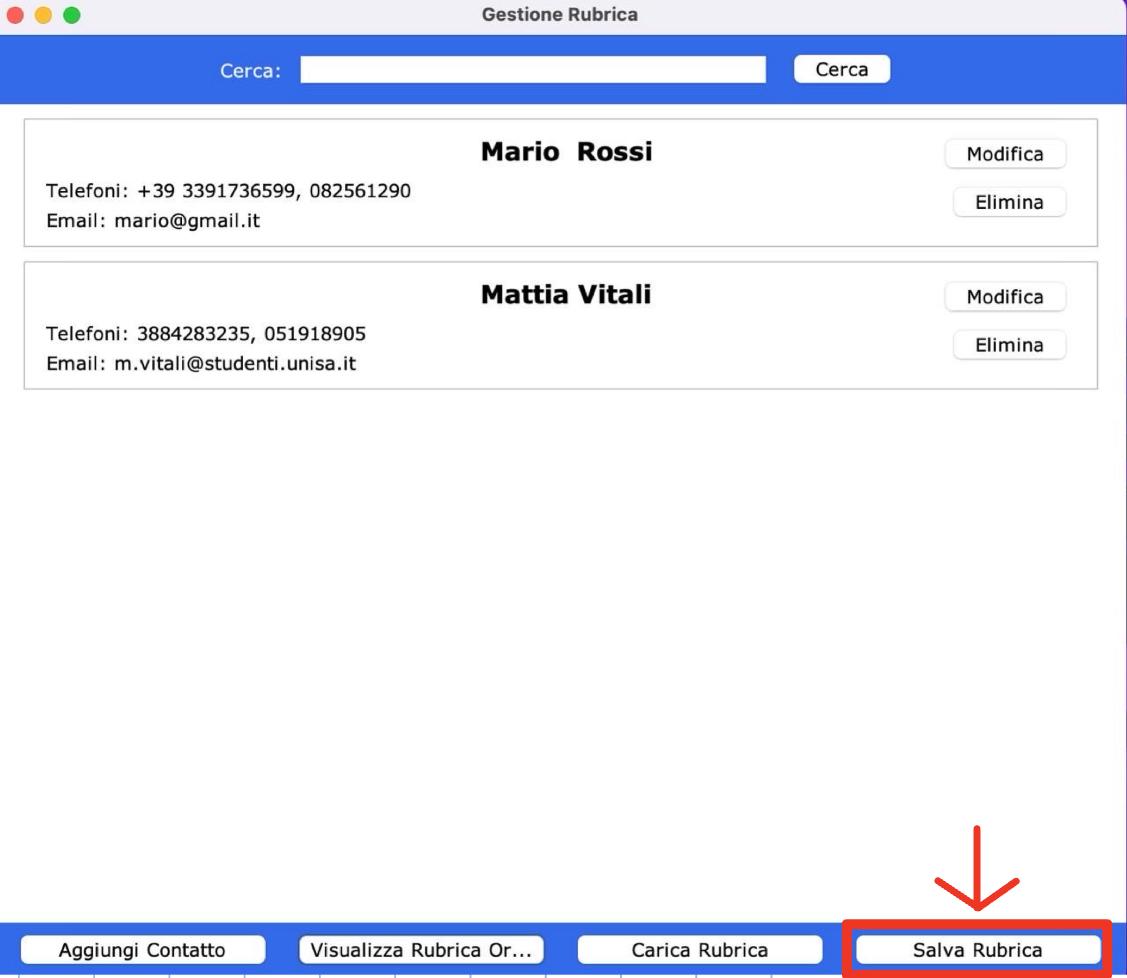
  
 *Figura 6*

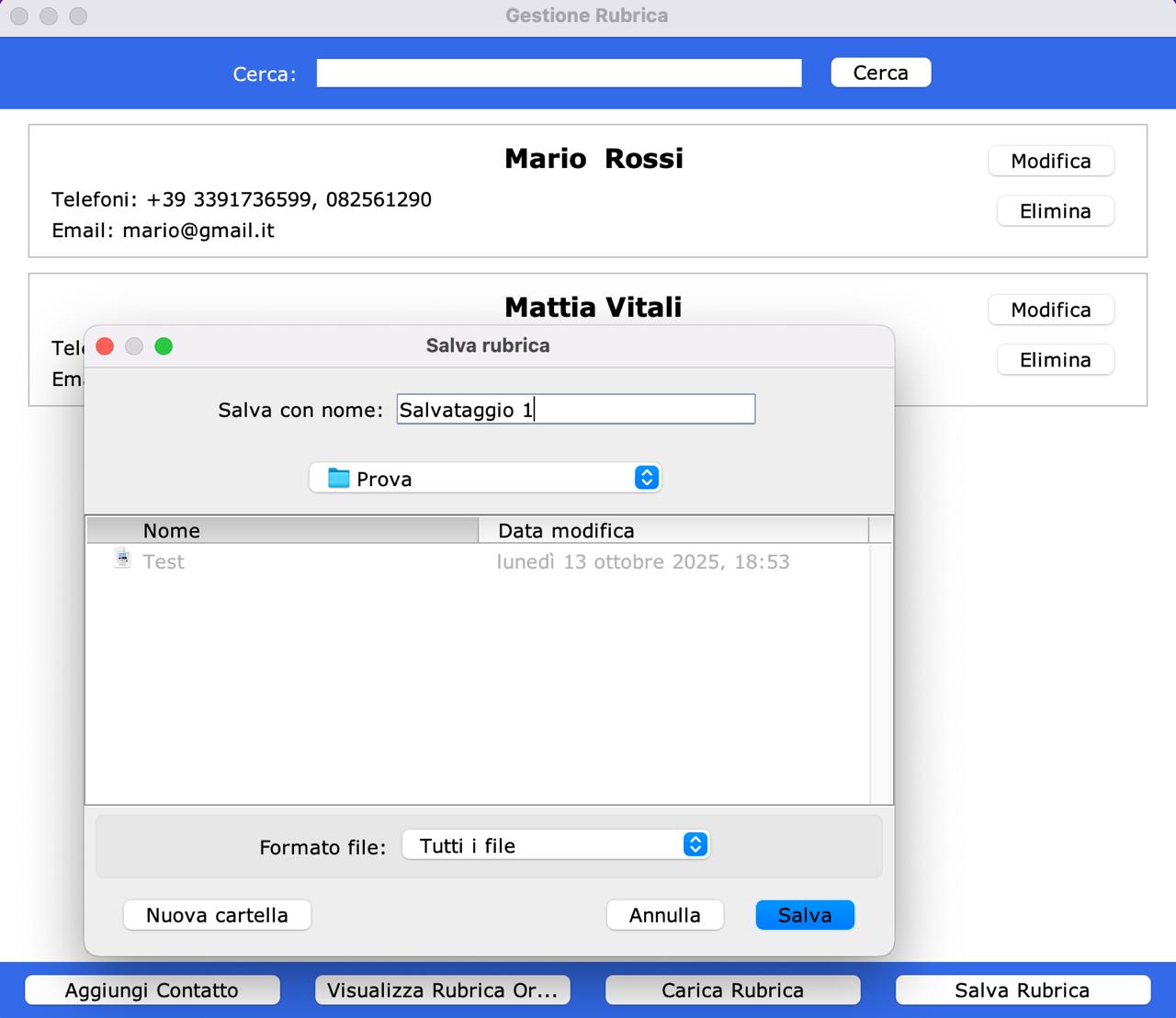
  
