

RUBRICA GRUPPO 15

**DOCUMENTO DI
PROGETTAZIONE**

Luigi Perone, Matteo Pepe, Claudio Panico, Umberto Scassillo

Contents

1	INTRODUZIONE	3
1.1	DESCRIZIONE DELLE INFORMAZIONI CONTENUTE NEL DOCUMENTO	3
1.2	SCOPO DEL DOCUMENTO	3
2	ARCHITECTURAL DESIGN	3
2.1	PATTERN MVC	3
3	DETAILED DESIGN	4
3.1	PACKAGE DIAGRAM	4
3.2	DETAILED CLASS DIAGRAM	5
3.2.1	RUBRICA CLASS	6
3.2.2	FILE MANAGER INTERFACE	7
3.2.3	CONTATTO CLASS	8
3.2.4	NUMERO TELEFONO CLASS	10
3.2.5	PREFISSO CLASS	11
3.2.6	CHECK NUMERO TELEFONO INTERFACE	12
3.2.7	EMAIL CLASS	13
3.2.8	CHECK EMAIL INTERFACE	14
3.2.9	CHECK LUNGHEZZA INTERFACE	14
3.2.10	CHECK CONTATTO VALIDO INTERFACE	14
3.2.11	HOME PAGE CONTROLLER CLASS	15
3.2.12	MODIFICA CONTROLLER CLASS	17
3.2.13	Aggiungi CONTROLLER CLASS	19
3.2.14	VISUALIZZA CONTROLLER CLASS	21
3.2.15	CHECK ALERT INTERFACE	22
3.3	SEQUENCE DIAGRAMS	23
3.3.1	FUNZIONE IMPORTA	23
3.3.2	FUNZIONE ESPORTA	24
3.3.3	FUNZIONE MODIFICA CONTATTO	25
3.3.4	FUNZIONE ELIMINA CONTATTO	27
3.3.5	FUNZIONE INSERISCI	28
4	USER INTERFACE	30
4.1	RICERCA CONTATTO	30
4.2	CAMBIA ORDINE RUBRICA	31
4.3	VISUALIZZA CONTATTO	32
4.4	MODIFICA CONTATTO	33
4.5	ELIMINA CONTATTO	34
4.6	AGGIUNGI NUOVO CONTATTO	35
4.7	SALVA CONTATTO	36
4.8	IMPORTA	37
4.9	ESPORTA	38
5	MATRICE TRACCIABILITA' AGGIORNATA	39
6	SPIEGAZIONE DELLE MODIFICHE APPORTATE	40

1 INTRODUZIONE

1.1 DESCRIZIONE DELLE INFORMAZIONI CONTENUTE NEL DOCUMENTO

Un documento di design di un progetto software è un elemento cruciale nel processo di sviluppo di quest'ultimo, che serve a guidare il team di sviluppo attraverso la progettazione e l'implementazione del sistema software.

Questo documento includerà:

1. **Architettura del sistema:** questa sezione descrive l'architettura di alto livello del sistema, comprese le relazioni tra i vari componenti di quest'ultimo.
2. **Detailed design:** questa sezione fornisce un'elaborazione dettagliata di ogni componente, inclusi i sequence diagrams, activity diagrams ed eventuali altri diagrammi di supporto.
3. **Interfaccia:** questa sezione descrive l'interfaccia tra il sistema in sviluppo e l'utente, incluso il modo in cui l'utente interagisce con essa.

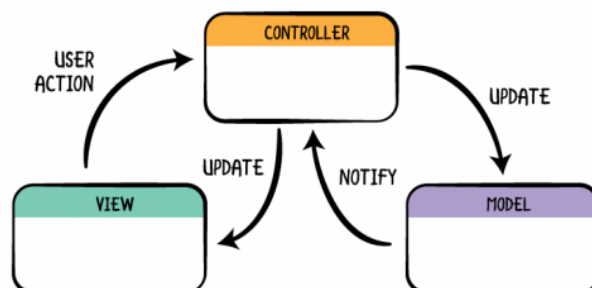
1.2 SCOPO DEL DOCUMENTO

Lo scopo principale di un documento di design è di fornire una visione d'insieme dettagliata del sistema in sviluppo, in modo che tutti possano comprendere come il sistema funzionerà e come sarà sviluppato. Questo documento è un riferimento essenziale per il team di sviluppo durante tutto il processo, in quanto definisce una base comune da cui attingere nelle successive fasi di implementazione e testing.

2 ARCHITECTURAL DESIGN

2.1 PATTERN MVC

Questo progetto sarà implementato come struttura modulare, l'architettura utilizzata è quella MVC. Per questo progetto è stato scelto il design orientato agli oggetti, quest'ultimo favorisce eventuali modifiche future, garantendo la manutenibilità del software.

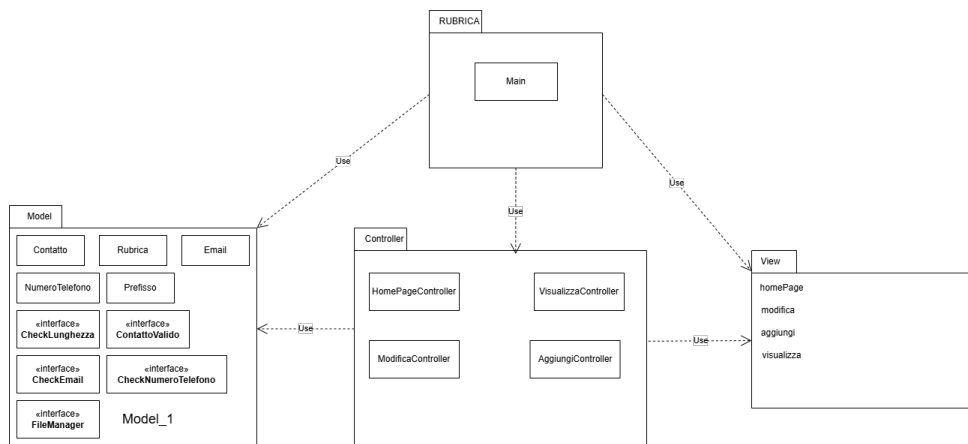


- **Model:** incapsula i dati e fornisce i metodi che permettono di interagire con essi e manipolarli. Non conosce nulla dell'interfaccia utente.
- **View:** è responsabile della presentazione dei dati all'utente e gestisce i componenti dell'interfaccia.

- **Controller:** agisce come un intermediario tra il modello e la vista:
 - Comunica con il modello per richiedere i dati o effettuare modifiche in base alle azioni richieste dall'utente.
 - Aggiorna la vista per riflettere sull'interfaccia tutti i cambiamenti nel modello avvenuti a seguito di un'azione dell'utente.

