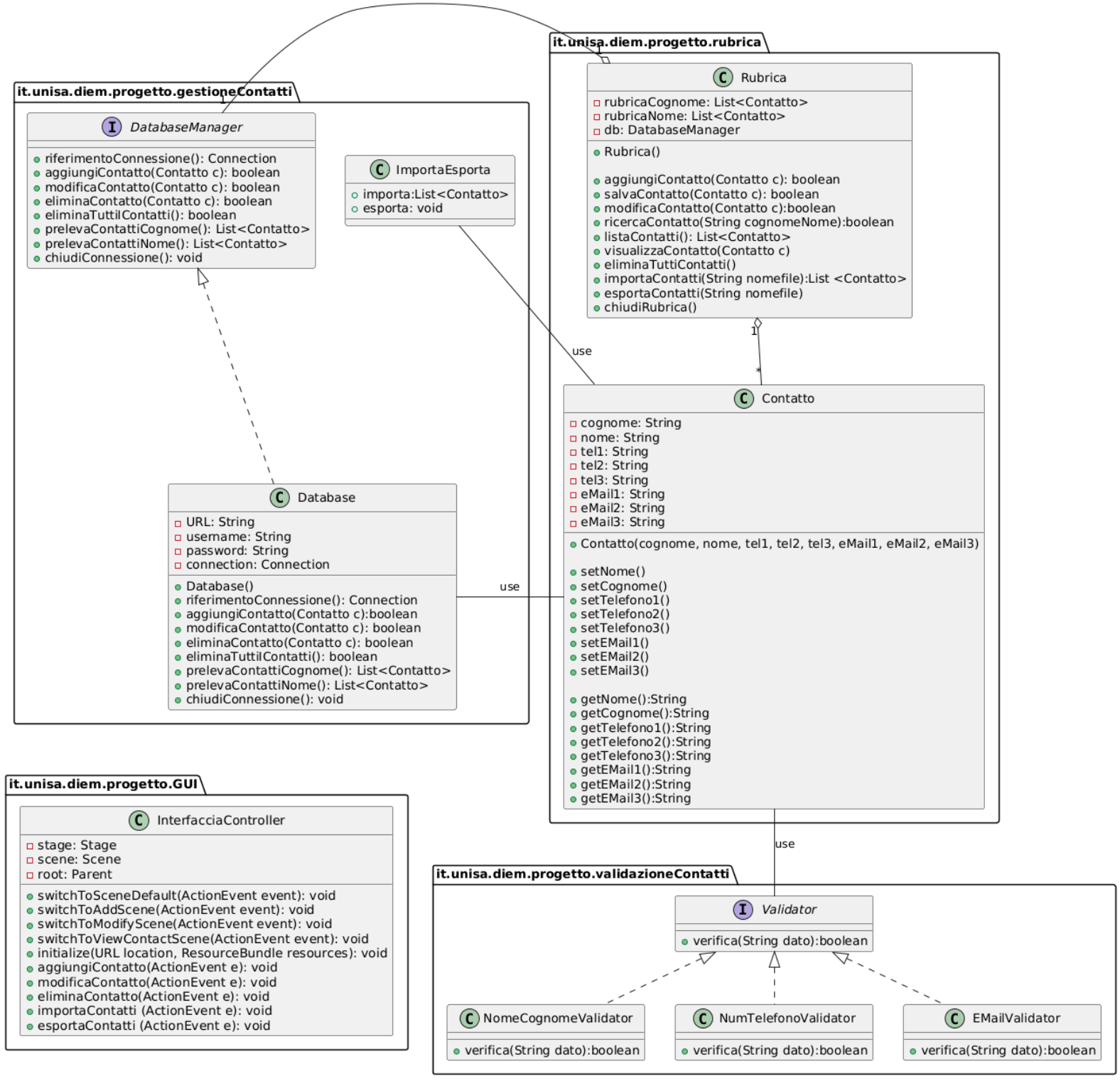
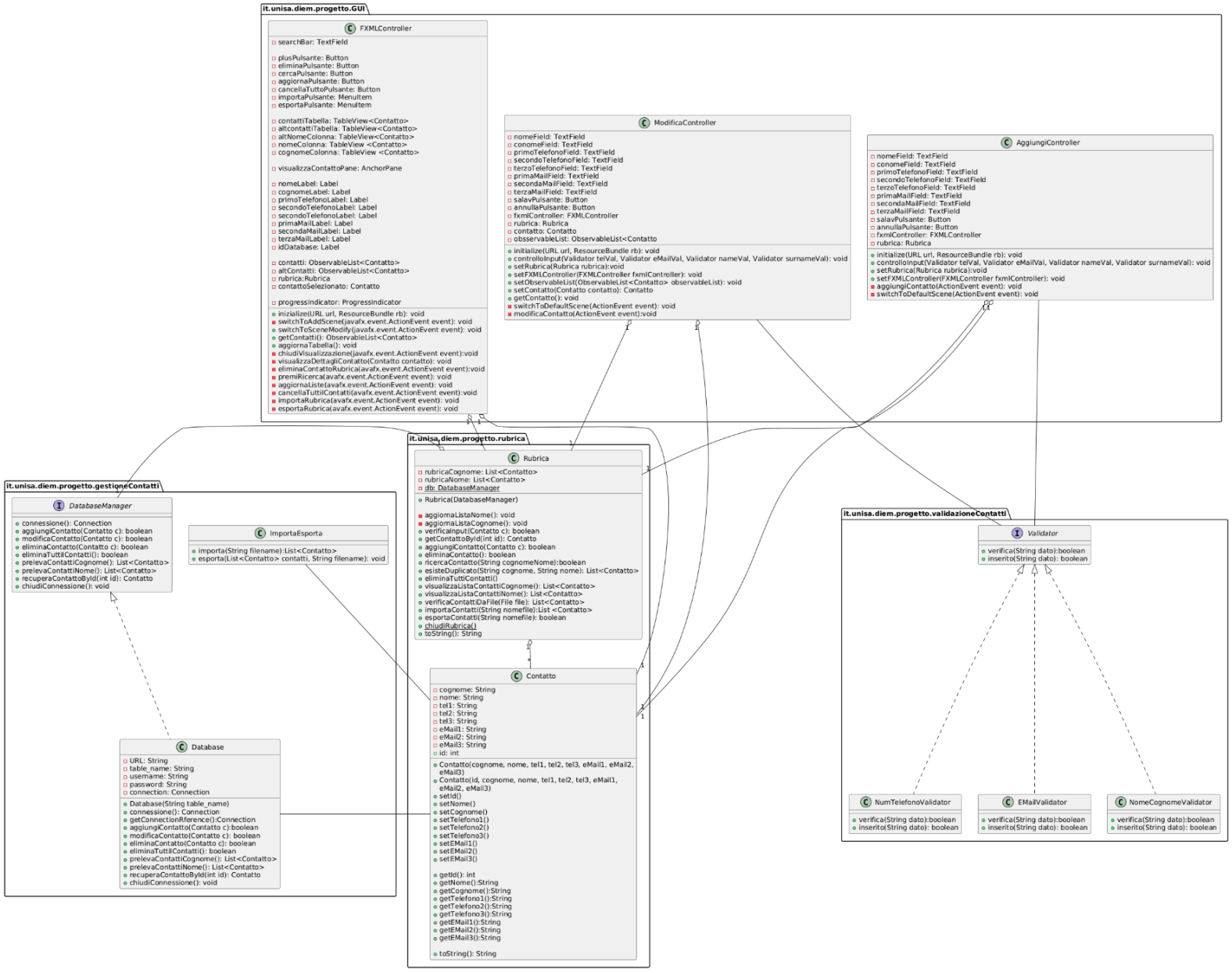
1. Diagramma delle classi(vecchio)