 *Figura 7*

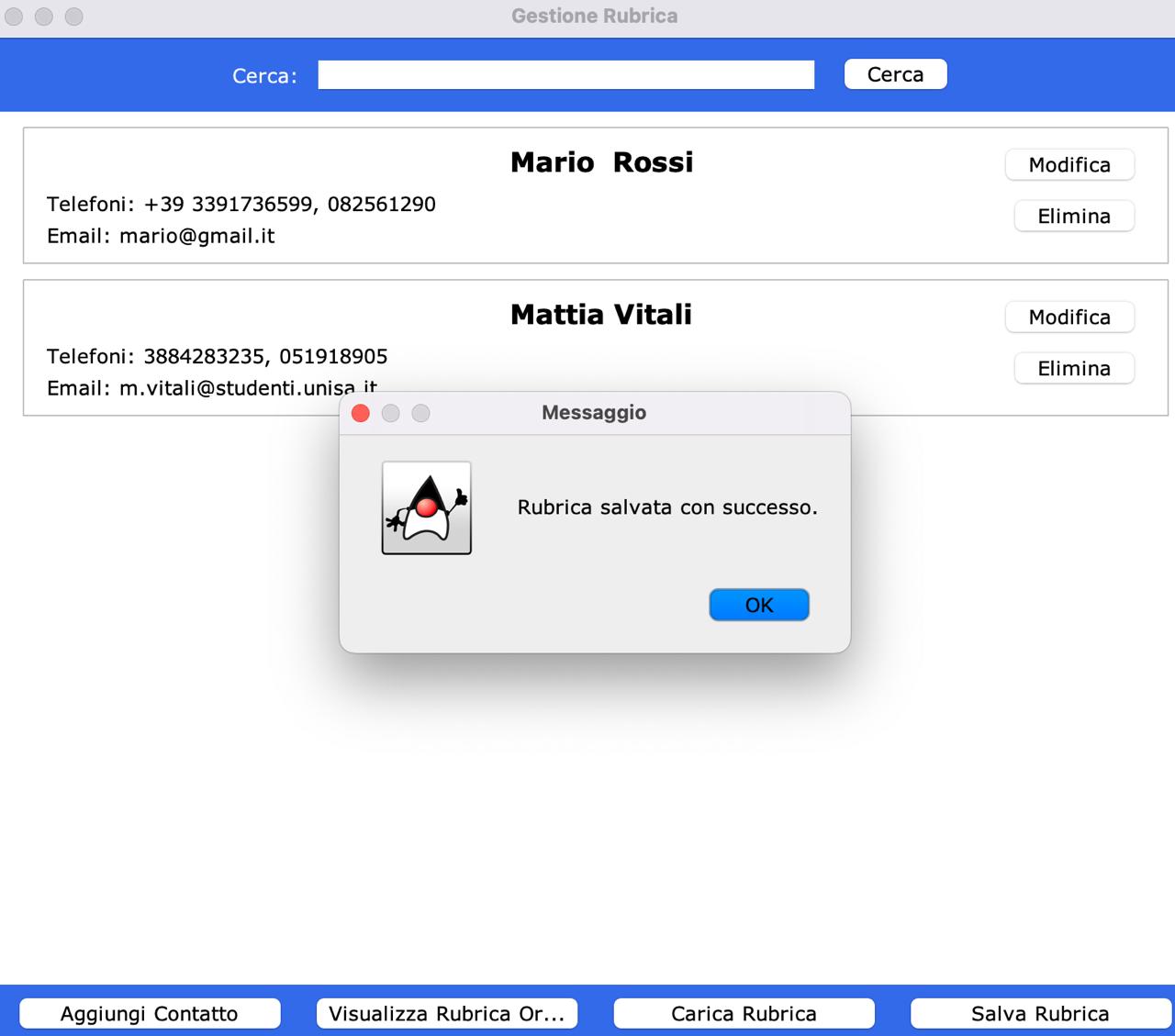
* **Visualizzare rubrica ordinata**: premendo il pulsante relativo, il sistema ordinerà i contatti in ordine alfabetico, dando la priorità al cognome.