3 DETAILED DESIGN

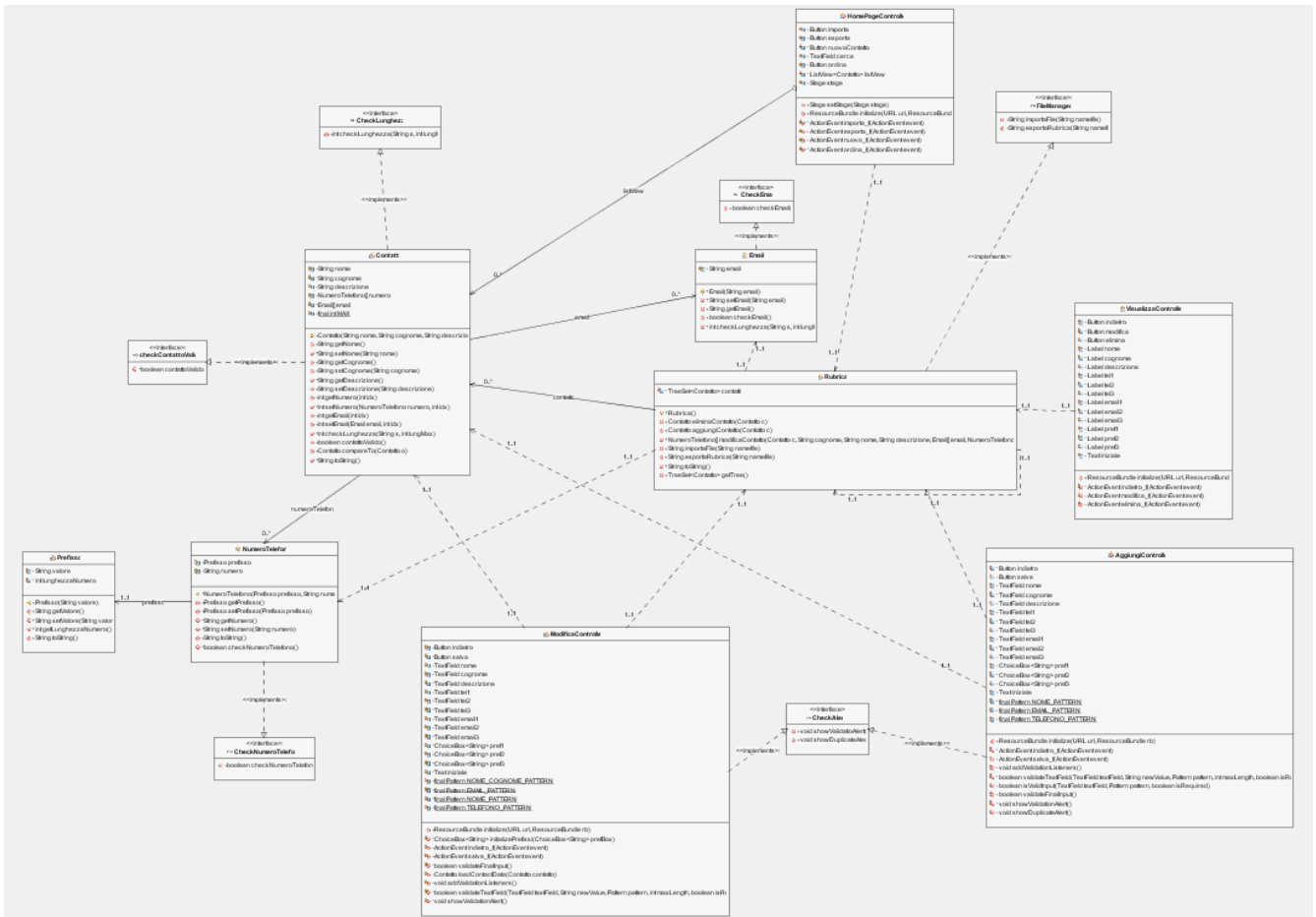
3.1 PACKAGE DIAGRAM




Rubrica è il package principale dell'applicazione e rappresenta il livello superiore della struttura. Da esso dipendono tre sottocartelle principali e il file main:

- **Main.java**
- **Model:** Contatto.java, Rubrica.java, Email.java, NumeroTelefono.java, Prefisso.java, CheckLunghezza.java, ContattoValido.java, CheckEmail.java, CheckNumeroTelefono.java, FileManager.java;
- **View:** homePage.fxml, aggiungi.fxml, modifica.fxml, visualizza.fxml;
- **Controller:** homePageController.java, aggiungiController.java, modificaController.java, visualizzaController.java;

3.2 DETAILED CLASS DIAGRAM



3.2.1 RUBRICA CLASS

Rubrica
 -TreeSet<Contatto> contatti
<ul style="list-style-type: none">+Rubrica()+void eliminaContatto(Contatto c)+void aggiungiContatto(Contatto c)+Contatto modificaContatto(Contatto c, String cognome, String nome, String descrizione, Email[] email, NumeroTelefono[] num)+Rubrica importaFile(String namefile)+void esportaRubrica(String namefile)+String toString()+TreeSet<Contatto> getTree()

- **Attributi:**

- TreeSet<Contatto> contatti: struttura dati che memorizza i contatti, garantendo l'ordinamento e la ricerca efficienti.

- **Metodi:**

- Rubrica(): costruttore per istanziare l'oggetto Rubrica.
- eliminaContatto(Contatto c): metodo utilizzato per la rimozione di un contatto specifico dalla rubrica.
- aggiungiContatto(Contatto c): metodo utilizzato per aggiungere un contatto alla rubrica.
- modificaContatto(Contatto c, String nome, String cognome, String descrizione, Email email[], NumeroTelefono[] n): metodo utilizzato per aggiornare i dettagli di un contatto esistente.
- toString(): metodo che restituisce l'insieme dei contatti, con i propri dettagli, come stringa.
- getTree(): metodo che permette di ritornare l'insieme dei contatti.

- **Implementa l' interfaccia:**

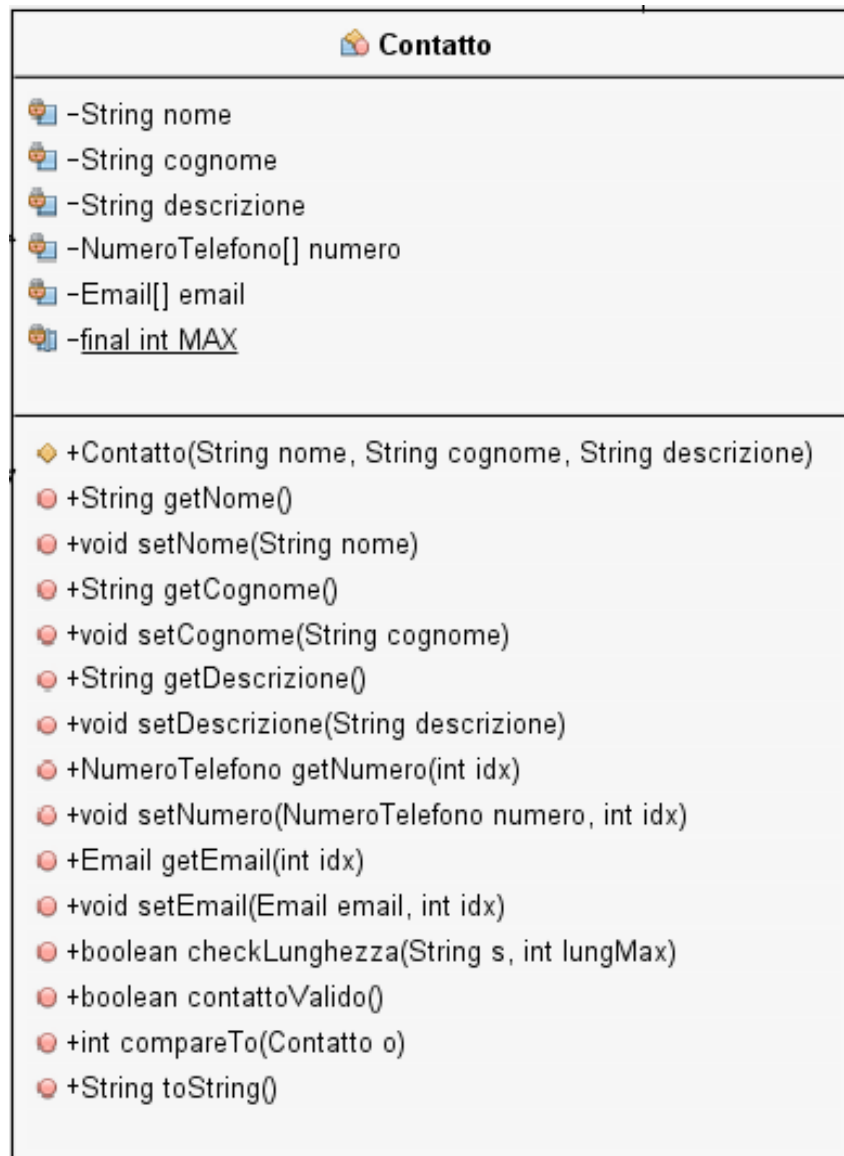
- FileManager

3.2.2 FILE MANAGER INTERFACE



- Metodi:
 - **importaFile(String nameFile)**:metodo per l'importo di un file contenente una rubrica di contatti.
 - **esportaRubrica(String nameFile)**:metodo per l'esportazione, salvataggio dei contatti della rubrica presente nel sistema.

3.2.3 CONTATTO CLASS



- **Attributi:**

- **String nome:** rappresenta il nome del contatto.
- **String cognome:** rappresenta il cognome del contatto.
- **String descrizione:** breve descrizione del contatto, utile per note personali.
- **NumeroTelefono[] numero:** array di oggetti **NumeroTelefono**, che memorizzano i numeri di telefono associati al contatto.
- **Email[] email:** array di oggetti **Email**, che che memorizza l'insieme delle email associate al contatto.
- **final int MAX:** costante che definisce il numero massimo di Email e NumeroTelefono.

