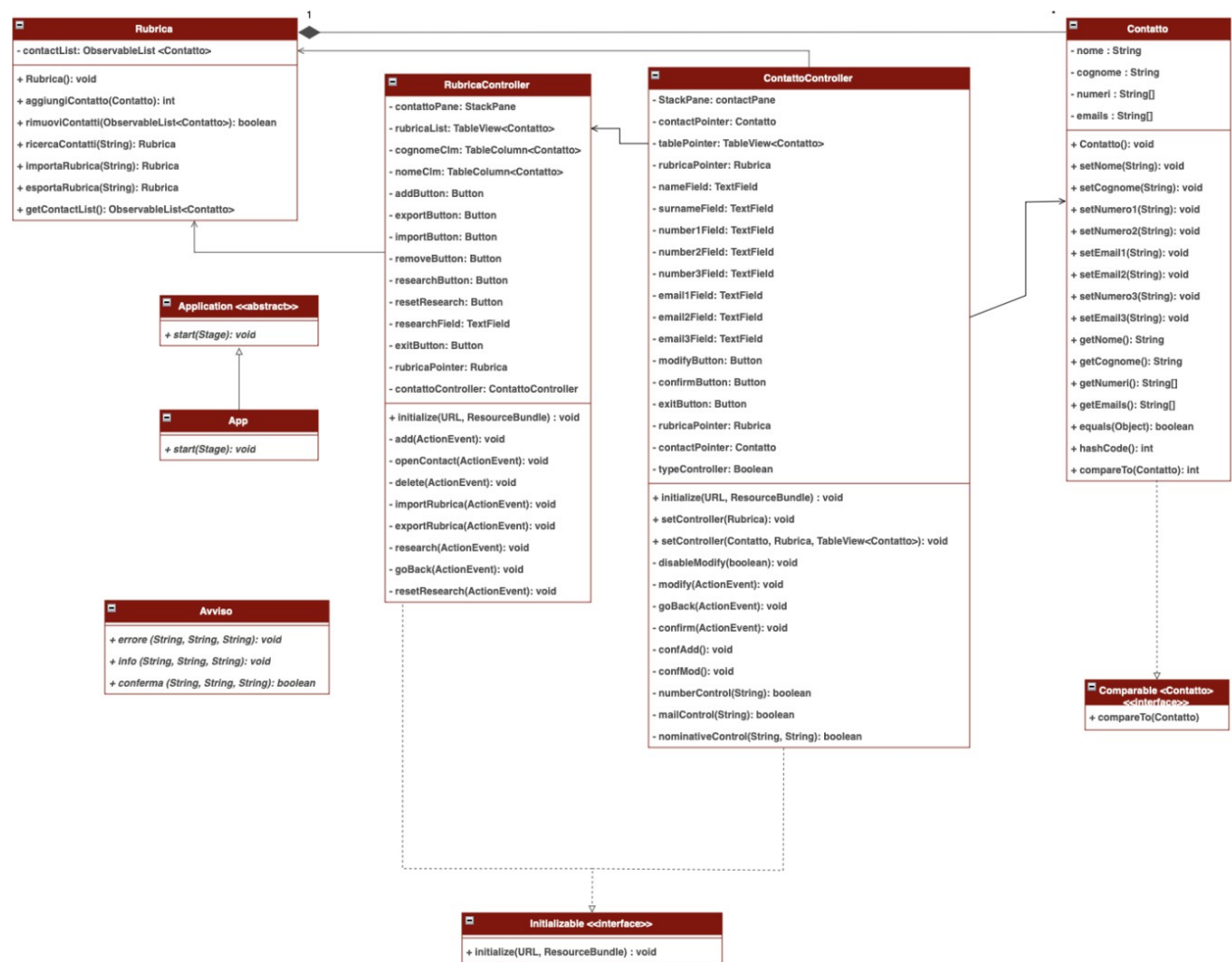


# Documento Design

## Diagramma delle classi



## Coesione delle classi

Rubrica
- contactList: ObservableList <Contatto>
+ Rubrica(): void
+ aggiungiContatto(Contatto): int
+ rimuoviContatti(ObservableList<Contatto>): boolean
+ ricercaContatti(String): Rubrica
+ importaRubrica(String): Rubrica
+ esportaRubrica(String): Rubrica
+ getContactList(): ObservableList<Contatto>


La classe Rubrica ha un livello di coesione funzionale, poiché si occupa di gestire un unico compito: le operazioni su una lista di contatti.

RubricaController
- contattoPane: StackPane
- rubricaList: TableView<Contatto>
- cognomeCim: TableColumn<Contatto>
- nomeCim: TableColumn<Contatto>
- addButton: Button
- exportButton: Button
- importButton: Button
- removeButton: Button
- researchButton: Button
- resetResearch: Button
- researchField: TextField
- exitButton: Button
- rubricaPointer: Rubrica
- contattoController: ContattoController
+ initialize(URL, ResourceBundle) : void
- add(ActionEvent): void
- openContact(ActionEvent): void
- delete(ActionEvent): void
- importRubrica(ActionEvent): void
- exportRubrica(ActionEvent): void
- research(ActionEvent): void
- goBack(ActionEvent): void
- resetResearch(ActionEvent): void

Il livello di coesione per RubricaController è comunicazionale poiché i metodi lavorano sugli stessi dati (oggetti di tipo rubrica).

ContattoController
<ul style="list-style-type: none"> <li>- StackPane: contactPane</li> <li>- contactPointer: Contatto</li> <li>- rubricaPointer: Rubrica</li> <li>- nameField: TextField</li> <li>- surnameField: TextField</li> <li>- number1Field: TextField</li> <li>- number2Field: TextField</li> <li>- number3Field: TextField</li> <li>- email1Field: TextField</li> <li>- email2Field: TextField</li> <li>- email3Field: TextField</li> <li>- modifyButton: Button</li> <li>- confirmButton: Button</li> <li>- exitButton: Button</li> <li>- rubricaPointer: Rubrica</li> <li>- contactPointer: Contatto</li> <li>- typeController: Boolean</li> <li>- tablePointer: TableView&lt;Contatto&gt;</li> </ul>
<ul style="list-style-type: none"> <li>+ initialize(URL, ResourceBundle) : void</li> <li>+ setController(Rubrica): void</li> <li>+ setController(Contatto, Rubrica, TableView&lt;Contatto&gt;): void</li> <li>- disableModify(boolean): void</li> <li>- modify(ActionEvent): void</li> <li>- goBack(ActionEvent): void</li> <li>- confirm(ActionEvent): void</li> <li>- confAdd(): void</li> <li>- confMod(): void</li> <li>- numberControl(String): boolean</li> <li>- mailControl(String): boolean</li> <li>- nominativeControl(String, String): boolean</li> </ul>

