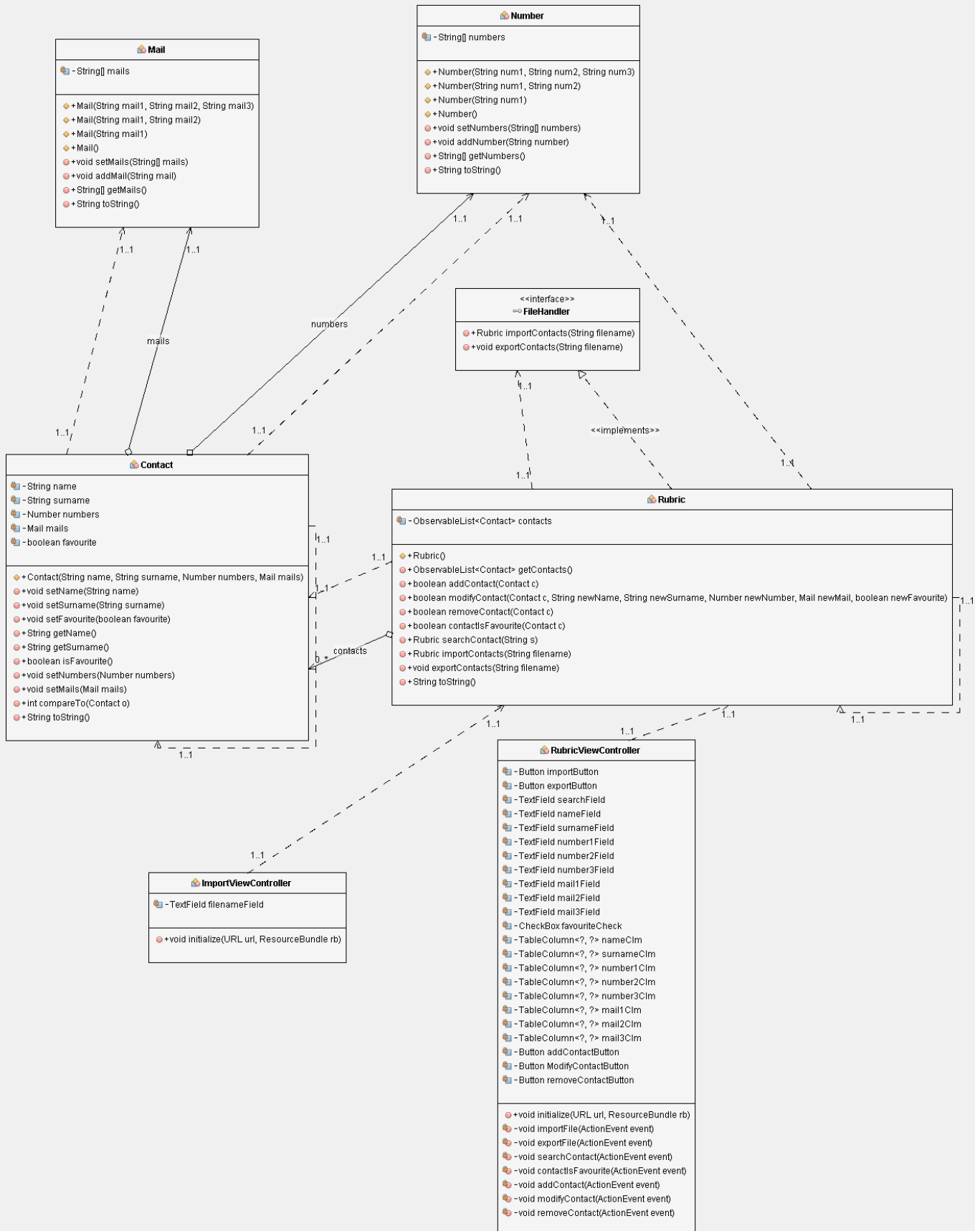


# Documento di design

## Diagramma delle classi



# Valutazione di coesione e accoppiamento

## Coesione

### Classe Contact

La classe Contact ha una coesione **funzionale**:

la classe gestisce tutti gli attributi e i metodi per rappresentare un contatto (nome, cognome, numeri, email, ecc.) e offre metodi per gestire questi attributi.