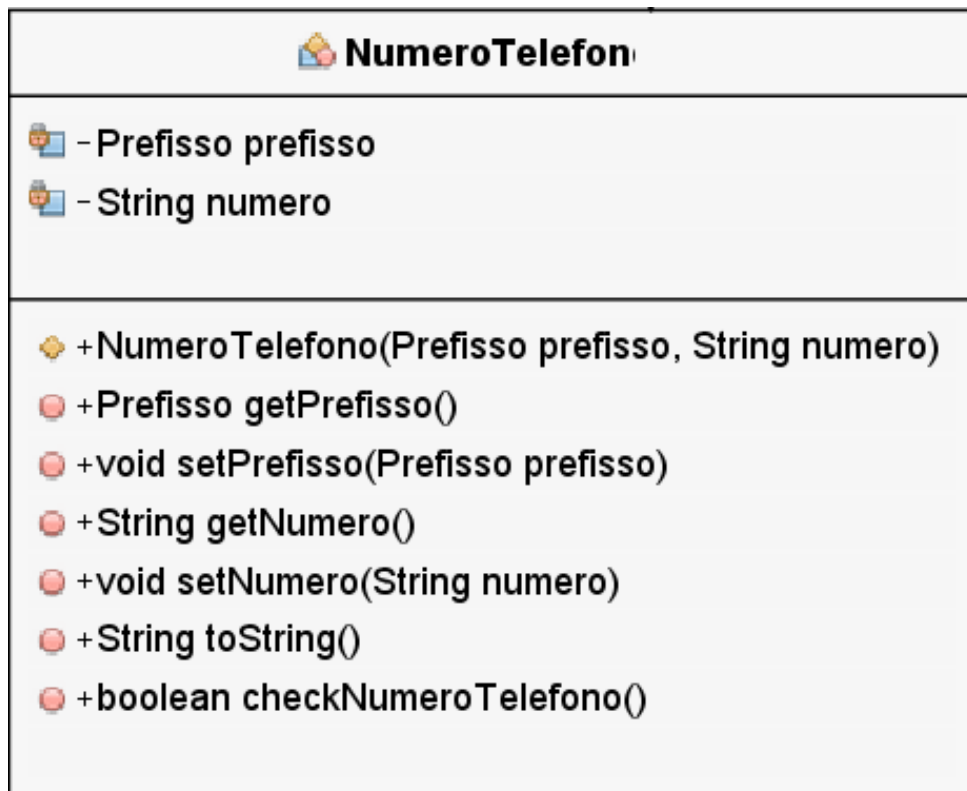
- **Metodi:**

- `Contatto(String nome, String cognome, String descrizione)`: costruttore della classe, assegnandogli un nome, cognome e una descrizione, inoltre verrà istanziato un array di `Email` e `NumeroTelefono`.
- `setNome(String n)`: imposta il nome del contatto.
- `setCognome(String c)`: imposta il cognome del contatto.
- `setDescrizione(String d)`: imposta la descrizione del contatto.
- `getNome()`: ritorna il nome del contatto.
- `getCognome()`: ritorna il cognome del contatto.
- `getDescrizione()`: ritorna la descrizione del contatto.
- `getNumero(int idx)`: restituisce l'idx-esimo numero di telefono.
- `getEmail(int idx)`: restituisce l'idx-esima email.
- `setNumeroTelefono(NumeroTelefono numero, int idx)`: imposta l'idx-esimo numero di telefono del contatto.
- `setEmail>Email email, int idx)`: imposta l'idx-esima email del contatto.
- `compareTo(Contatto c)`: permette di confrontare contatti tra loro, basandosi principalmente sul cognome dei contatti, a partià di cognome si confronta per nome.
- `toString()`: metodo che permette di avere una rappresentazione testuale del contatto: `Cognome Nome`.

- **Implementa le interfacce:**

- `checkContattoValido`, `CheckLunghezza`,.

3.2.4 NUMERO TELEFONO CLASS



- **Attributi:**

- **Prefisso prefisso:** oggetto Prefisso che rappresenta il prefisso del numero di telefono.
- **String numero:** rappresenta il numero di telefono.









- **Metodi:**

- **NumeroTelefono(Prefisso prefisso, String numero):** costruttore che istanzia l'oggetto passando il relativo prefisso e numero.
- **setNumero(String numero):** imposta un numero di telefono.
- **getNumero():** restituisce il numero di telefono.
- **setPrefisso(Prefisso prefisso):** imposta il prefisso.
- **getPrefisso():** ritorna il prefisso associato al numero di telefono.
- **toString():** permette la visualizzazione del Numero di Telefono come una stringa.

- **Implementa l'interfaccia:**

- **CheckNumeroTelefono.**

3.2.5 PREFISSO CLASS

 Prefisso	
 - String valore	
 - int lunghezzaNumero	
 + Prefisso(String valore)	
 + String getValore()	
 + void setValore(String valore)	
 + int getLunghezzaNumero()	
 + String toString()	

- **Attributi:**

- **String valore:** rappresenta il prefisso numerico del numero di telefono.
- **int lunghezzaNumero:** indica la lunghezza massima del numero di telefono che il prefisso può gestire.

- **Metodi:**

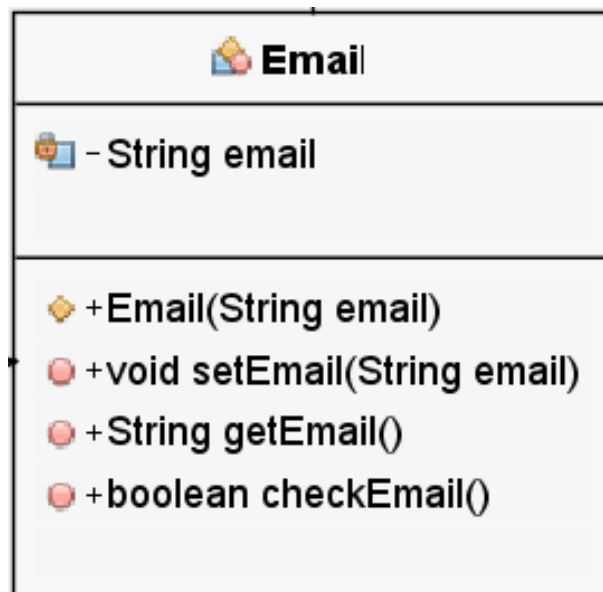
- **Prefisso(String valore):** istanzia il prefisso passandogli l'apposito valore e calcolando lunghezza massima associata.
- **getValore():** restituisce il valore del prefisso.
- **getLunghezzaNumero():** restituisce la lunghezza massima consentita per il numero associato al prefisso.
- **setValore(String prefisso):** imposta il valore del prefisso.
- **toString():** permette di visualizzare il prefisso come una stringa.

3.2.6 CHECK NUMERO TELEFONO INTERFACE



- Metodo: `checkNumeroTelefono()`
Verifica che la lunghezza del numero di telefono, sia conforme a quella dettata dal Prefisso.

3.2.7 EMAIL CLASS



- **Attributi:**

- `String email`: rappresenta l'indirizzo email del contatto.

- **Metodi:**

- `Email(String email)`: costruttore della classe `Email`.
- `setEmail(String email)`: imposta l'indirizzo email.
- `getEmail()`: restituisce l'indirizzo email.

- **Implementa l'interfaccia:**

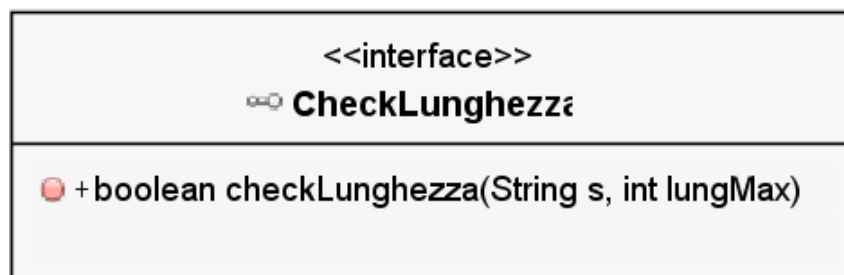
- `CheckEmail`.

3.2.8 CHECK EMAIL INTERFACE



- Metodo: `checkEmail(Email e)`
Valida la sintassi dell'email, verificando la presenza della @ e di un successivo dominio, creato dalla presenza di un punto seguito da almeno 2 caratteri.

3.2.9 CHECK LUNGHEZZA INTERFACE



- Metodo: `checkLunghezza()`
Controlla che la lunghezza del nome, cognome o descrizione rispetti i limiti predefiniti, prevenendo errori legati ad attributi troppo lunghi.

3.2.10 CHECK CONTATTO VALIDO INTERFACE



- Metodo: `contattoValido()`
Verifica se i campi obbligatori sono stati compilati correttamente e controlla se gli altri campi sono validi.

3.2.11 HOME PAGE CONTROLLER CLASS



- **Attributi:**

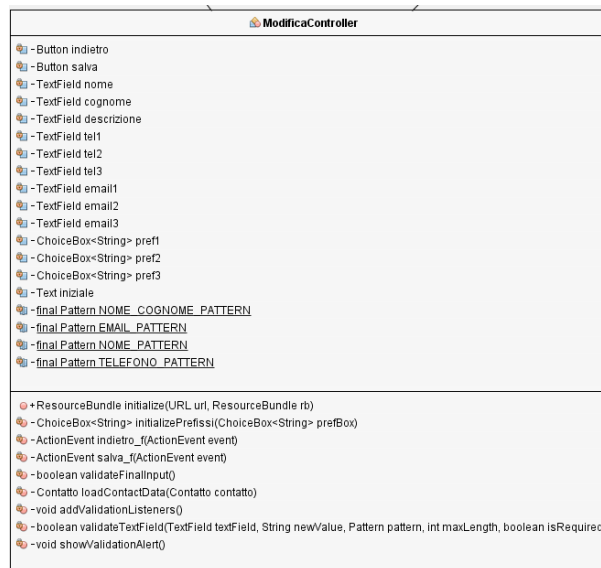
- **Button importa:** rappresenta il bottone per importare un file.
- **Button esporta:** rappresenta il bottone per esportare la rubrica in un file di output.
- **Button nuovoContatto:** rappresenta il bottone per la creazione di un nuovo Contatto.
- **TextField cerca:** rappresenta la barra di ricerca per filtrare la rubrica quindi cercare contatti esistenti.
- **Button ordina:** rappresenta il bottone per cambiare l'ordine di visualizzazione della rubrica.
- **ListView<Contatto> listView:** rappresenta l'insieme di contatti che si andranno a visualizzare nella homePage.
- **Stage stage:** rappresenta la finestra da visualizzare.

- **Metodi:**

- **setStage(Stage stage):** il metodo utilizzato dal Controller per impostare lo stage da visualizzare.
- **initialize(URL url, ResourceBundle rb):** il metodo serve per inizializzare il Controller.

- `importa_f(Action event)`: il metodo utilizzato dal Controller per importare una rubrica da file di input.
- `esporta_f(Action event)`: il metodo utilizzato dal Controller per esportare la rubrica in un file di output.
- `nuovo_f(Action event)`: il metodo utilizzato dal Controller per aggiungere un contatto nella rubrica.
- `ordina_f(Action event)`: il metodo utilizzato dal Controller per cambiare l'ordine di visualizzazione della rubrica.