  
 *Figura 8*

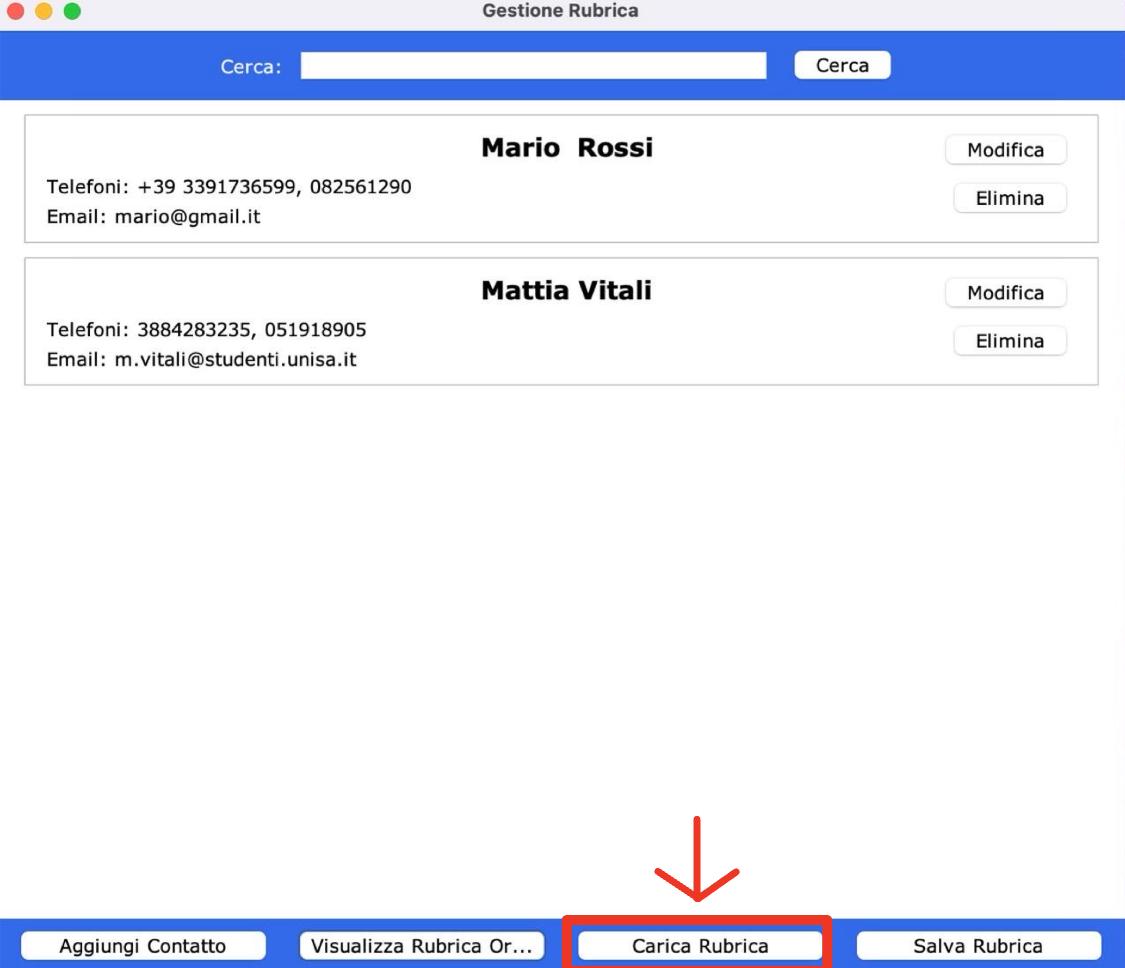
* **Salvare una rubrica**: premendo il pulsante relativo apparirà una finestra dove sarà possibile selezionare il percorso per il salvataggio della rubrica.