Le funzionalità sono raggruppate per rappresentare un'entità ben definita, ovvero un contatto.

### Classe Rubric

La classe Rubric ha una coesione **funzionale**:

gestisce un insieme di contatti e fornisce metodi per aggiungere, modificare, rimuovere e contrassegnare i contatti come preferiti.

Anche in questo caso, le funzionalità rendono la classe ben focalizzata sull'entità rappresentata, ovvero una rubrica.

### Classe Mail e Number

Sia la classe Mail che la classe Number hanno una coesione **funzionale**:

entrambe forniscono attributi e metodi per rappresentare email e numeri di telefono relativi ad un contatto.

- La classe Mail gestisce direttamente le email associate, con metodi per impostare e ottenere le email.
- La classe Number fa lo stesso per i numeri di telefono.

Dunque ciascuna delle due classi racchiude le funzionalità della singola entità.

### Interfaccia FileHandler

L'interfaccia FileHandler ha una coesione **funzionale**:

si occupa di astrarre le operazioni di import ed export, che vengono poi specializzate da Rubric nel caso specifico dell'import e l'export dei contatti.

## Accoppiamento

### Rubric-Contact

#### Data Coupling - accoppiamento per dati:

La classe Rubric utilizza direttamente oggetti di tipo Contact in modo chiaro e ben strutturato, passando solo i dati necessari per i metodi addContact, removeContact, modifyContact, contactIsFavourite.

L'accoppiamento è dunque limitato al minimo indispensabile per il funzionamento.

### Contact-Mail/Number

#### Data Coupling - accoppiamento per dati:

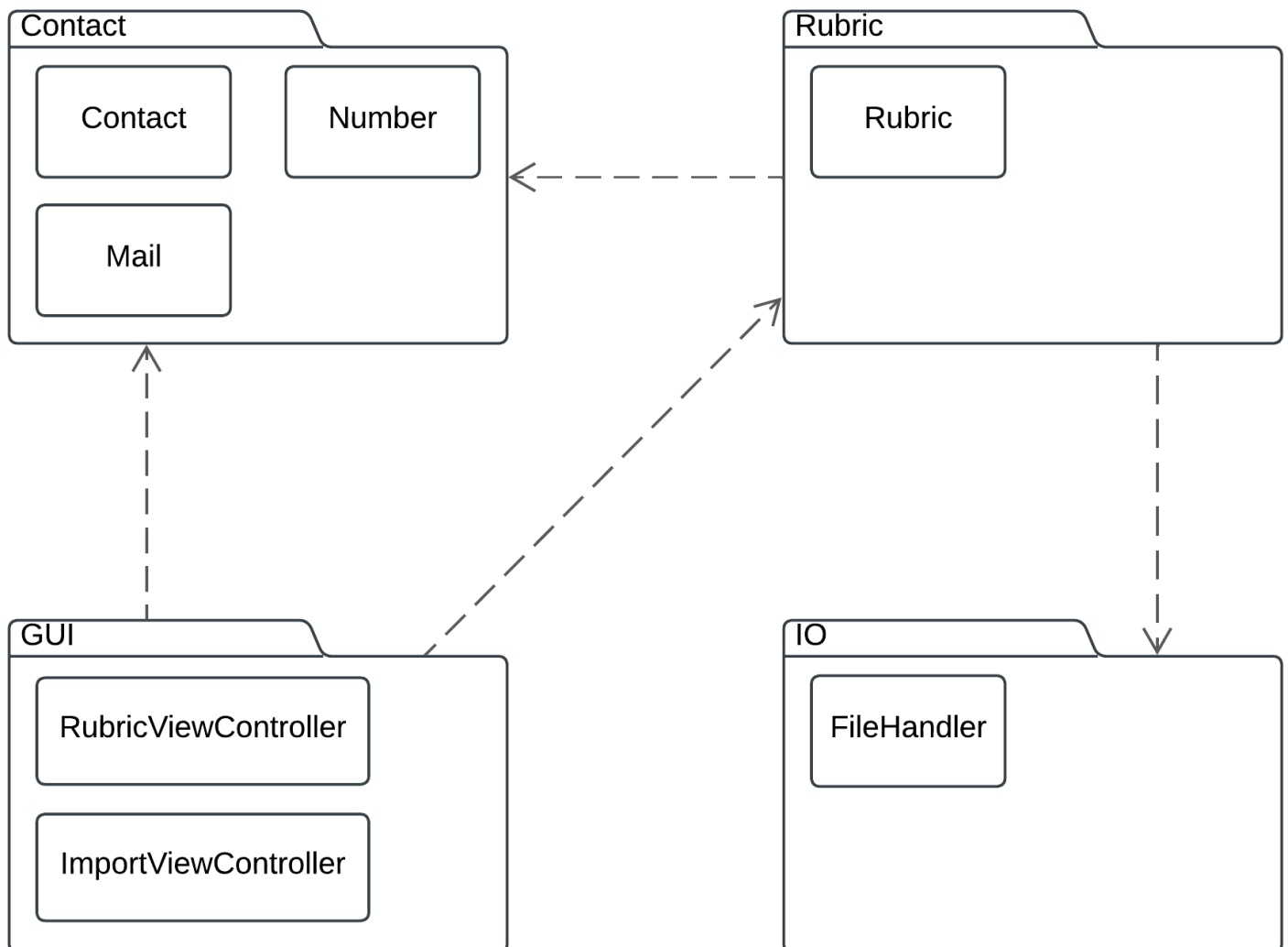
La classe Contact utilizza oggetti di tipo Mail e Number per rappresentare i dati di un contatto, in modo chiaro evitando di creare dipendenze ridondanti.