3.2.12 MODIFICA CONTROLLER CLASS



• Attributi:

- **Button indietro**: rappresenta pulsante per tornare alla visualizzazione dei dati del contatto.
- **Button salva**: rappresenta il pulsante per salvare le modifiche apportate al contatto.
- **TextField nome**: rappresenta la casella di testo per la modifica del nome..
- **TextField cognome**: rappresenta la casella di testo per la modifica del cognome.
- **TextField descrizione**: rappresenta la casella di testo per la modifica della descrizione.
- **TextField tel1**: rappresenta la casella di testo per la modifica del primo numero di telefono.
- **TextField tel2**: rappresenta la casella di testo per la modifica del secondo numero di telefono.
- **TextField tel3**: rappresenta la casella di testo per la modifica del terzo numero di telefono.
- **TextField email1**: rappresenta la casella di testo per modificare la prima email.
- **TextField email2**: rappresenta la casella di testo per modificare la seconda email.
- **TextField email3**: rappresenta la casella di testo per modificare la terza email.
- **ChioceBox<String> pref1**: rappresenta il menu a tendina per modificare il primo prefisso.
- **ChioceBox<String> pref2**: rappresenta il menu a tendina per modificare il secondo prefisso.

- `ChioceBox<String> pref3`: rappresenta il menu a tendina per modificare il terzo prefisso.
- `Text iniziale`: rappresenta l’iniziale del nome del contatto che comparirà sul cerchio grigio.
- `final int MAX LENGHT`: costante che rappresenta la lunghezza massima dei campi.
- `final Pattern NOME_COGNOME_PATTERN`: costante pattern utile per validare nome e cognome.
- `final Pattern EMAIL_PATTERN`: costante pattern utile per validare i campi e-mail.
- `final Pattern NOME_PATTERN`: costante pattern utile per validare i campi nome o cognome.
- `final Pattern TELEFONO_PATTERN`: costante pattern utile per validare i campi numeri di telefono.

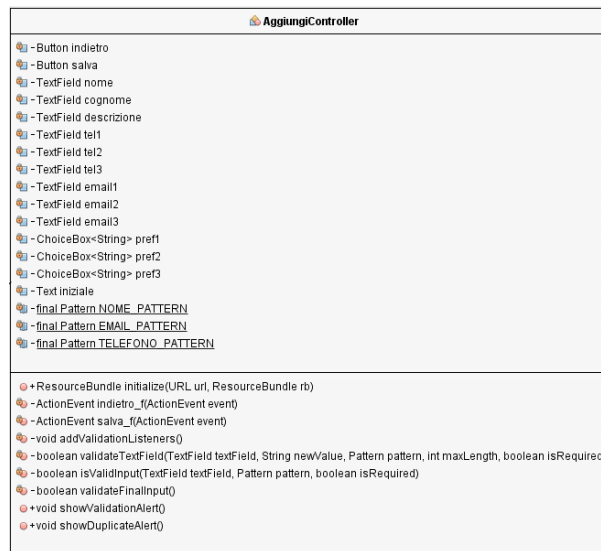
- **Metodi:**

- `initialize(URL url, ResourceBundle rb)`: il metodo serve per inizializzare il controller.
- `initializePrefissi(ChoiceBox<String> prefBox)`: inizializza i prefissi per un ChoiceBox.
- `indietro_f(Action event)`: il metodo utilizzato dal Controller per tornare alla visualizzazozione dei dettagli del contatto, annullando le modifiche.
- `salva_f()`: il metodo utilizzato dal Controller per salvare le modifiche apportate al contatto.
- `validateFinalInput()`: valida i dati inseriti prima del salvataggio .
- `loadContactData(Contatto contatto)`: metodo per caricare il contatto.
- `addValidationListeners()`: aggiunge Listener per la validazione dei campi di input.
- `validateTextField(TextField textField, String newValue, Pattern pattern, int maxLenght, boolean isRequired)`: verifica la correttezza del dato inserito in una casella di testo.

- **Implementa l’interfaccia:**

- `CheckAlert`.

3.2.13 Aggiungi CONTROLLER CLASS



- **Attributi:**

- **Menu indietro:** rappresenta pulsante per tornare alla visualizzazione dei dati del contatto.
- **Menu salva:** rappresenta il pulsante per salvare le modifiche apportate al contatto.
- **TextField nome:** rappresenta la casella di testo per impostare il nome..
- **TextField cognome:** rappresenta la casella di testo per impostare il cognome.
- **TextField descrizione:** rappresenta la casella di testo per impostare la descrizione.
- **TextField tel1:** rappresenta la casella di testo per impostare il primo numero di telefono.
- **TextField tel2:** rappresenta la casella di testo per impostare il secondo numero di telefono.
- **TextField tel3:** rappresenta la casella di testo per impostare il terzo numero di telefono.
- **TextField email1:** rappresenta la casella di testo per impostare la prima email.
- **TextField email2:** rappresenta la casella di testo per impostare la seconda email.
- **TextField email3:** rappresenta la casella di testo per impostare la terza email.
- **ChioceBox<String> pref1:** rappresenta il menu a tendina per impostare il primo prefisso.
- **ChioceBox<String> pref2:** rappresenta il menu a tendina per impostare il secondo prefisso.
- **ChioceBox pref3:** rappresenta il menu a tendina per impostare il terzo prefisso.

- `Text iniziale`: rappresenta l'iniziale del nome del contatto che comparirà sul cerchio grigio.

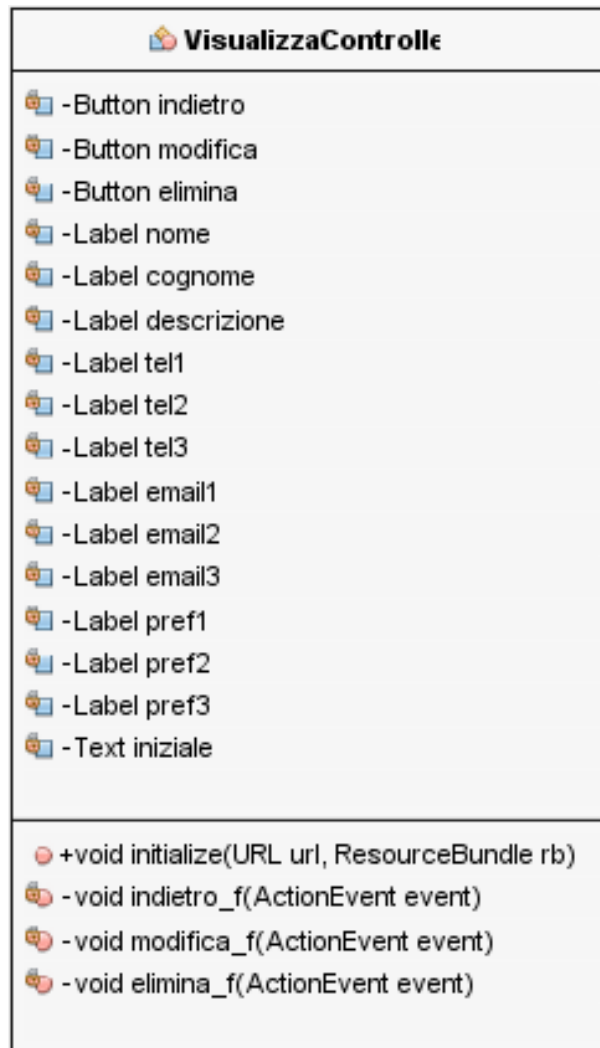
- **Metodi:**

- `initialize(URL url, ResourceBundle rb)`: il metodo serve per inizializzare il controller.
- `indietro_f(Action event)`: il metodo utilizzato dal Controller per tornare alla visualizzazione dei dettagli del contatto, annullando le modifiche.
- `salva_f()`: il metodo utilizzato dal Controller per salvare i dati inseriti.
- `validateFinalInput()`: valida i dati inseriti prima del salvataggio .
- `addValidationListeners()`: aggiunge Listener per la validazione dei campi di input.
- `validateTextField(TextField textField, String newValue, Pattern pattern, int maxLength, boolean isRequired)`: verifica la correttezza del dato inserito in una casella di testo.
- `isValidInput(TextField textField, Pattern pattern, boolean isRequired)`: verifica la correttezza del dato inserito.

- **Implementa l'interfaccia:**

- `CheckAlert`.

