Università degli Studi di Salerno

.DIEM

Dipartimento di Ingegneria dell'Informazione ed Elettrica e Matematica Applicata

Corso di Laurea in Ingegneria Informatica

Ingegneria del Software a.a. 2024/2025

Project Work

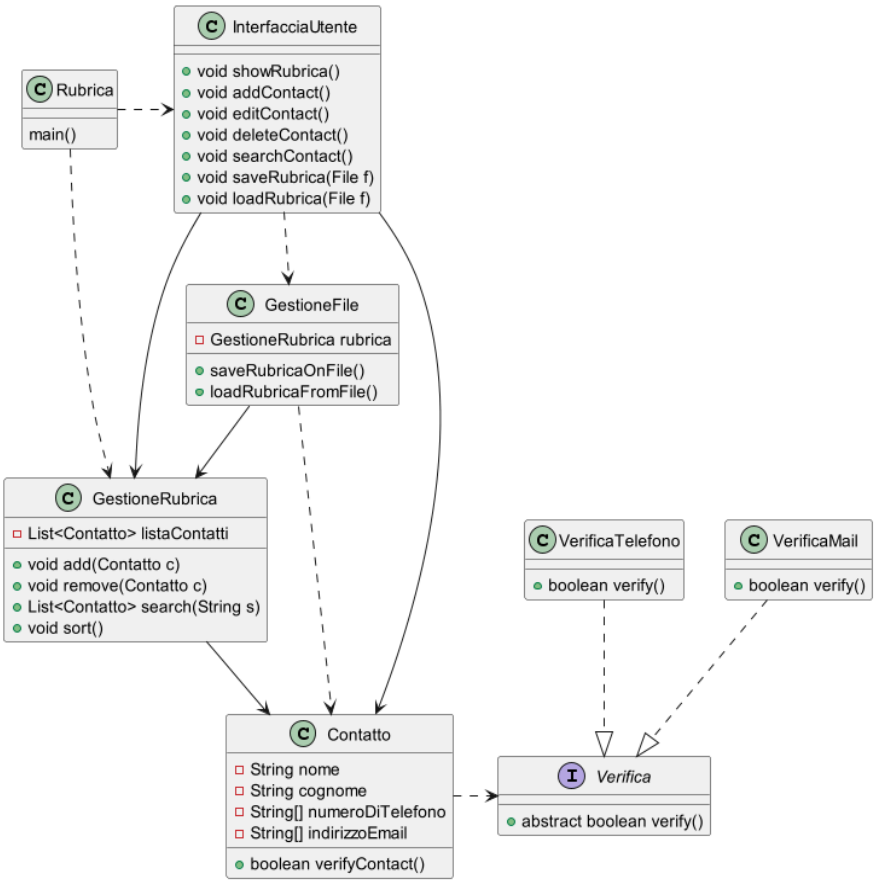
Canale A-H

Gruppo 1

Cognome e nome	Matricola	email
Vincenzo Citro	0612705328	v.citro26@studenti.unisa.it
Antonio Bellofatto	0612708323	a.bellofatto4@studenti.unisa.it
Genco Davide	0612705464	d.genco1@studenti.unisa.it
Cynthiachiara Gambardella	0612707276	c.gambardella13@studenti.unisa.it