Il livello di coesione per ContattoController è comunicazionale perché include funzionalità che lavorano sugli stessi dati (singolo contatto)

 Contatto
<ul style="list-style-type: none"> <li>- nome : String</li> <li>- cognome : String</li> <li>- numeri : String[]</li> <li>- emails : String[]</li> </ul>
<ul style="list-style-type: none"> <li>+ Contatto(): void</li> <li>+ setNome(String): void</li> <li>+ setCognome(String): void</li> <li>+ setNumero1(String): void</li> <li>+ setNumero2(String): void</li> <li>+ setEmail1(String): void</li> <li>+ setEmail2(String): void</li> <li>+ setNumero3(String): void</li> <li>+ setEmail3(String): void</li> <li>+ getNome(): String</li> <li>+ getCognome(): String</li> <li>+ getNumeri(): String[]</li> <li>+ getEmails(): String[]</li> <li>+ equals(Object): boolean</li> <li>+ hashCode(): int</li> <li>+ compareTo(Contatto): int</li> </ul>

La classe Contatto ha un livello di coesione funzionale poiché si focalizza sulla gestione dei dati del singolo contatto

 Avviso
<ul style="list-style-type: none"> <li>+ errore (String, String, String): void</li> <li>+ info (String, String, String): void</li> <li>+ conferma (String, String, String): boolean</li> </ul>

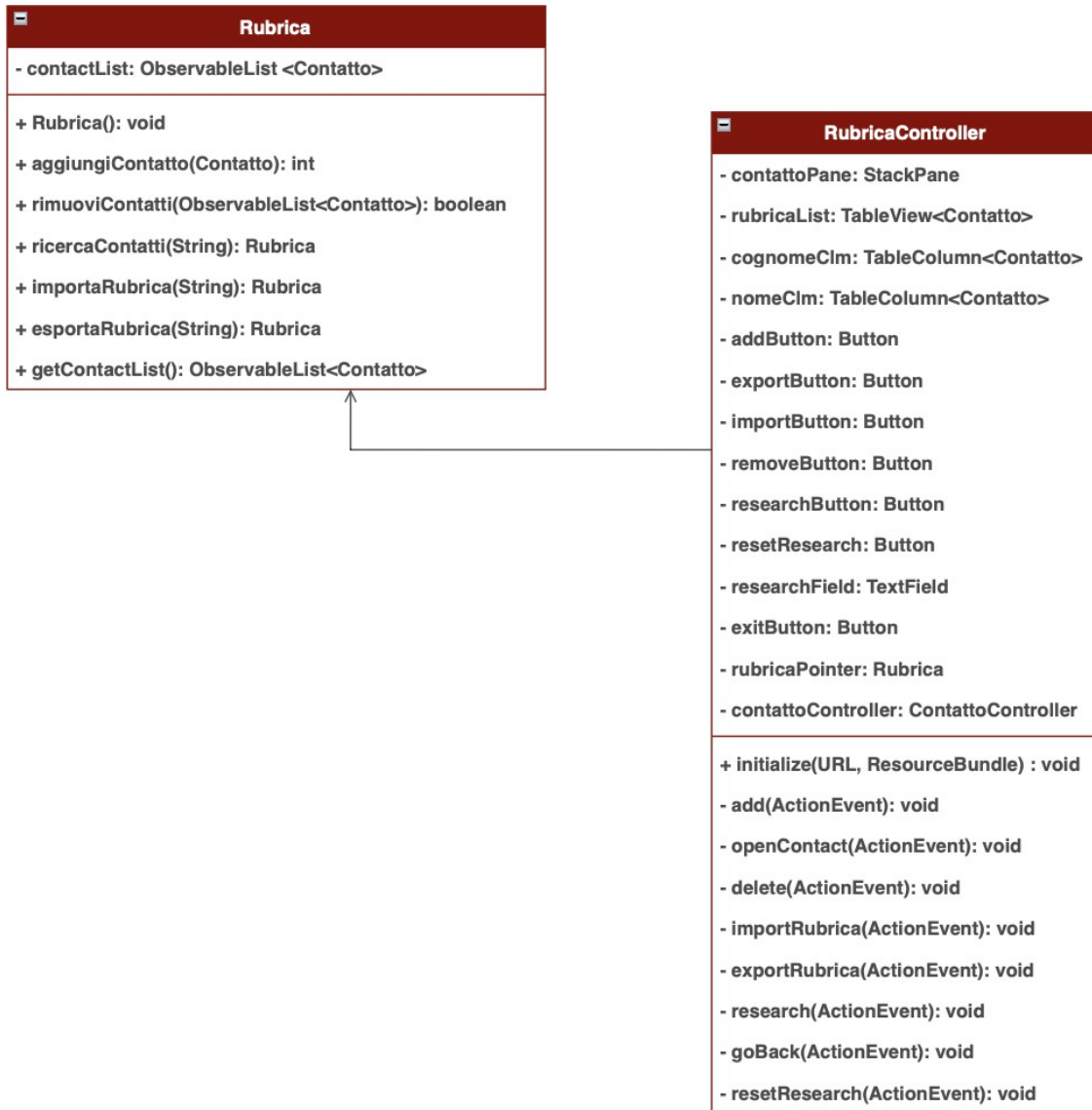
La coesione è logica poiché i metodi, pur essendo correlati tra loro in quanto si occupano tutti della gestione di pop-up per l'interfaccia utente, non condividono dati comuni né dipendono l'uno dall'altro. I metodi sono raggruppati in base al tipo di operazione che svolgono, mantenendo un legame concettuale

senza essere direttamente interconnessi.