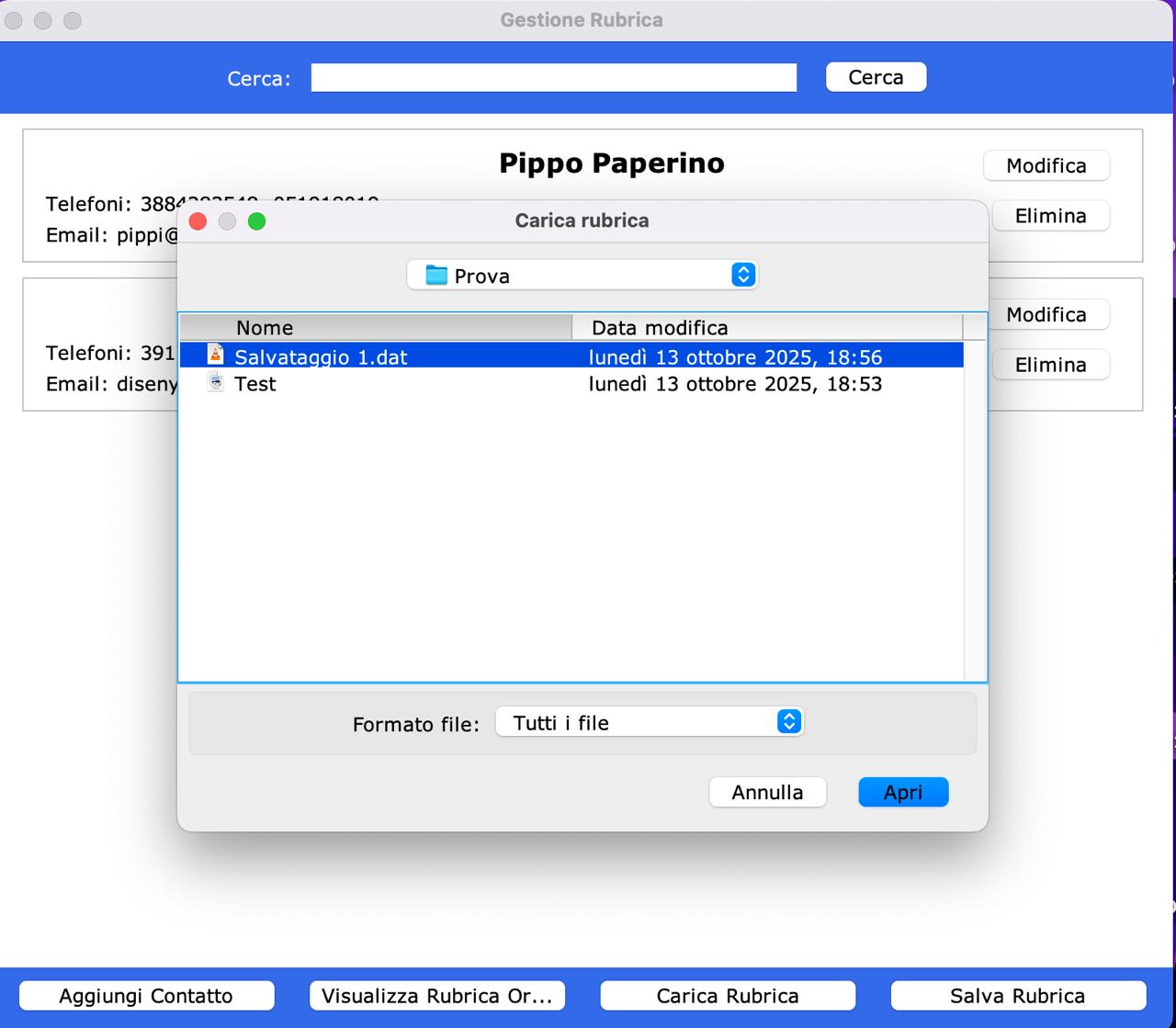
  
 *Figura 9*

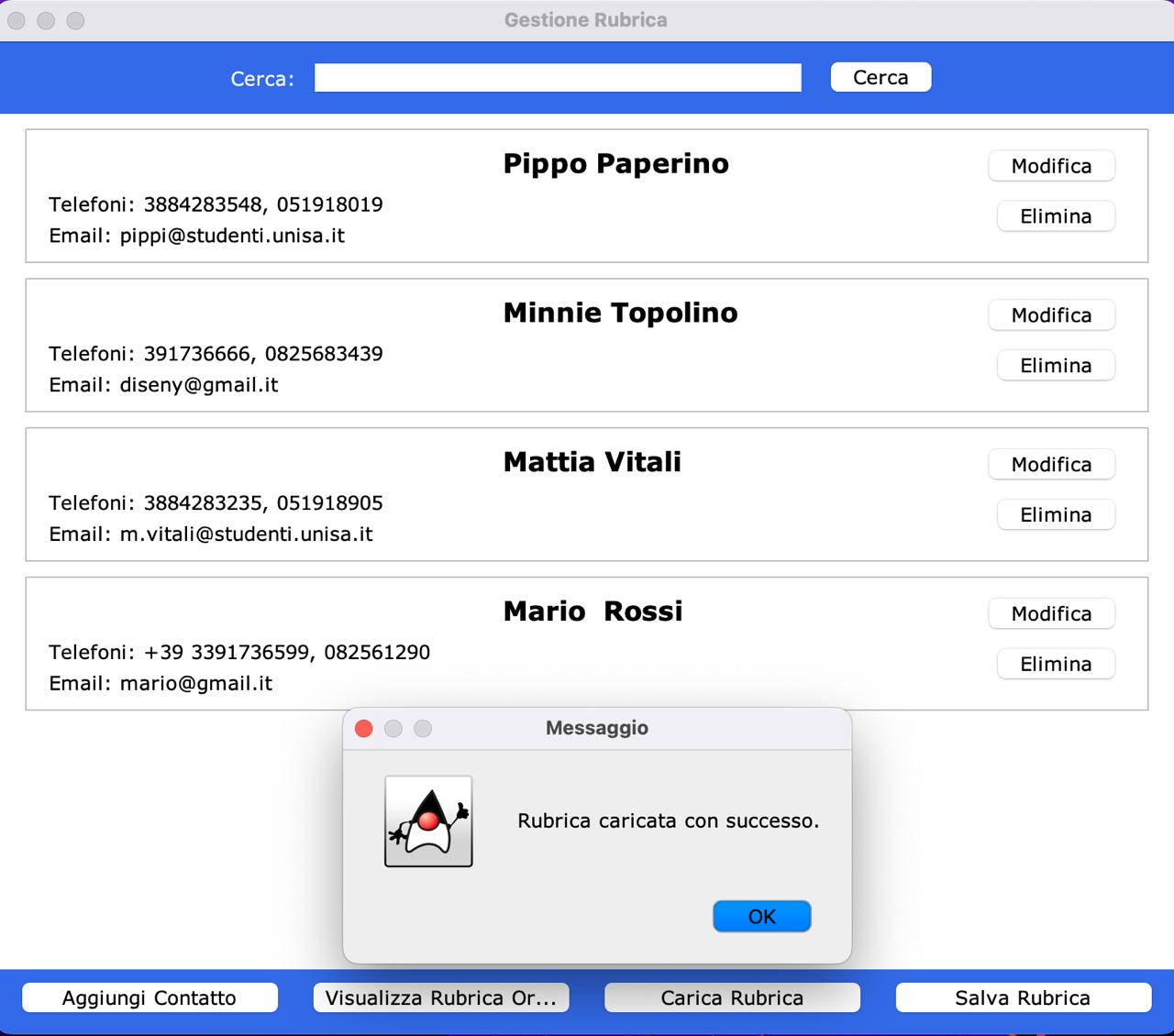
  
 *Figura 10*

  
 *Figura 11*

* **Caricare una rubrica**: premendo il pulsante relativo apparirà una finestra dove sarà possibile selezionare il percorso di un file desiderato, per il caricamento di una rubrica precedentemente salvata.

