

Corso di Ingegneria del Software Deliverable di progetto	2024-2025
---	-----------

# “Ingegneria del Software” 2024-2025

Docente: Prof. Angelo Furfaro

## Gestore di Libreria Personale

Data	27/06/2025
Documento	Documento Finale

Team Members		
Nome e Cognome	Matricola	E-mail address
Francesco Pio Ruffo	240044	rffnc03h10d086z@stud enti.unical.it

# Sommario

---

**List of challenging/risky requirements or tasks .....3**

**A. Stato dell’Arte .....4**

**B. Raffinamento dei Requisiti.....5**

    B.1 Servizi (con prioritizzazione) .....5

    B.2 Requisiti non funzionali.....7

    B.3 Scenari d’uso dettagliati .....8

    B.4 Excluded Requirements .....13

    B.5 Assunzioni .....13

    B.6 Use Case Diagram .....14

**C. Architettura Software.....15**

    C.1 Static view of the system: .....15

    C.2 Dinamic view of the software architecture: .....17

**D. Dati e loro modellazione .....19**

**E. Scelte Progettuali (Design Decisions).....20**

**F. Progettazione di Basso Livello.....22**

**G. Soddisfacimento dei requisiti funzionali (FRs) e non funzionali(NFRs) .....28**

**Appendix. Prototype**

## List of Challenging/Risky Requirements or Tasks

Challenging Task	Date the task is identified	Date the challenge is resolved	Explanation on how the challenge has been managed
Implementazione della classe Libro (considerando l'elevato numero di attributi)	16-05-2025	16-05-2025	Utilizzando il design pattern Builder si è reso il costruttore della classe Libro (centrale nel progetto) facilmente utilizzabile e immediatamente chiaro.
Supportare la funzionalità di ricerca dei libri in modo robusto	18-05-2025	18-05-2025	Implementazione di un metodo di 'cleaning' dell'input, al fine di rimuovere caratteri speciali, segni di punteggiatura, spazi...
Scegliere dinamicamente la strategia per ordinamento e filtraggio della libreria	20-05-2025	21-05-2025	Utilizzo del FactoryMethod per selezionare dinamicamente la strategia adeguata di ordinamento e filtraggio in base all'input scelto dall'utente
Implementazione dell'interfaccia grafica	20-05-2025	24-05-2025	Utilizzo di Java Swing

## A. Stato dell'Arte

I software che consentono di gestire e catalogare una collezione personale di libri rappresentano degli strumenti largamente diffusi e spesso indispensabili per la corretta fruizione dell'attività di lettura.

La possibilità di consultare e apportare delle modifiche ai propri volumi, di visualizzarli secondo criteri personalizzati e dinamici, tenendo traccia delle diverse caratteristiche di ogni libro, rientrano tra le esigenze comuni di un'ampia varietà di utenti.

Negli anni, sono stati lanciati sul mercato diversi sistemi per la gestione di una libreria. Quelli maggiormente utilizzati e conosciuti sono:

- ❖ Calibre: oltre a consentire l'organizzazione e l'ordinamento dei volumi, implementa funzionalità avanzate per la modifica dei metadati
- ❖ LibraryThing: la classica aggiunta manuale dei libri viene ottimizzata con uno strumento di ricerca avanzato su varie piattaforme online (Amazon Books, Overcat, British library). Ciò consente di automatizzare la fase di inserimento delle informazioni di ogni libro.
- ❖ Libib: immettendo il codice ISBN recupera automaticamente le informazioni del libro che si vuole inserire. Presenta un'interfaccia grafica moderna e arricchisce l'esperienza generale di utilizzo con varie statistiche (numero di elementi aggiunti per mese e per anno...)
- ❖ BookCollector: consente l'aggiunta di un libro scannerizzando il codice a barre (scaricando la versione mobile) e supporta la visualizzazione dei libri riproducendo una libreria reale.

I punti di forza dei sistemi descritti vertono quindi principalmente sull'automatizzare l'inserimento dei libri, provvedendo in alcuni casi anche funzionalità avanzate.

L'applicazione implementata si pone l'obiettivo di coniugare in un unico sistema software tutte le operazioni fondamentali atte a gestire la propria libreria con un'esperienza d'uso semplice e intuitiva, supportata da una user interface moderna ed essenziale.

## B. Raffinamento dei Requisiti

### *B.1 Servizi (con prioritizzazione)*

I servizi descritti saranno prioritizzati in base a due scale:

- Importanza (alta, media, bassa)
- Complessità (alta, media, bassa)

#### 1. Aggiunta libro

Importanza: ALTA, Complessità: MEDIA

L'utente deve essere in grado di poter inserire un libro, specificandone i dettagli (titolo, autore, ISBN, genere, stato lettura e valutazione).

#### 2. Modifica libro

Importanza: MEDIA, Complessità: BASSA

L'utente deve poter modificare i dettagli dei libri precedentemente inseriti

#### 3. Rimozione libro

Importanza: MEDIA, Complessità: BASSA

L'utente deve poter rimuovere un libro precedentemente inserito

#### 4. Visualizzazione libreria

Importanza: ALTA, Complessità: MEDIA

L'utente deve poter visualizzare dinamicamente i libri catalogati in una vista d'insieme (tabellare o lista), al fine di interagire efficacemente con il sistema.

#### 5. Ordinamento libreria

Importanza: MEDIA, Complessità: MEDIA

L'utente deve essere in grado di personalizzare l'ordinamento della libreria sulla base di vari criteri (in base all'ordine di inserimento, all'autore, al titolo o alla valutazione dell'utente).

**6. Ricerca di un libro**

Importanza: ALTA, Complessità: ALTA

L'utente deve essere in grado di ricerca un libro in base all'autore, al titolo o all'ISBN

**7. Filtraggio dei libri**

Importanza: MEDIA, Complessità: MEDIA

L'utente deve essere in grado di personalizzare la visualizzazione della libreria sulla base di meccanismo di filtro (per genere, per stato lettura).

**8. Salvataggio e ripristino da file system**

Importanza: ALTA, Complessità: MEDIA

L'utente deve poter effettuare il salvataggio ed il ripristino della libreria dalla memoria secondaria

## *B.2 Requisiti non Funzionali*

### ❖ **Correttezza**

Il sistema deve soddisfare le specifiche funzionali e non funzionali stabilite durante l'analisi dei requisiti

### ❖ **Usabilità**

Il sistema software deve prestarsi ad essere utilizzato in modo facile e intuitivo da parte degli utenti, accrescendo la soddisfazione dell'utente stesso nell'effettuare le operazioni supportate dal sistema

### ❖ **Robustezza**

Il sistema deve comportarsi in modo accettabile anche nelle circostanze non previste dalle specifiche

### *B.3 Scenari d'uso dettagliati*

#### ❖ *Aggiunta di un libro*

Precondizione: l'utente si trova nella schermata principale della libreria

Postcondizione: l'utente visualizzerà il libro aggiunto all'interno della libreria

Descrizione:

1. L'utente clicca sul bottone 'Aggiungi libro'
2. Viene aperto un pannello di dialogo contenente i campi obbligatori e facoltativi
3. L'utente inserisce le informazioni del libro
4. L'utente clicca su 'Salva'
5. Il libro comparirà nella tabella e contemporaneamente si aggiornerà di +1 il contatore

Estensione:

- 4.a L'utente non ha inserito uno dei parametri obbligatori
  - 4.a.1 Viene visualizzato il messaggio di errore 'Compila tutti i campi obbligatori'
  - 4.a.2 L'utente clicca su 'Ok'
  - 4.a.3 Viene riaperto il pannello contenente i campi del libro

#### ❖ **Modifica di un libro**

Precondizione: l'utente si trova nella schermata principale della libreria e ha precedentemente inserito il libro da voler modificare

Postcondizione: l'utente visualizzerà il libro al quale ha apportato le modifiche

Descrizione:

1. L'utente clicca sulla riga della tabella per selezionare il libro che vuole modificare (la riga selezionata diventerà con sfondo nero)
2. L'utente clicca sul bottone 'Modifica libro'
3. Viene aperto un pannello di dialogo contenente le informazioni del libro inserite in fase di aggiunta
4. L'utente effettua la modifica
5. L'utente clicca su 'Modifica'
6. Il libro comparirà nella tabella con le informazioni appena modificate



Estensione:

- 4.a L'utente inserisce le informazioni di un libro già presente
  - 4.a.1 Viene visualizzato il messaggio di errore 'Libro con questi dati è già presente!'
  - 4.a.2 L'utente clicca su 'Ok'
  - 4.a.3 Viene riaperto il pannello contenente i campi del libro

❖ **Rimozione di un libro**

Precondizione: l'utente si trova nella schermata principale della libreria ed è almeno presente il libro che si vuole rimuovere

Postcondizione: l'utente visualizzerà la libreria senza il libro da rimuovere

Descrizione:

1. L'utente clicca sulla riga della tabella per selezionare il libro che vuole rimuovere
2. L'utente clicca sul bottone 'Elimina libro'
3. Il libro verrà eliminato dalla visualizzazione tabellare ed il contatore verrà decrementato di 1

Estensione

- 1.a L'utente non ha selezionato la riga
  - 1.a.1 Non viene apportata alcuna modifica ai libri; l'utente rimane nella schermata iniziale

❖ **Salvataggio su file system**

Verrà descritto come un unico caso d'uso, in quanto il salvataggio tra formato json e csv differisce solo per la scelta, da parte dell'utente, di una delle due modalità di salvataggio.