3.2.14 VISUALIZZA CONTROLLER CLASS



- **Attributi:**

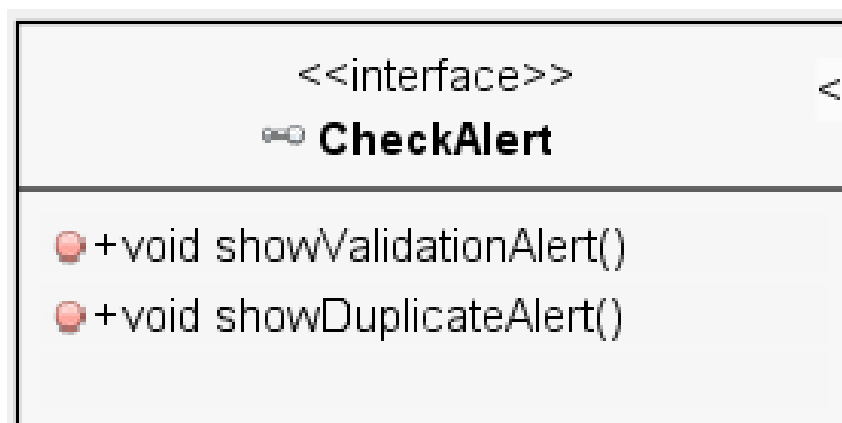
- Menu indietro: rappresenta pulsante per tornare alla homePage.
- Menu modifica: rappresenta il pulsante per modificare i dati del contatto.
- Menu elimina: rappresenta il pulsante per eliminare il contatto.
- Label nome: rappresenta l’etichetta contenente il nome..
- Label cognome: rappresenta l’etichetta contenente il cognome.
- Label descrizione: rappresenta l’etichetta contenente la descrizione.
- Label tel1: rappresenta l’etichetta contenente il primo numero di telefono.
- Label tel2: rappresenta l’etichetta contenente il secondo numero di telefono.
- Label tel3: rappresenta etichetta contenente il terzo numero di telefono.
- Label email1: rappresenta l’etichetta contenente la prima email.
- Label email2: rappresenta l’etichetta contenente la seconda email.
- Label email3: rappresenta l’etichetta contenente la terza email.
- Label pref1: rappresenta l’etichetta contenente il primo prefisso.

- Label `pref2`: rappresenta l’etichetta contenente il secondo prefisso.
- Label `pref3`: rappresenta l’etichetta contenente il terzo prefisso.
- Text `iniziale`: rappresenta l’iniziale del nome del contatto che comparirà sul cerchio grigio.

- **Metodi:**

- `initialize(URL url, ResourceBundle rb)`: il metodo serve per inizializzare il controller.
- `indietro_f(Action event)`: il metodo utilizzato dal Controller per tornare alla `homePage`.
- `modifica_f(Action event)`: il metodo utilizzato dal Controller per apportare modifiche al contatto.
- `elimina_f(Action event)`: il metodo utilizzato dal Controller per eliminare il contatto.

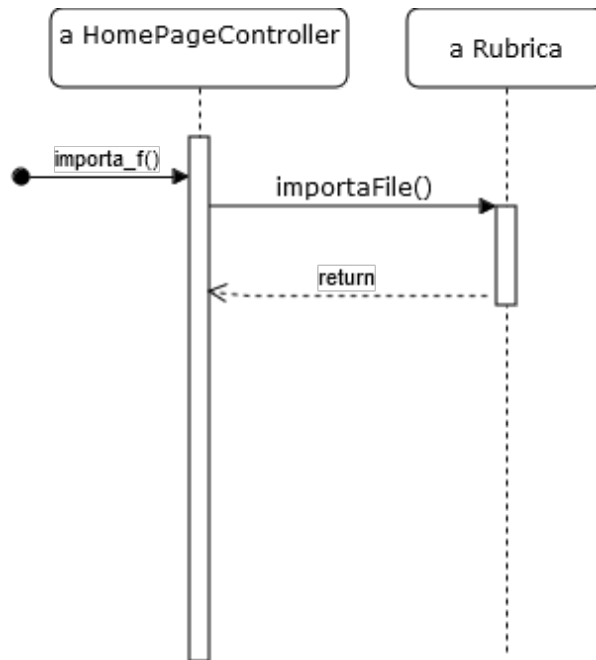
3.2.15 CHECK ALERT INTERFACE



- `showValidationAlert()`: metodo che mostra un messaggio di errore nel caso la validazione non va a buon fine.
- `showDuplicateAlert()`: metodo che mostra un messaggio di errore nel caso esiste già un contatto nella rubrica con stesso Nome e Cognome.

3.3 SEQUENCE DIAGRAMS

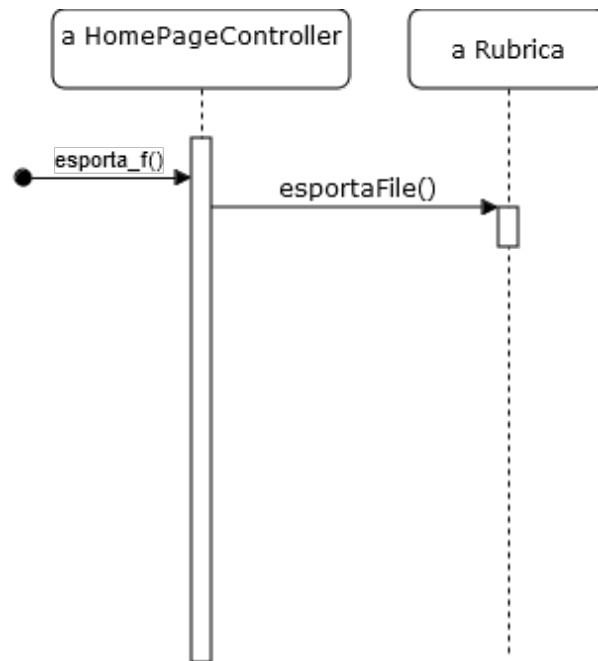
3.3.1 FUNZIONE IMPORTA



Il diagramma di sequenza riportato rappresenta l'interazione tra due classi: **HomeController** e **Rubrica**. I passi dell'interazione sono descritti di seguito:

- L'interazione inizia con una chiamata al metodo `importa_f()` sulla classe **HomeController**.
- A questo punto, **HomeController** interagisce con la classe **Rubrica** tramite la funzione `importaFile()`, che consente all'utente di selezionare il file da cui importare la rubrica desiderata.
- La funzione `importaFile()` restituirà alla classe **HomeController** la rubrica contenuta all'interno del file selezionato in precedenza.

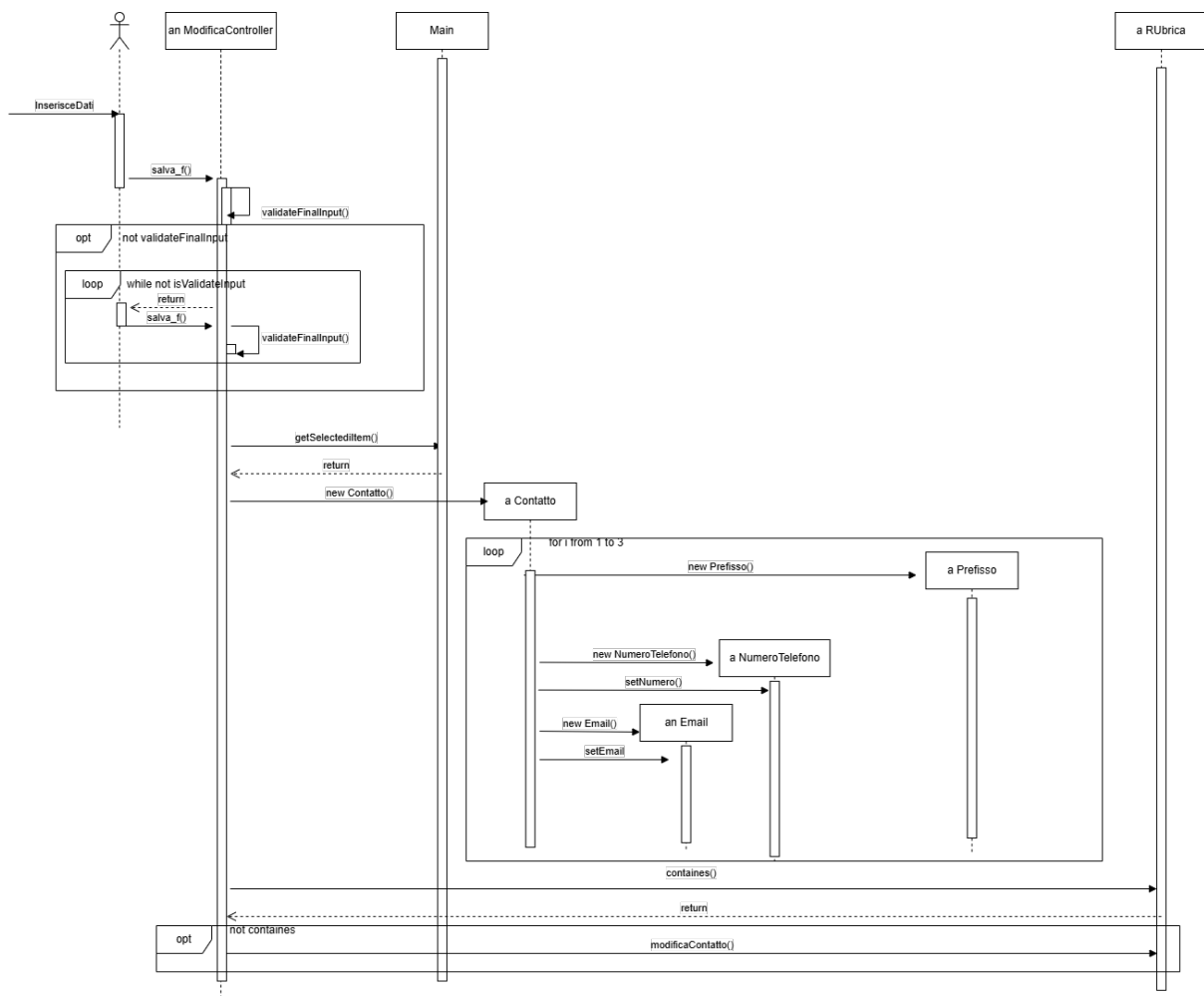
3.3.2 FUNZIONE ESPORTA



Il diagramma di sequenza riportato rappresenta l'interazione tra due classi: **HomeController** e **Rubrica**. I passi dell'interazione sono descritti di seguito:

- L'interazione inizia con una chiamata al metodo `esporta_f()` sulla classe **HomeController**.
- A questo punto, **HomeController** interagisce con la classe **Rubrica** tramite la funzione `esportaFile()`, che esporta la rubrica in un file.