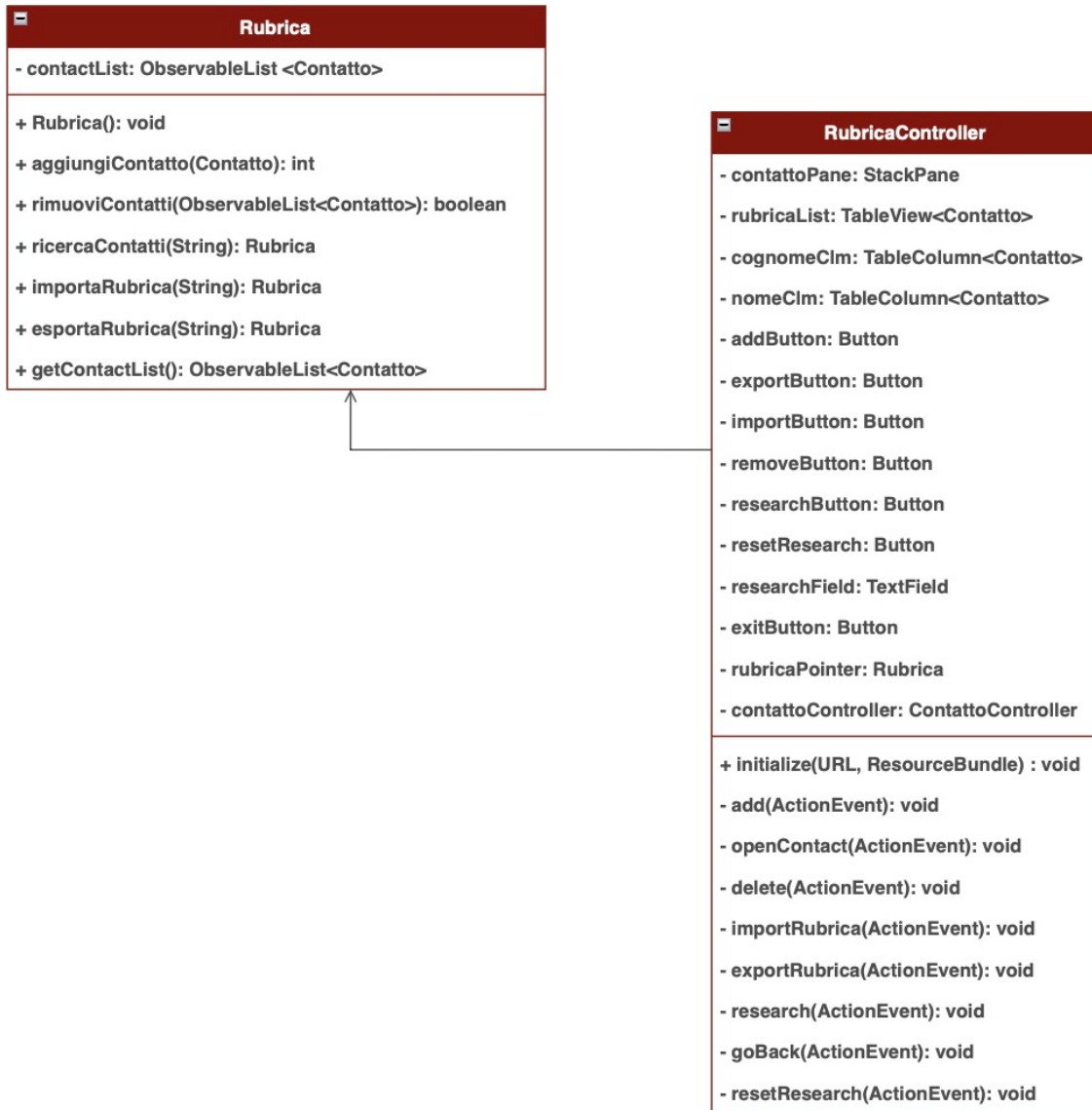
## Accoppiamento



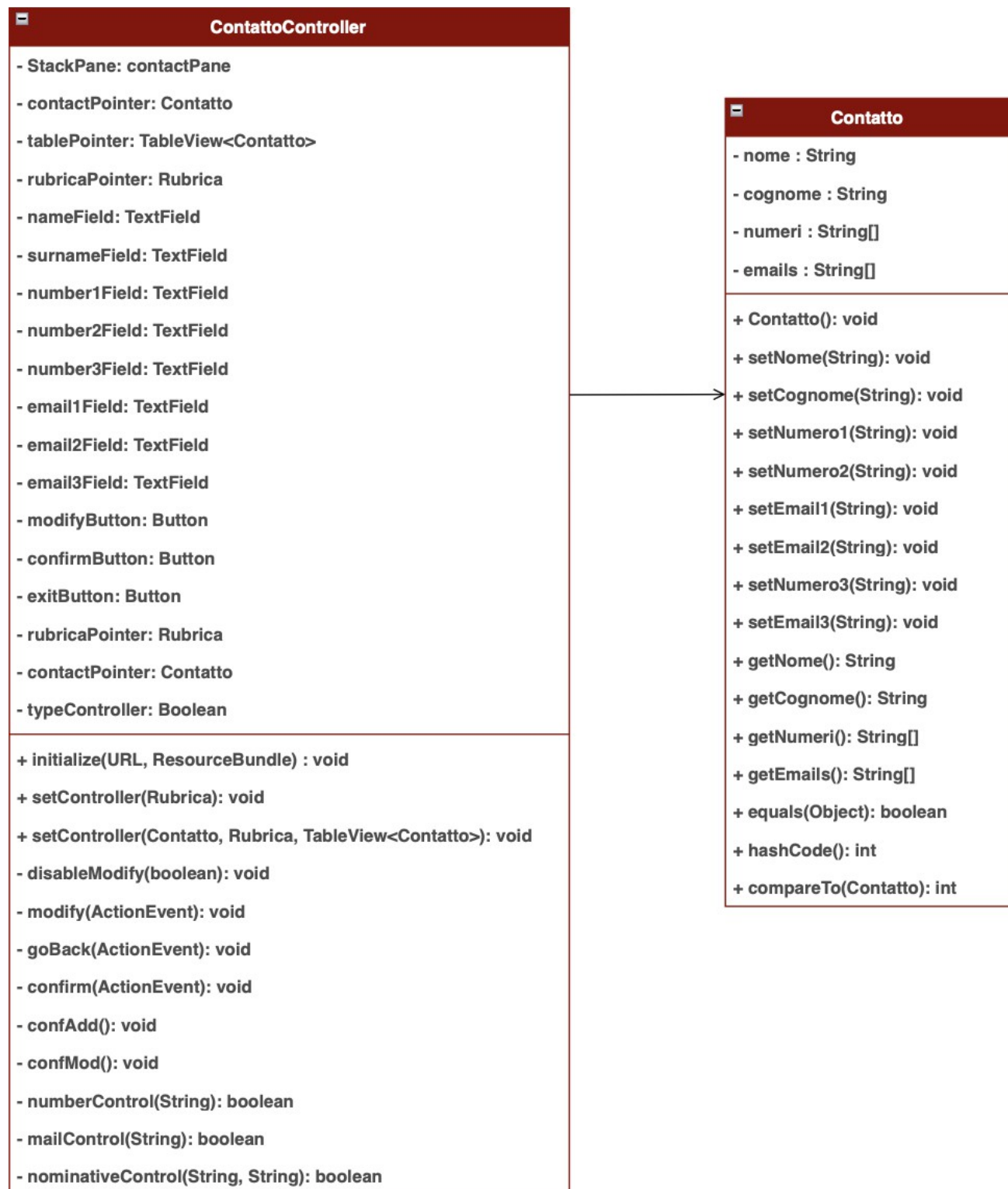
L'accoppiamento tra **Rubrica** e **Contatto** è **accoppiamento per dati**, poiché la comunicazione tra le due classi avviene passando solo le informazioni necessarie tramite metodi pubblici.