Il diagramma delle classi proposto è solo un diagramma concettuale, potrebbe subire delle modifiche nella prossima fase.



2. Diagramma delle classi(nuovo)

Il diagramma delle classi proposto è quello definitivo.



Discussione delle scelte di implementazione

Il progetto presenta alta coesione e basso accoppiamento, rendendolo facile da mantenere e ben strutturato. Vi è una chiara suddivisione delle responsabilità e una generale indipendenza tra i vari componenti.
La gestione delle responsabilità è visibile soprattutto dalle classi **Database** e **Contatto,** che si concentrano solo e unicamente, rispettivamente, sulle operazioni del database e sulla rappresentazione delle informazioni del contatto.
L' interfaccia **DatabaseManager** è stata introdotta con lo scopo di ridurre la dipendenza tra la classe **Rubrica** e la classe **Database**. In questo modo la classe Rubrica può essere riutilizzata in altri contesti insieme all'interfaccia DatabaseManager.
L'interfaccia **Validator** ha un livello di coesione di tipo funzionale poiché ha il compito di verificare che i dati inseriti dall'utente siano corretti.
In tal modo la classe **Rubrica** o **Contatto** non dovranno occuparsene rendendo il codice più leggibile.
In conclusione vi è una buona coesione tra le classi riuscendo comunque a mantenere un basso accoppiamento, garantendo efficienza operativa e sostenibilità a lungo termine.

➔ Principi di buona progettazione orientata agli oggetti

Le classi definite rispettano il Principio di Singola Responsabilità:

- Classe Conttato: contiene le informazioni del singolo contatto;
- Classe Rubrica: si occupa di riunire le informazioni di più contatti delegando la gestione degli stessi alla classe Database;
- Classe DataBase: si occupa solo della gestione della persistenza dei dai dei contatti;
- Classi che implementano Validator: si occupano di verificare la correttezza dei dati dei singoli contatti.

Le classi definite rispettano il Principio Aperto/Chiuso:

- Interfaccia DatabaseManager: permette di gestire la logica del DataBase, senza la modifica di altre classi
- Interfaccia Validator: permette la gestione di validazioni di campi senza che venga necessariamente mostrata la logica nei metodi delle classi che la utlizzano.

Le classi definite rispettano il Principio di Sostituzione di Liskov:

- I validatori “NomeCognomeValidator”, “NumTelefonoValidator”, “EMailValidator” possono essere usati al posto dell’interfaccia Validator senza compromettere il funzionamento

Le classi definite rispettano il Principio di Segregazione delle Interfacce:

- L’interfaccia DatabaseManager fornisce solo i metodi necessari per la gestione dei contatti su database e di apertura e chiusura della connessione al , è quindi molto specifica
- L’interfaccia Validator fornisce solo i metodi necessari per la validazione dei campi, è quindi molto specifica

Le classi definite rispettano il Principio di Inversione della Dipendenza:

- L’interfaccia DatabaseManager: permette di ridurre le dipendenze dalla classe Database, nascondendo tutto ciò che riguarda il database e le operazioni da effettuare su di esso.