### Rubric-FileHandler

#### Data Coupling - accoppiamento per dati:

La classe Rubric non dipende dall'implementazione specifica dei metodi siccome FileHandler è un'interfaccia. Rubric implementa i metodi forniti da FileHandler passando solo i dati necessari.

# Diagramma dei package

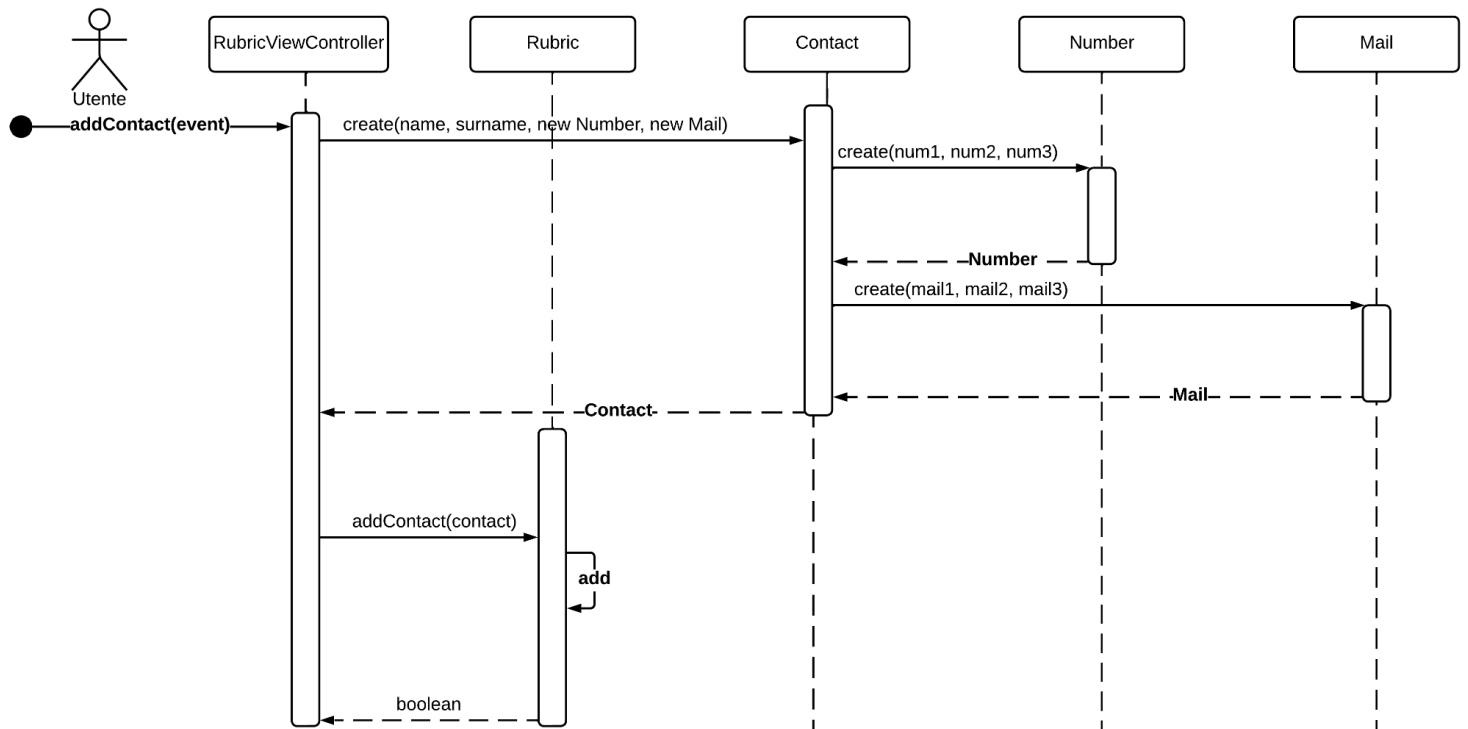


Dal diagramma di evincono le dipendenza tra i vari package:

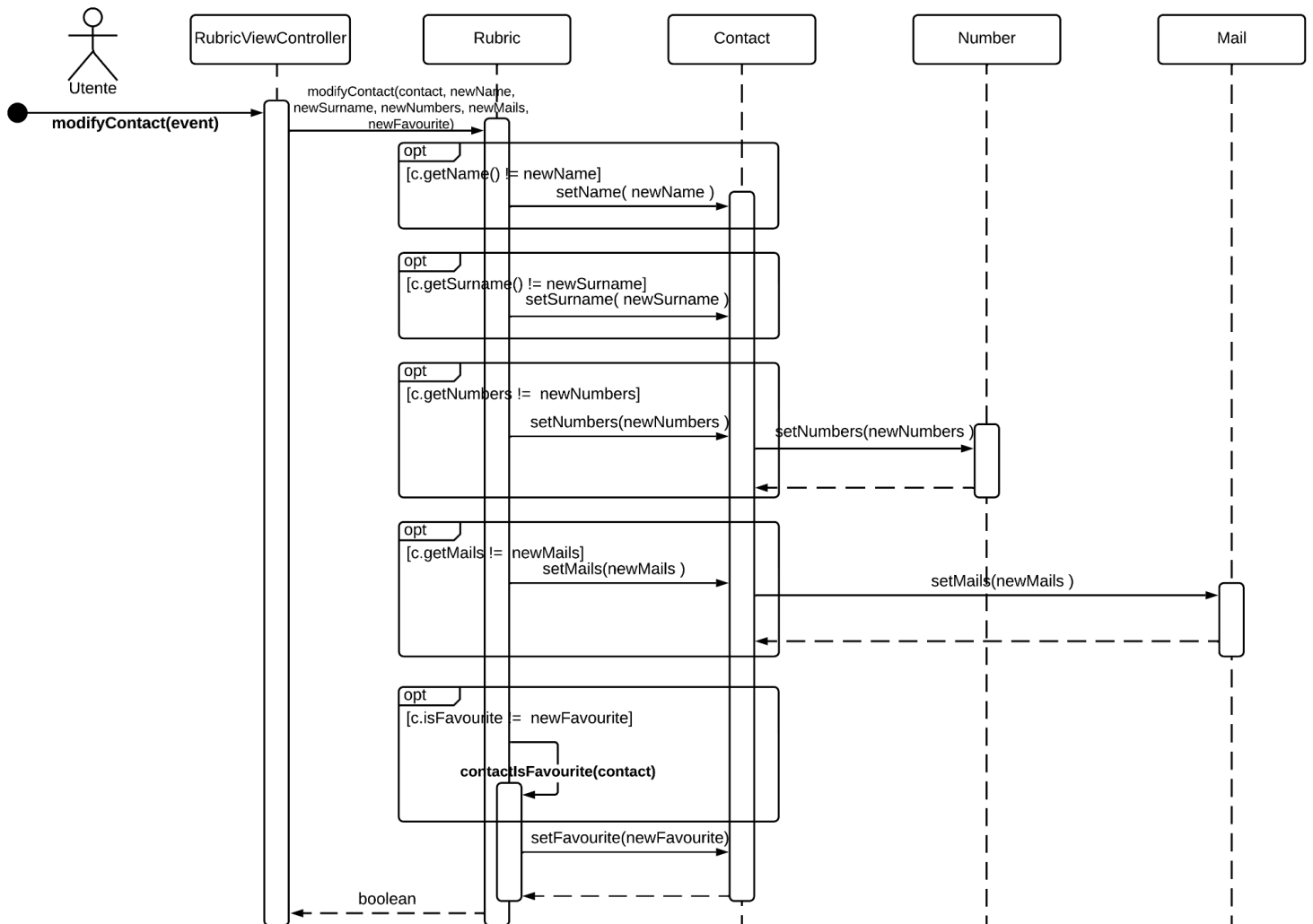
- package **Rubric**: contiene la classe **Rubric**. Per gestire i contatti della rubrica, dipende dal package **Contact**.
- package **IO**: fornisce funzionalità al package **Rubric**, dunque **Rubric** dipende anche da questo.
- package **GUI**: contiene i controller per l'interfaccia grafica. Dipende dal package **Rubric** per accedere ai dati della rubrica e alla logica applicativa; dal package **Contact** per accedere ai singoli dettagli dei contatti.

