# **Design della Rubrica Telefonica**

## **Scelte Architetturali**

### **Pattern MVC**

Il sistema utilizza il pattern Model-View-Controller per separare:

* **Model (Rubrica, Contatto):** gestione dei dati
* **View (FXML):** interfaccia utente
* **Controller:** logica applicativa e gestione eventi

### **Principi SOLID applicati**

* **Single Responsibility:**
  + **Contatto:** gestisce solo i dati di un contatto
  + **Rubrica:** gestisce solo la collezione di contatti
  + **NumeroTel/Email:** validazione e formattazione
* **Open/Closed:**
  + Facile aggiungere nuovi tipi di contatti o informazioni estendendo le classi base
* **Interface Segregation:**
  + Interfacce minimali per ogni componente
  + Separazione tra logica di business e presentazione

### **Coesione e Accoppiamento**

#### **Alta Coesione**

* Ogni classe ha responsabilità ben definite
* Contatto: gestisce solo i dati relativi a un singolo contatto
* Rubrica: gestisce solo operazioni sulla collezione

#### **Basso Accoppiamento**

* Controller comunica con il Model tramite interfacce
* View completamente separata dalla logica di business
* Utilizzo di eventi per la comunicazione View-Controller

## **Decisioni di Design**

### **Validazione**

* Validazione dei numeri di telefono e email a livello di classe specifica
* Controlli di business logic nel controller

### **Persistenza**

* Serializzazione CSV per salvare/caricare la rubrica

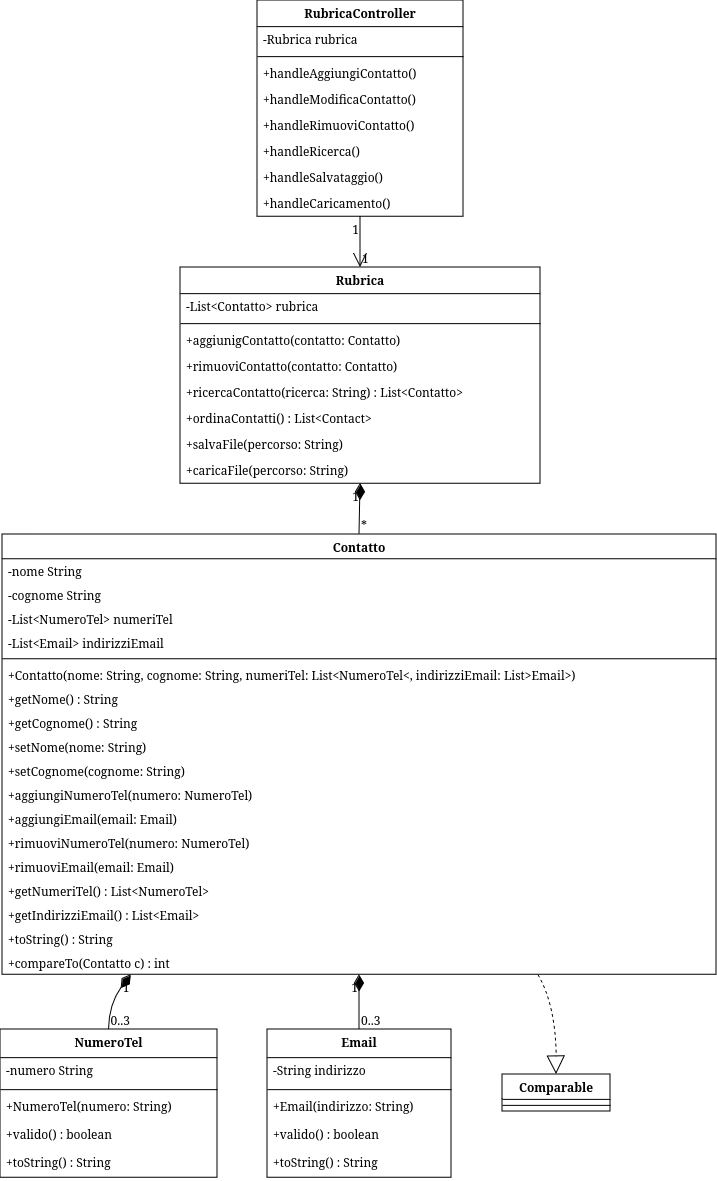
### **Ricerca e Ordinamento**

* Implementazione di Comparable per ordinamento naturale
* Ricerca case-insensitive su nome/cognome

## **Gestione Errori**

* Utilizzo di eccezioni custom per gestire errori di business
* Validazione input utente nel controller

## **Diagramma delle classi**



## **Diagramma delle sequenze**