L'accoppiamento tra RubricaController e Rubrica è di tipo **controllo**, perché RubricaController passa richieste che determinano come Rubrica debba comportarsi, influenzandone la logica operativa.

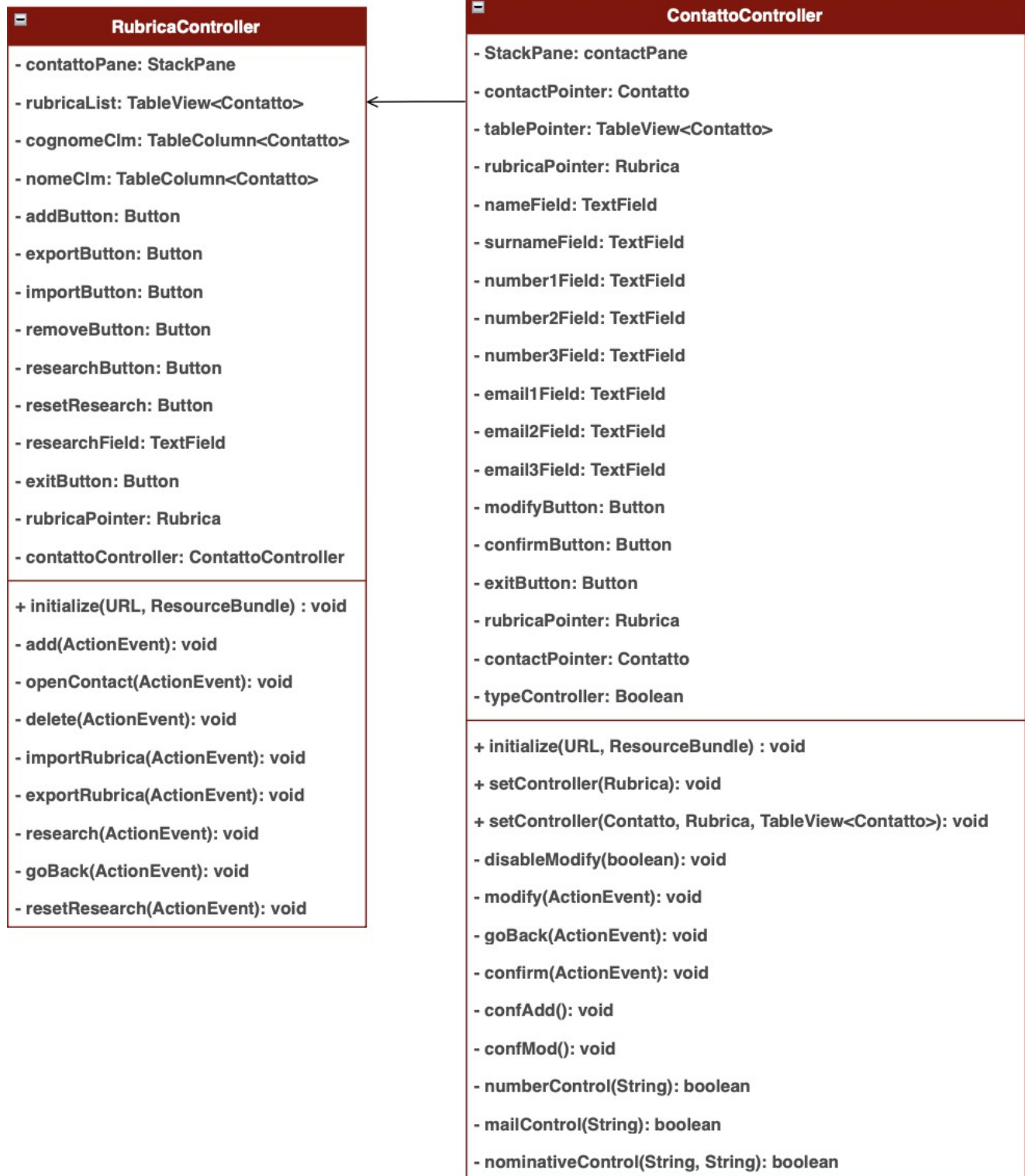


L'**accoppiamento** tra Rubrica e ContattoController è per **timbro**, in quanto il controller mantiene informazioni riguardo la totalità della rubrica, senza accedere però a tutte le sue funzioni (metodi di esportazione/importazione e di ricerca non vengono usati, ovvero sono informazioni extra che il controller possiede).



La classe Contatto si relaziona con la classe ContattoController unicamente tramite metodi pubblici che restituiscono singole informazioni: pertanto preserva un tipo di accoppiamento sui dati.