# Diagrammi di sequenza

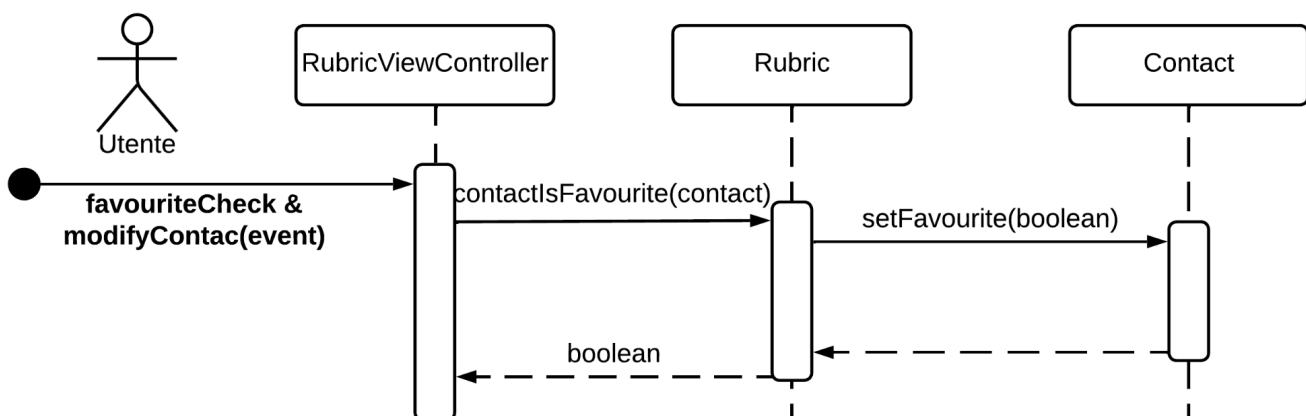
## Scenario: aggiungere un contatto



## Scenario: modificare un contatto



## Scenario: contrassegnare il contatto come preferito



## Scenario: eliminare un contatto