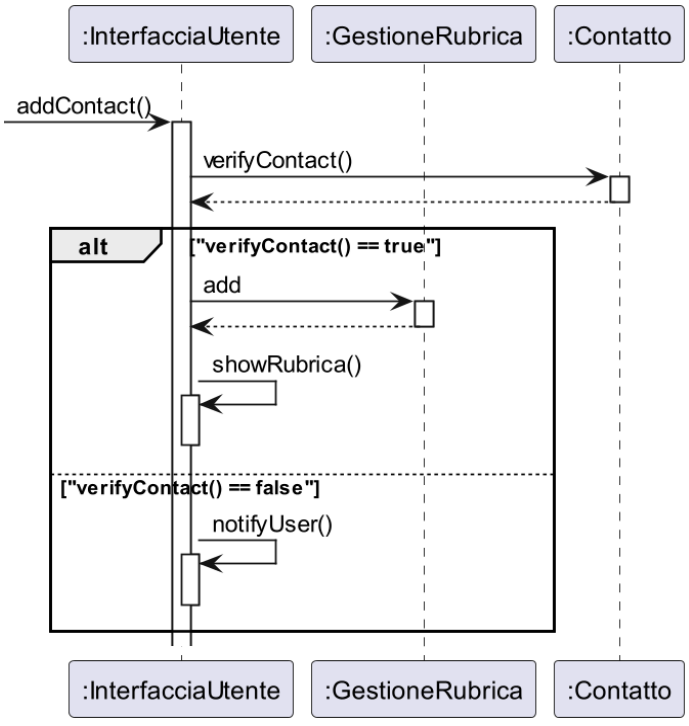
PROGETTAZIONE
DI
DETTAGLIO

1. Diagramma delle classi



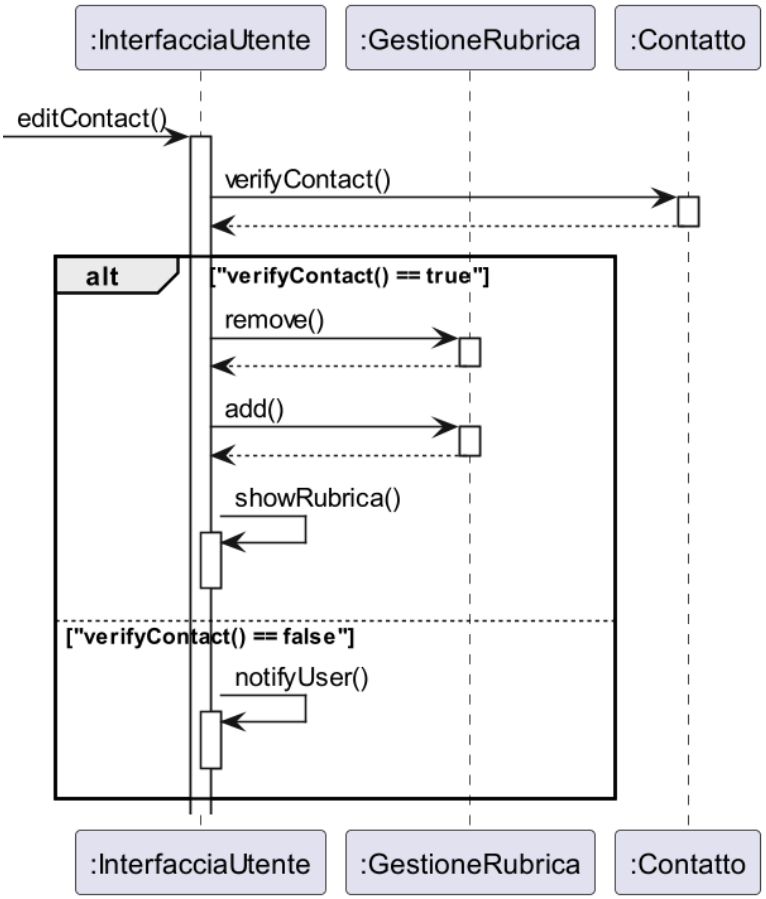
2. Diagrammi di sequenza

a. Aggiunta di un contatto

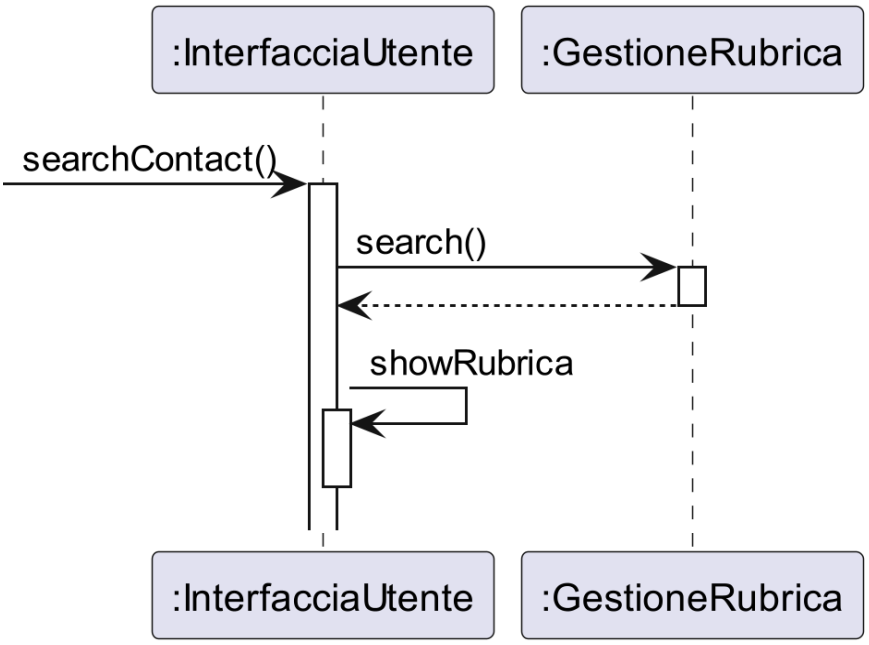


Il metodo privato `notifyUser()` notifica all'utente che il contatto inserito non è valido.