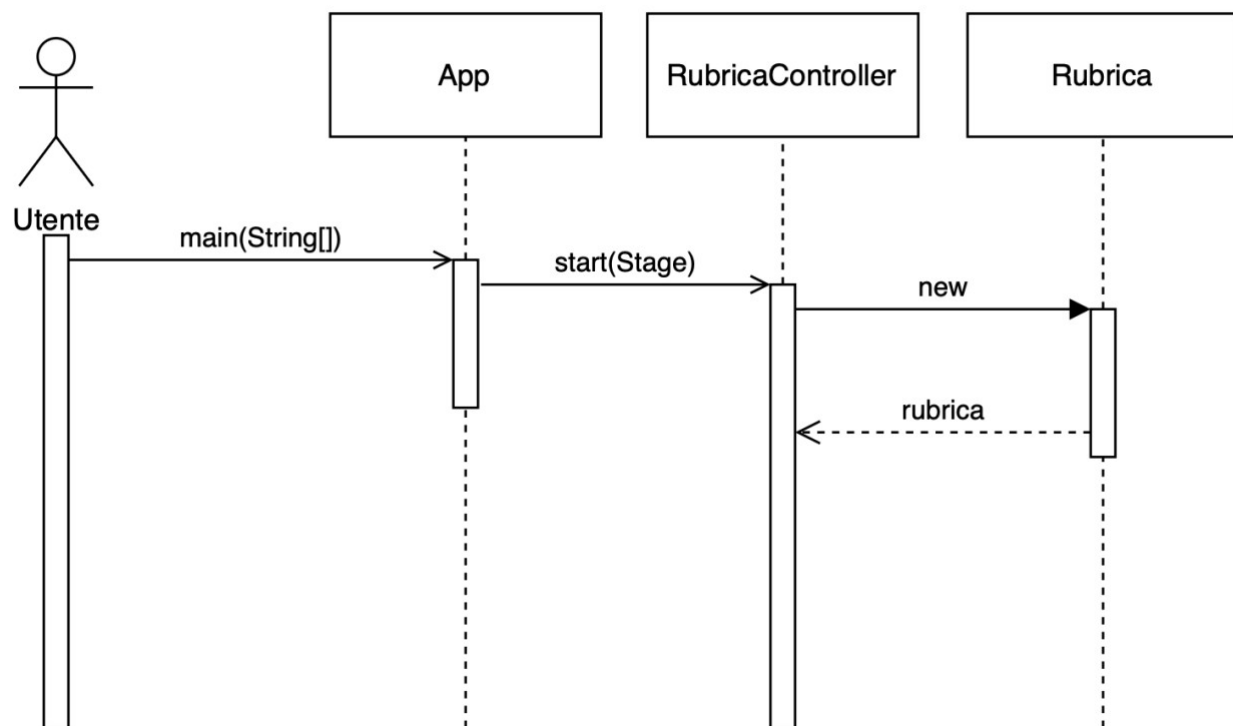




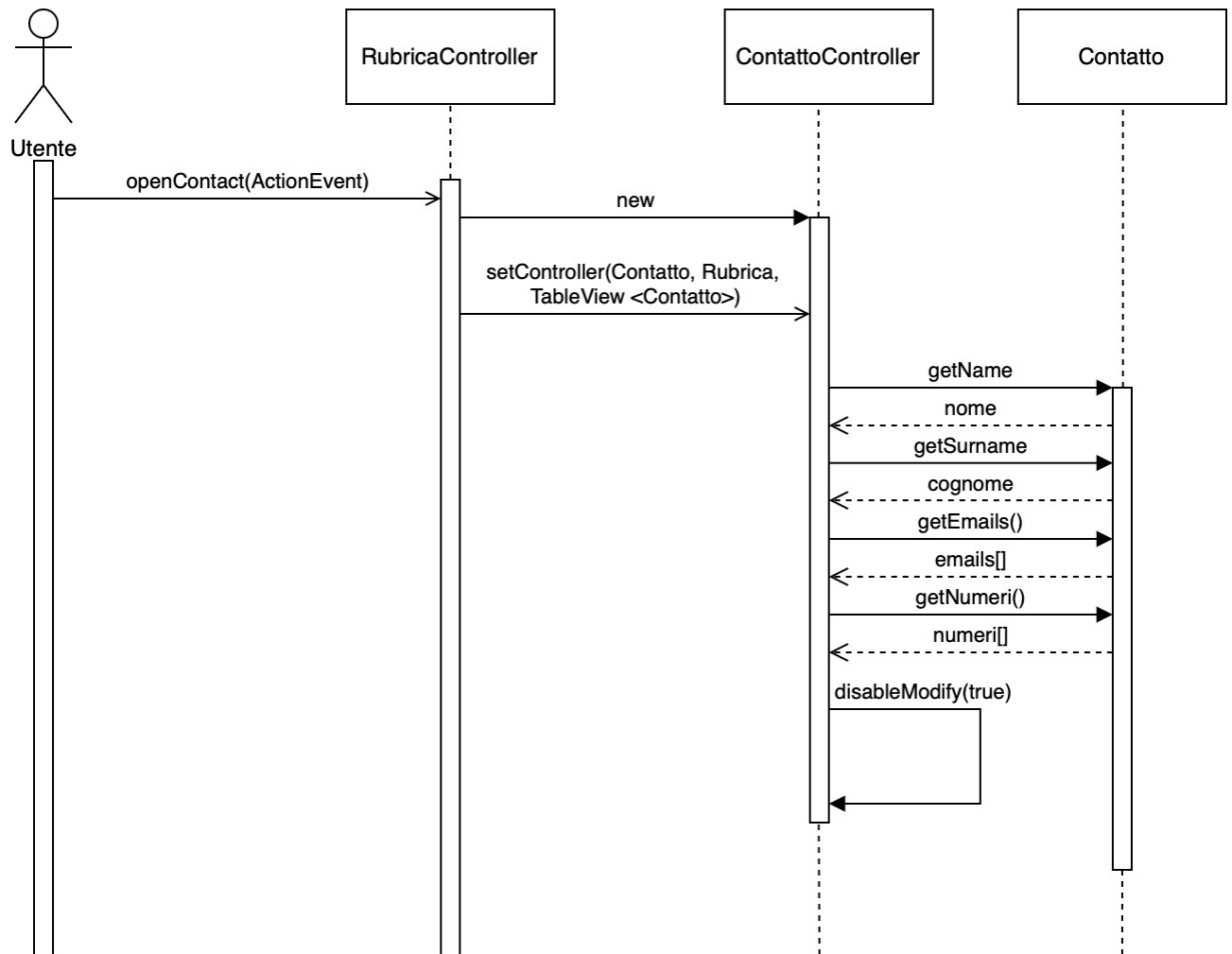
L'accoppiamento tra ContattoController e RubricaController è **basato sui dati**. ContattoController interagisce con i dati gestiti da RubricaController tramite i metodi setController, senza dipendere direttamente dalla logica di RubricaController.