3.3.3 FUNZIONE MODIFICA CONTATTO

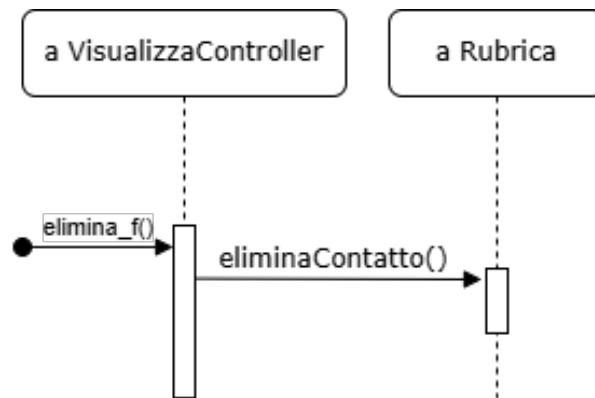


Il diagramma di sequenza riportato rappresenta l'interazione tra le classi: **ModificaController**, **Rubrica**, **Contatto**, **Email**, **NumeroTelefono** e **Prefisso**.
L'interazione prevede i seguenti passi:

- La sequenza inizia con l' **Utente** che inserisce i dati del contatti.
- Una volta terminato tramite il pulsante Salva viene invocata la funzione `salva_f()` su **AggiungiController**.
- **AggiungiController** tramite `salva_f()` verifica i dati inseriti.
- Nel caso in cui questi non sono validi si ritorna all'**Utente** che dovrà riinserire i dati, ciò si ripeterà finchè i dati siano validi.
- Per comprendere il contatto da modificare allora viene invocato il metodo `getSelectedItem()`.
- A questo punto **ModificaController** istanzia un nuovo **Contatto** che sarà quello con i campi nuovi, serve a capire se il contatto è già presente.
- **Contatto** invoca il metodo `new Prefisso()` per creare un prefisso che sarà poi associato ad un numero di telefono.

- Successivamente sarà istanziato un **NumeroTelefono**, per poi impostarlo come numero associato al contatto, tramite `setNumeroTelefono()` .
- Si farà la stessa operazione precedente ma con la classe **Email**, quindi viene istanziata, `new Email()`, e poi impostata al contatto, `setEmail()`.
- Queste ultimi operazioni sono ripetute per 3 volte essendo che ad ogni contatto possono essere associati 3 numeri di telefono e 3 e-mail.
- **ModifcaController** per poi verifica se il contatto è già presente nella rubrica tramite `contains()`.
- Nel caso il contatto non è presente il **ModifcaController** invoca il metodo `modificaContatto()`, per effettuare la modifica.

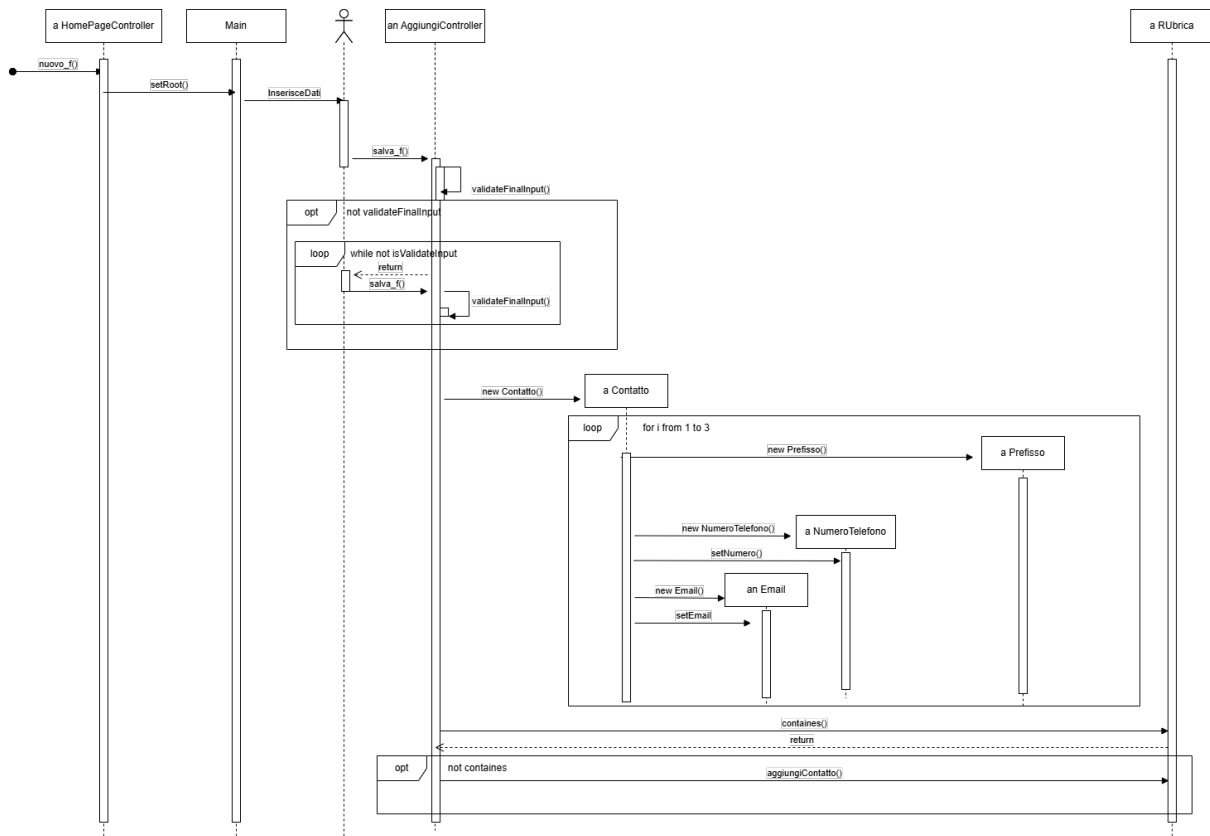
3.3.4 FUNZIONE ELIMINA CONTATTO



L'interazione riportata dal diagramma di sequenza rappresenta un'interazione che interessa 2 classi. I passi dell'interazione sono descritti di seguito:

- L'interazione inizia con una chiamata al metodo `elimina_f()` sulla classe **VisualizzaController**.
- Il metodo `elimina_f()` invia alla rubrica la richiesta di rimuovere il contatto selezionato tramite la funzione `eliminaContatto()`.
- A questo punto, è possibile scegliere di confermare o meno l'eliminazione del contatto.
- Confermando l'eliminazione, il contatto selezionato non sarà più visualizzato nella rubrica.

3.3.5 FUNZIONE INSERISCI



Il diagramma di sequenza riportato rappresenta l'interazione tra le classi: **HomePageController**, **Main**, **AggiungiController**, **Contatto**, **NumeroTelefono**, **Prefisso** ed **Email**, **Rubrica**. L'interazione prevede i seguenti passi:

- L'interazione inizia con una chiamata al metodo **nuovo_f()** sulla classe **HomePageController**.
- **HomePageController** invoca il metodo **setRoot** sul **Main** per far visualizzare la nuova schermata all'utente.
- A questo punto, **Utente** può inserire i dati del contatto, sulle apposite cassette di testo.
- Una volta terminato tramite il pulsante Salva viene invocata la funzione **salva_f()** su **AggiungiController**. **AggiungiController** tramite **salva_f()** verifica i dati inseriti.
- Nel caso in cui questi non sono validi si ritorna all'**Utente** che dovrà riinserire i dati, ciò si ripeterà finchè i dati siano validi.
- A questo punto **AggiungiController** istanzia un nuovo **Contatto**
- **Contatto** invoca il metodo **new Prefisso()** per creare un prefisso che sarà poi associato ad un numero di telefono.
- Successivamente sarà istanziato un **NumeroTelefono**, per poi impostarlo come numero associato al contatto, tramite **setNumeroTelefono()**.

- Si farà la stessa operazione precedente ma con la classe **Email**, quindi viene istanziata, `new Email()`, e poi impostata al contatto, `setEmail()`.
- Queste ultimi operazioni sono ripetute per 3 volte essendo che ad ogni contatto possono essere associati 3 numeri di telefono e 3 e-mail.
- Successivamente **AggiungiController** verifica tramite il metodo `contains()`, se il contatto è già esistente.
- Nel caso in cui il contatto non è presente **AggiungiController** viene aggiunto tramite `aggiungiContatto()`.

4 USER INTERFACE

In questa sezione viene mostrata una sequenza di screen nei quali sono visualizzate le possibili interazioni dell'utente con la rubrica.