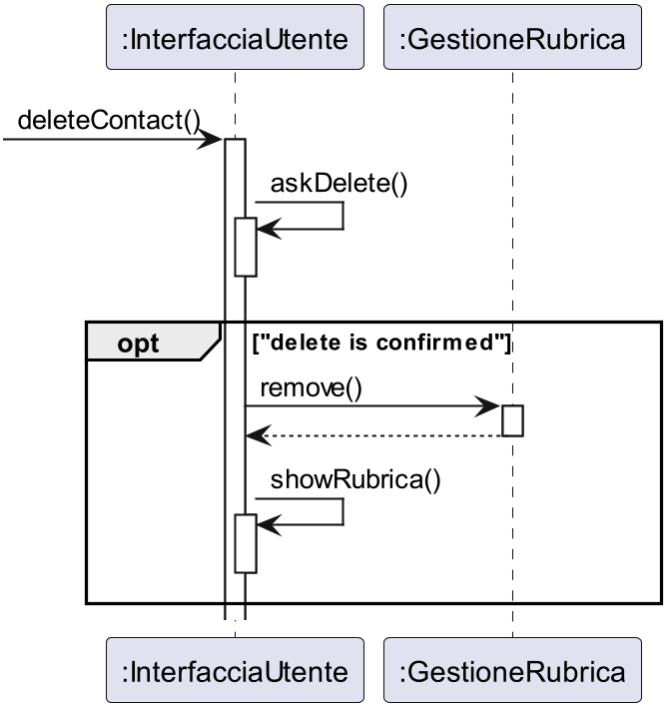
b. Modifica di un contatto



c. Ricerca di un contatto

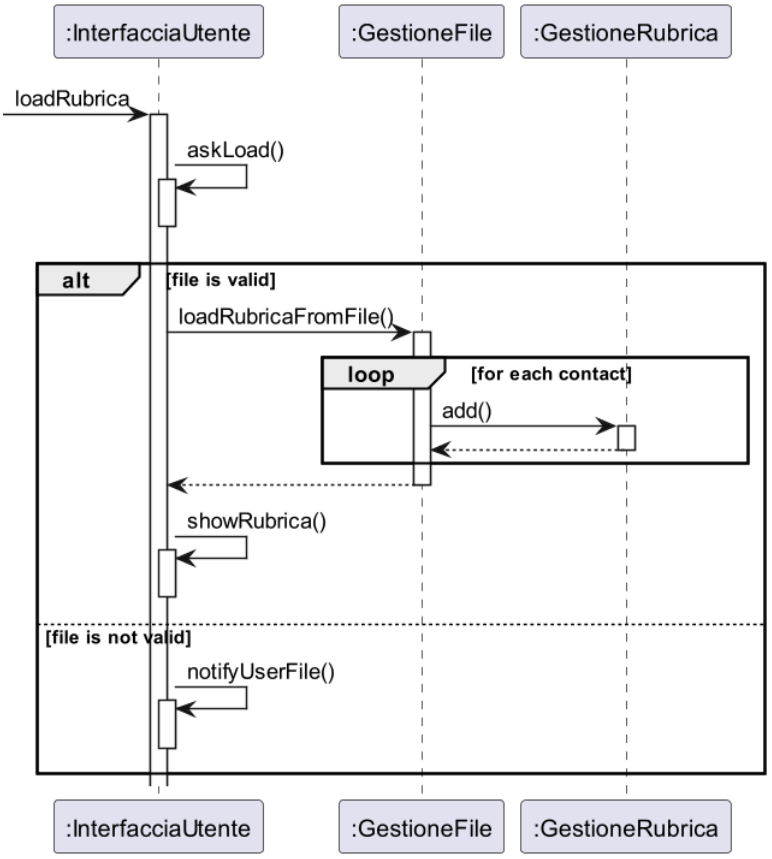


d. Eliminazione di un contatto



Il metodo privato `askDelete()` chiede all'utente