## Diagrammi di sequenza

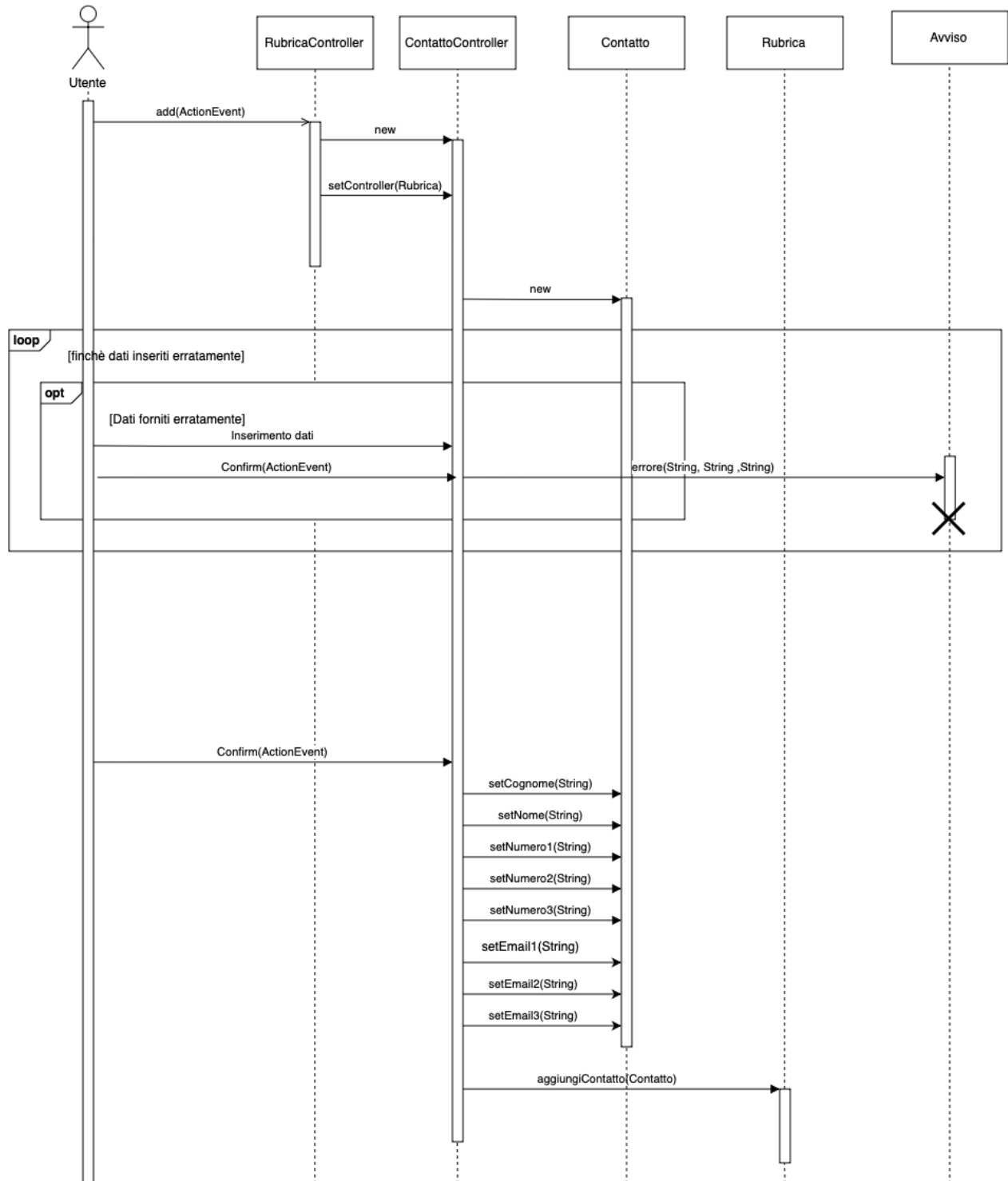
### VisualizzaRubrica



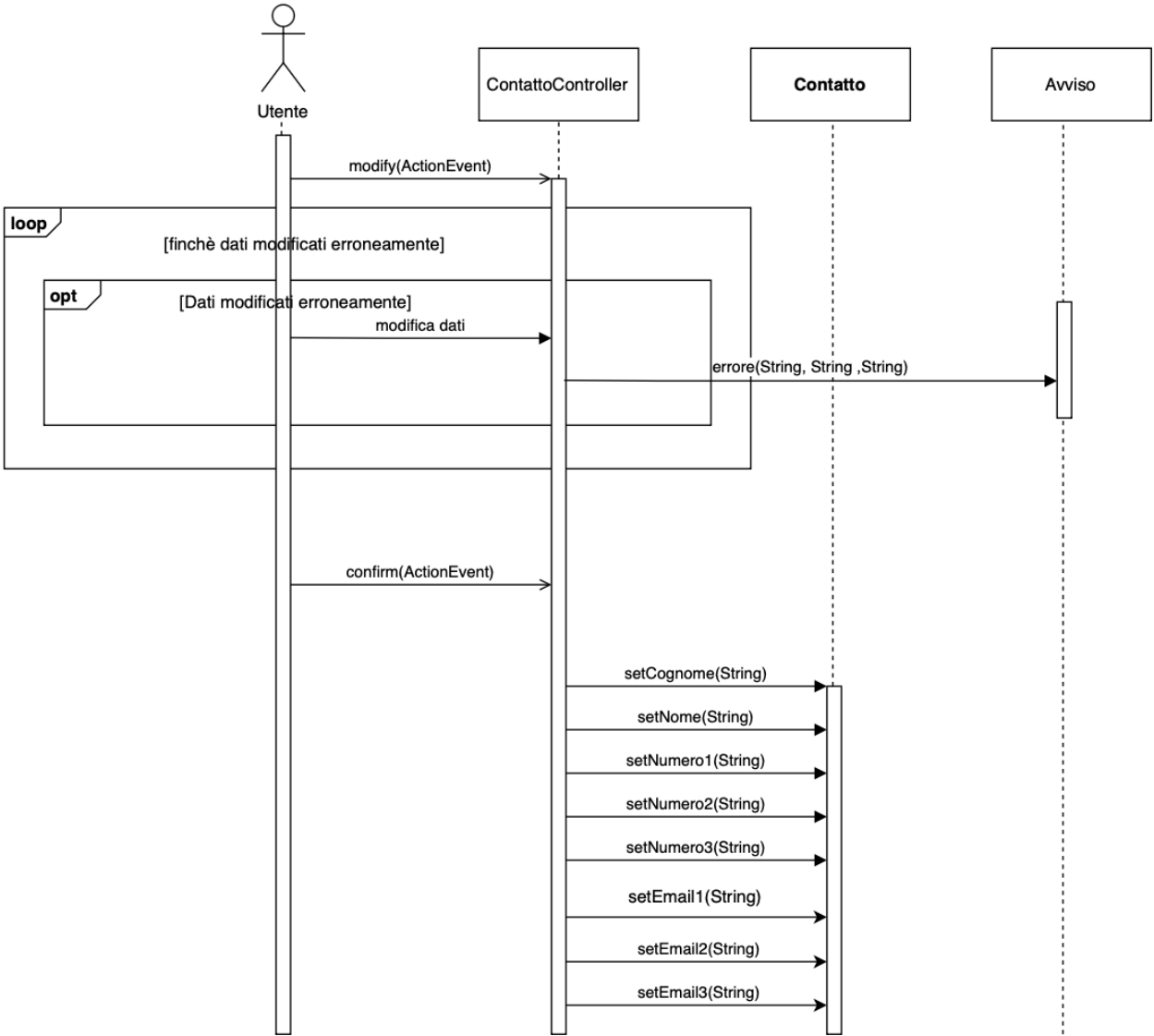
## AccessoContatto