  
 *Figura 12*

  
 *Figura 13*

  
 *Figura 14*

# Tabella di Tracciabilità

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID Requisito | Progettazione | Implementazione | Testing | Requisiti correlati |
| IF.1.1 | RubricaController, Rubrica, SD\_UC.1.1 | RubricaController.java,  Rubrica.java |  | DF.1.1, DF.1.2, DF.1.3, ER.1.1, ER.1.4, US.1.3 |
| IF.1.2 | RubricaController, Rubrica, SD\_UC.1.2 | RubricaController.java,  Rubrica.java |  | DF.1.1, DF.1.2, DF.1.3, ER.1.1, ER.1.3, US.3.2 |
| IF.1.3 | RubricaController, Rubrica, SD\_UC.1.3 | RubricaController.java,  Rubrica.java |  | ER.3.1 |
| DF.1.1 | Contatto, ValidatoreDati, SD\_UC.1.1, SD\_UC.1.2 | Contatto.java, ValidatoreDati.java |  | IF.1.1, IF.1.2, ER.1.4, IF.3.1, IF.3.2 |
| DF.1.2 | Contatto, ValidatoreDati, SD\_UC.1.1, SD\_UC.1.2 | Contatto.java, ValidatoreDati.java |  | IF.1.1, IF.1.2, ER.1.1, ER.1.3 |
| DF.1.3 | Contatto, ValidatoreDati, SD\_UC.1.1, SD\_UC.1.2 | Contatto.java, ValidatoreDati.java |  |  |
| IF.2.1 | RubricaController, GestoreFile, SD\_UC.3.1 | RubricaController.java,  GestoreFile.java |  | ER.2.2. ER.2.3, US.1.2 |
| IF.2.2 | RubricaController, GestoreFile, SD\_UC.3.2 | RubricaController.java,  GestoreFile.java |  | ER.2.1, US.1.1 |
| IF.3.1 | Rubrica, SD\_UC.2.1, SD\_UC.2.3 | Rubrica.java |  | IF.3.2, ER.3.1, US.2.1 |
| IF.3.2 | Rubrica, SD\_UC.2.2 | Rubrica.java |  | ER.3.1, ER.3.2, US.2.2 |
| US.1.1 | PanelStrumenti | MainView.java, PanelStrumenti.java |  | IF.2.2 |
| US.1.2 | PanelStrumenti | MainView.java, PanelStrumenti.java |  | IF2.1 |
| US.1.3 | PanelStrumenti, DialogoContatto | MainView.java, PanelStrumenti.java, DialogoContatto.java |  | IF1.1 |
| US.2.1 | PanelListaContatti | MainView.java |  | IF3.1 |
| US.2.2 | PanelListaContatti | PanelListaContatti.java |  | IF.3.2 |
| US.3.1 | RubricaController. PanelListaContatti | MainView.java, PanelListaContatti.java |  | ER.3.1 |
| US.3.2 | PanelListaContatti, DialogoContatto | MainView.java, PanelListaContatti.java, DialogoContatto.java |  | IF.1.2 |
| ER.1.1 | ValidatoreDati, MainView, SD\_UC.1.1, SD\_UC.1.2 | ValidatoreDati.java. DialogoContatto.java |  | DF.1.2, IF.1.1, IF.1.2 |
| ER.1.2 | ValidatoreDati, MainView, SD\_UC.1.1, SD\_UC.1.2 | ValidatoreDati.java. DialogoContatto.java |  | DF.1.3, IF.1.1, IF.1.2 |
| ER.1.3 | ValidatoreDati, MainView, SD\_UC.1.1, SD\_UC.1.2 | ValidatoreDati.java. DialogoContatto.java |  | DF.1.2, DF.1.3, IF.1.1, IF.1.2 |
| ER.1.4 | ValidatoreDati, MainView, SD\_UC.1.1, SD\_UC.1.2 | ValidatoreDati.java. DialogoContatto.java |  | DF.1.1, IF.1.1 |
| ER.2.1 | GestoreFile, MainView, SD\_UC.3.2 | MainView.java, GestoreFile.java |  | IF.2.2 |
| ER.2.2 | RubricaController, MainView, SD\_UC.3.1, SD\_UC.2.1, SD\_UC.2.2, SD\_UC.2.3 | MainView.java, RubricaController.java |  | IF.2.1 |
| ER.2.3 | GestoreFile, MainView, SD\_UC.3.1 | MainView.java, GestoreFile.java |  | IF.2.1 |
| ER.3.1 | RubricaController, MainView, SD\_UC.2.1 | MainView.java, RubricaController.java |  | IF.1.3, IF.3.1, IF.3.2, US.3.1 |
| ER.3.2 | RubricaController, MainView, SD\_UC.2.2 | Rubrica.java, RubricaController.java, PanelStrumenti.java |  | IF.3.2 |