

Design

INDICE

- 1 Diagrammi delle classi
- 2 Diagrammi di sequenza
- 3 Altri diagrammi

1 Diagramma delle classi



Il design del sistema segue il **pattern MVC** per separare in modo chiaro le responsabilità tra i componenti principali.

Le scelte progettuali sono state fatte con l'obiettivo di massimizzare la coesione, mantenendo ogni classe focalizzata su un singolo compito:

- La classe Rubrica si occupa della gestione dei contatti nel loro insieme.
- La classe Contatto si occupa della gestione e validazione dei dati di un singolo contatto.
- La classe FileManager si occupa delle operazioni di input/output su file, senza sovrapporsi con logiche di business.
- Le classi controller sono responsabili della gestione delle interazioni utente, separando la logica di interfaccia utente da quella di gestione dei dati.
- Le classi view implementano la GUI dell'applicazione.

Questa separazione delle responsabilità assicura un alto livello di coesione per ciascuna classe, che facilita la manutenzione e l'estensibilità del sistema.

Le scelte progettuali sono state inoltre finalizzate ad ottenere dipendenze chiare tra le classi e un basso livello di accoppiamento, al fine di ridurre la propagazione dei cambiamenti e migliorare la modularità.

Il design del sistema segue i principi di buona progettazione per garantire che il software sia robusto, manutenibile ed estensibile. In particolare, sono stati applicati i principi SOLID, che promuovono la separazione delle responsabilità, la modularità e la gestione efficace delle dipendenze. Questi principi sono stati adottati per ottenere un sistema ben strutturato, che possa adattarsi a future modifiche senza compromettere l'integrità del design.

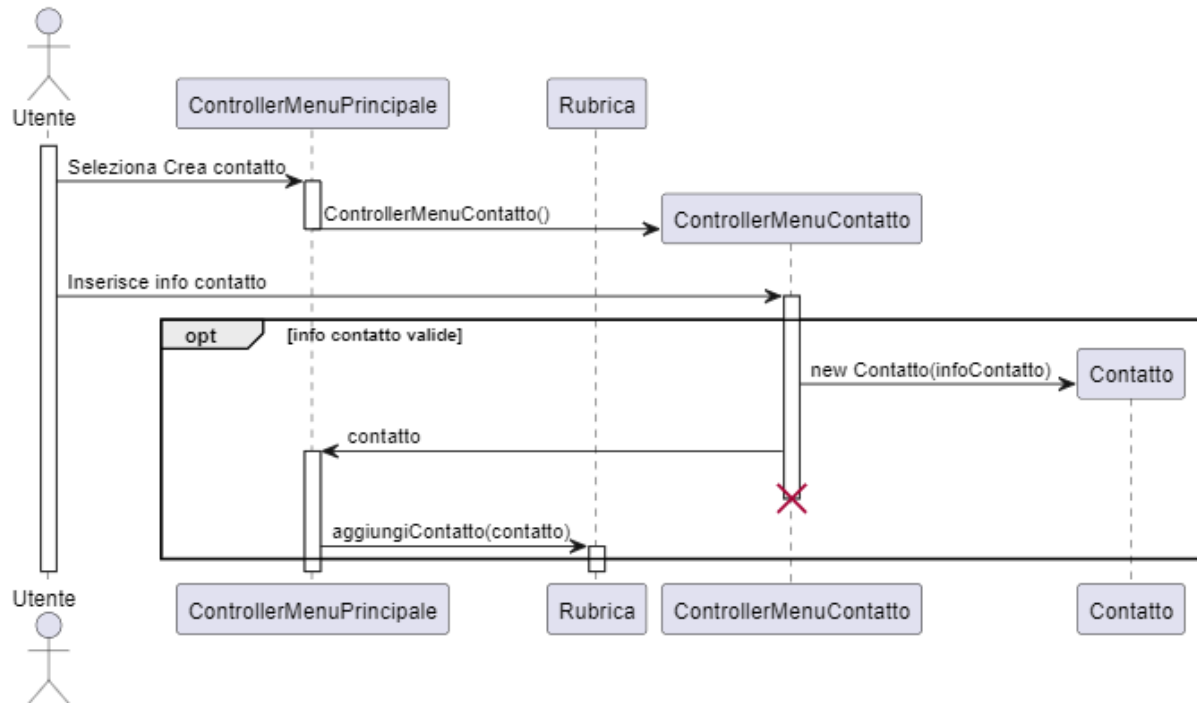
Ad esempio:

- Il principio della singola responsabilità (SRP) è rispettato separando le funzionalità in classi che svolgono funzioni specifiche, questo approccio evita sovrapposizioni di responsabilità e semplifica la manutenzione.
- Il principio di robustezza è garantito dall'utilizzo dell'eccezione `InfoContattoException`, che permette di gestire eventuali errori o dati non validi in modo strutturato. Questa eccezione assicura che il sistema possa rilevare e segnalare errori relativi ai dati, migliorando l'affidabilità.
- Il principio di inversione della dipendenza è applicato nella progettazione del `FileManager`, definito come un'astrazione. Questo permette di separare la logica di gestione dei dati dalla logica di business, consentendo la sostituzione o estensione del meccanismo di salvataggio senza impattare altre parti del sistema.

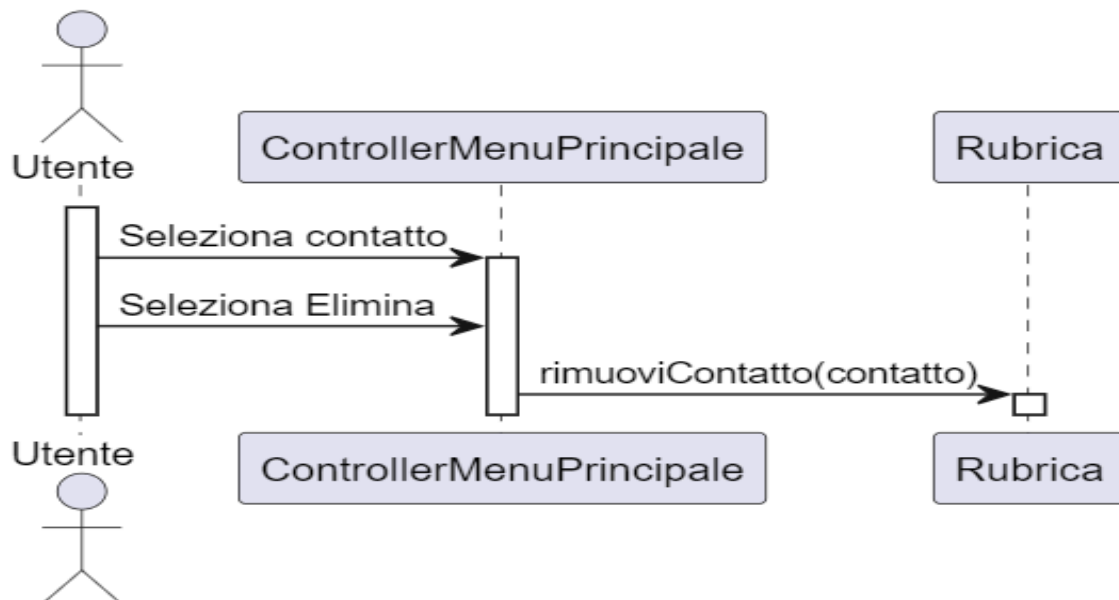
2 Diagrammi di sequenza

I seguenti diagrammi modellano il comportamento degli oggetti nei vari casi d'uso.

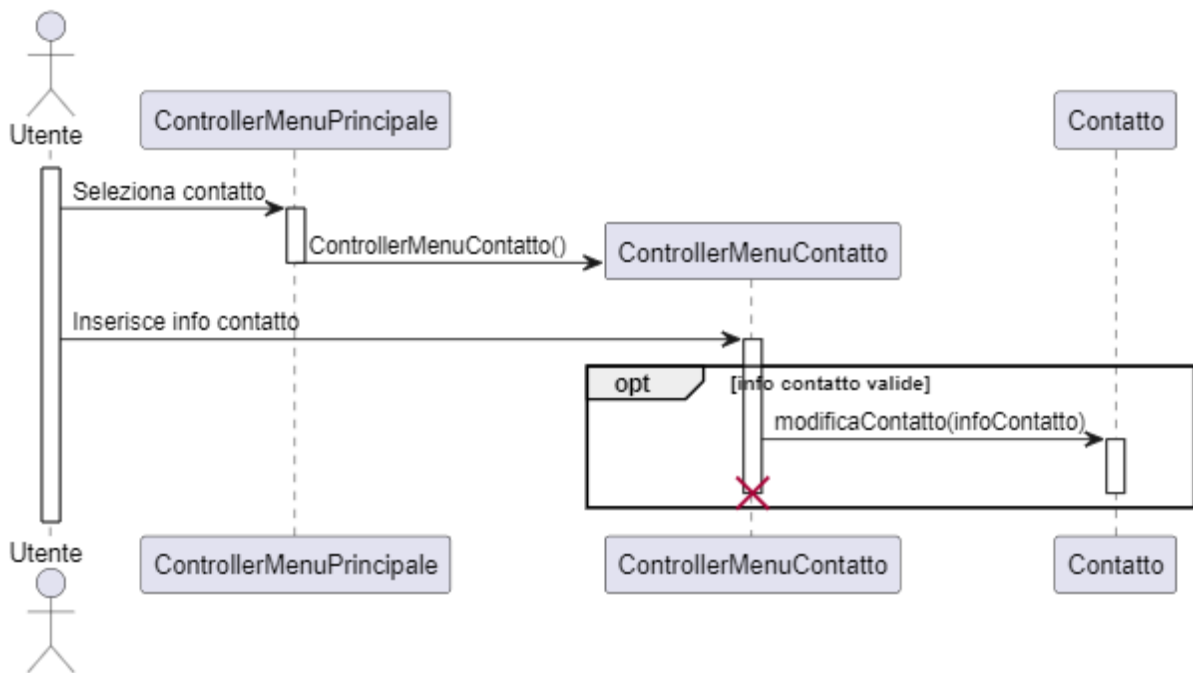
Creazione contatto:



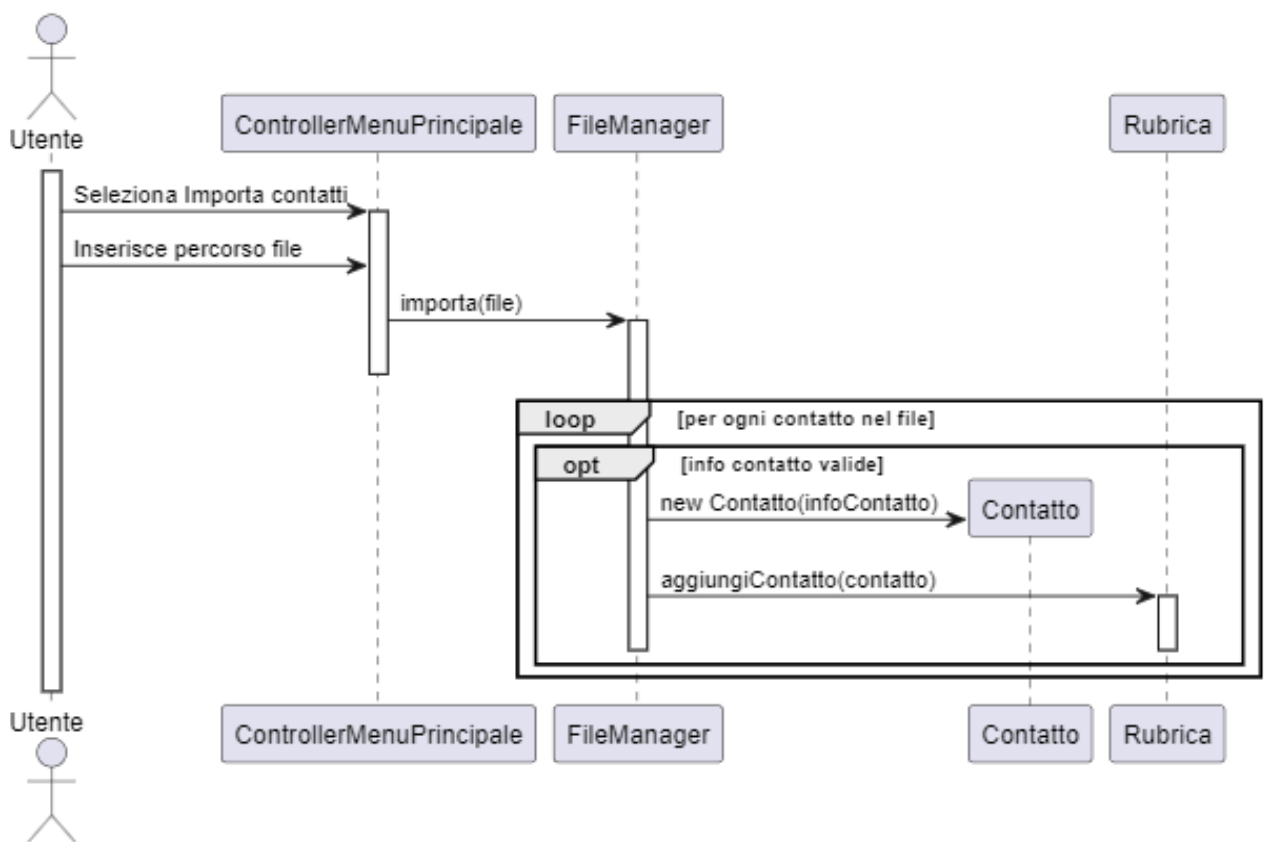
Rimozione contatto:



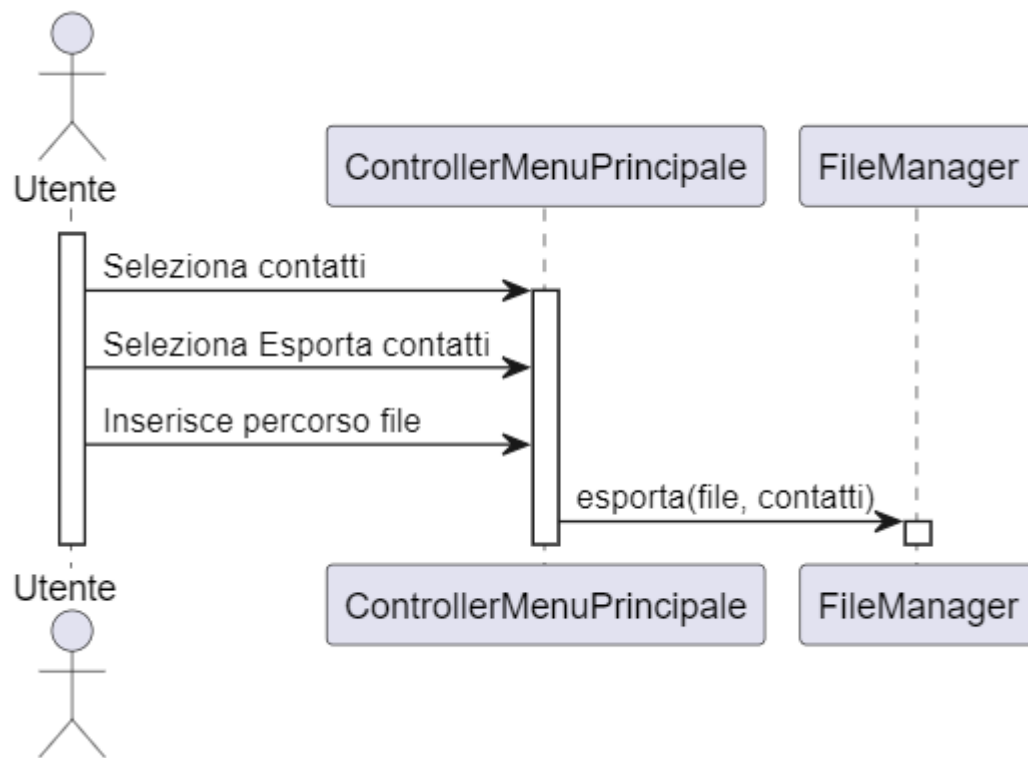
Modifica contatto:



Importa contatti:

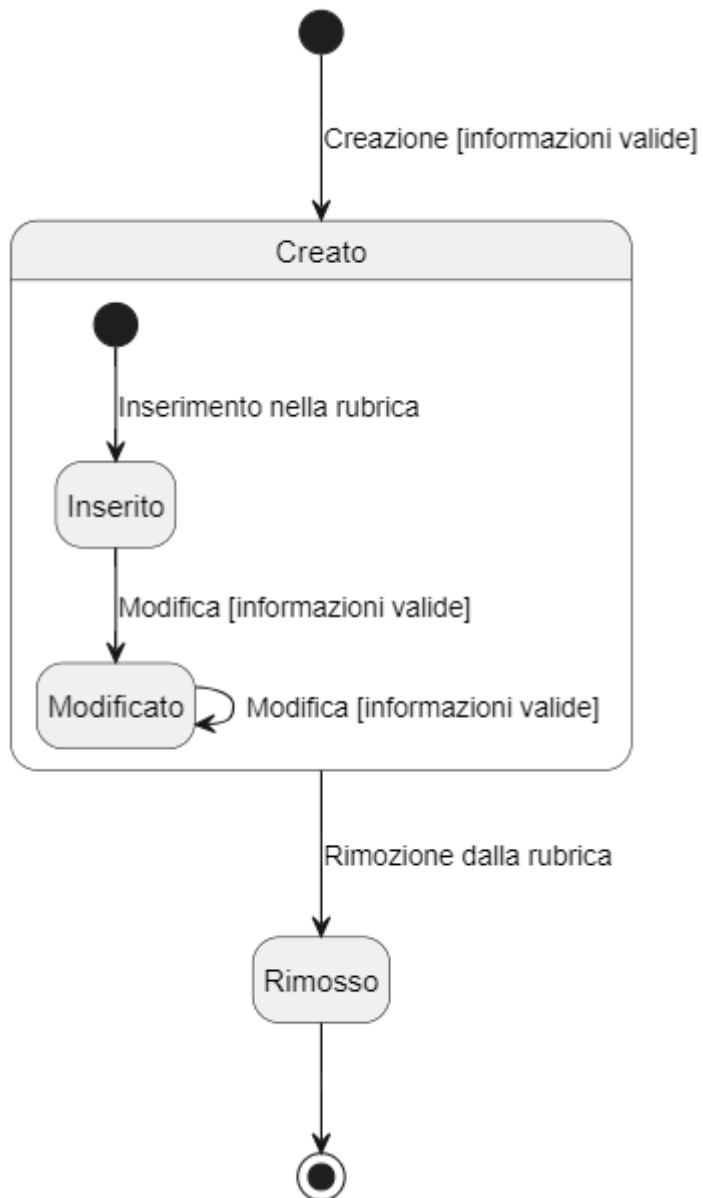


Esporta contatti:



3 Altri diagrammi

Il seguente diagramma a stati mostra il comportamento di un oggetto Contatto durante il suo ciclo di vita.



Il seguente diagramma di attività mostra l'intero workflow dell'applicazione.