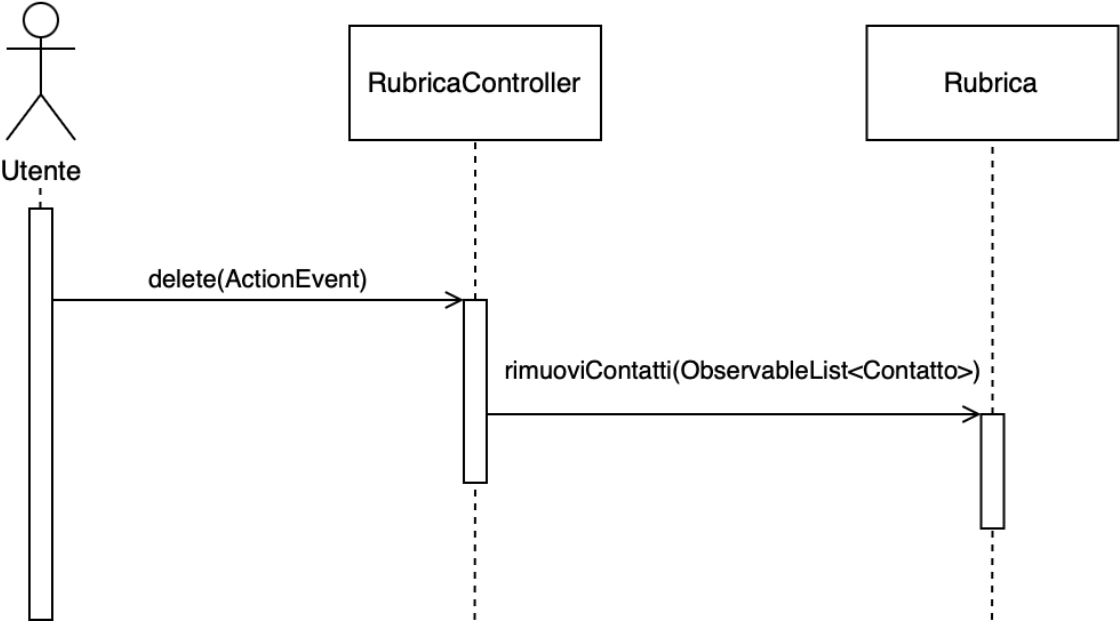
## AggiuntaContatto



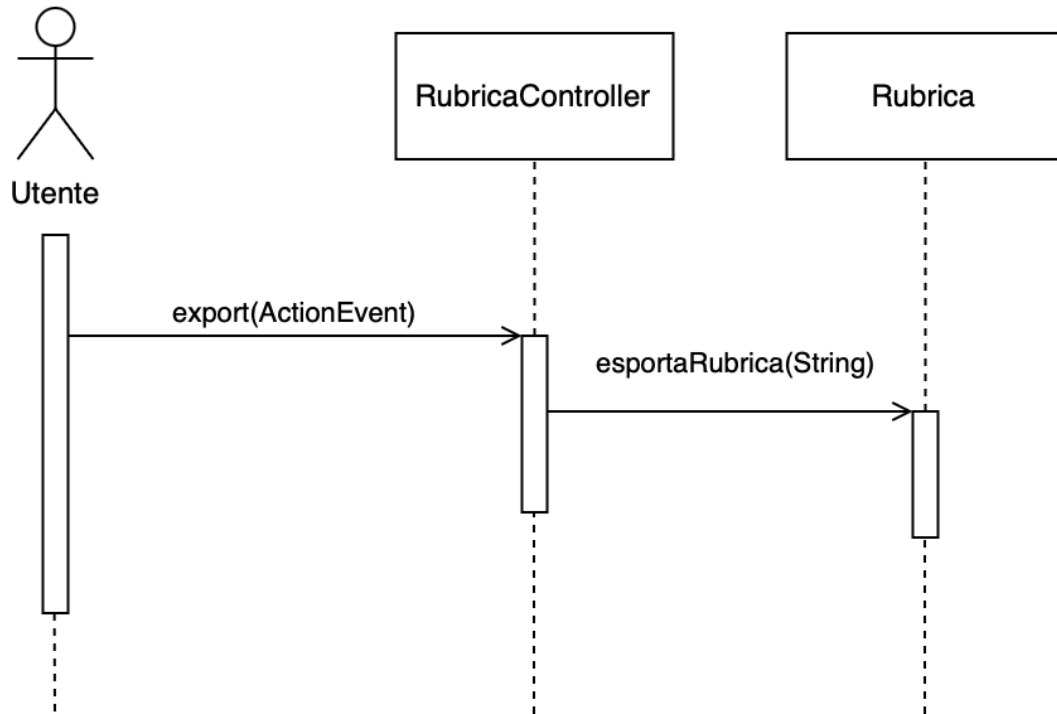
# ModificaContatto



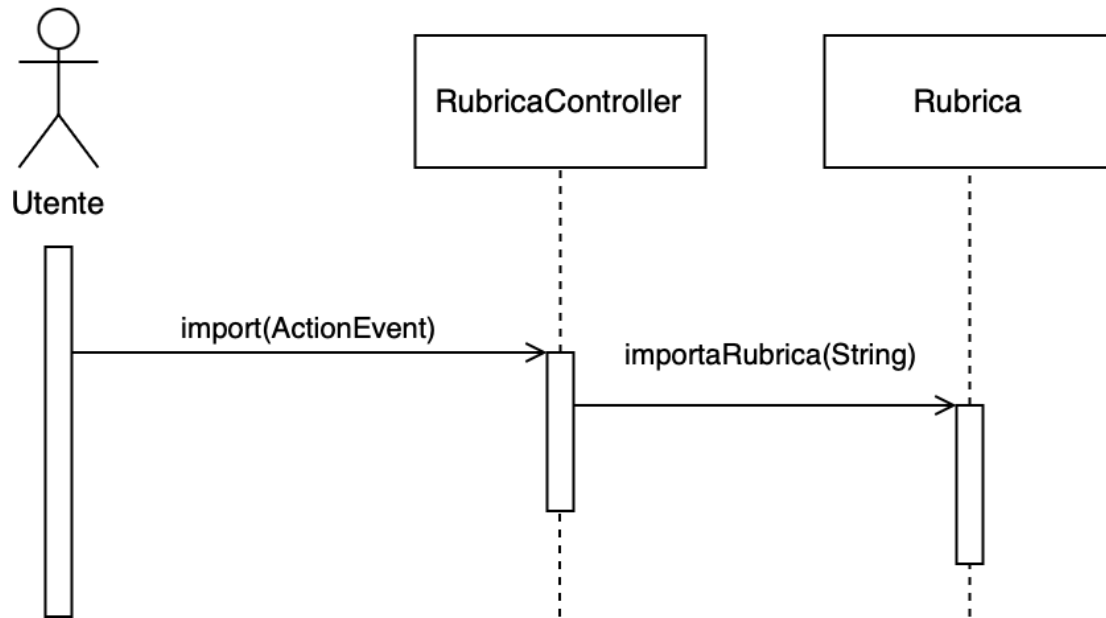