se desidera cancellare il contatto.

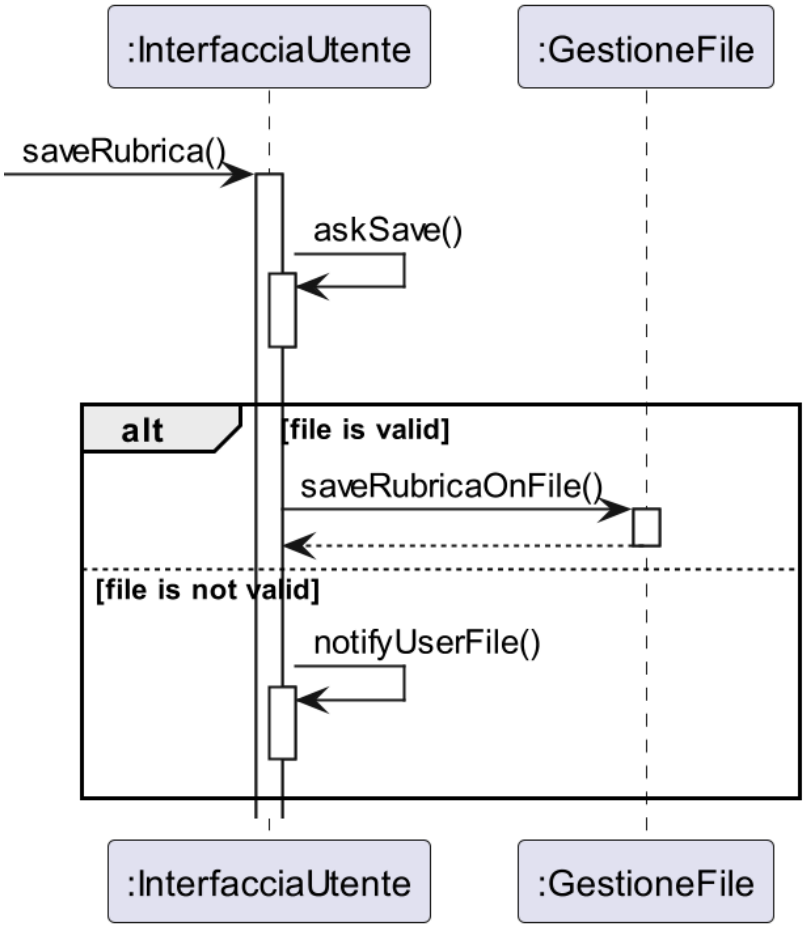
e. Caricamento rubrica da file



Il metodo privato `askLoad()` chiede all'utente da quale file bisogna caricare la rubrica sull'applicazione.