Precondizione: l'utente si trova nella schermata principale della libreria

Postcondizione: la libreria verrà salvata su file .json/.csv

Descrizione:

1. L'utente clicca sul bottone JSON/CSV
2. Compare il messaggio di 'operazione avvenuta con successo'
3. L'utente clicca sul bottone 'OK'
4. L'utente ritorna nella schermata principale

❖ ***Caricare le informazioni da file system***

Verrà descritto come un unico caso d'uso, in quanto il ripristino differisce solo per la scelta, da parte dell'utente, del file.

Precondizione: il file esista e sia valido

Postcondizione: la libreria verrà caricata e visualizzata nella schermata principale del programma

Descrizione:

1. L'utente clicca sul bottone 'Carica libreria'
2. Viene visualizzata la finestra di dialogo per la selezione del file json/csv, aperta di default nella cartella Downloads
3. L'utente seleziona il file da voler ripristinare
4. L'utente clicca su 'Open'
5. Viene visualizzato il messaggio di 'operazione avvenuta con successo'
6. L'utente preme 'OK'
7. Viene visualizzata la schermata principale

Estensione:

- 3a L'utente non vuole più importare la libreria
  - 3.a.1 L'utente clicca su 'Cancel'
  - 3.a.2 Viene visualizzato il messaggio 'Errore nell'importazione'
  - 3.a.3 L'utente preme 'OK'
  - 3.a.4 Viene visualizzata la schermata principale

❖ ***Ordinamento***

Precondizione: l'utente si trova nella schermata principale della libreria

Postcondizione: la libreria verrà ordinata sulla base del criterio selezionato

Descrizione:

1. L'utente clicca sul menù a tendina
2. Vengono visualizzate le possibili opzioni di ordinamento (per ordine di inserimento, per autore, per titolo, per valutazione)
3. L'utente seleziona il criterio
4. Viene visualizzata la libreria con i libri ordinati

❖ **Filtraggio**

Precondizione: l'utente si trova nella schermata principale della libreria

Postcondizione: i libri verranno filtrati sulla base del criterio selezionato

Descrizione:

1. L'utente clicca sul menù a tendina
2. Vengono visualizzate le possibili opzioni di filtraggio (nessun filtro\*, per genere, per stato lettura)
3. L'utente seleziona il criterio
4. Viene visualizzato, di fianco al criterio scelto, un secondo menù a tendina che consente di selezionare sulla base di quale genere (nel caso in cui l'utente scelga *per genere*) o sulla base di quale stato lettura (nel caso in cui l'utente scelga *per stato lettura*) effettuare il filtraggio.
5. L'utente esplicita la sua scelta
6. Viene visualizzata la libreria con i soli libri che rispettano le scelte effettuate dall'utente

\*Oss.

'Nessun filtro' consente di visualizzare per intero la libreria, considerando tutti i libri inseriti

❖ **Ricerca**

Precondizione: l'utente si trova nella schermata principale della libreria ed è presente almeno un libro

Postcondizione: verranno visualizzati il/i libri conformi alla stringa query

Descrizione:

1. Cliccando su uno dei tre criteri proposti (per autore, per titolo, per ISBN) l'utente seleziona in base a quale criterio effettuare la ricerca.
2. L'utente scrive la stringa query all'interno della barra di ricerca .
3. L'utente clicca sul bottone 'lente ingrandimento'
4. Viene visualizzata la libreria con i soli libri che rispettano le scelte effettuate dall'utente

Estensione:

3a L'utente ricerca un attributo non presente in alcun campo della libreria

3.a.1 Viene visualizzata una schermata vuota

❖ ***Ripristina vista***

Precondizione: l'utente si trova nella schermata principale della libreria

Postcondizione: verrà ripristinata la vista originale della tabella (non visualizzando più i risultati derivanti dall'applicazione di eventuali criteri di filtraggio/ordinamento/ricerca)

Descrizione:

1. L'utente clicca 'Ripristina vista'
2. Viene visualizzata la vista originale della libreria

#### ***B.4 Excluded Requirements***

Tutti i requisiti presenti all'interno della specifica sono stati implementati

#### ***B.5 Assunzioni***

Si assume che un libro *non* possa essere inserito all'interno della libreria se non sono specificati almeno i parametri obbligatori: *titolo*, *cognome dell'autore* e codice *ISBN*. Gli attributi sopraindicati sono stati infatti considerati indispensabili per la corretta e non ambigua catalogazione del libro nel sistema