# RimuoviContatto



# EsportaRubrica

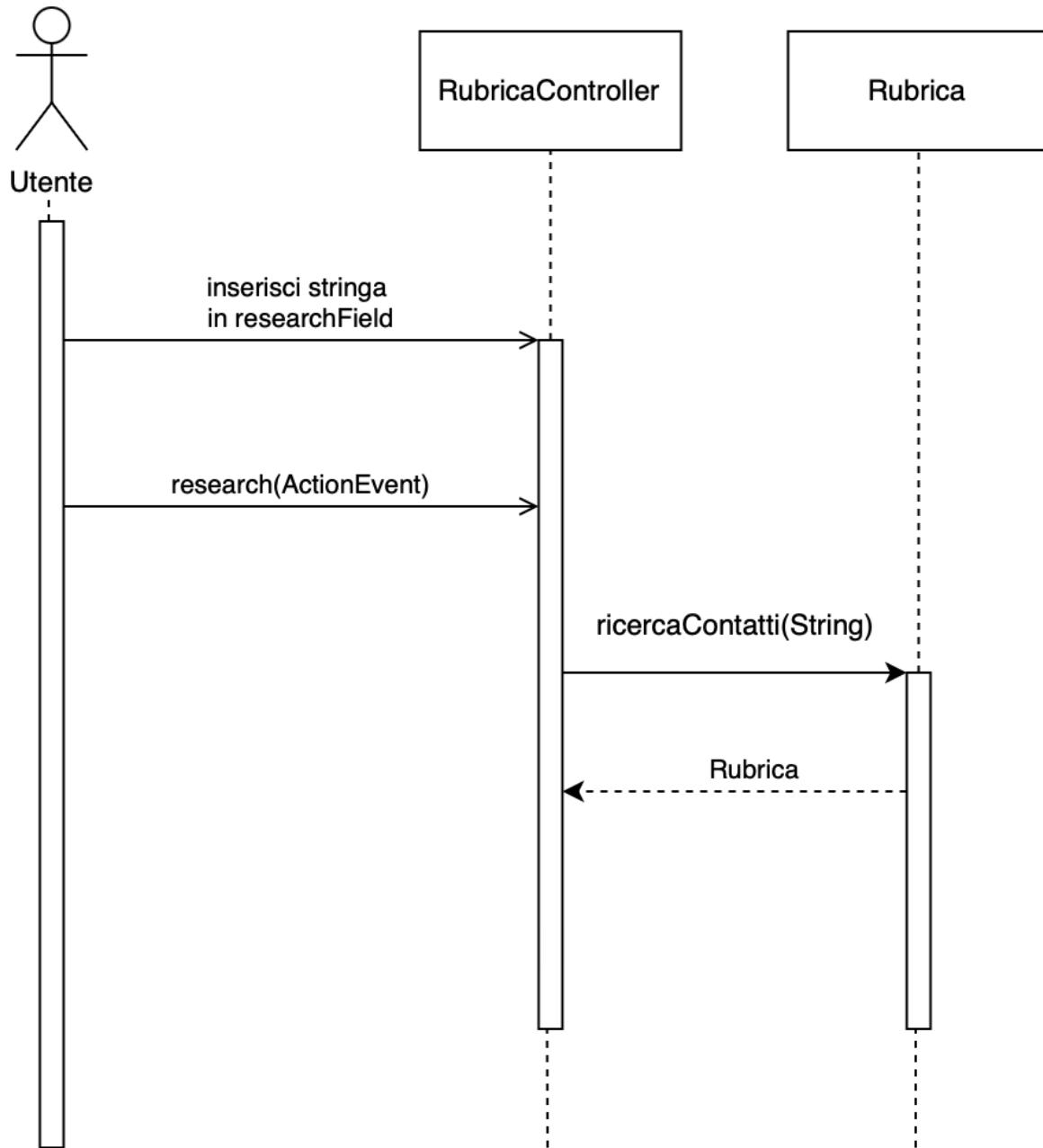


# ImportaRubrica





# RicercaContatto



## Diagramma dei package

