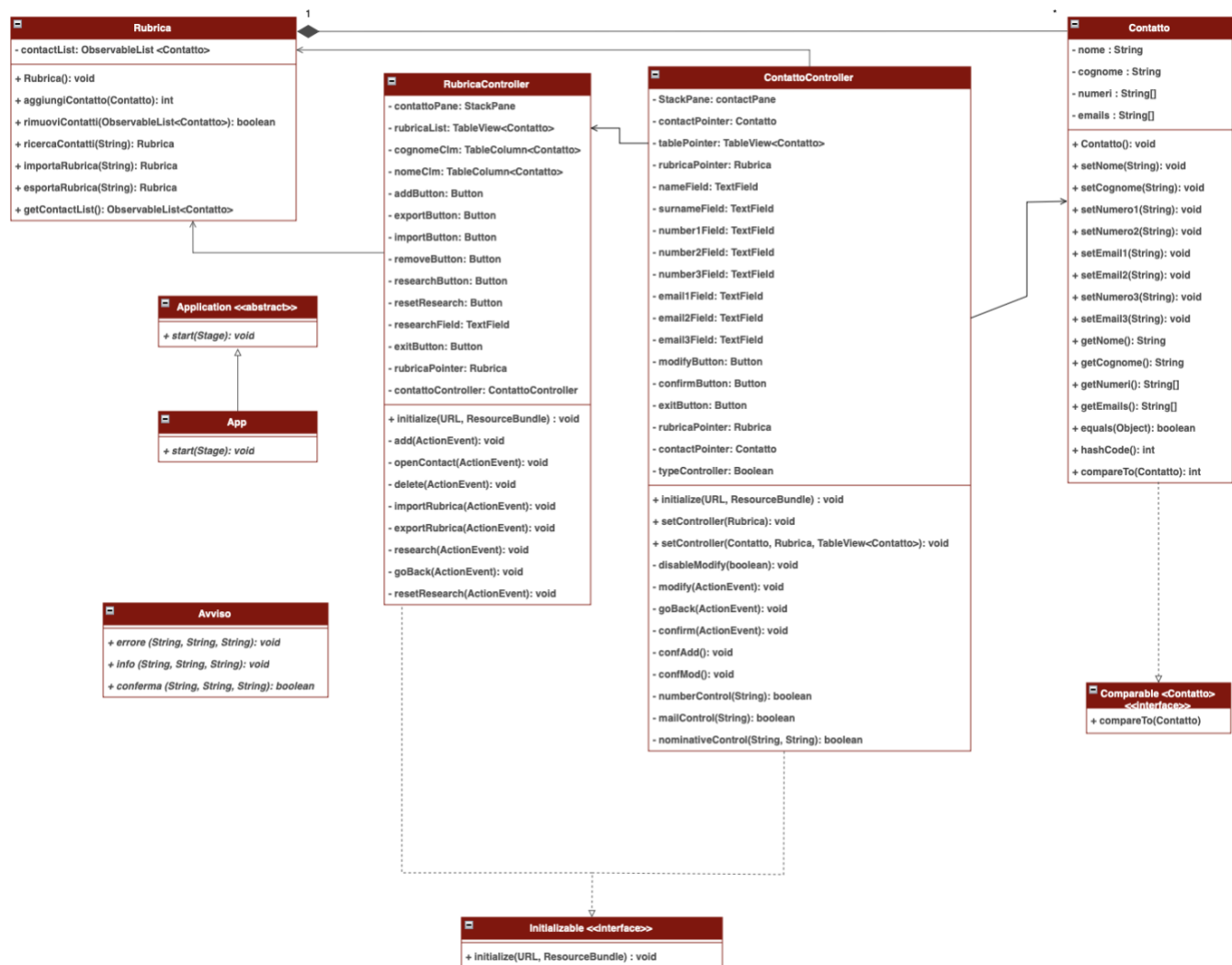


Documento Design

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

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

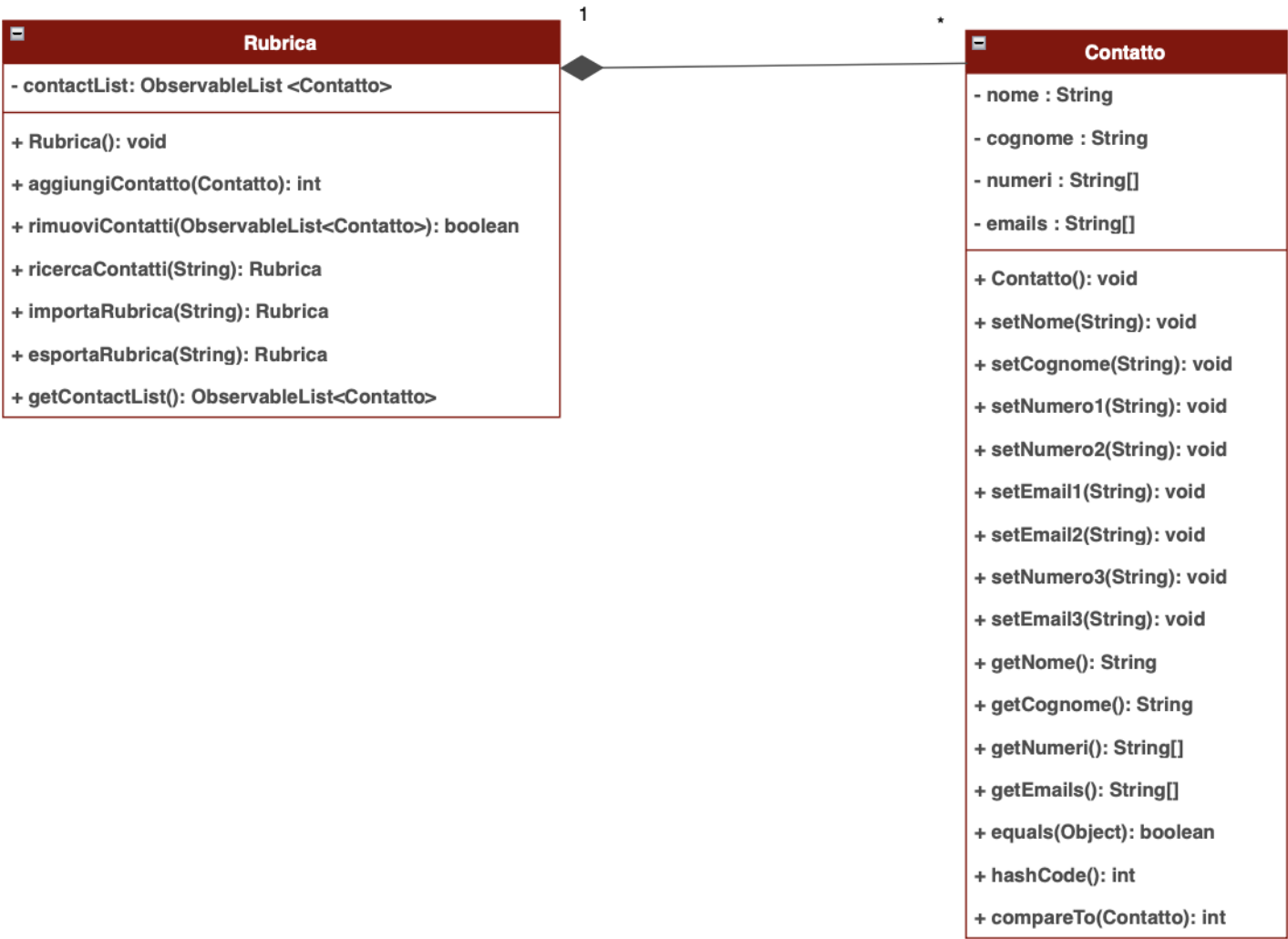
La classe Contatto ha un livello di coesione funzionale poiché si focalizza sulla gestione dei dati del singolo contatto

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

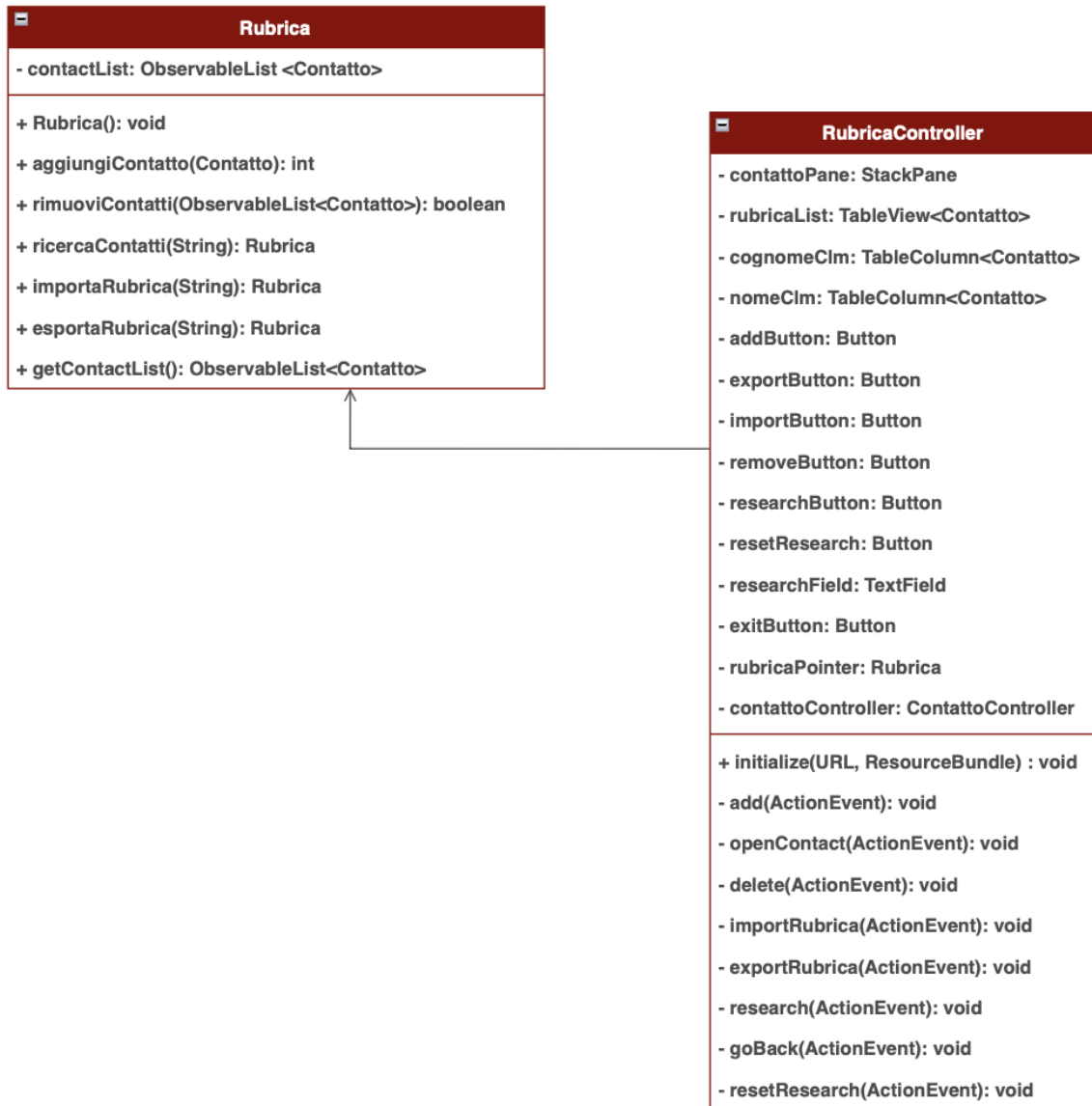
La coesione è logica poiché i metodi, pur essendo correlati tra loro in quanto si occupano tutti della gestione di pop-up per l'interfaccia utente, non condividono dati comuni né dipendono l'uno dall'altro. I metodi sono raggruppati in base al tipo di operazione che svolgono, mantenendo un legame concettuale

senza essere direttamente interconnessi.

Accoppiamento



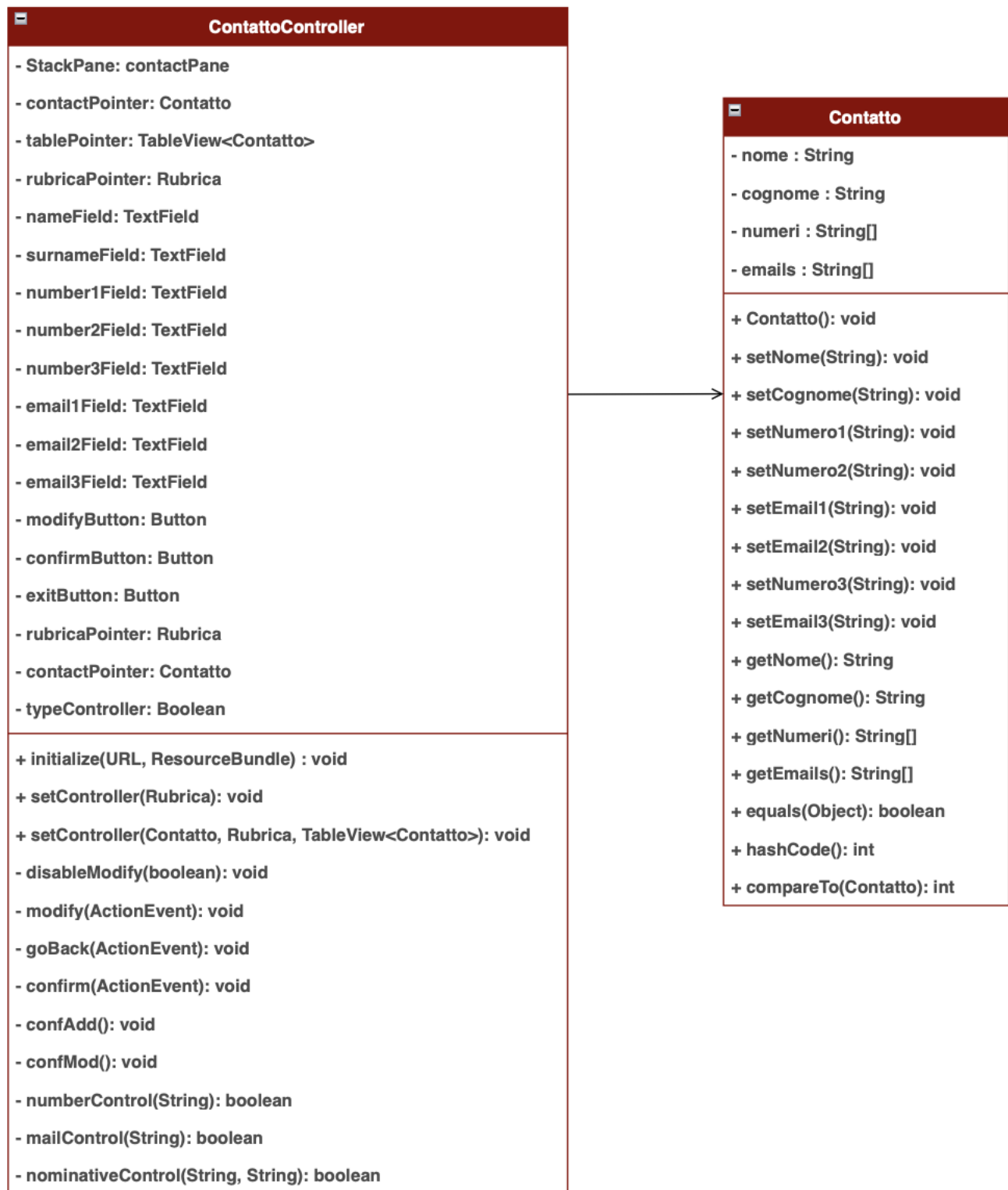
L'accoppiamento tra Rubrica e Contatto è **accoppiamento per dati**, poiché la comunicazione tra le due classi avviene passando solo le informazioni necessarie tramite metodi pubblici.



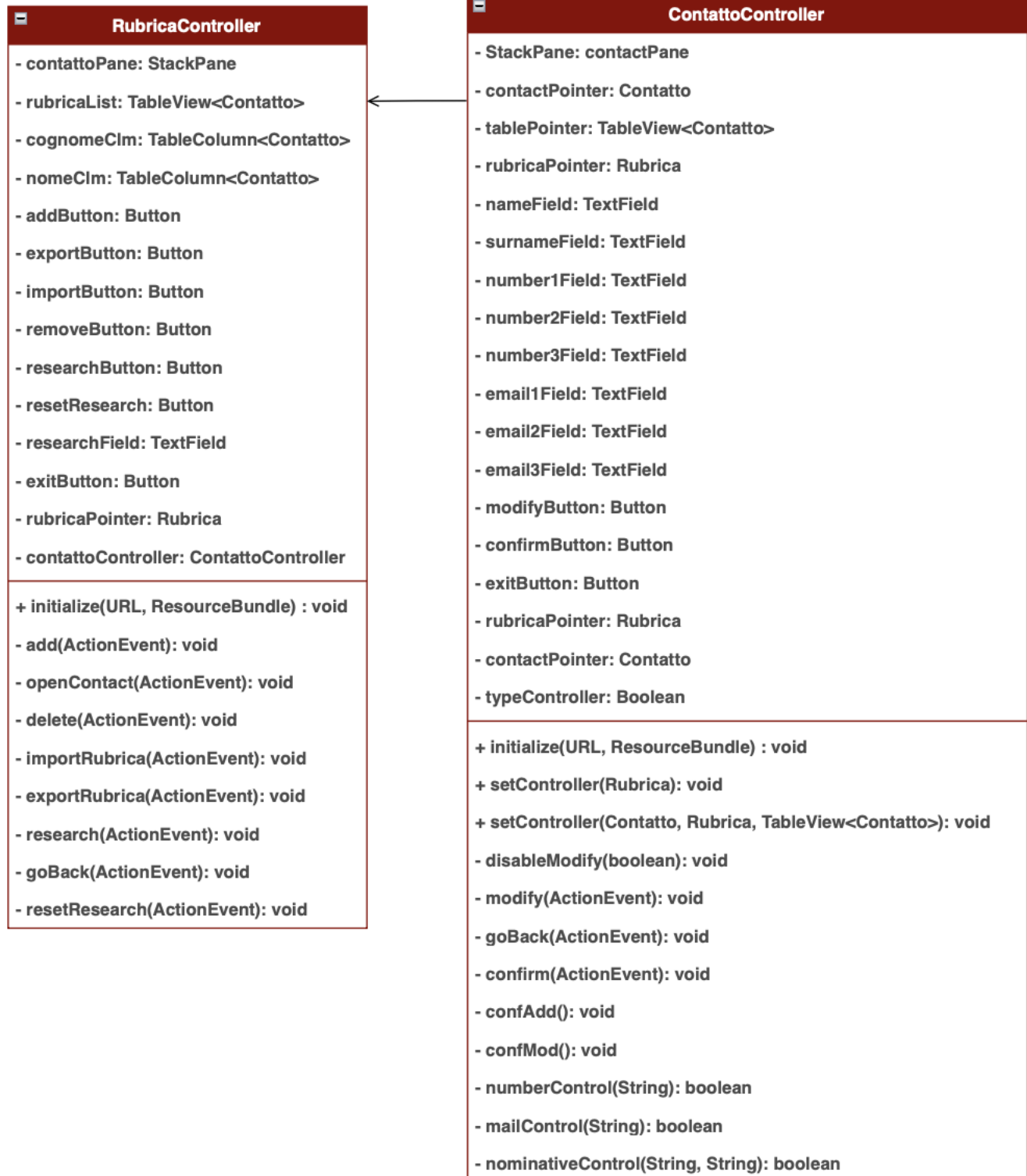
L'**accoppiamento** tra **RubricaController** e **Rubrica** è di tipo **controllo**, perché **RubricaController** passa richieste che determinano come **Rubrica** debba comportarsi, influenzandone la logica operativa.



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).



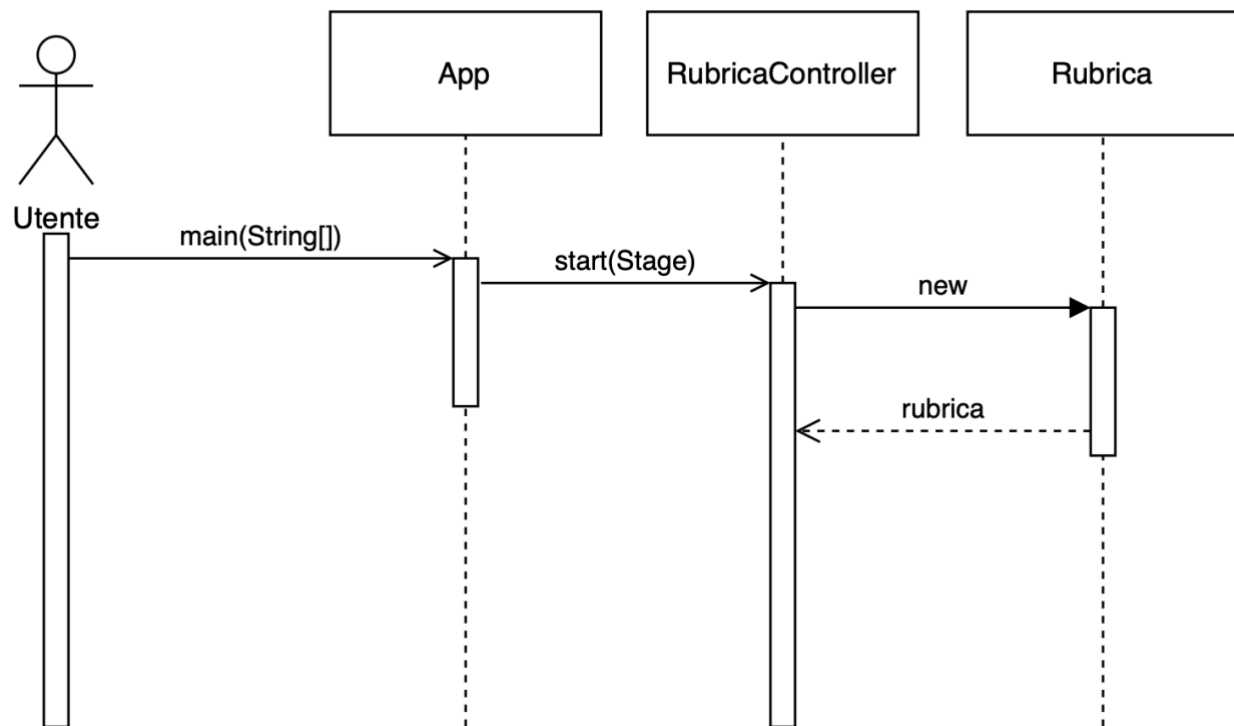
La classe Contatto si relaziona con la classe ContattoController unicamente tramite metodi pubblici che restituiscono singole informazioni: pertanto preserva un tipo di **accoppiamento sui dati**.



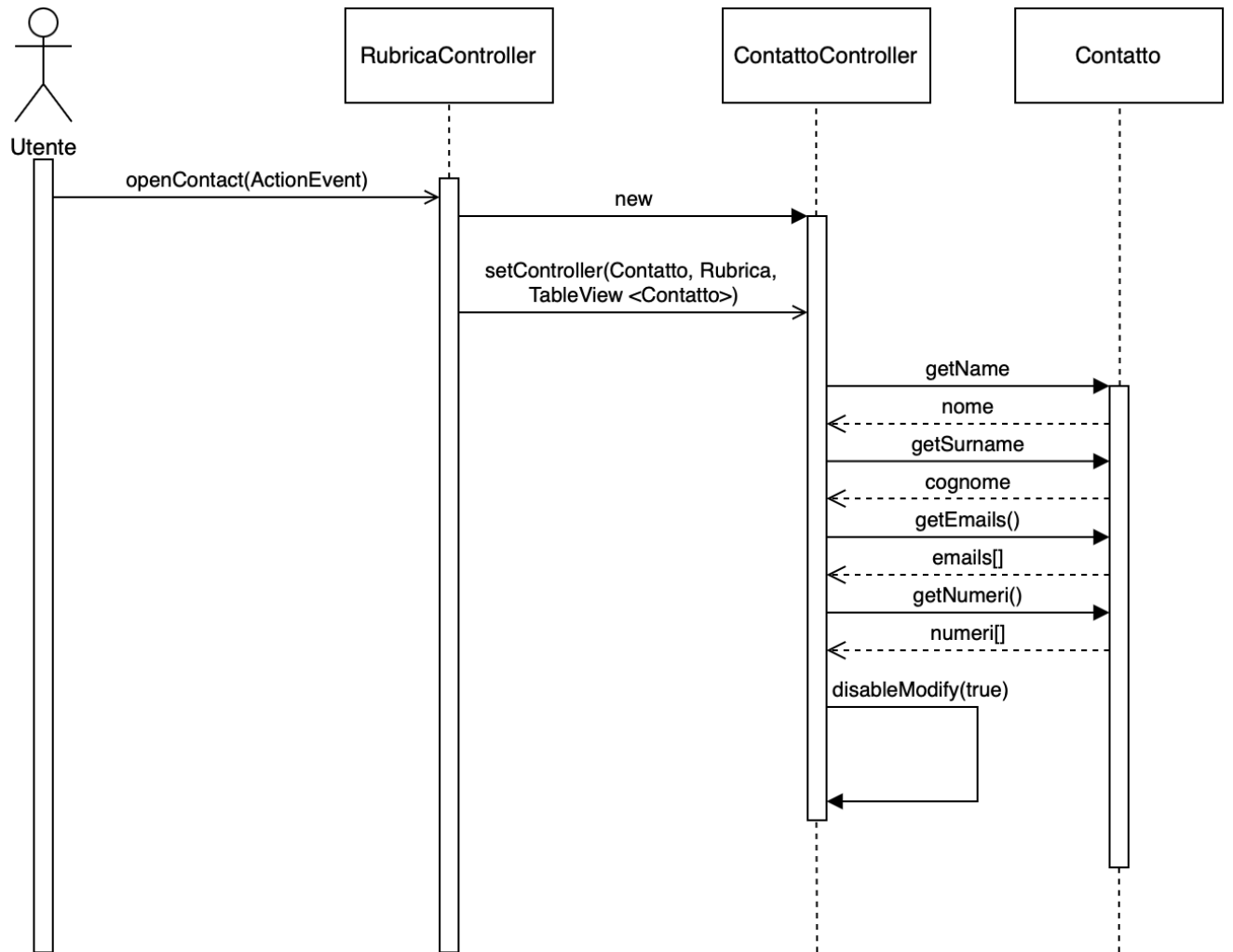
L'**accoppiamento** tra ContattoController e RubricaController è **basato sui dati**. ContattoController interagisce con i dati gestiti da RubricaController tramite i metodi setController, senza dipendere direttamente dalla logica di RubricaController.

Diagrammi di sequenza

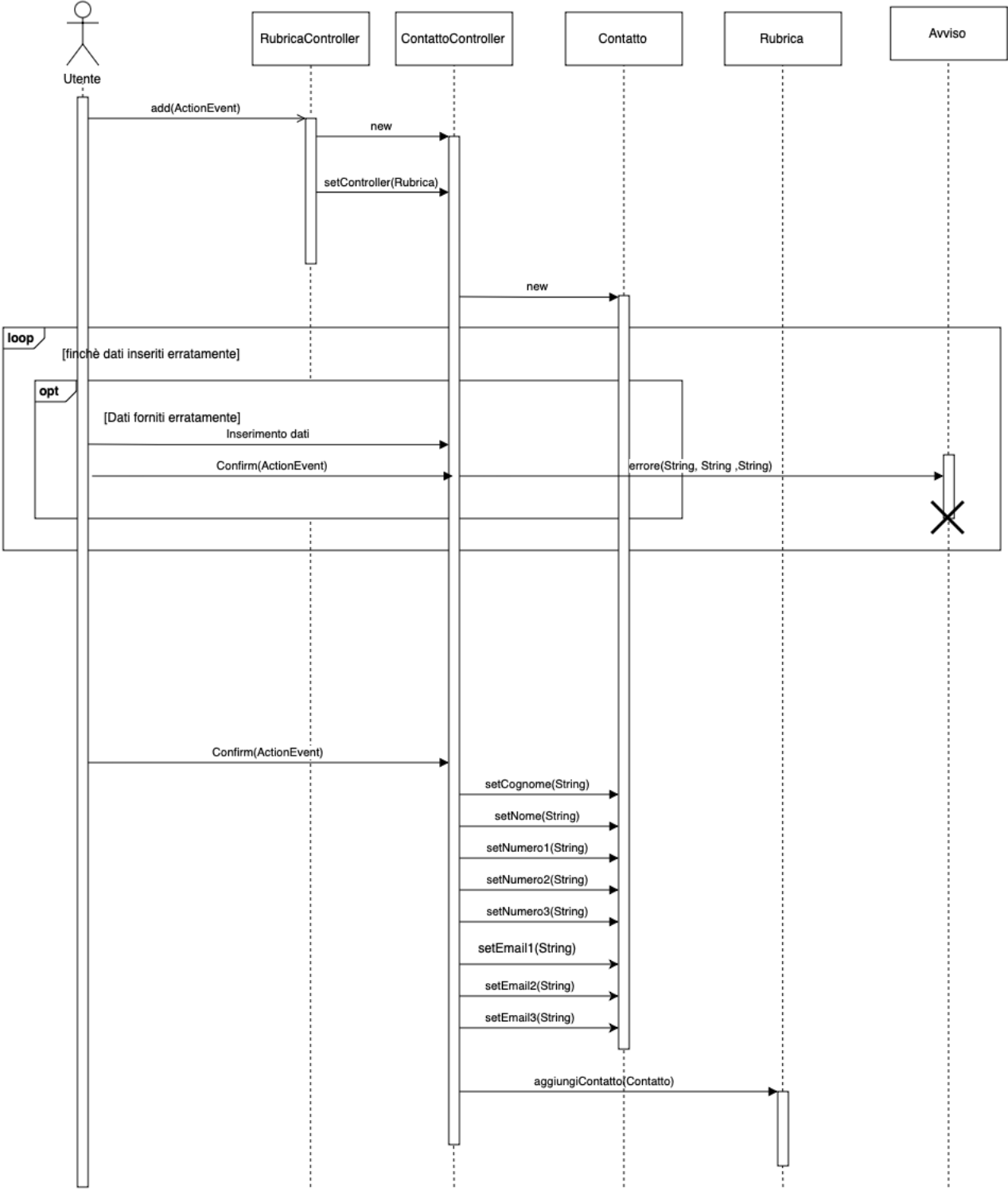
VisualizzaRubrica



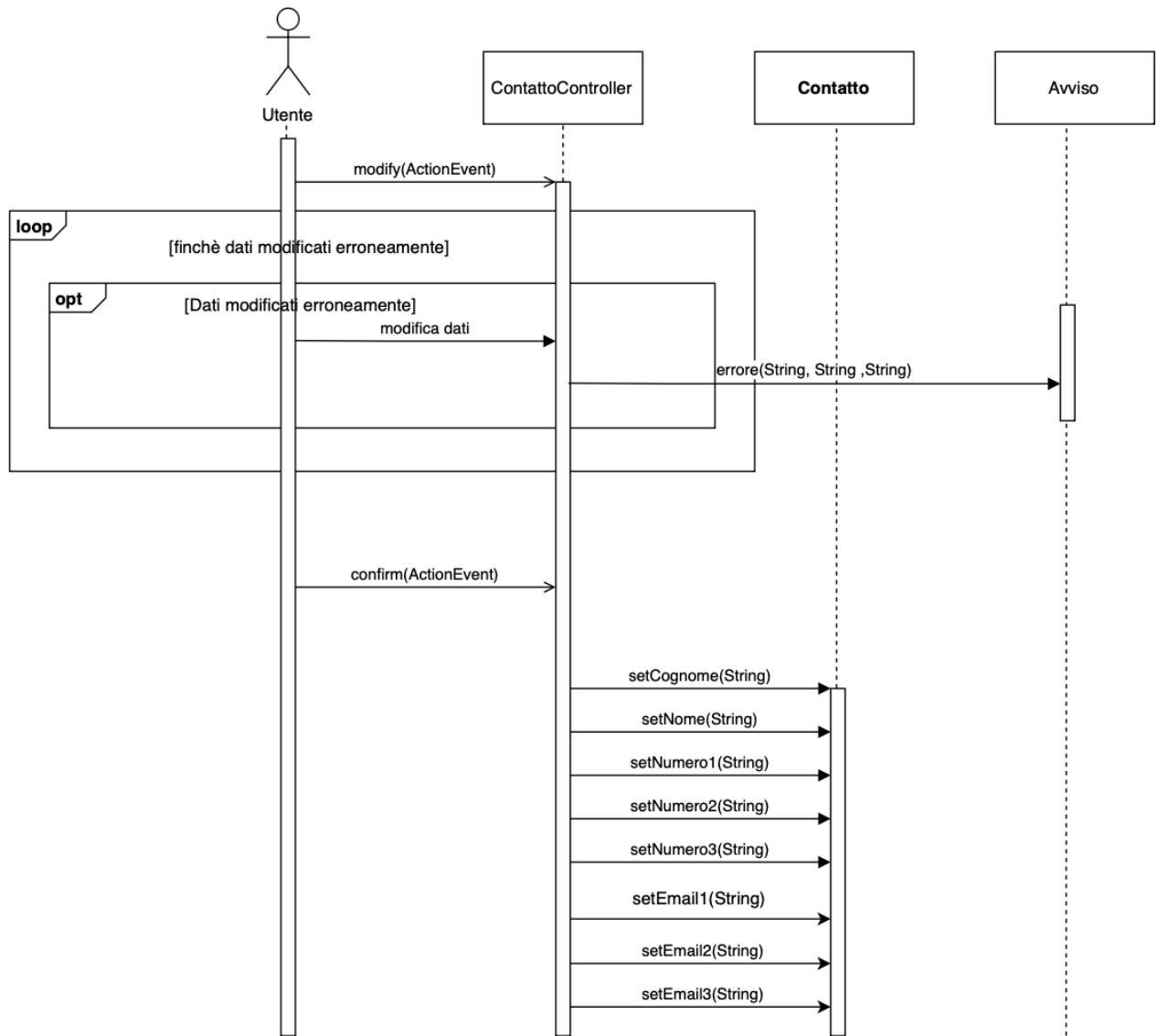
AccessoContatto



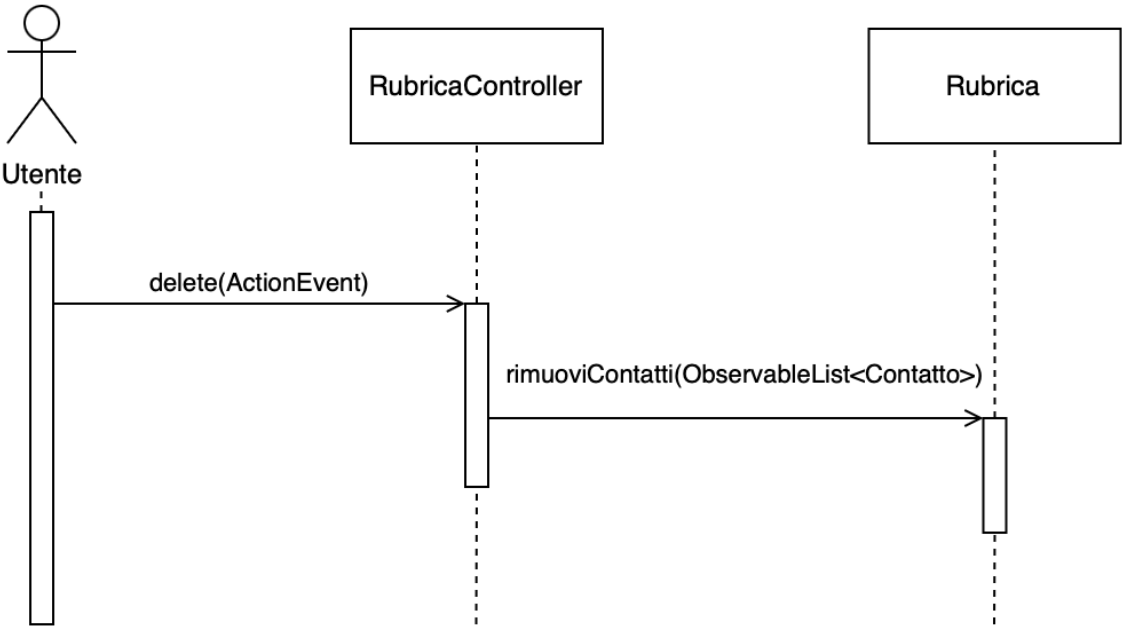
AggiuntaContatto



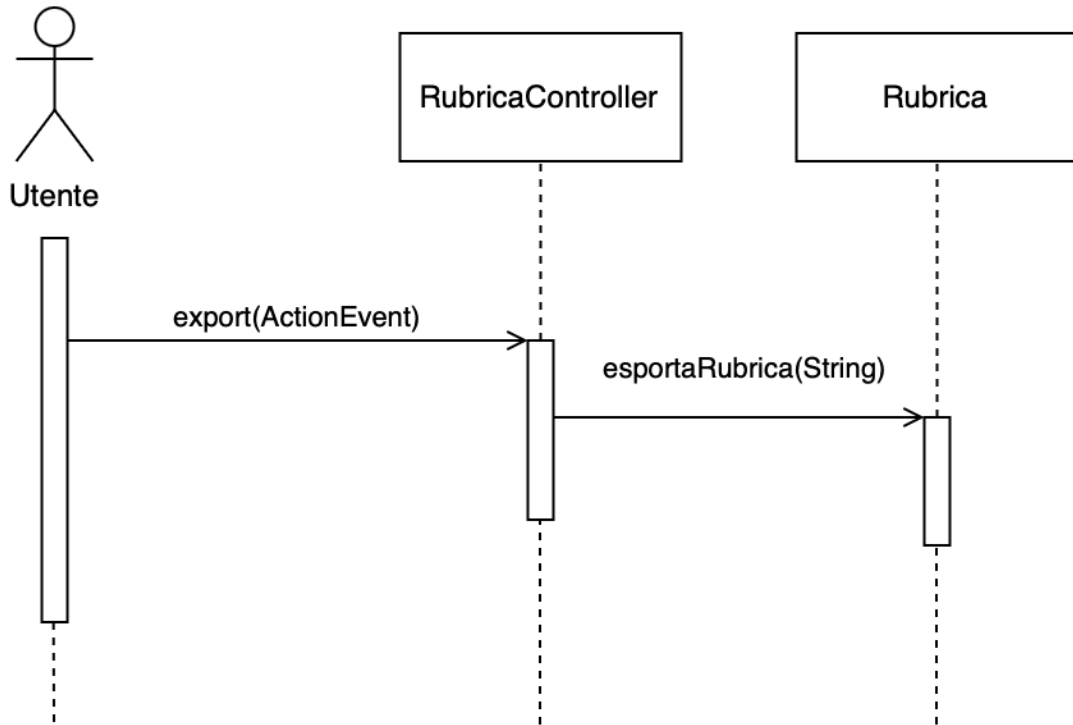
ModificaContatto



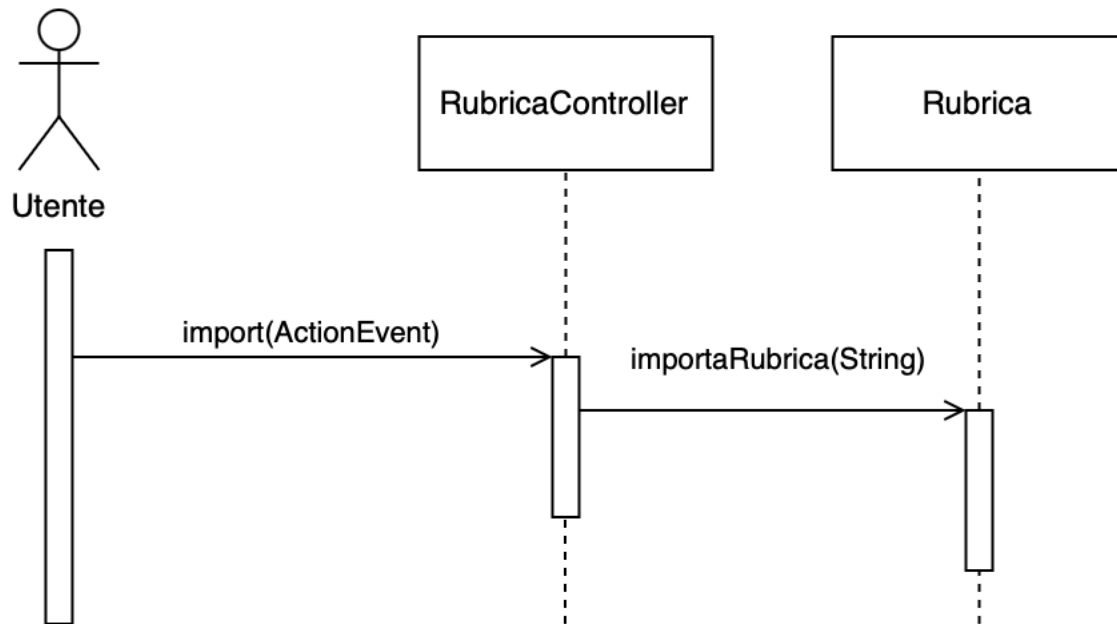
RimuoviContatto



EsportaRubrica



ImportaRubrica



RicercaContatto

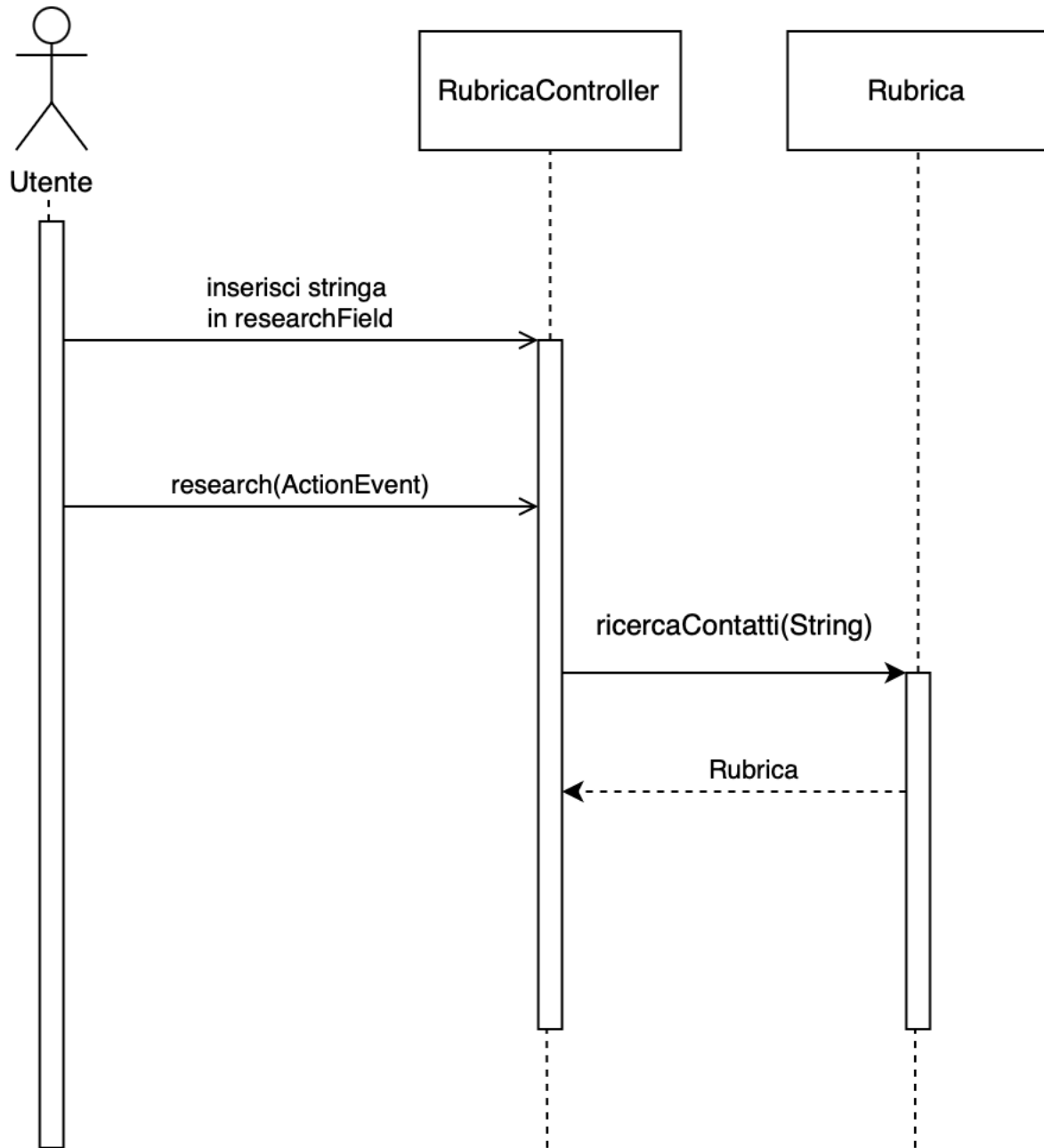


Diagramma dei package