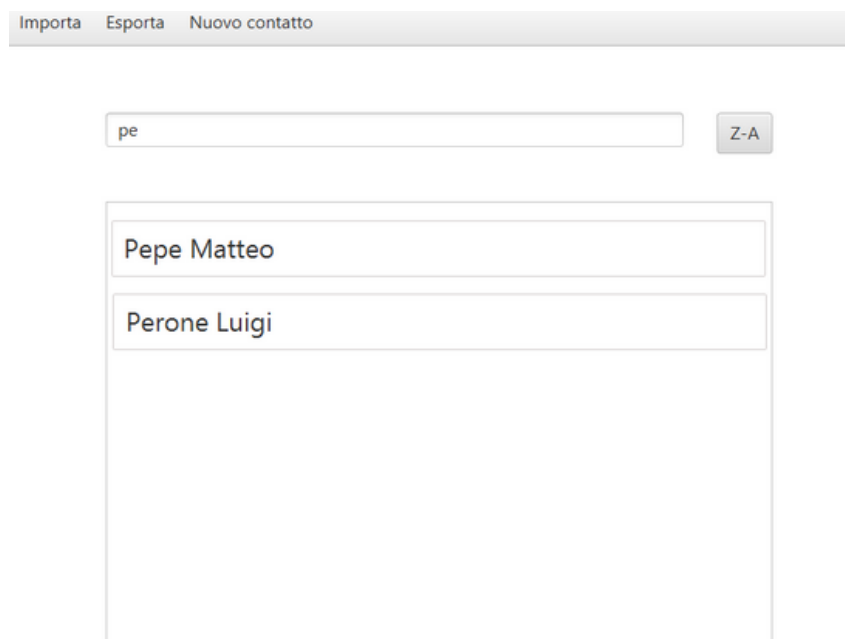
4.1 RICERCA CONTATTO

Prima: l'utente inserisce nel textField una sequenza di caratteri.



The screenshot shows a user interface for a contact book. At the top, there is a horizontal bar with three buttons: "Importa", "Esporta", and "Nuovo contatto". Below this bar, there is a search section. It features a text input field with the placeholder text "Cerca.." and a button labeled "Z-A" to its right. The text input field is circled in red. Below the search section is a large, empty rectangular area, which is the list view for search results.


Dopo: visualizzazione nella listView di tutti i contatti che iniziano con la sequenza di caratteri inserito nel textField.



The screenshot shows the same user interface as before, but now the search results are displayed. The text input field contains the characters "pe". The "Z-A" button is still present. Below the search section, the list view now displays two contact names: "Pepe Matteo" and "Perone Luigi".

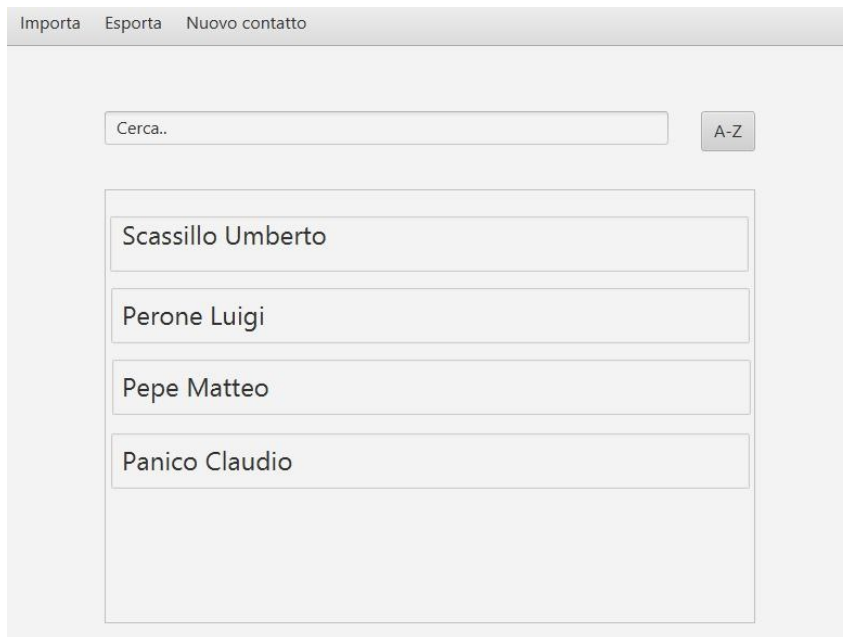
4.2 CAMBIA ORDINE RUBRICA

Prima: l'utente clicca sul pulsante Z-A.



The screenshot shows a web interface for managing contacts. At the top, there is a navigation bar with three buttons: 'Importa', 'Esporta', and 'Nuovo contatto'. Below this, there is a search bar labeled 'Cerca..' and a button labeled 'Z-A' which is circled in red. Below the search bar, there is a list of four contact names: 'Panico Claudio', 'Pepe Matteo', 'Perone Luigi', and 'Scassillo Umberto'.

Dopo: visualizzazione della listView al contrario e quindi non più in ordine alfabetico da A a Z ma dalla Z alla A.



The screenshot shows the same web interface as before, but now the 'A-Z' button is highlighted. The list of contact names is now in reverse alphabetical order: 'Scassillo Umberto', 'Perone Luigi', 'Pepe Matteo', and 'Panico Claudio'.

4.3 VISUALIZZA CONTATTO

Prima: l'utente clicca sul contatto di cui vuole visualizzare tutte le informazioni



Importa Esporta Nuovo contatto

Cerca.. Z-A

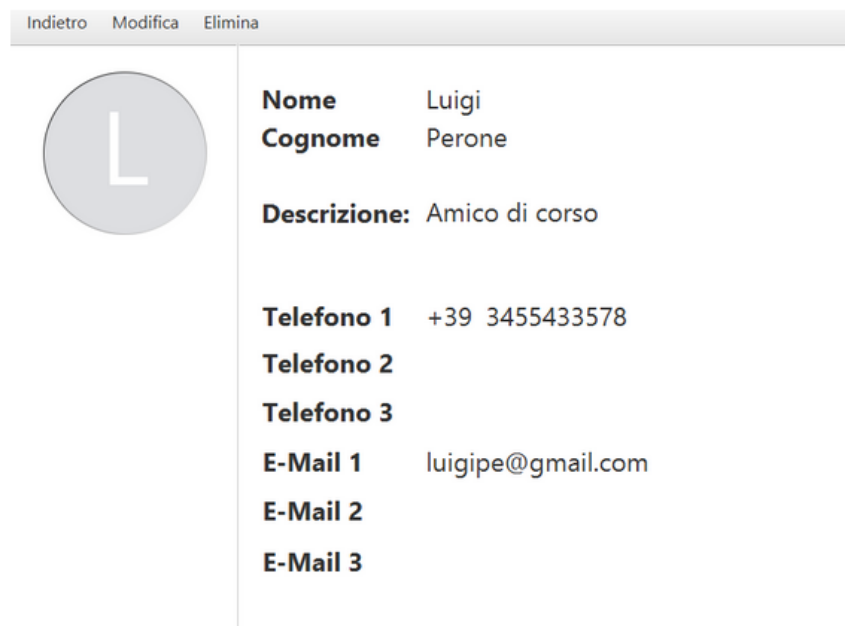
Panico Claudio

Pepe Matteo

Perone Luigi

Scassillo Umberto

Dopo: visualizzazione della schermata di dettaglio del contatto



Indietro Modifica Elimina

Nome Luigi

Cognome Perone

Descrizione: Amico di corso

Telefono 1 +39 3455433578

Telefono 2

Telefono 3

E-Mail 1 luigipe@gmail.com


E-Mail 2

E-Mail 3

4.4 MODIFICA CONTATTO

Prima: l'utente clicca sul pulsante modifica

[Indietro](#) [Modifica](#) [Elimina](#)



Nome Luigi

Cognome Perone

Descrizione: Amico di corso

Telefono 1 +39 3455433578

Telefono 2

Telefono 3


E-Mail 1 luigipe@gmail.com

E-Mail 2

E-Mail 3

Dopo: visualizzazione della schermata di modifica del contatto

[Indietro](#) [Salva](#)



Nome

Cognome

Descrizione:

Telefono 1

Telefono 2

Telefono 3

E-Mail 1


E-Mail 2

E-Mail 3

4.5 ELIMINA CONTATTO

Prima: l'utente clicca sul pulsante elimina

[Indietro](#) [Modifica](#) [Elimina](#)



Nome Luigi

Cognome Perone

Descrizione: Amico di corso

Telefono 1 +39 3455433578

Telefono 2

Telefono 3

E-Mail 1 luigipe@gmail.com

E-Mail 2

E-Mail 3

Dopo: il contatto viene eliminato previa conferma

[Importa](#) [Esporta](#) [Nuovo contatto](#)

[Z-A](#)

