# Documentazione del Progetto Software

### 1. Diagramma dei Package

Il diagramma dei package suddivide il sistema in moduli principali: ‘main’, ‘interface’, ‘data’ e ‘annullaoperazione’. Questa struttura mostra una buona separazione delle preoccupazioni (Separation of Concerns). Ogni package ha una responsabilità chiara: ‘main’ avvia l’applicazione, ‘interface’ gestisce i controller e le interfacce grafiche, ‘data’ contiene le entità principali come Rubrica e Contatto, e ‘annullaoperazione’ gestisce l’operazione di annullamento.

### 2. Diagramma delle Classi

Il diagramma delle classi mostra un’implementazione che segue i principi SOLID. Ogni classe segue il principio della singola responsabilità (SRP): ad esempio, MainInterfaceController si occupa di gestire le operazioni dell'interfaccia principale, mentre Rubrica si occupa della gestione dei contatti.

Le relazioni tra classi sono ben definite e seguono i principi SOLID:

* SRP (Single Responsibility Principle): Ogni classe ha una responsabilità chiara, ad esempio MainInterfaceController gestisce l'interfaccia principale, mentre Rubrica si occupa della gestione dei contatti.
* OCP (Open-Closed Principle): Le classi sono progettate per essere estese senza necessità di modifiche dirette, favorendo la scalabilità.
* LSP (Liskov Substitution Principle): Le classi derivanti rispettano il contratto delle classi base, garantendo sostituibilità.
* ISP (Interface Segregation Principle): Le interfacce sono specifiche e ridotte, riducendo la dipendenza dai metodi non utilizzati.
* DIP (Dependency Inversion Principle): Le classi di alto livello dipendono da astrazioni, migliorando la modularità.

3. Diagrammi di Sequenza

I diagrammi di sequenza descrivono in dettaglio i flussi principali: aggiunta, modifica, eliminazione, importazione, esportazione e visualizzazione dei contatti.

I diagrammi di sequenza ci mostrano una chiara separazione dei ruoli tra le classi, seguendo i principi di:

* Ortogonalità: Ogni classe esegue solo le operazioni di sua competenza.
* Regola del Boy-Scout: Ogni interazione migliora la chiarezza e riduce il debito tecnico.
* KISS: Ogni flusso (aggiunta, modifica, eliminazione, importazione, esportazione) è rappresentato con semplicità e linearità.

### 4. Diagramma delle Attività

Il diagramma delle attività rappresenta un flusso operativo robusto, conforme al:

* Principio di robustezza: Gestione chiara degli errori e percorsi definiti per prevenire stati inconsistenti.
* DRY: Le decisioni e attività sono rappresentate una sola volta, evitando duplicazioni.
* Evitare l’ottimizzazione precoce: Il focus è sulla funzionalità corretta prima di eventuali ottimizzazioni.

## Decomposizione in Moduli

Decomposizione in Moduli

1. Modulo Main

Il modulo ‘Main’ contiene il punto di ingresso dell’applicazione, rappresentato dalla classe ‘Applicazione’. Responsabilità principale: avviare il sistema e coordinare gli altri moduli.

Coesione: Funzionale – il modulo realizza un compito ben definito (avvio del sistema).

Accoppiamento: Per dati – comunica con altri moduli tramite parametri essenziali, riducendo le dipendenze.

2. Modulo Interface

Il modulo ‘Interface’ gestisce l'interazione utente ed è suddiviso in due sottosezioni: ‘Controller’ e ‘FXML’. I controller gestiscono la logica, mentre la sezione FXML si occupa della rappresentazione grafica.

Coesione: Procedurale – i controller gestiscono operazioni frequentemente usate insieme, come input e feedback.

Accoppiamento: Per controllo – dipende dalle specifiche dei moduli ‘Data’ per accedere ai dati.

3. Modulo Data

Il modulo ‘Data’ contiene le entità principali del sistema: ‘Rubrica’ e ‘Contatto’. La Rubrica fornisce operazioni come aggiunta, modifica, eliminazione e ricerca, mentre Contatto rappresenta ogni singolo elemento.

Coesione: Comunicazionale – tutte le operazioni lavorano sugli stessi dati (contatti e rubrica).

Accoppiamento: Per dati – espone solo i dati necessari tramite metodi specifici, minimizzando dipendenze dirette.

4. Modulo Annulla Operazione

Il modulo ‘Annulla Operazione’ include una classe progettata per gestire l'annullamento di operazioni, prevenendo stati inconsistenti e garantendo robustezza.

Coesione: Temporale – tutte le operazioni vengono eseguite durante una specifica fase del processo.

Accoppiamento: Nessuno – il modulo opera indipendentemente, interagendo solo quando richiesto dai controller.