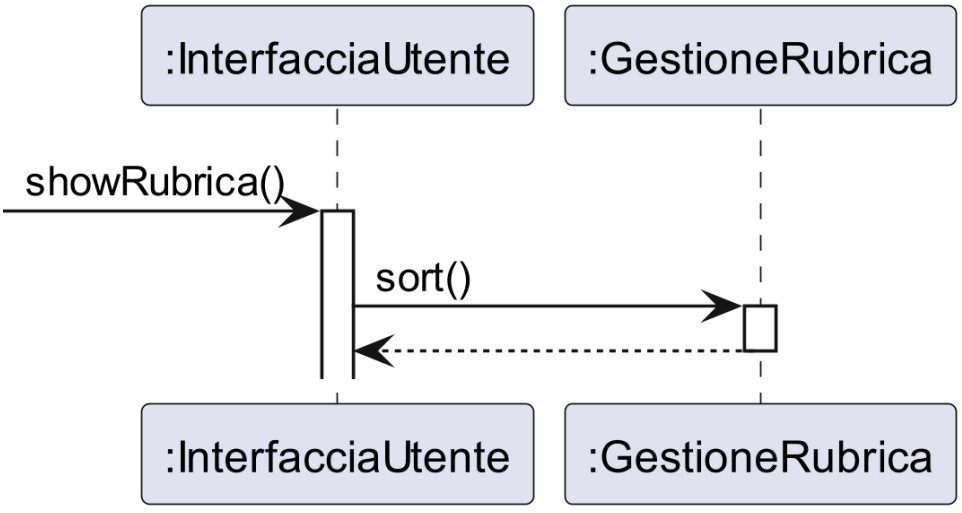
Il metodo privato `notifyUserFile()` notifica all'utente che il file selezionato non è valido.

f. Salvataggio rubrica su file



Il metodo privato `askSave()` chiede all'utente su quale file salvare la rubrica.

g. Visualizzazione rubrica sull'interfaccia



3. Scelte progettuali

a. Coesione

Non tutte le classi individuate presentano un livello di coesione funzionale. La classe `GestioneRubrica`, che include metodi per la gestione della struttura dati che rappresenta la lista dei contatti, presenta un livello di coesione funzionale, mentre `GestioneFile` si occupa esclusivamente delle operazioni su file, quindi presenta un livello di coesione comunicazionale. La classe `InterfacciaUtente` si occupa della parte grafica dell'applicazione e dell'interazione dell'utente con quest'ultima utilizzando i servizi offerti dalle altre classi per effettuare le principali operazioni sulla rubrica.

b. Accoppiamento

La classe `Contatto` è indipendente dai dettagli di implementazione di `VerificaTelefono` e `VerificaEmail`, poiché interagisce con esse tramite l'interfaccia `Verifica`. Questo design consente di modificare o sostituire le implementazioni delle verifiche senza impatti sulla classe `Contatto`. L'accoppiamento tra l'`InterfacciaUtente` e le classi `GestioneRubrica` e `GestioneFile` è ridotto scambiando solo i dati strettamente necessari (riferimenti alla lista di contatti e riferimenti dei file su cui effettuare operazioni di input/output). L'accoppiamento tra la classe `GestioneFile` e la classe `GestioneRubrica` è anch'esso ridotto scambiando solo i dati necessari per il funzionamento delle operazioni su file.

c. Principi di buona progettazione

Abbiamo applicato il principio di *separation of concerns* relativamente alle classi `GestioneRubrica` e `GestioneFile`, separando quindi la operazioni dei contatti da quelle su file. Abbiamo applicato il principio di *singola responsabilità* in modo tale che ogni classe abbia un singolo compito ben definito: la classe `GestioneRubrica` opera sui contatti, la classe `GestioneFile` opera sui file, la classe `InterfacciaUtente` gestisce le interazioni dell'utente con l'applicazione. E' stato applicato il principio di *sostituzione di Liskov* alle classi `VerificaMail` e `VerificaTelefono`, utilizzando anche il principio di *segregazione delle interfacce* con l'interfaccia `Verifica`, che presenta l'unico metodo verify(). E' stato applicato il principio di *inversione della dipendenza* tra la classe `Contatto` e le classi `VerificaMail` e `VerificaTelefono` in modo tale che la classe `Contatto` non dipenda da queste classi ma da un astrazione che è l'interfaccia `Verifica`.