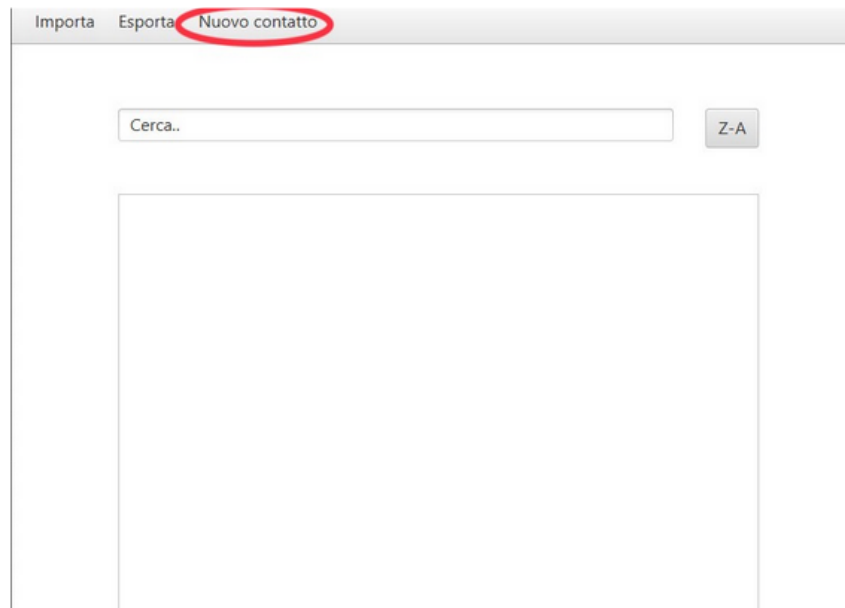
Panico Claudio

Pepe Matteo

Scassillo Umberto

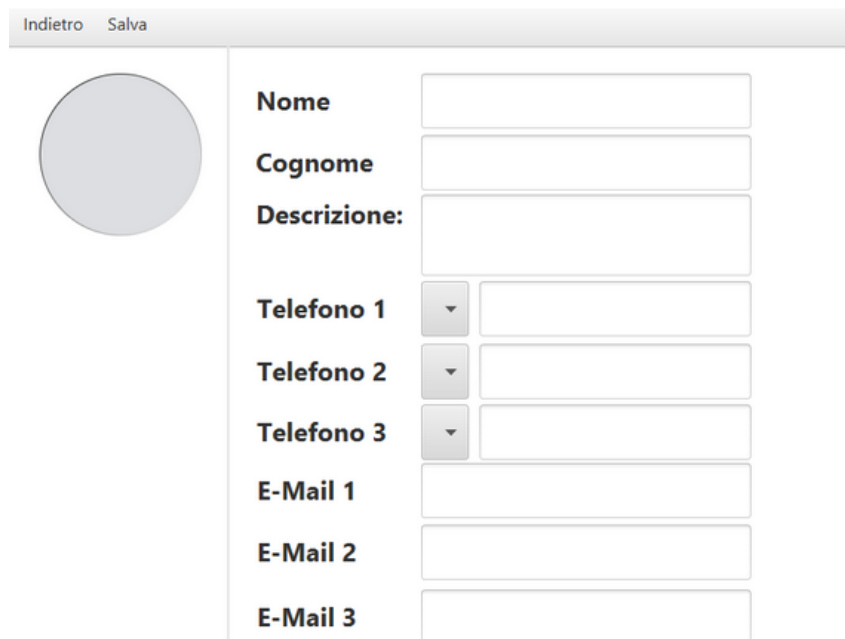
4.6 AGGIUNGI NUOVO CONTATTO

Prima: l'utente clicca sul bottone Nuovo contatto



The screenshot shows a web interface with a top navigation bar containing three buttons: 'Importa', 'Esporta', and 'Nuovo contatto'. The 'Nuovo contatto' button is circled in red. Below the navigation bar is a search area with a text input field labeled 'Cerca..' and a 'Z-A' button. The main content area is a large, empty rectangular box.

Dopo: visualizzazione della schermata di di aggiunta di un nuovo contatto




The screenshot shows a web interface for adding a new contact. At the top is a navigation bar with 'Indietro' and 'Salva' buttons. On the left is a large, empty circular profile picture placeholder. To the right of the placeholder is a form with the following fields: 'Nome' (text input), 'Cognome' (text input), 'Descrizione:' (text input), 'Telefono 1' (dropdown menu with a downward arrow and a text input), 'Telefono 2' (dropdown menu with a downward arrow and a text input), 'Telefono 3' (dropdown menu with a downward arrow and a text input), 'E-Mail 1' (text input), 'E-Mail 2' (text input), and 'E-Mail 3' (text input).

4.7 SALVA CONTATTO

Prima: l'utente clicca sul bottone salva

[Indietro](#) [Salva](#)



Nome

Cognome

Descrizione:

Telefono 1

Telefono 2

Telefono 3


E-Mail 1

E-Mail 2

E-Mail 3

Dopo: le modifiche vengono salvate se i dati inseriti rispettano il formato

[Indietro](#) [Modifica](#) [Elimina](#)



Nome Luigi

Cognome Perone

Descrizione: Amico di corso

Telefono 1 +39 3455433578

Telefono 2

Telefono 3

E-Mail 1 luigipe@gmail.com

E-Mail 2

E-Mail 3

4.8 IMPORTA

Prima: l'utente clicca sul bottone importa e seleziona il file da cui importare



The screenshot shows a web interface with a top navigation bar containing three buttons: 'Importa', 'Esporta', and 'Nuovo contatto'. The 'Importa' button is circled in red. Below the navigation bar, there is a search input field with the placeholder text 'Cerca..' and a 'Z-A' button to its right. A large, empty rectangular box is positioned below the search field, intended for displaying the imported contacts.

Dopo: la rubrica importata è presente



The screenshot shows the same web interface as before, but now the large rectangular box contains a list of four contacts, each in its own row:

Panico Claudio
Pepe Matteo
Perone Luigi
Scassillo Umberto

The search input field and the 'Z-A' button remain above the list.

4.9 ESPORTA

Prima: l'utente clicca sul bottone esporta e seleziona il file su cui esportare



The screenshot shows a web interface with a top navigation bar containing three buttons: 'Importa', 'Esporta', and 'Nuovo contatto'. The 'Esporta' button is circled in red. Below the navigation bar is a search section with a text input field labeled 'Cerca..' and a 'Z-A' button. The main content area displays a list of four contact names, each in its own box: 'Panico Claudio', 'Pepe Matteo', 'Perone Luigi', and 'Scassillo Umberto'. There is an empty box below the last name.

Dopo: visualizzazione della schermata home e file salvato con successo



The screenshot shows the same web interface as before, but now the 'Esporta' button in the top navigation bar is highlighted with a grey background, indicating it is the active selection. The rest of the interface, including the search bar and the list of contact names, remains unchanged.

5 MATRICE TRACCIABILITA' AGGIORNATA

La matrice di tracciabilità è uno strumento utilizzato nella gestione dei requisiti per tracciare e documentare le relazione tra diversi elementi di un sistema. Consente nelle diverse fasi di sviluppo del progetto di verificare che tutti i requisiti siano stati sviluppati, testati, confezionati e consegnati. La tabella verrà aggiornata nelle prossime fasi di sviluppo.

ID REQ.	DESIGN	CODE	TEST	STATUS
IF 1.1	Rubrica, Contatto	setNome(),setCognome()	ContattoTest	STARTED
IF 1.2	NumeroTelefono, Prefisso, Contatto, Rubrica	setNumeroTelefono()	PrefissoTest, NumeroTelefonoTest	STARTED
IF 1.3	Email, Contatto, Rubrica	setEmail()	EmailTest, ContattoTest	STARTED
IF 1.4	Contatto, Rubrica	setDescrizione()	ContattoTest	STARTED
DF 1.1	Contatto	/	ContattoTest	STARTED
DF 1.2	NumeroTelefono, Prefisso	checkNumeroTelefono()	NumeroTelefonoTest	STARTED
DF 1.3	Email	checkEmail()	EmailTest	STARTED
DF 1.4	Contatto	/	ContattoTest	STARTED
IF 2.1	Contatto, Rubrica	modificaContatto()	RubricaTest	STARTED
IF 2.2	NumeroTelefono, Prefisso, Contatto, Rubrica	modificaContatto()	RubricaTest	STARTED
IF 2.3	Email, Contatto, Rubrica	modificaContatto()	RubricaTest	STARTED
IF 2.4	Contatto, Rubrica	modificaContatto()	RubricTest	STARTED
IF 3	Rubrica	eliminaContatto()	RubricaTest	STARTED
IF 4	VisualizzaController	/	/	STARTED
IF 5	Rubrica, HomePageController	/	/	STARTED
IF 6	Rubrica, HomePageController	ordina_f()	/	STARTED
IF 7	Rubrica	importaFile()	RubricaTest	STARTED
DF 7.1	HomePageController	/	RubricaTest	STARTED
DF 7.2	Rubrica	importaFile()	RubricaTest	STARTED
IF 8	Rubrica	esportaRubrica()	RubricaTest	STARTED
IF 9	Rubrica, HomePageController	/	/	STARTED

Table 1: Matrice tracciabilità

6 SPIEGAZIONE DELLE MODIFICHE APPORTATE

- aggiornamento funzionalità di `checkEmail()` verificando la presenza di `@` e di un successivo dominio, ossia un punto seguito da almeno 2 caratteri .
- per dividere i compiti del `ModificaController` e `AggiungiController` si è deciso di implementare una nuova interfaccia `CheckAlert`.
- per facilità di implementazione sono stati aggiunti nuovi metodi o attributi ai controller.
- matrice di tracciabilità aggiornata.