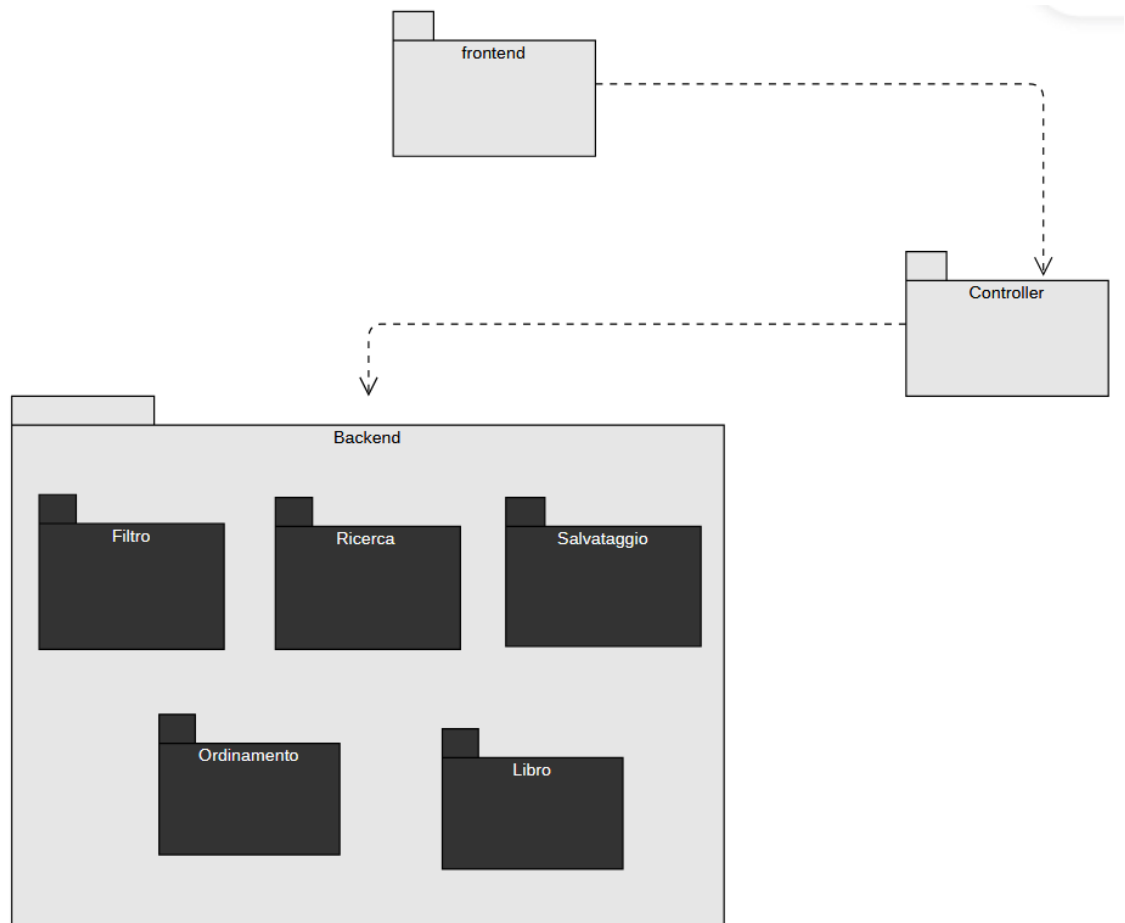
### A.6 Use Case Diagrams



## C. Architettura Software

### *C.1 The static view of the system: Package and Class Diagram*

#### Package diagram



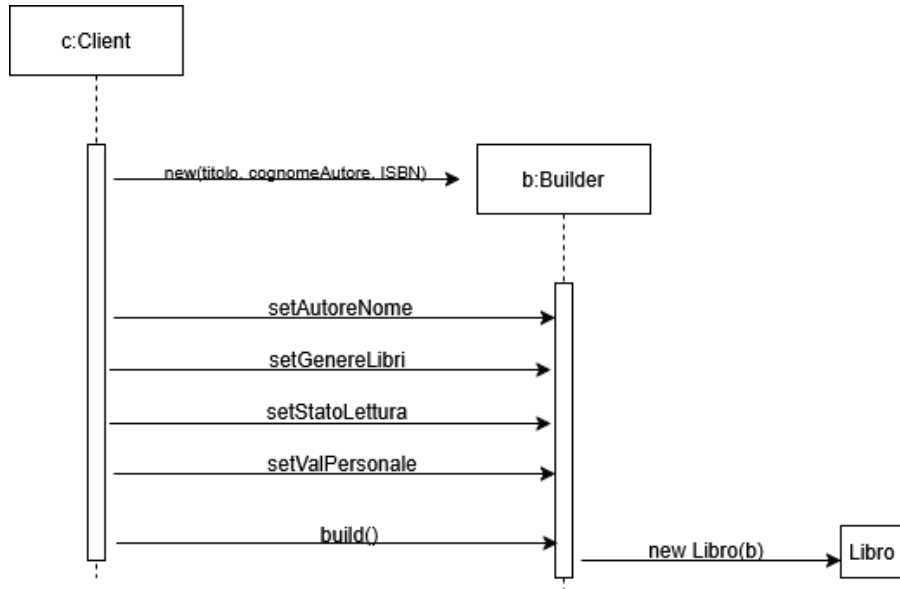
The flowchart illustrates the proposed algorithm for solving the multi-objective problem. It begins with 'Problem Formulation' and 'Initial Population Generation'. The process then branches into two main paths: 'Multi-Objective Evolutionary Algorithm' and 'Multi-Objective Genetic Algorithm'. The 'Multi-Objective Evolutionary Algorithm' path involves 'Non-dominated Sorting', 'Crowding Distance Calculation', and 'Selection'. The 'Multi-Objective Genetic Algorithm' path involves 'Crossover', 'Mutation', and 'Selection'. Both paths lead to 'Pareto Frontier Extraction' and 'Final Solution Set'. The flowchart is divided into two main sections: 'Multi-Objective Evolutionary Algorithm' and 'Multi-Objective Genetic Algorithm'.

\*Il class diagram può essere visualizzato nelle sue dimensioni originali al seguente link:  
<https://drive.google.com/file/d/1yY36Kppjcx2sDWplZft2KAphz6KbGo1e/view?usp=sharing>

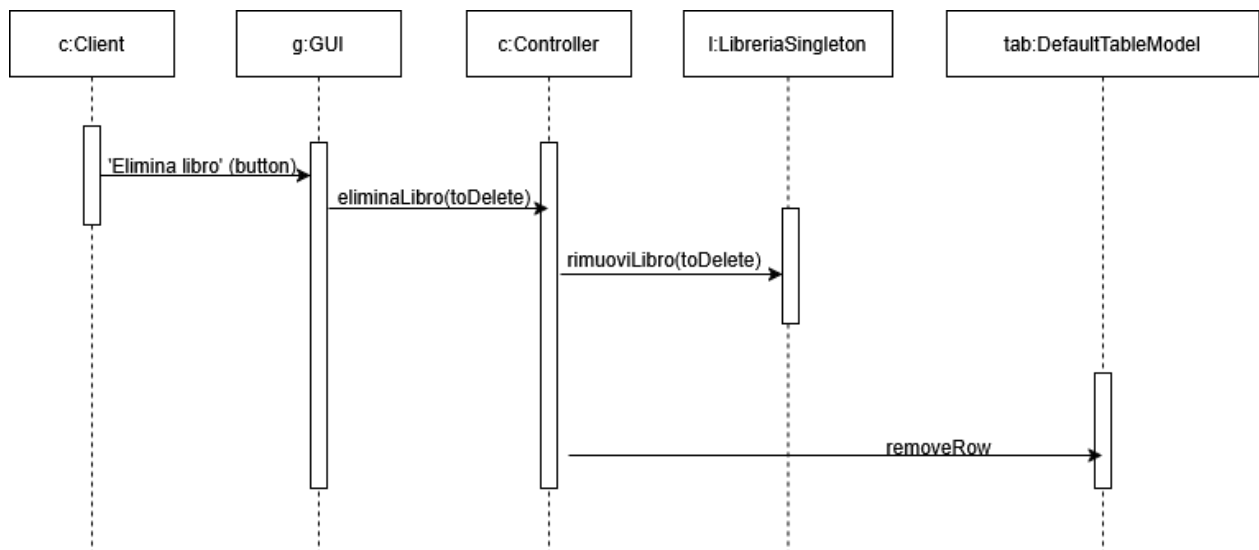


## *C.2 The dynamic view of the software architecture: Sequence Diagram*

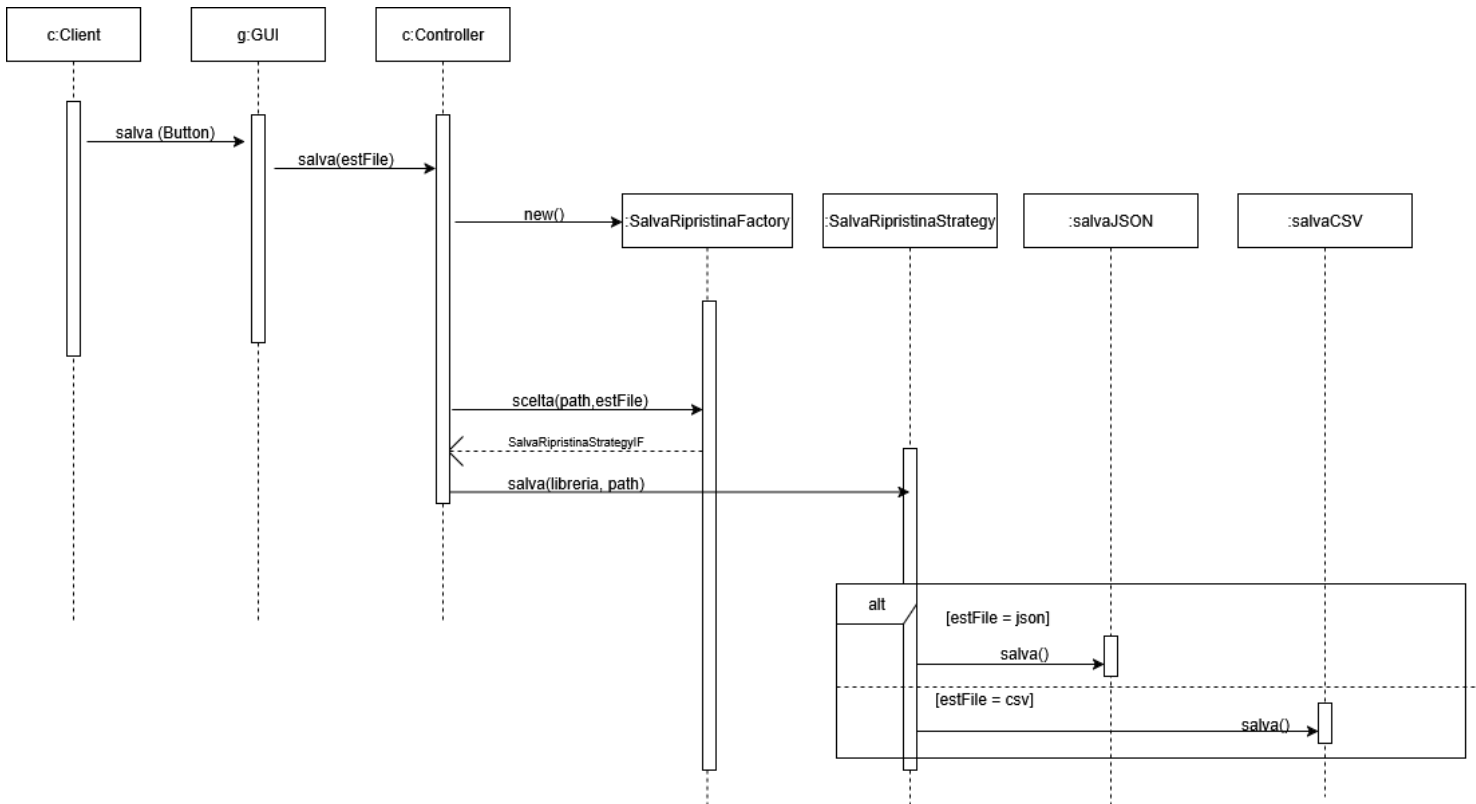
### Creazione di un Libro con Builder



### Eliminazione



### Salvataggio di un libro



L'esemplificazione relativa al salvataggio del Libro è del tutto simile alle operazioni di filtraggio, ordinamento e ricerca.

## D. Dati e loro modellazione (se il sistema si interfaccia con un DBMS)

Il sistema supporta il salvataggio ed il ripristino della libreria sia in formato JSON (avvalendosi dell'utilizzo dell'apposita libreria '*Gson*') che in formato CSV, al fine di garantire all'utente massima flessibilità per la gestione della persistenza su memoria secondaria.

Di default il file contenente il salvataggio della libreria si trova all'interno della cartella Download, al fine di favorire l'usabilità ed evitare che l'utente debba specificare manualmente un path. Il tutto avviene in modo compatibile con i principali sistemi operativi, in quanto il percorso della directory viene rilevato dinamicamente.

## E. Scelte Progettuali (Design Decisions)

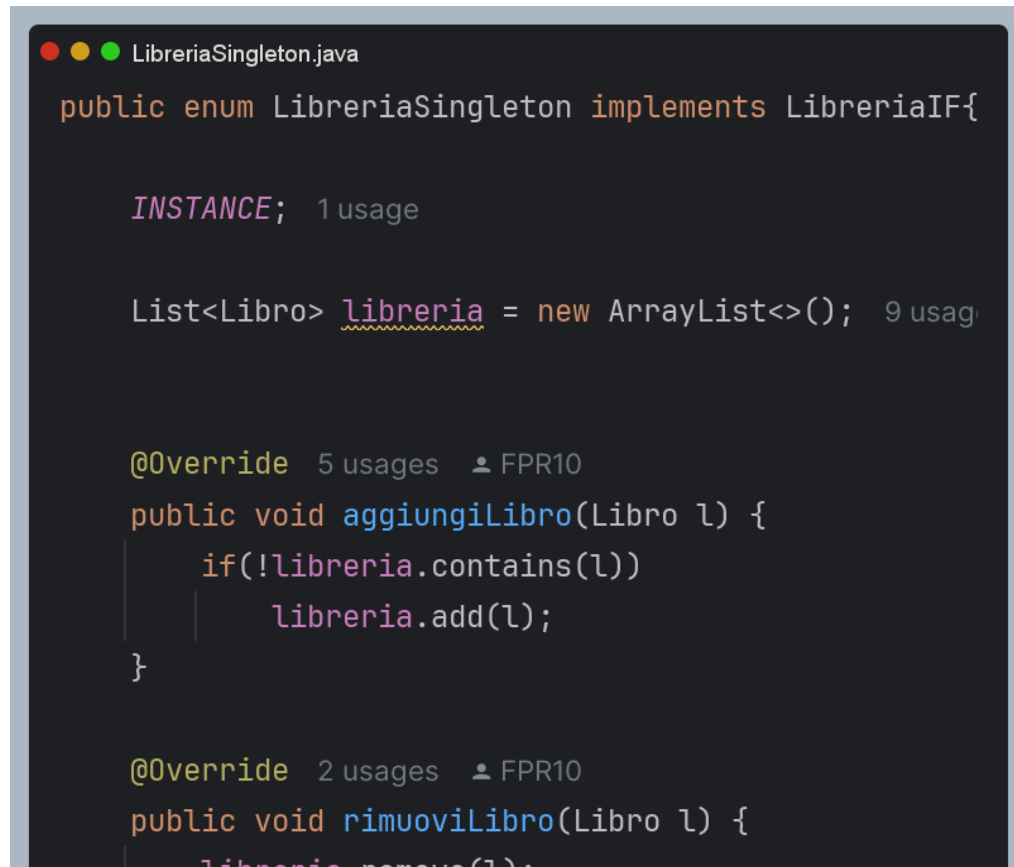
---

- ❖ L'oggetto Libreria rappresenta il fulcro dell'intera applicazione. Si è scelto di implementarla come **Singleton** al fine di garantire l'esistenza di un'unica istanza condivisa. Questo approccio consente di semplificare la gestione della libreria stessa ed evita l'insorgenza di possibili inconsistenze dei dati.
- ❖ Tra le funzionalità principali che il sistema offre, sono presenti diversi possibili meccanismi di ordinamento, filtraggio e ricerca. Le diverse strategie sono accumulate dalla stessa logica di base, che si diversifica solo per la decisione dell'utente.  
Si è dunque definita, mediante l'utilizzo del pattern **Strategy**, una serie di strategie concrete, che implementano un'interfaccia comune.  
Questa scelta consente di:
  - mantenere il codice leggibile e facilmente manutenibile (in quanto ogni strategia ha una sua classe concreta)
  - facilitare l'aggiunta di una nuova strategia senza dover in alcun modo modificare le implementazioni delle strategie già presenti
- ❖ La creazione di oggetti Libro rappresenta un'operazione fondamentale all'interno del sistema.  
La numerosità degli attributi che caratterizzano un Libro e la necessità di 'forzare' l'inserimento dei parametri obbligatori, mantenendo un adeguato tasso di flessibilità per quelli opzionali ha reso necessario l'utilizzo del pattern **Builder**.  
Si *evita* dunque potenziale ambiguità nell'inserire nell'ordine corretto gli attributi del Libro che stiamo creando (tipico di altre soluzioni, come quella dei costruttori telescopici), prediligendo la leggibilità.

- ❖ Si è reso necessario utilizzare il **FactoryMethod** per istanziare dinamicamente la strategia di ordinamento/filtraggio in base alla scelta effettuata dall'utente.  
Il Controller non viene così a conoscenza di quali siano le classi concrete di ordinamento/filtraggio, delegando la logica di creazione al Factory.
- ❖ Per garantire una corretta separazione tra la logica applicativa e la componente grafica, si è scelto di disaccoppiare le due.  
Tale approccio, coerente con il principio di separazione delle responsabilità, consente di rendere il sistema più modulare e flessibile, facilitando la manutenzione e l'evoluzione indipendente delle due componenti.  
Si favorisce così la riusabilità del codice di backend anche in contesti diversi.

## F. Progettazione di Basso Livello

### Libreria - Singleton



```
LibreriaSingleton.java

public enum LibreriaSingleton implements LibreriaIF{

    INSTANCE; 1 usage

    List<Libro> libreria = new ArrayList<>(); 9 usag

    @Override 5 usages ⓘ FPR10
    public void aggiungiLibro(Libro l) {
        if(!libreria.contains(l))
            libreria.add(l);
    }

    @Override 2 usages ⓘ FPR10
    public void rimuoviLibro(Libro l) {
        libreria.remove(l);
    }
}
```

L'implementazione della Libreria utilizzando il pattern Singleton consente di assicurare che la classe abbia una sola istanza e che l'accesso globale ai dati sia centralizzato in un unico punto, garantendo integrità.

Si è optato per l'impiego dell'enumeration, in quanto essa garantisce sicurezza nelle fasi di serializzazione e deserializzazione e assicura la thread safety (*senza* quindi rendere necessari metodi ausiliari del tipo `readResolve()` o il `getInstance()` `synchronized`).

## Libro - Builder

```
Libro.java

public class Libro {  FPR10
    private String titolo;  4 usages
    private String autoreNome;  6 usages
    private String autoreCognome;  6 usages
    private String ISBN;  7 usages
    private Genere_Libri genLib;  4 usages
    private Valutazione_Personale valPers;
    private Stato_Lettura statLett;  4 usag
```

Per la creazione dell'oggetto Libro si è reso necessario l'utilizzo del pattern Builder, che ne facilita le operazioni di costruzione (trattandosi di un oggetto complesso con un elevato numero di parametri).

```
Libro.java

//Definizione del Builder come inner class
public static class Builder{  28 usages  FPR10
    //Parametri obbligatori
    private String titolo;  2 usages
    private String autoreCognome;  2 usages
    private String ISBN;  2 usages

    //Opzionali
    private String autoreNome;  2 usages
    private Genere_Libri genLib;  2 usages
    private Stato_Lettura statLett;  2 usages
    private Valutazione_Personale valPers;  2 usages

    public Builder (String titolo,String autoreCognome, String ISBN){
        this.titolo = titolo;
        this.autoreCognome = autoreCognome;
        this.ISBN = ISBN;
    }
}
```

```
public Builder setAutoreNome (String an){ 4 usages  FPR10
    autoreNome=an;
    return this;
}

public Builder setGenereLibri (Genere_Libri gl){ 15 usages  FPR10
    genLib=gl;
    return this;
}

public Builder setStatoLettura (Stato_Lettura sl){ 13 usages  FPR
    statLett = sl;
    return this;
}

public Builder setValutazionePersonale (Valutazione_Personale vp){
    valPers = vp;
    return this;
}

public Libro build() { return new Libro( b: this); }
}
```

Il Builder viene definito come inner class e contiene tutti i campi della classe Libro: il costruttore include i soli parametri obbligatori, mentre quelli opzionali possono essere aggiunti mediante gli appositi setter.

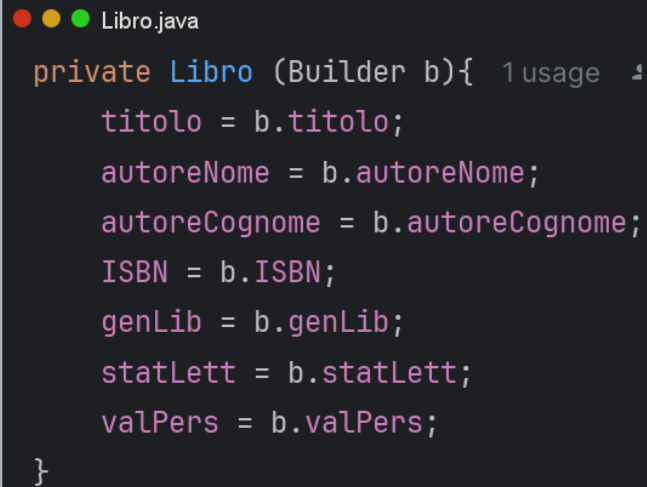
La particolarità di questa tipologia di implementazione riguarda il fatto che i setter, contrariamente al solito, non siano void ma di tipo Builder.

Questo consente, in fase di creazione dell'oggetto, di invocare i vari setter in sequenza:

```
LibriEsempio.java
Libro esempio = new Libro.Builder( titolo: "NomeLibro", autoreCognome: "CognomeAutore", ISBN: "1")
    .setStatoLettura(Stato_Lettura.IN_LETTURA)
    .setGenereLibri(Genere_Libri.SCIENTIFICO)
    .build();
```

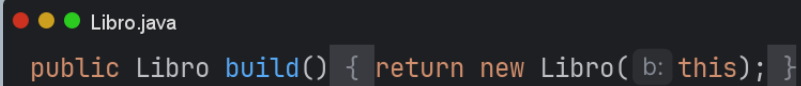


Il costruttore di Libro è private:



```
Libro.java  
  
private Libro (Builder b){  
    titolo = b.titolo;  
    autoreNome = b.autoreNome;  
    autoreCognome = b.autoreCognome;  
    ISBN = b.ISBN;  
    genLib = b.genLib;  
    statLett = b.statLett;  
    valPers = b.valPers;  
}
```

possiamo infatti invocarlo solo dal Builder:



```
Libro.java  
  
public Libro build() { return new Libro(b: this); }
```

### Salvataggio, ricerca, ordinamento, filtro – Factory Method e Strategy

La combinazione dei due pattern viene utilizzata per tutte le funzionalità sopracitate; poiché la logica di base è sostanzialmente simile, verrà di seguito descritta la progettazione e l'implementazione nel caso del Filtro.

```
FiltroFactoryIF.java  
  
public interface FiltroFactoryIF { 2 usages 1 implementation  FPR10  
  
    FiltroStrategyIF setStrategy (String paramFiltro,String tipoFiltro);  
}
```

L'interfaccia FiltroFactoryIF definisce il metodo setStrategy, che ha il compito di restituire un oggetto che implementa l'interfaccia FiltroStrategyIF e che cambierà dinamicamente in base alla tipologia di filtro che si vuole impiegare:

```
FiltroStrategyIF.java  
  
public interface FiltroStrategyIF {  
    List<Libro> filtra (); 7 usages  
}
```

La classe concreta FiltroFactory ‘sceglie’ quale strategia di filtraggio adottare, delegando poi la creazione dell'oggetto concreto:

```
FiltroFactory.java  
  
public class FiltroFactory implements FiltroFactoryIF{ 1 usage  FPR10  
  
    @Override 1 usage  FPR10  
    public FiltroStrategyIF setStrategy(String paramFiltro, String tipoFiltro) {  
        switch(tipoFiltro){  
            case "genere": return new FiltroGenere(Genere_Libri.valueOf(paramFiltro));  
            case "stato lettura": return new FiltroStatoLettura(Stato_Lettura.valueOf(paramFiltro.replace(target: " ", replacement: "_")));  
            default: return new NessunFiltro();  
        }  
    }  
}
```

Le classi concrete implementano l'operazione filtra() in modo indipendente, in base alla tipologia di filtraggio che è stata scelta:

```
FiltroGenere.java
public class FiltroGenere implements FiltroStrategyIF { 9 usages  FPR10

    private final Genere_Libri genere; 2 usages

    public FiltroGenere(Genere_Libri genere) { this.genere = genere; }

    @Override 7 usages  FPR10
    public List<Libro> filtra() {
        LibreriaSingleton l = LibreriaSingleton.INSTANCE;
        List<Libro> ret = new ArrayList<>();
        for (Libro lib : l.getLibreria()){
            if (lib.getGenLib() != null && lib.getGenLib().equals(genere))
                ret.add(lib);
        }
        return ret;
    }
}
```

```
FiltroStatoLettura.java
public class FiltroStatoLettura implements FiltroStrategyIF { 8 usages  FPR10

    private final Stato_Lettura sl; 2 usages

    public FiltroStatoLettura (Stato_Lettura sl) { this.sl = sl; }

    @Override 7 usages  FPR10
    public List<Libro> filtra() {
        LibreriaSingleton l = LibreriaSingleton.INSTANCE;
        List<Libro> ret = new ArrayList<>();
        for (Libro lib : l.getLibreria()){
            if (lib.getStatLett() != null && lib.getStatLett().equals(sl))
                ret.add(lib);
        }
        return ret;
    }
}
```

Utilizzando questa logica, la selezione viene centralizzata in un solo punto ed il client non viene a conoscenza di quelli che sono i dettagli implementativi della strategia adottata.

## G. Spiegare come il progetto soddisfa i requisiti funzionali (FRs) e quelli non funzionali (NFRs)

### Requisiti funzionali (FRs)

1. Aggiunta libro

L'utente può inserire le informazioni caratterizzanti del libro all'interno dell'apposito pannello di dialogo. Una volta terminato il processo di inserimento, verrà richiamato il metodo SalvaLibro() del Controller, che a sua volta richiamerà il metodo aggiungiLibro() di LibreriaSingleton, che implementa la logica di base dell'inserimento.

2. Modifica libro

L'utente, similmente all'aggiunta, avrà a disposizione le informazioni (già inserite precedentemente) del Libro da modificare mediante apposito pannello di dialogo. Anche in questo caso, il metodo ModificaLibro() del Controller richiamerà il metodo modificaLibro() di LibreriaSingleton.

3. Rimozione libro

L'eliminazione del libro viene garantita dall'invocazione del metodo eliminaLibro() del Controller, che richiama il metodo rimuoviLibro() di LibreriaSingleton.

4. Visualizzazione libreria

La visualizzazione dinamica e sempre aggiornata della libreria, essenziale per utilizzare e dialogare efficacemente con il sistema, viene garantita mediante i diversi metodi che vengono invocati a seguito delle principali operazioni effettuate (aggiungiLibroGUI(), aggiornaLibroGUI(), aggiornaTabellaGUI()).

5. Ricerca, Ordinamento e Filtraggio della libreria

Sono le operazioni che vengono performati mediante l'utilizzo incrociato dei pattern Strategy e FactoryMethod (come discusso al punto E). A seguito della scelta di ordinamento/filtraggio/tipologia di ricerca adoperata dall'utente, la selezione viene passata al FactoryMethod, che si occupa di istanziare dinamicamente la strategia corretta.

6. Salvataggio e ripristino da file system

Come illustrato nel punto D, il salvataggio avviene mediante la deserializzazione della libreria nei formati json/csv; viene richiamato il metodo salva() che invocherà a sua volta la logica di deserializzazione presente nel package 'salvataggio'.

Avendo a disposizione un file valido, è possibile poi ripristinare la libreria.

**Requisiti non funzionali (NFRs)**

1. Correttezza

I vari componenti del sistema sono stati testati sia singolarmente che tenendo conto del sistema nel suo complesso, verificando che il software funzionasse come previsto.

2. Usabilità

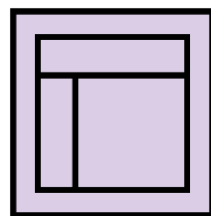
L'interfaccia grafica è stata implementata avendo come punto di riferimento la semplicità e la compattezza: tutte le funzionalità sono immediatamente visibili non appena ci si trova nella schermata principale.

3. Robustezza

Il sistema gestisce le possibili anomalie che potrebbero presentarsi durante l'esecuzione. L'utente, anche in caso di errore, viene 'guidato', mediante appositi box in cui viene segnalato il problema, nella risoluzione dello stesso.

## Appendix. Prototype

Si riporta di seguito una visualizzazione dell'applicazione in esecuzione, mediante l'utilizzo di screenshots, al fine di mostrare visivamente il soddisfacimento dei requisiti funzionali e non funzionali



### Panoramica generale

Appena avviata, l'applicazione si presenta così:

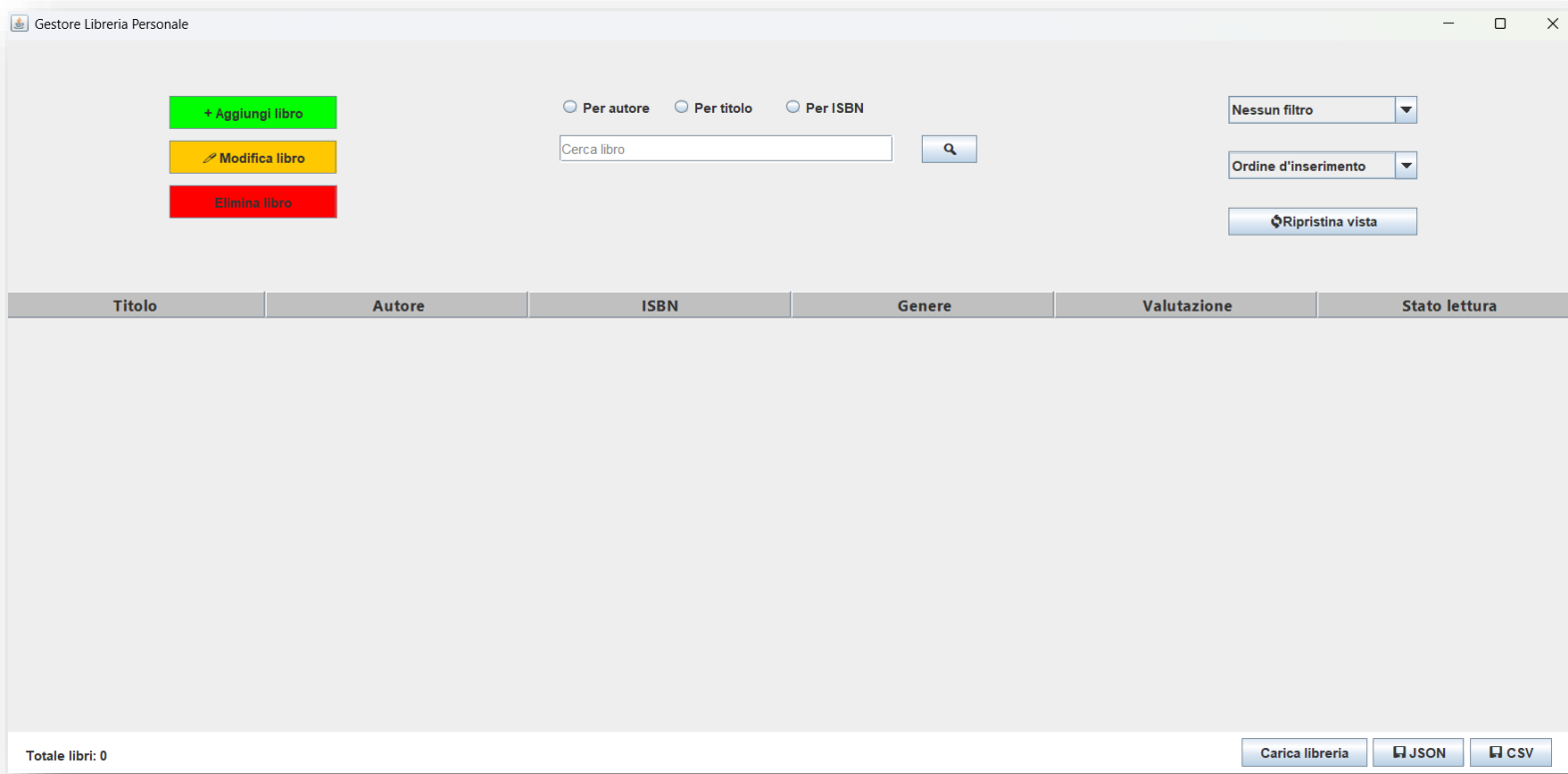


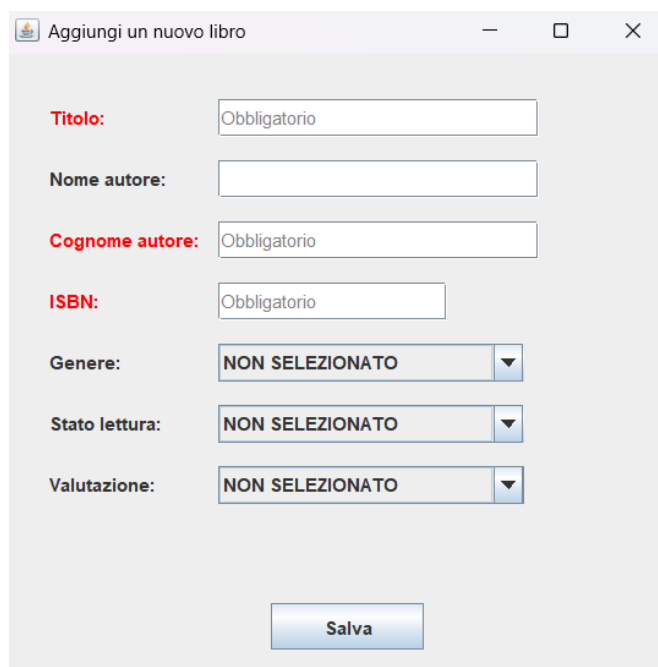
Figura 1: Avvio applicazione

Come si evince, tutte le funzionalità sono immediatamente visibili nella schermata principale: così facendo, si accresce la facilità con la quale un utente può, anche per la prima volta, interfacciarsi con il sistema e rispettare quindi il requisito non funzionale dell'usabilità.

Le principali operazioni di aggiunta, modifica ed eliminazione del libro vengono evidenziate con colori che ne richiamano l'importanza.

### *Aggiunta di un libro*

Per aggiungere un libro, una volta premuto il button ‘Aggiungi libro’, comparirà un pannello dedicato, con tutti i campi che caratterizzano il volume che si vuole aggiungere:



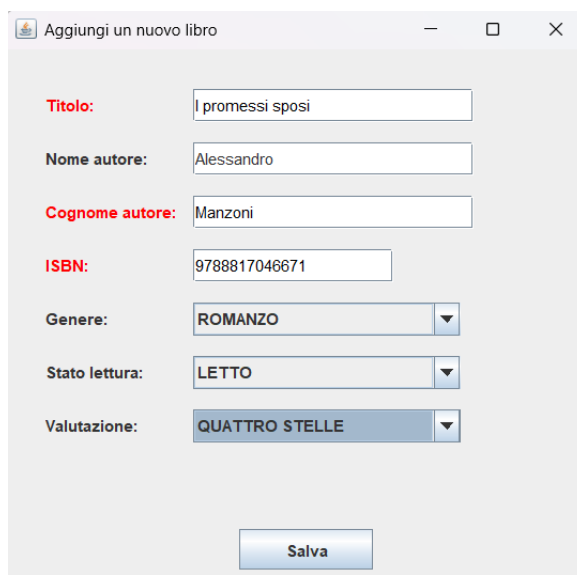
The screenshot shows a window titled "Aggiungi un nuovo libro" with a standard Windows-style title bar. Inside the window, there is a form with the following fields:

- Titolo:** A text input field with the placeholder text "Obbligatorio".
- Nome autore:** A text input field.
- Cognome autore:** A text input field with the placeholder text "Obbligatorio".
- ISBN:** A text input field with the placeholder text "Obbligatorio".
- Genere:** A dropdown menu currently showing "NON SELEZIONATO".
- Stato lettura:** A dropdown menu currently showing "NON SELEZIONATO".
- Valutazione:** A dropdown menu currently showing "NON SELEZIONATO".

At the bottom center of the form is a button labeled "Salva".

*Figura 2: Panel per inserimento libro*

Per i campi obbligatori, unitariamente al colore rosso, si è optato anche per l'utilizzo della dicitura 'Obbligatorio', che dinamicamente scompare quando l'utente inserisce i dettagli del campo specifico: si è voluto quindi impiegare un approccio visivamente chiaro ed efficace, evitando ambiguità su quali campi possano essere inseriti e quali no.



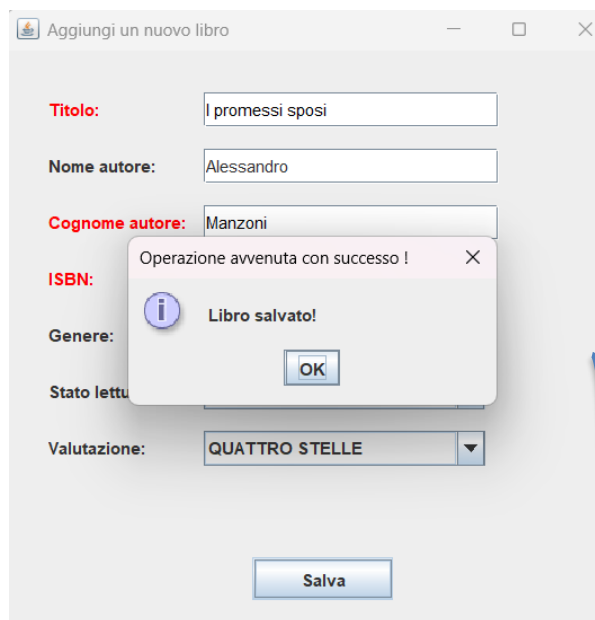
The screenshot shows a window titled "Aggiungi un nuovo libro" with the following fields and values:

- Titolo:** I promessi sposi
- Nome autore:** Alessandro
- Cognome autore:** Manzoni
- ISBN:** 9788817046671
- Genere:** ROMANZO
- Stato lettura:** LETTO
- Valutazione:** QUATTRO STELLE

A "Salva" button is located at the bottom right of the form.

Figura 3: Campi compilati

Premendo il bottone 'Salva', se i parametri obbligatori sono tutti stati esplicitati, avverrà l'inserimento del libro all'interno della libreria:



The screenshot shows the same "Aggiungi un nuovo libro" window as in Figure 3, but with a small dialog box overlaid in the center. The dialog box has a title bar "Operazione avvenuta con successo !" and contains the text "Libro salvato!" with an "OK" button. A blue arrow points from the "Salva" button in the background window to the dialog box.

Figura 4: Salvataggio avvenuto con successo



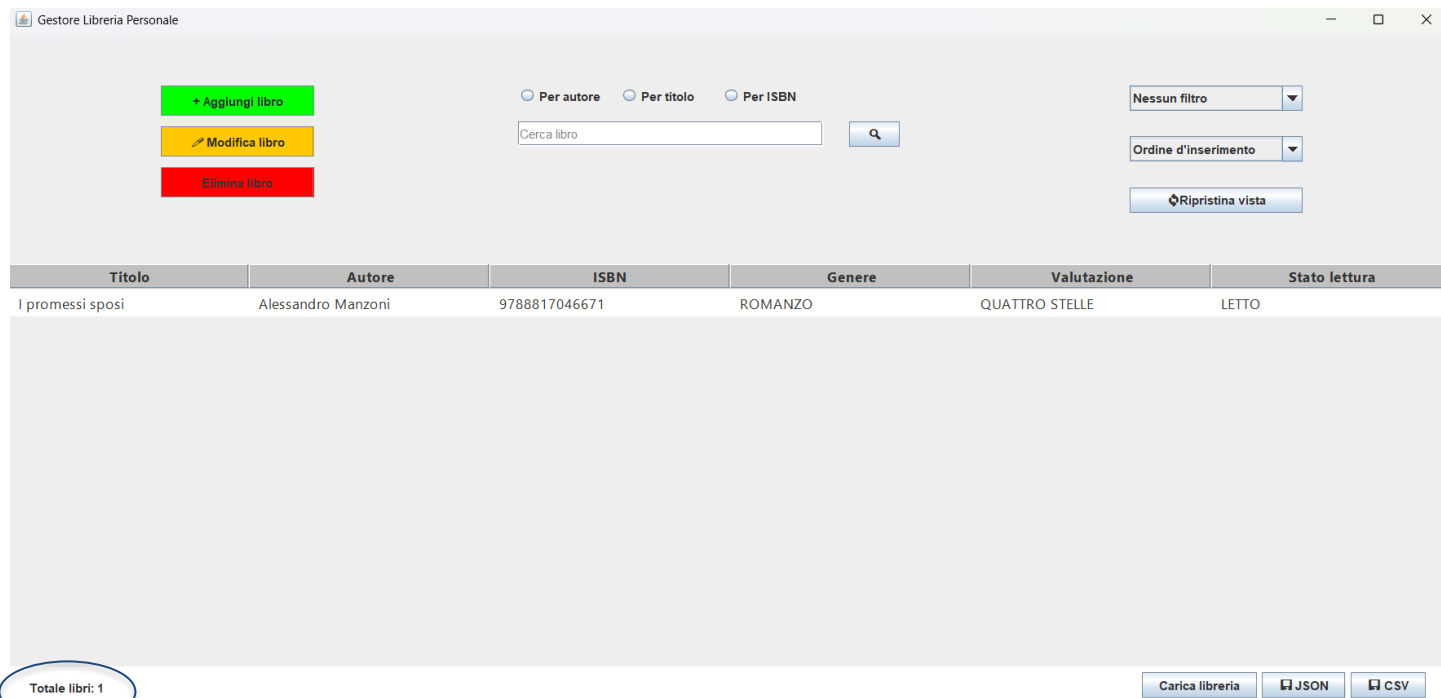


Figura 5: Visualizzazione del libro appena inserito

Dinamicamente si aggiorna anche il 'Totale libri' che consente di tener traccia del numero di volumi presenti all'interno della collezione, senza doverli contare manualmente.

### Modifica e rimozione di un libro

Per poter effettuare la modifica di un libro, è necessario selezionare il volume sulla tabella e poi premere il bottone ‘Modifica’:

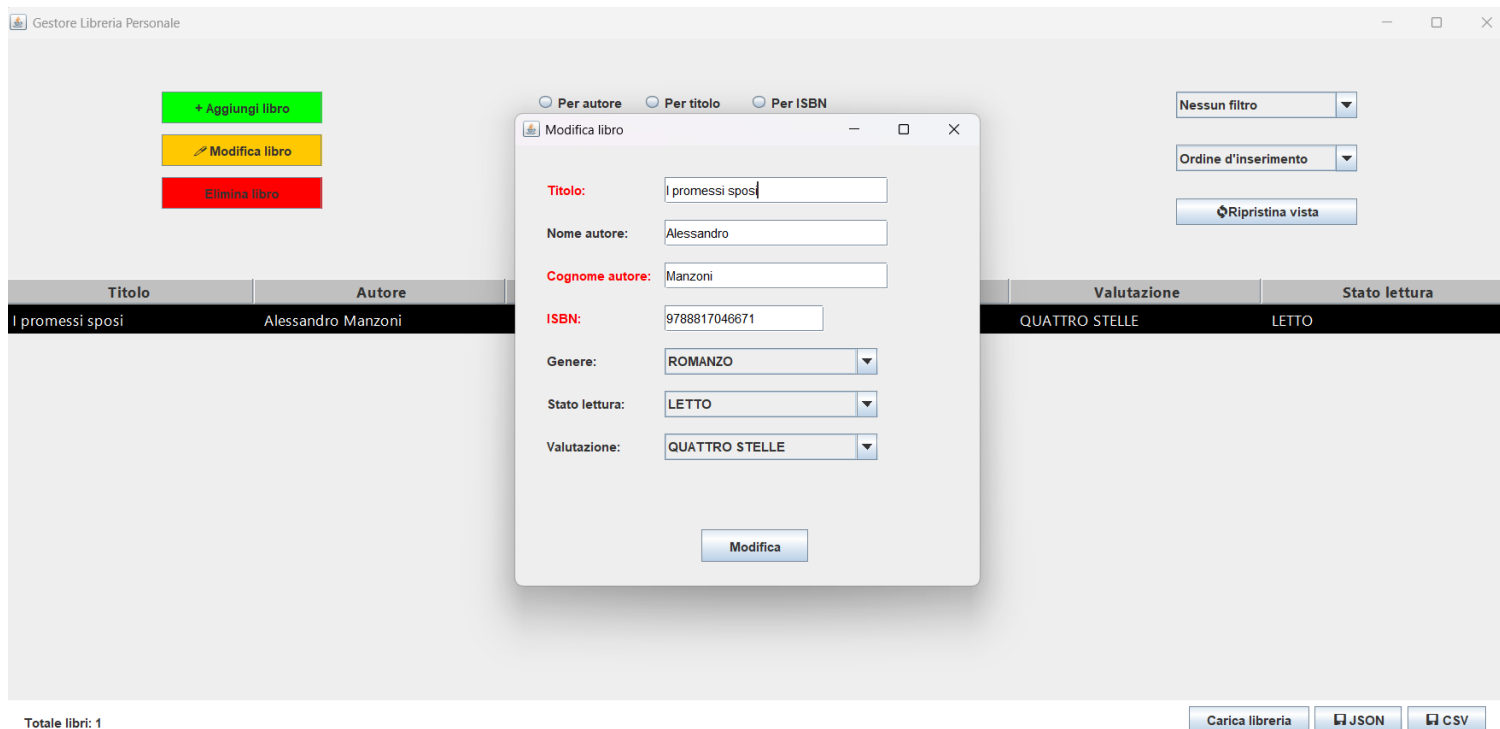


Figura 6: Modifica libro

Comparirà un pannello contenente le informazioni del libro che si vuole modificare fornite in fase di aggiunta.

Non appena terminata la fase di modifica, basterà premere il corrispondente tasto; apparirà nella libreria il libro modificato:

Titolo	Autore	ISBN	Genere	Valutazione	Stato lettura
I promessi sposi	Alessandro Manzoni	9788817046671	AVVENTURA	UNA STELLA	DA LEGGERE

Figura 7: Attributi modificati

Per la rimozione, similmente all’operazione di modifica, basterà selezionare il libro da eliminare e premere su ‘Elimina’. L’operazione comporta l’eliminazione del libro da quelli visualizzati e il decremento del contatore.

### *Salvataggio e ripristino*

Per la persistenza della libreria, effettuando il salvataggio con l'apposito bottone, otterremo nella cartella 'Downloads' rispettivamente un file 'salvataggio.csv' ed un file 'salvataggio.json', che si presentano così:

```
C: > Users > frff1 > Downloads > {} salvataggio.json > ...
1  [
2    {
3      "titolo": "I promessi sposi",
4      "autoreNome": "Alessandro",
5      "autoreCognome": "Manzoni",
6      "ISBN": "9788817046671",
7      "genLib": "AVVENTURA",
8      "valPers": "UNA_STELLA",
9      "statLett": "DA_LEGGERE"
10   }
11 ]
12
```

Figura 8: Salvataggio in JSON

	A	B	C	D	E	F	G	H	I	J	K
1	I promessi sposi,Alessandro,Manzoni,9788817046671,AVVENTURA,UNA_STELLA,DA_LEGGERE										
2											
3											
4											
5											

Figura 9: salvataggio in csv

Premendo su 'Carica libreria', verrà aperta di default la stessa cartella sulla quale vengono salvati i file:

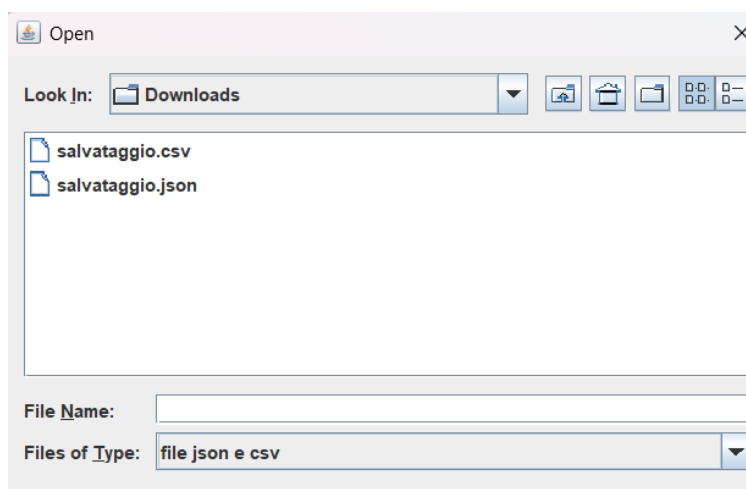


Figura 10: Ripristino libreria da file

### Ricerca per autore, genere e ISBN

Gestore Libreria Personale

+ Aggiungi libro  
✎ Modifica libro  
Elimina libro

☒ Per autore ☐ Per titolo ☐ Per ISBN

Cay

Nessun filtro  
Ordine d'inserimento  
Ripristina vista

Titolo	Autore	ISBN	Genere	Valutazione	Stato lettura
Fondamenti di Python	Cay Horstmann	9788891635433	SCIENTIFICO	QUATTRO STELLE	LETTO
Fondamenti di Java	Cay Horstmann	9788891639431	SCIENTIFICO	CINQUE STELLE	LETTO

Totale libri: 10

Carica libreria JSON CSV

Figura 12: ricerca per autore

Gestore Libreria Personale

+ Aggiungi libro  
✎ Modifica libro  
Elimina libro

☐ Per autore ☒ Per titolo ☐ Per ISBN

dina

Nessun filtro  
Ordine d'inserimento  
Ripristina vista

Titolo	Autore	ISBN	Genere	Valutazione	Stato lettura
Sistemi dinamici	Luca Benvenuti	9788838665387	SCIENTIFICO	CINQUE STELLE	NON SELEZIONATO

Totale libri: 10

Carica libreria JSON CSV

Figura 11: ricerca per titolo

La tipologia di ricerca da effettuare, trovandosi immediatamente sopra la barra di ricerca è facilmente selezionabile ed intuitiva.

La ricerca gestisce sia i casi in cui l'utente non immette come parametro in input l'autore/il titolo/ISBN per intero, che i casi in cui vengono mancati caratteri speciali presenti nel volume memorizzato all'interno della libreria.

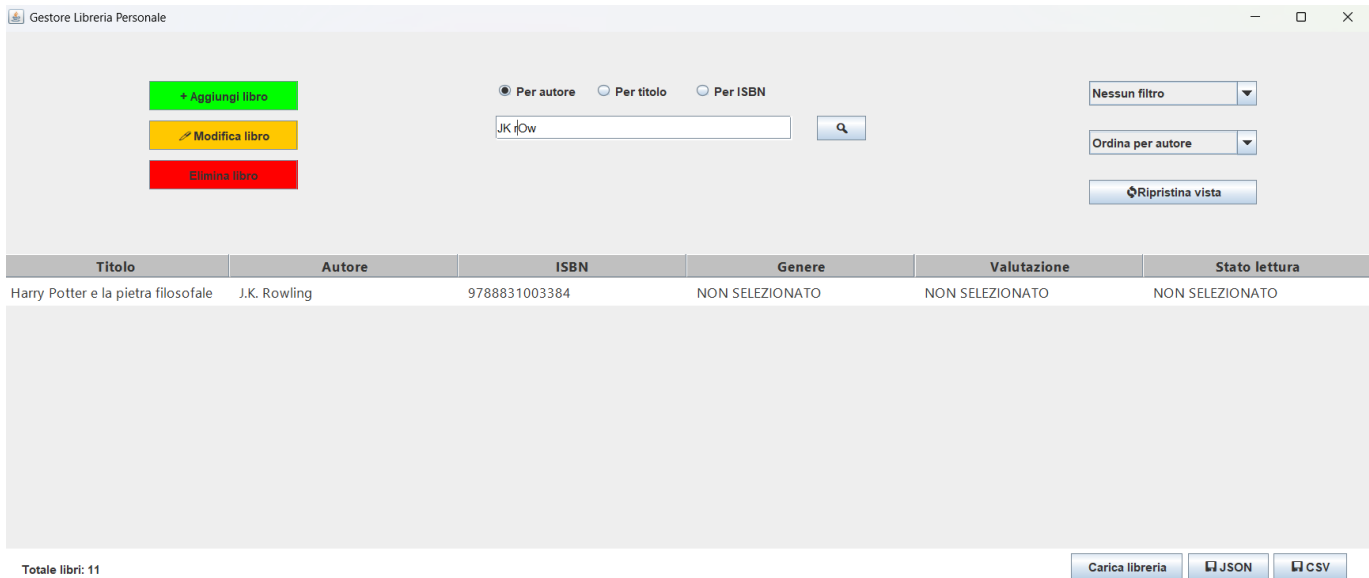


Figura 13: ricerca

### Meccanismi di filtro e di ordinamento

Possono essere impiegati scegliendo la tipologia che si vuole applicare dall'apposito menù a tendina:

