

# Applicazione per la gestione di una Biblioteca Universitaria

Progetto finale Ingegneria del Software a.a. 2025/2026

Alessandro Altieri  
Alessia Casaburi  
Annamaria Conte  
Antonietta Franzese

# Contents

<b>Descrizione introduttiva</b>	<b>3</b>
<b>Mockup</b>	<b>3</b>
<b>1 Ingegneria dei Requisiti - SRS</b>	<b>4</b>
1.1 Requisiti Funzionali .....	4
1.2 Tabella di categorizzazione dei requisiti.....	8
1.3 Diagramma dei Casi d'Uso .....	11
1.4 Casi d'Uso formato testuale.....	12
<b>2 Design</b>	<b>18</b>
2.1 Diagrammi di classi.....	18
2.1.1 Livello di dettaglio alto.....	19
2.2 Diagrammi dei package.....	20
2.3 Diagrammi di sequenza.....	22
2.3.1 C1 – Inserire libro.....	23
2.3.2 C4 – Cercare libro .....	24
2.3.3 C5 – Visualizzare elenco libri.....	25
2.3.4 C6 – Inserire utente .....	26
2.3.5 C9 – Cercare utente .....	27
2.3.6 C11 – Registrare prestito .....	28
2.3.7 C15 – Visualizzare prestiti attivi .....	29
2.4 Principi di buona progettazione .....	30
2.4.1 Livelli di coesione.....	30
2.4.2 Livelli di accoppiamento .....	21
2.4.3 Relazioni tra le Classi .....	32
2.4.4 Riduzione dell'Accoppiamento .....	32
2.4.5	

## Descrizione introduttiva

Il software gestisce una biblioteca universitaria, permettendo di archiviare informazioni sui **libri** (titolo, autori, anno, codice univoco e copie disponibili) e sugli **utenti** (nome, cognome, matricola, e-mail e prestiti attivi). L'applicazione, dotata di **interfaccia grafica**, consente di inserire, modificare, cancellare e cercare libri e utenti, registrare prestiti e restituzioni, e visualizzare liste ordinate dei libri, degli utenti e dei prestiti. Tutti i dati sono salvati su file per garantire la persistenza delle informazioni.

Si rende disponibile un'interfaccia grafica che permette all'utente di usufruire delle funzionalità offerte dal sistema.

## Mockup

Si inseriscono alcuni mockup dell'interfaccia grafica del software per scopo illustrativo, finalizzati ad una stesura dei requisiti più dettagliata e ad un supporto alla comprensione dei requisiti.

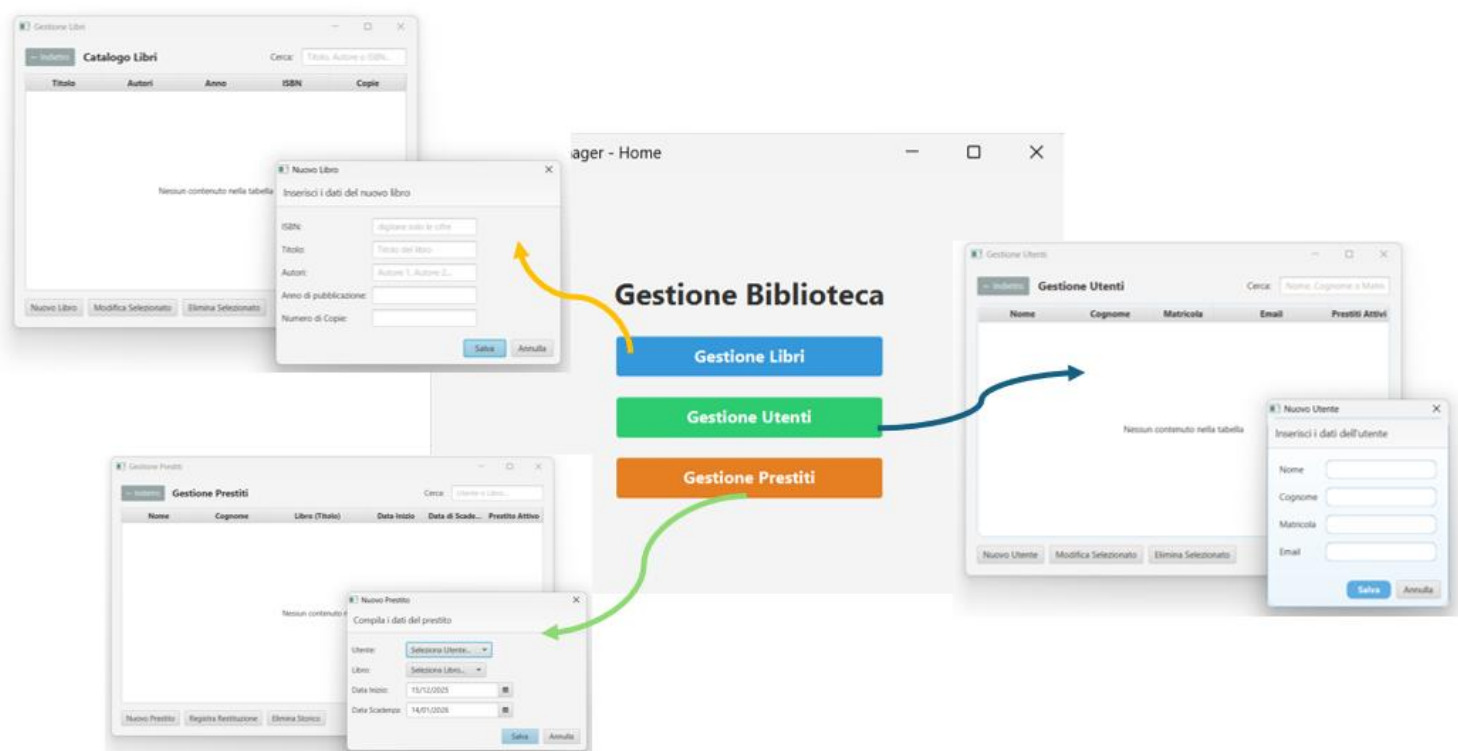


Figure 1: Mockup

# 1 Ingegneria dei Requisiti - SRS

## 1.1 Requisiti Funzionali

### Funzionalità Individuali

#### IF-1.1 Inserimento libro

L'utente ha la possibilità di inserire un nuovo libro all'interno dell'archivio della biblioteca. Attraverso l'apposita sezione di gestione libri, viene mostrato un form in cui è possibile compilare tutti i campi definiti in DF-1.1: titolo, autori, anno di pubblicazione, codice identificativo univoco e numero di copie. Al momento del salvataggio il sistema verifica che il codice identificativo (es. ISBN) non sia già presente in archivio (vedi FC-1.2). E' possibile annullare l'operazione tramite l'apposito pulsante.

#### IF-1.2 Modifica libro

L'utente può modificare i dati di un libro già presente in archivio. Selezionando un libro dall'elenco e cliccando su "Modifica" viene mostrata una schermata dove tutti i campi possono essere aggiornati. Dopo la modifica è possibile confermare l'operazione o annullarla tramite i pulsanti "Salva" e "Annulla".

#### IF-1.3 Eliminazione libro

L'utente può eliminare un libro dall'archivio, selezionando e premendo "Elimina". L'operazione richiede una conferma tramite conferma di dialogo. L'eliminazione è consentita solo se nessuna copia del libro risulta attualmente in prestito (vedi FC-1.3).

#### IF-1.4 Ricerca libro

L'utente può ricercare un libro inserendo nella barra di ricerca una sottostringa del titolo, il nome di un autore o il codice identificativo del libro. Il sistema restituisce tutti i libri che soddisfano il criterio di ricerca, ordinati alfabeticamente secondo quanto definito in IF-1.5.

#### IF-1.5 Visualizzazione elenco libri

L'elenco dei libri è mostrato in forma tabellare e ordinato alfabeticamente per titolo. L'utente può scorrere, filtrare e selezionare i libri. Il sistema garantisce che l'ordinamento rimanga coerente in tutte le viste.

#### IF-1.6 Inserimento utente

L'utente amministratore può aggiungere un nuovo utente della biblioteca. Durante l'inserimento vengono richiesti i dati definiti in DF-1.2: nome, cognome, matricola ed e-mail istituzionale. La matricola deve essere unica all'interno del sistema e deve contenere 10 cifre rispettando i criteri d'ateneo (vedi FC-1.1). L'operazione può essere confermata o annullata tramite appositi pulsanti.

#### IF-1.7 Modifica utente

L'amministratore può modificare i dati di un utente già registrato. Attraverso la schermata dedicata è possibile modificare ciascun campo (vedi DF-1.2). Le modifiche possono essere modificate o annullate.

#### IF-1.8 Eliminazione utente

L'amministratore può eliminare un utente dall'archivio. L'operazione richiede una conferma esplicita. Non è possibile eliminare un utente che risulta avere prestiti attivi (vedi FC-1.3)

**IF-1.9 Ricerca utente**

E' possibile ricercare un utente digitando una sottostringa del cognome o la matricola.  
Il sistema mostra i risultati ordinati per cognome e nome, secondo quanto definito in IF-2.1.

**IF-2.1 Visualizzazione elenco utenti**

Il sistema mostra l'elenco completo degli utenti, ordinato alfabeticamente per cognome e nome. La vista è coerente e costante in tutte le sezioni dell'applicazione

**IF-2.2 Restituzione libro**

L'amministratore può registrare la restituzione di un libro. Selezionando il prestito attivo, il sistema aggiorna le copie disponibili del libro e rimuove l'associazione tra utente e prestito. L'operazione che riflette sia sull'archivio dei libri che su quello degli utenti.

**Dati e formato dei dati****DF-1 Struttura libro**

Per ogni libro sono memorizzati i seguenti dati obbligatori:

- titolo;
- lista degli autori (nome e cognome);
- anno di pubblicazione;
- codice identificativo univoco (ISBN);
- numero di copie disponibili;

Tutti questi dati sono utilizzati dalle funzionalità IF 1.1, IF 1.2 e dal flusso BF 1.X.

**DF-2 Struttura utente**

Per ogni utente il sistema conserva:

- nome;
- cognome;
- matricola univoca;
- e-mail istituzionale;
- lista dei prestiti attivi e data prevista di restituzione;

Questi campi sono utilizzati nelle funzionalità IF-1.6, IF-1.7, IF-1.9, e BF-1.X.

**DF-3 Struttura prestito**

Un prestito contiene:

- utente;
- libro;
- data del prestito;
- data prevista di restituzione.

Questi dati sono usati nei requisiti BF-1.1, BF-1.5 e IF-2.2.

**DF-4 Persistenza dei dati**

L'intero archivio della biblioteca deve essere salvato su file, comprendendo libri, utenti e prestiti. Alla chiusura dell'applicazione tutti i dati vengono memorizzati, e vengono caricati automaticamente all'avvio (vedi IS-1.2).

**Business flow****BF 1.1- Registrazione prestito**

Per registrare un prestito, l'amministratore seleziona un utente e un libro disponibile. Viene quindi indicata la data prevista di restituzione. Il prestito viene salvato secondo la struttura definita in DF-1.3.

**BF 1.2- Controllo copie disponibili**

Durante l'operazione di prestito il sistema verifica che il numero di copie disponibili del libro sia maggiore di zero; in caso contrario l'operazione viene bloccata e viene mostrato un messaggio di errore (vedi UI-1.7).

**BF 1.3- Limiti di prestiti per utente**

Il sistema verifica che l'utente non abbia già tre prestiti attivi. Se il limite è stato raggiunto, il prestito non viene registrato e l'utente riceve una notifica (vedi UI-1.7).

**BF 1.4- Lista prestiti attivi**

Il sistema mostra tutti i prestiti attivi ordinati per data prevista di restituzione, dalla più prossima alla più lontana. Questa vista permette un controllo immediato della situazione dei prestiti.

**BF 1.5- Evidenziazione ritardi**

I prestiti la cui data prevista di restituzione è superata vengono evidenziati graficamente. L'utente può quindi riconoscere immediatamente le situazioni di ritardo.

**Interfacce con Sistemi esterni****IS 1.1- Salvataggio archivio**

Il sistema consente di salvare l'intero archivio su file. Il salvataggio viene attivato automaticamente e può essere eseguito anche manualmente dall'utente.

**IS 1.2- Caricamento archivio**

All'avvio dell'applicazione, il sistema carica l'archivio precedentemente salvato; se il file non è presente o risulta corrotto, il sistema mostra un messaggio di errore (vedi IS-1.4).

**IS 1.3- Formato dei dati**

Il sistema può importare o esportare archivi in formati comuni come JSON, CSV o file di testo.

**IS 1.4- Gestione errori I/O**

Eventuali errori in fase di lettura o scrittura del file (mancanza permessi, file danneggiato, percorso non valido) vengono notificati tramite l'interfaccia utente.

## Interfaccia utente

### UI 1.1- GUI generale

L'applicazione deve fornire un'interfaccia grafica che permetta una navigazione intuitiva tra le varie sezioni (libri, utenti, prestiti). Tutte le funzionalità principali devono essere accessibili tramite pulsanti, menù o pannelli dedicati.

### UI 1.2- Form inserimento/modifica libri

Le operazioni di inserimento e modifica di un libro avvengono attraverso form dedicati. Tutti i campi previsti da DF-1.1 sono visibili e modificabili.

### UI 1.3- Form inserimento/modifica utenti

Analogamente a UI-1.2, anche gli utenti vengono gestiti tramite form che mostrano i campi previsti da DF-1.2.

### UI 1.4- Vista elenco utenti

I libri sono visualizzati in una tabella ordinata per titolo. L'utente può selezionare un libro per iniziare un prestito.

### UI 1.5- Form inserimento/modifica utenti

Gli utenti sono visualizzati in una tabella ordinata per cognome e nome. L'interfaccia permette una rapida selezione per operazioni successive.

### UI 1.6- Form inserimento/modifica utenti

I prestiti attivi sono mostrati in ordine di data prevista di restituzione. I prestiti in ritardo sono evidenziati graficamente.

## Ulteriori vincoli

### FC 1.1- Unicità matricola

Ogni utente deve avere una matricola univoca. Il sistema effettua un controllo prima del salvataggio dei dati.

### FC 1.2- Unicità codice libro

Ogni libro deve possedere un codice univoco. Il sistema impedisce l'inserimento di duplicati.

### FC 1.3- Integrità dei dati

Non è possibile eliminare utenti o libri che risultano coinvolti in prestiti attivi. Questa regola preserva la coerenza dell'archivio.

## 1.2 Tabella di categorizzazione dei Requisiti

Prefisso	Requisito	Priorità	Categoria	Informazioni	Note
<b>IF-1.1</b>	Inserimento libro	Alta	Funzionalità individuale	Inserire un nuovo libro con titolo, autori, anno, codice univoco e copie disponibili.	Verifica unicità codice (es. ISBN).
<b>IF-1.2</b>	Modifica libro	Alta	Funzionalità individuale	Modificare i dati di un libro esistente.	Tutti i campi editabili.
<b>IF-1.3</b>	Eliminazione libro	Alta	Funzionalità individuale	Eliminare un libro dal catalogo.	Possibile solo se nessuna copia è in prestito.
<b>IF-1.4</b>	Ricerca libro	Alta	Funzionalità individuale	Cercare un libro per titolo, autore o codice identificativo.	Ricerca con testo parziale.
<b>IF-1.5</b>	Visualizzazione elenco libri	Media	Funzionalità individuale	Visualizzare la lista dei libri ordinata per titolo.	Ordinamento alfabetico.
<b>IF-1.6</b>	Inserimento utente	Alta	Funzionalità individuale	Inserire un nuovo utente con nome, cognome, matricola, e-mail istituzionale.	Matricola univoca.
<b>IF-1.7</b>	Modifica utente	Alta	Funzionalità individuale	Modificare i dati di un utente esistente.	-
<b>IF-1.8</b>	Eliminazione utente	Alta	Funzionalità individuale	Eliminare un utente dal sistema.	Ammesso solo senza prestiti attivi
<b>IF-1.9</b>	Ricerca utente	Alta	Funzionalità individuale	Cercare un utente per cognome o matricola.	-
<b>IF-2.1</b>	Visualizzazione elenco utenti	Media	Funzionalità individuale	Visualizzare la lista utenti ordinata per cognome e nome.	-
<b>IF-2.2</b>	Restituzione libro	Alta	Funzionalità individuale	Registrare la restituzione di un libro e aggiornare le copie disponibili.	-



<b>BF-1.1</b>	Registrazione prestito	Alta	Business flow	Selezionare utente e libro e impostare la data prevista di restituzione	-
<b>BF-1.2</b>	Controllo copie disponibili	Alta	Business flow	Il prestito è consentito solo se esistono copie disponibili	-
<b>BF-1.3</b>	Limite prestiti utenti	Alta	Business flow	Un utente non può avere più di tre libri in prestito contemporaneamente	-
<b>BF-1.4</b>	Lista prestiti attivi	Alta	Business flow	Visualizzare l'elenco dei prestiti attivi ordinati per data prevista di restituzione	-
<b>BF-1.5</b>	Sollecito restituzione	Media	Business flow	I prestiti oltre le data prevista di restituzione devono essere sollecitati	-
<b>DF-1.1</b>	Struttura libro	Alta	Dati e formato	Definizione dei campi obbligatori per entità libro	Titolo, autori, anno, codice, copie
<b>DF-1.2</b>	Struttura utente	Alta	Dati e formato	Definizione dei campi obbligatori per entità utente	Nome, cognome, matricola, email.
<b>DF-1.3</b>	Struttura prestito	Alta	Dati e formato	Definizione dei dati relativi ai prestiti	Utente, libro, data prestito, data restituzione prevista
<b>DF-1.4</b>	Persistenza dati	Alta	Dati e formato	L'intero archivio deve essere salvato su file	Libri, utenti, prestiti
<b>UI-1.1</b>	GUI generale	Alta	Interfaccia Utente	L'applicazione deve avere un'interfaccia grafica	-
<b>UI-1.2</b>	Form inserimento/modifica libri	Alta	Interfaccia Utente	Le operazioni sui libri devono disporre di appositi form	-
<b>UI-1.3</b>	Form inserimento/modifica utenti	Alta	Interfaccia Utente	Le operazioni sugli utenti devono disporre di appositi form	-
<b>UI-1.4</b>	Vista elenco libri	Media	Interfaccia Utente	Mostrare in tabella i libri ordinati per titolo	-

<b>UI-1.5</b>	Vista elenco utenti	Media	Interfaccia Utente	Mostrare utenti ordinati per nome e cognome	-
<b>UI-1.6</b>	Vista prestiti attivi	Alta	Interfaccia Utente	Mostrare prestiti ordinati per data prevista di restituzione	Evidenziare ritardi di restituzione
<b>UI-1.7</b>	Messaggi di errore	Alta	Interfaccia Utente	Notificare errori	Copie insufficienti, limite prestiti superato.
<b>IS-1.1</b>	Salvataggio archivio	Alta	Interfaccia con sistemi esterni	Salvare l'intero archivio su file.	-
<b>IS-1.2</b>	Caricamento archivio	Alta	Interfaccia con sistemi esterni	Caricare l'archivio all'avvio dell'applicazione.	-
<b>IS-1.3</b>	Formato dati	Media	Interfaccia con sistemi esterni	Supportare formati come JSON / CSV / testo.	-
<b>IS-1.4</b>	Gestione errori I/O	Alta	Interfaccia con sistemi esterni	Segnalare errori di lettura/scrittura file.	-
<b>FC-1.1</b>	Unicità matricola	Alta	Ulteriori vincoli	La matricola utente deve essere unica.	-
<b>FC-1.2</b>	Unicità codice libro	Alta	Ulteriori vincoli	Ogni libro deve avere un codice univoco.	-
<b>FC-1.3</b>	Integrità dati	Alta	Ulteriori vincoli	Non è possibile eliminare entità collegate a prestiti attivi.	-

Figura 1: Tabella categorizzazione requisiti

### 1.3 Diagramma dei Casi d'Uso

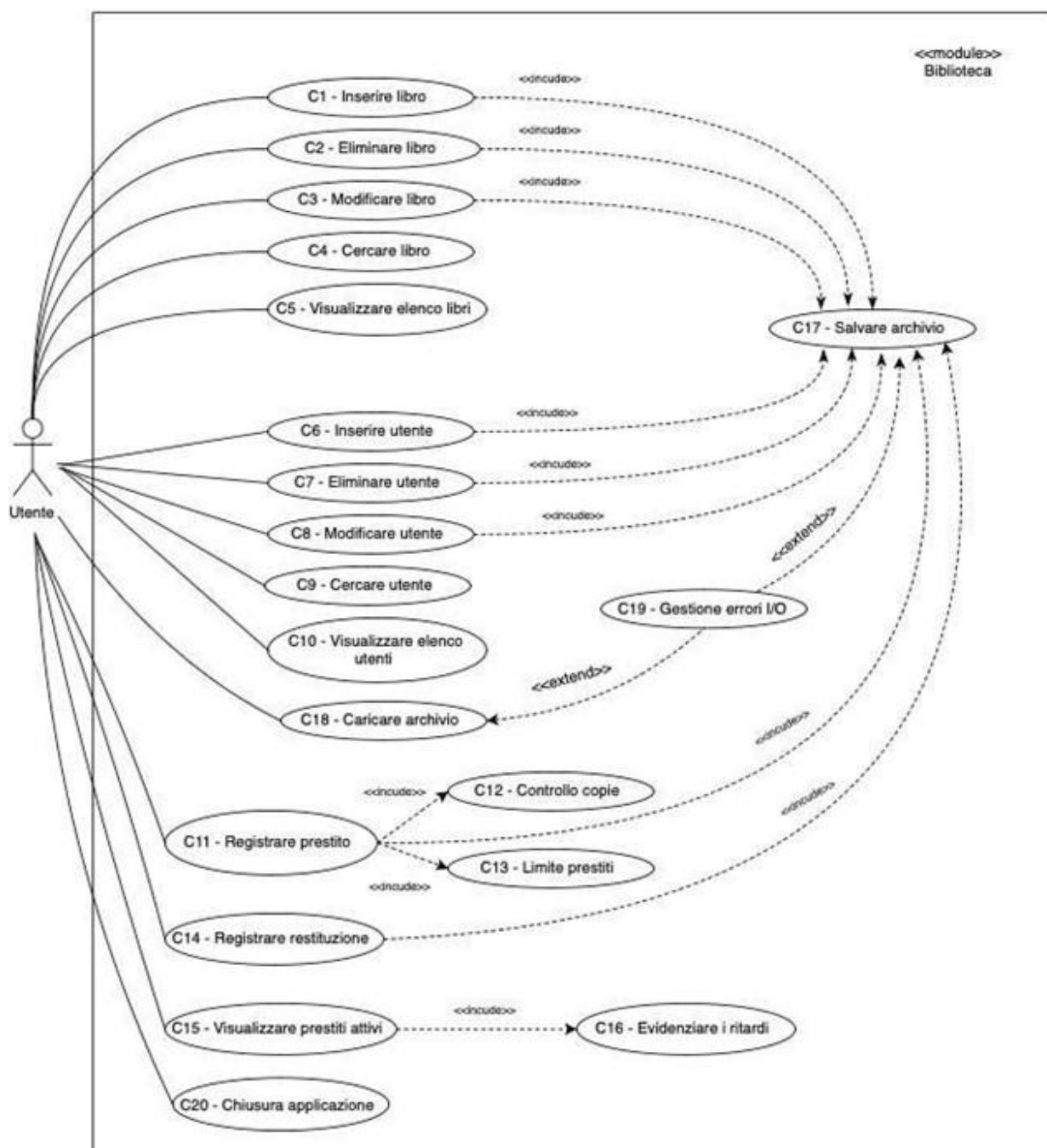


Figura 3: Diagramma dei casi d'uso

## 1.4 Casi d'Uso formato testuale

### C1 – Inserire libro

**Attore partecipante:** Utente

**Precondizioni:** Utente ha aperto la schermata di gestione libri.

**Postcondizioni:** Un nuovo libro viene aggiunto all'archivio.

- **Flusso di eventi normale:**
  1. Utente clicca il pulsante "Aggiungi libro";
  2. Utente inserisce titolo, autore, anno di pubblicazione, ISBN;
  3. Utente salva l'operazione.
- **Flusso di eventi alternativo:**
  - 4a. Utente annulla l'operazione;
    - 4a.1 L'esecuzione riprende dal passo 1

### C2 – Eliminare libro

**Attore partecipante:** Utente

**Precondizioni:** Esiste almeno un libro nell'archivio.

**Postcondizioni:** Il libro selezionato viene rimosso dall'archivio.

- **Flusso di eventi normale:**
  1. Utente seleziona il libro da eliminare;
  2. Utente clicca il pulsante "Elimina";
  3. Utente conferma l'operazione;
- **Flusso di eventi alternativo:**
  - 3a. Utente annulla l'operazione;
    - 3a.1 L'esecuzione riprende dal passo 1;

### C3 - Modificare libro

**Attore partecipante:** Utente

**Precondizioni:** Esiste almeno libro nell'archivio.

**Postcondizioni:** Le informazioni del libro selezionato vengono aggiornate.

- **Flusso di eventi normale:**
  1. Utente seleziona il libro da modificare;
  2. Utente clicca il pulsante "Modifica";
  3. Utente modifica i campi desiderati;
  4. Utente salva l'operazione;
- **Flusso di eventi alternativo:**
  - 4a. Utente annulla l'operazione;
    - 4a.1 L'esecuzione riprende dal passo 1;

**C4 - Cercare libro**

**Attore partecipante:** Utente

**Precondizioni:** Utente ha la schermata di gestione libri aperta.

**Postcondizioni:** I libri che rispettano il criterio di ricerca vengono visualizzati.

- **Flusso di eventi normale:**
  1. Utente inserisce titolo, autore o genere nella casella di ricerca;
- **Flusso di eventi alternativo:**
  - 1a. Utente applica filtri avanzati per anno o disponibilità.

**C5 – Inserire utente**

**Attore partecipante:** Utente

**Precondizioni:** Esiste almeno un libro nell'archivio.

**Postcondizioni:** L'utente visualizza tutti i libri registrati.

- **Flusso di eventi normale:**
  1. Utente apre la schermata "Elenco libri";
  2. Il sistema mostra tutti i libri disponibili.
- **Flusso di eventi alternativo:**
  - 2a. Utente applica filtri o ordinamenti (alfabetico, per autore, per anno).

**C6 – Inserire utente**

**Attore partecipante:** Utente

**Precondizioni:** Utente ha aperto la schermata di gestione utenti.

**Postcondizioni:** Un nuovo utente viene aggiunto all'archivio.

- **Flusso di eventi normale:**
  1. Utente clicca il pulsante "Aggiungi utente";
  2. Utente inserisce nome, cognome, codice utente, contatti;
  3. Utente salva l'operazione;
- **Flusso di eventi alternativo:**
  - 3a. Utente annulla l'operazione;
    - 3a.1 L'esecuzione riprende dal passo 1;

**C7 – Eliminare utente**

**Attore partecipante:** Utente

**Precondizioni:** Esiste almeno un utente nell'archivio.

**Postcondizioni:** L'utente selezionato viene rimosso dall'archivio.

- **Flusso di eventi normale:**
  1. Utente seleziona l'utente da eliminare;
  2. Utente clicca il pulsante "Elimina";
  3. Utente conferma l'operazione.
- **Flusso di eventi alternativo:**
  - 3a. Utente annulla l'operazione;
  - 3a.1 L'esecuzione riprende dal passo 1.

**C8 – Modificare utente**

**Attore partecipante:** Utente

**Precondizioni:** Esiste almeno un utente nell'archivio.

**Postcondizioni:** Le informazioni dell'utente selezionato vengono aggiornate.

- **Flusso di eventi normale:**
  1. Utente seleziona l'utente da modificare;
  2. Utente clicca il pulsante "Modifica";
  3. Utente modifica i campi desiderati;
  4. