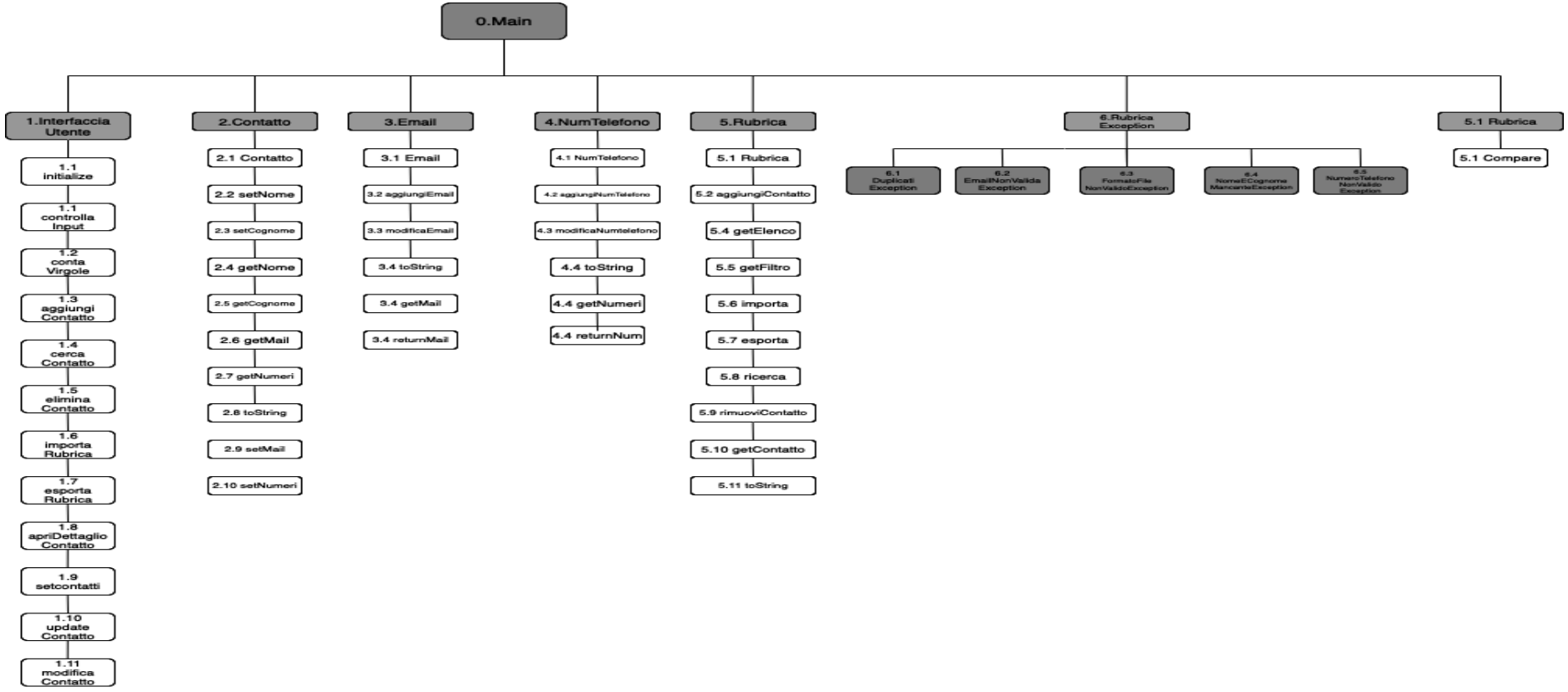
Design

1. Decomposizione in moduli

La decomposizione dei moduli ha l'obiettivo di semplificare l'implementazione del codice e migliorare la sua organizzazione, rendendolo più leggibile, efficiente e facilmente modificabile. Consiste nella suddivisione del modulo principale in sottomoduli, ciascuno dedicato a una funzione specifica. Il diagramma seguente illustra la struttura modulare del nostro progetto.
Per maggiore leggibilità visionare il PDF “DecomposizioneInModuli” presente nella cartella “Allegati”.



2. Interfaccia Utente

Qui vengono elencati alcuni esempi di come l'interfaccia si evolve in base alle diverse azioni che l'utente può compiere.
Sono stati considerati i casi generali, in particolare uno scenario con un tipo di operazione specifica per ognuno.

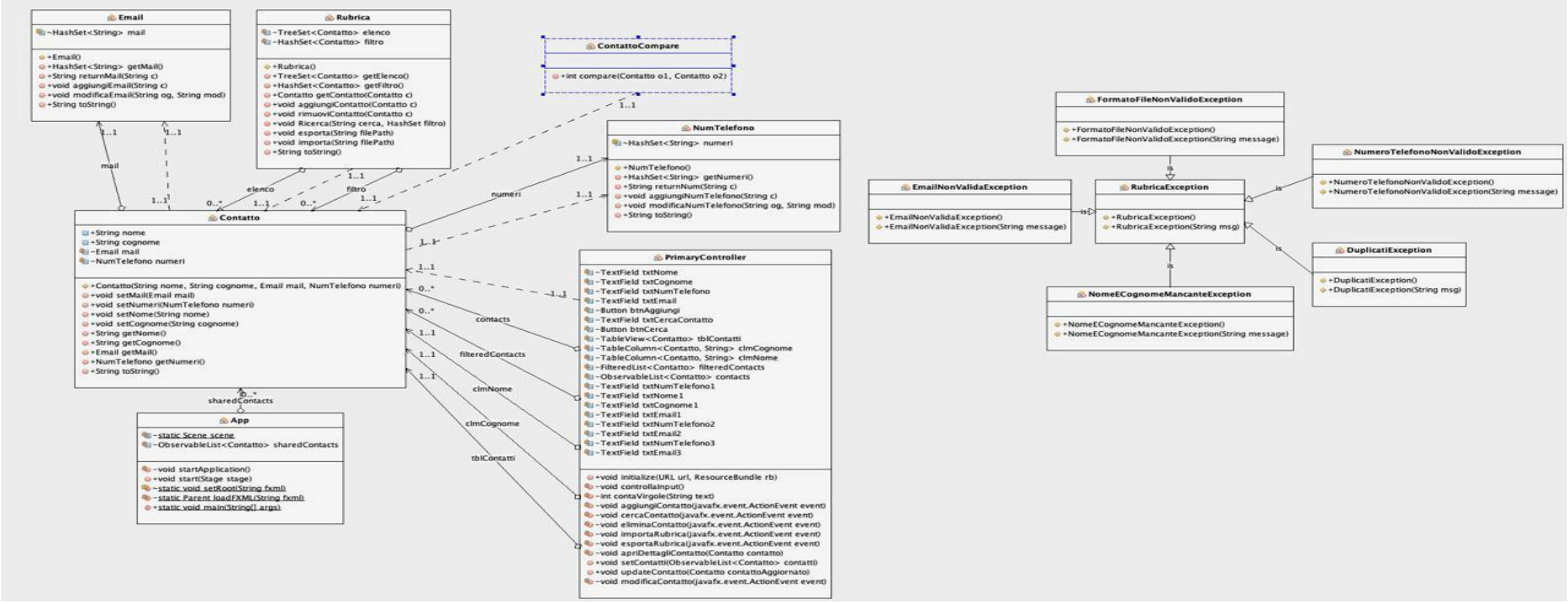
- I° Scenario: Creazione di un contatto
- II° Scenario: Modifica di un contatto
- III° Scenario: Esportazione della rubrica su un file
- IV° Scenario: Errore durante l'esportazione della rubrica su un file
- V° Scenario: Importazione della rubrica da un file
- VI° Scenario: Errore durante l'importazione della rubrica da un file
- VII° Scenario: Errore in caso di creazione di un contatto

3. Diagrammi UML

Class Diagram

Il seguente diagramma mostra la suddivisione del nostro sistema in “classi”, ognuna delle quali deve svolgere determinate operazioni. Il diagramma mostra come le classi si relazionano tra loro e eventuali dipendenze.

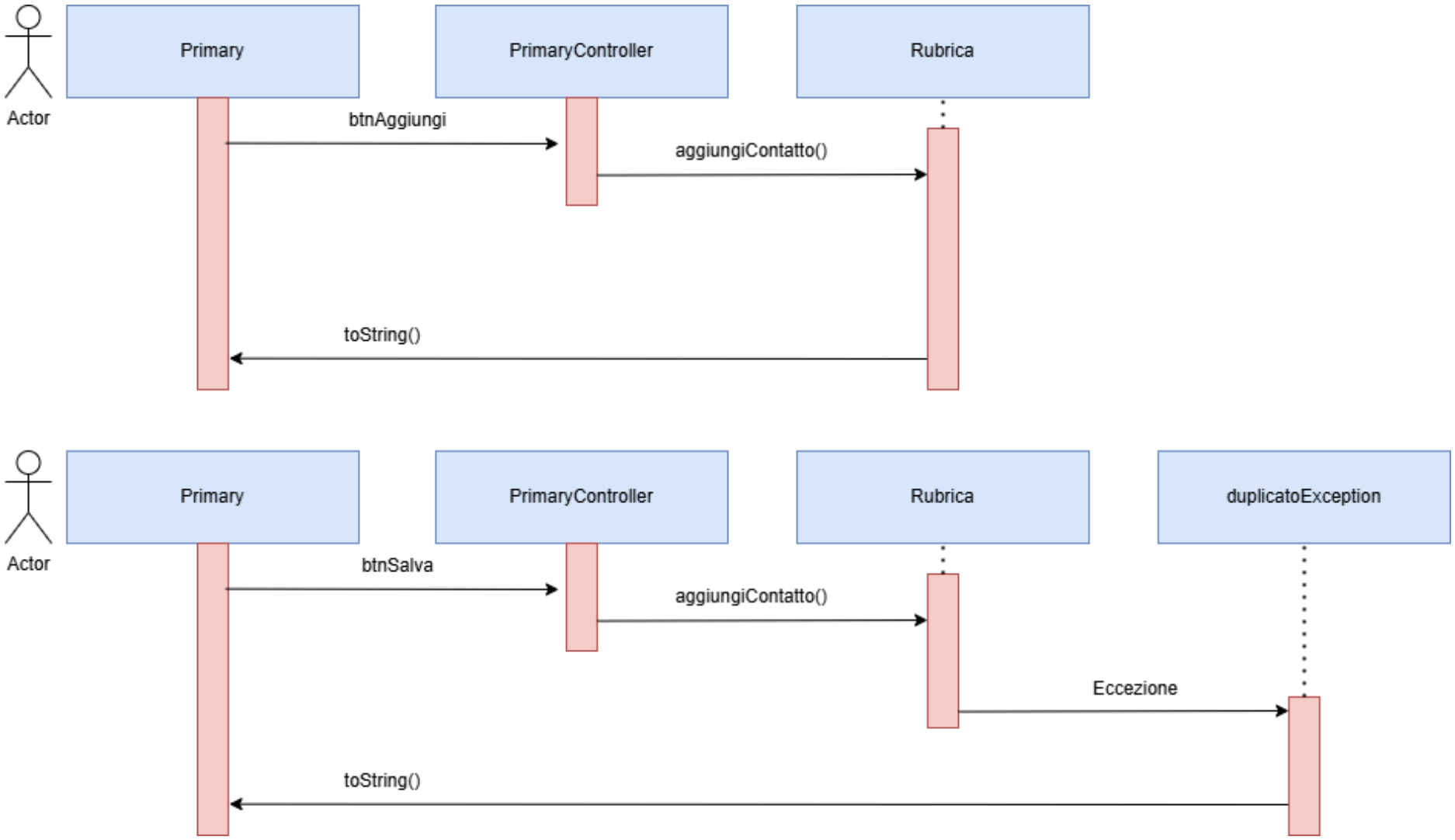
Per maggiore leggibilità visionare il PDF “ClassDiagram” presente nella cartella “Allegati”.

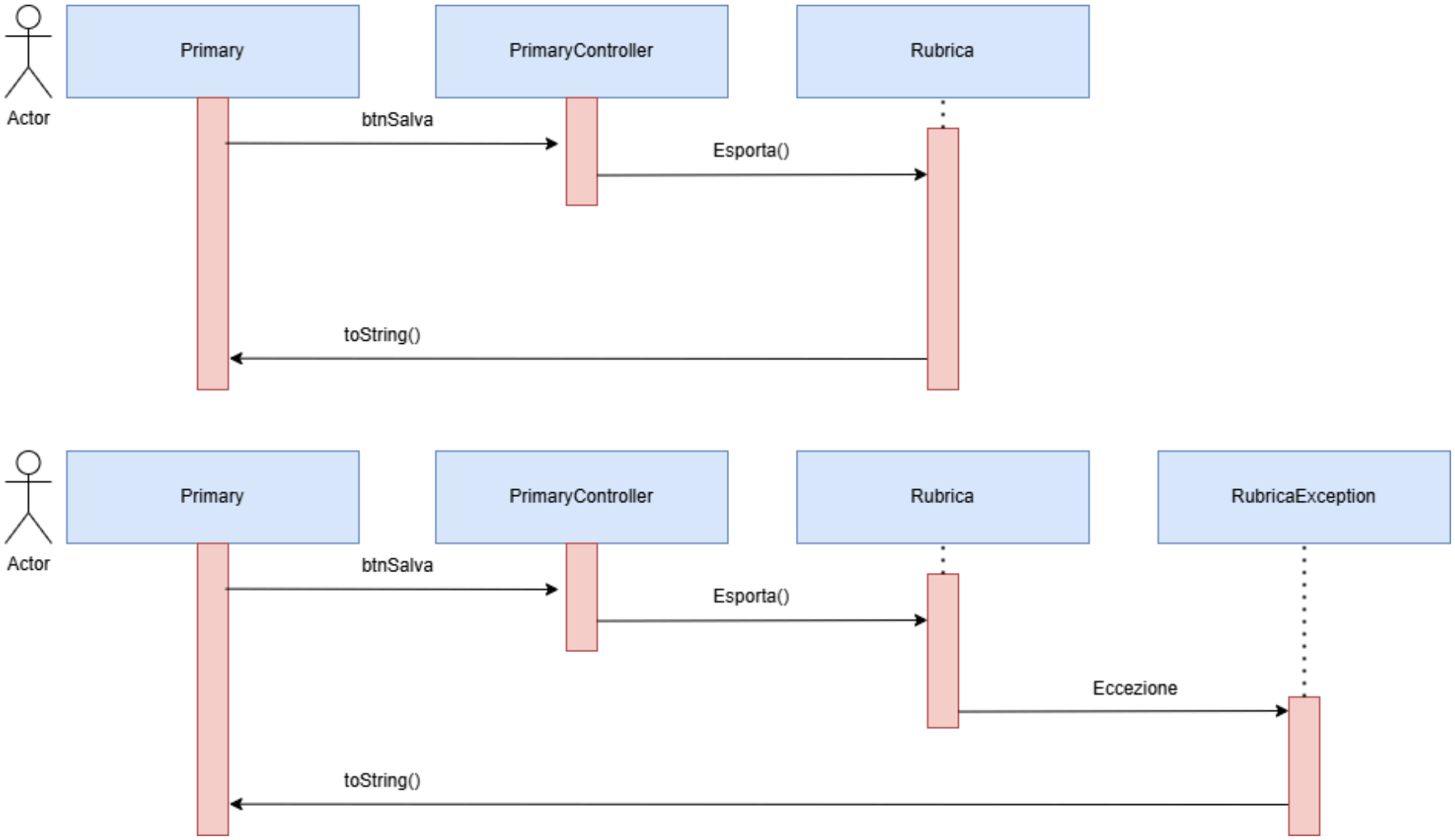


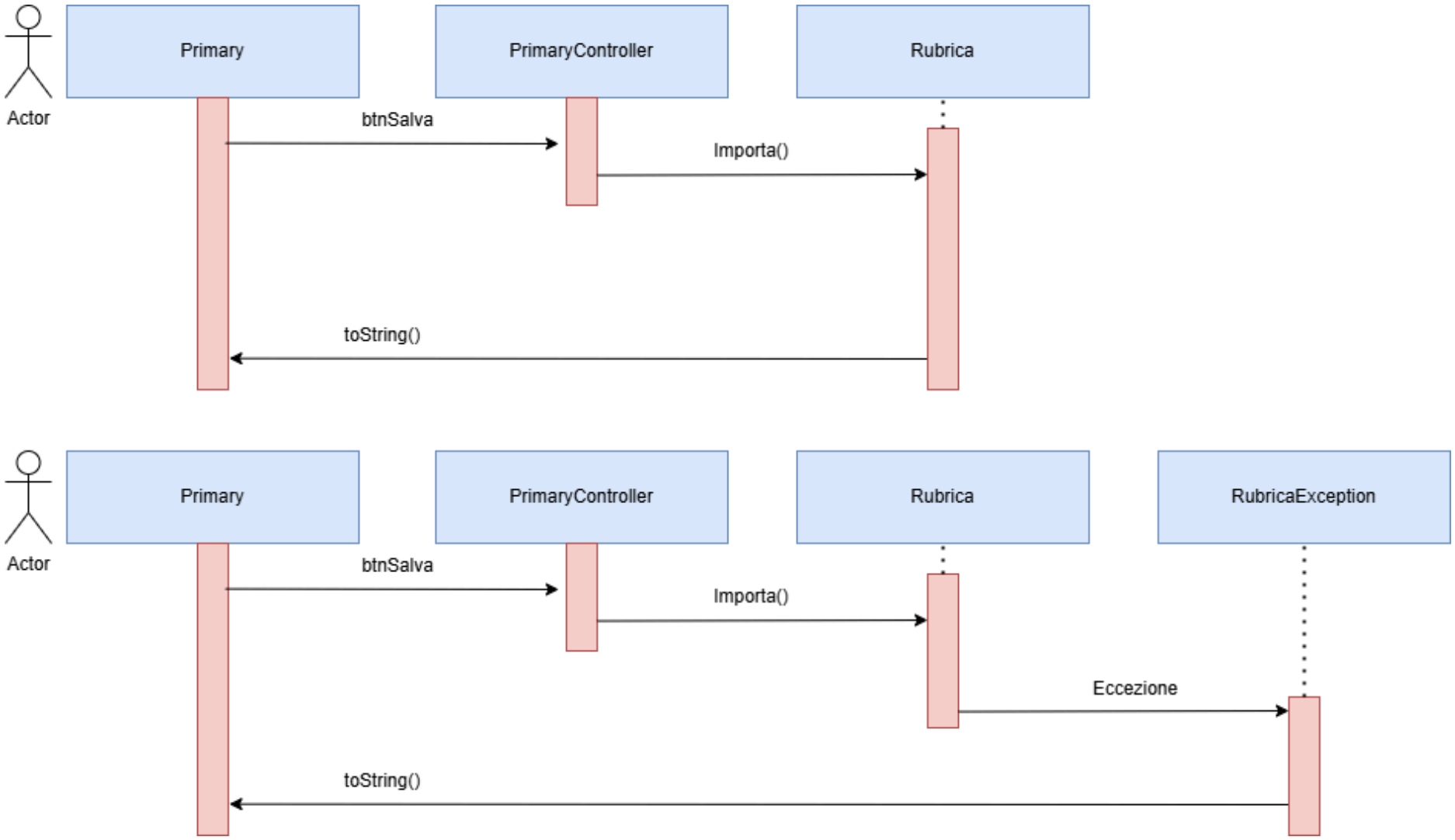
4. Interaction Overview Diagrams

I seguenti diagrammi sono degli Interaction Overview Diagrams, ovvero degli Activity diagrams le cui attività sono rimpiazzate da sequence diagrams.
Ogni diagramma presentato è volto a esplicitare il funzionamento del sistema in determinati scenari. Per ogni scenario verrà mostrata l'esecuzione di una possibile operazione che può essere effettuata dall'utente. In generale, per le operazioni non mostrate, le attività svolte dal sistema sono pressoché uguali, ma il comportamento complessivo, in ogni scenario, resta invariato.
Per maggiore leggibilità visionare il PDF "OverviewDiagrams" presente nella cartella "Allegati".
I diagrammi rappresentano rispettivamente:

1. Aggiunta di un contatto.
2. Aggiunta di un contatto con errore(nome o cognome, email, numero di telefono non valido).
3. Esportazione della rubrica su un file.
4. Esportazione della rubrica su un file con errore(File non trovato).
5. Importazione della rubrica da un file.
6. Importazione della rubrica da un file con errore(File non trovato).


Actor
Primary
PrimaryController
Rubrica
btnAggiungi
aggiungiContatto()
toString()
Actor
Primary
PrimaryController
Rubrica
duplicatoException
btnSalva
aggiungiContatto()
Eccezione
toString()


Actor
Primary
PrimaryController
Rubrica
btnSalva
Esporta()
toString()
Actor
Primary
PrimaryController
Rubrica
RubricaException
btnSalva
Esporta()
Eccezione
toString()


Actor
Primary
PrimaryController
Rubrica
btnSalva
Importa()
toString()
Actor
Primary
PrimaryController
Rubrica
RubricaException
btnSalva
Importa()
Eccezione
toString()

5.Coesione e accoppiamento

Rispettivamente, la prima misura quanto le parti che sono incluse nello stesso modulo sono legate tra di loro mentre la seconda misura il grado di interdipendenza tra moduli diversi.

- ## Contatto

  La classe Contatto beneficia di un livello di coesione comunicazionale e presenta i seguenti tipi di accoppiamento:
  - Per Timbro: A causa della dipendenza da oggetti complessi come Email e NumTelefono.
  - Per Dati: La classe comunica con altre classi tramite dati concreti e specifici.

- ## Email

  La classe Email beneficia di un livello di coesione comunicazionale e presenta i seguenti tipi di accoppiamento:
  - Per Timbro: Un po' presente a causa della dipendenza da HashSet per memorizzare le email.
  - Per Dati: La classe comunica principalmente attraverso dati specifici (stringhe di email).

- ## NumTelefono

  La classe NumTelefono beneficia di un livello di coesione comunicazionale e presenta i seguenti tipi di accoppiamento:
  - Per Timbro: A causa della dipendenza da HashSet per la gestione dei numeri di telefono.
  - Per Dati: La classe comunica con dati concreti (numeri di telefono) senza dipendere da altre classi in modo significativo.

- ## Rubrica

  La classe Rubrica beneficia di un livello di coesione comunicazionale e presenta i seguenti tipi di accoppiamento:
  - Per Controllo: Dovuto alla gestione diretta delle eccezioni specifiche.
  - Per Timbro: Causato dall'uso di oggetti complessi (Contatto, Email, e NumTelefono) quando solo parte dei loro dati è necessaria.

- Per Dati: Il livello più basso applicabile in questa progettazione, indicativo di una buona separazione tra le responsabilità delle classi.

7. Principi di buona progettazione

- S: Single Responsibility Principle (SRP):
  Un componente del codice deve svolgere un singolo compito ben definito.
  - Ogni classe rispetta il principio di singola responsabilità:
    - Contatto gestisce i dati personali.
    - Rubrica gestisce i contatti.
    - Email e NumTelefono sono dedicate a specifici dettagli di implementazione.
- O: Open/Closed Principle (OCP):

  Le unità software (classi, moduli, funzioni, ecc.) dovrebbero essere aperte all'estensione, ma chiuse alla modifica:

  - La struttura supporta estensioni (ad esempio, aggiungendo nuovi tipi di validazione o funzionalità) senza modificare le classi esistenti.
- L: Liskov Substitution Principle (LSP):

  Gli oggetti devono poter essere sostituiti con istanze dei loro sottotipi senza alterare la correttezza del programma.

  - Le eccezioni personalizzate rispettano il principio, poiché derivano da RubricaException e possono essere usate ovunque è atteso il tipo base.
- I: Interface Segregation Principle (ISP)
  - Non ci sono interfacce definite nel diagramma. Tuttavia, le classi sembrano evitare di imporre metodi inutili.
- D: Dependency Inversion Principle (DIP)

  I moduli di alto livello non devono dipendere dai moduli di basso livello, entrambi devono dipendere da astrazioni.

  - Il progetto potrebbe beneficiare di astrazioni per ridurre la dipendenza diretta tra controller e rubrica.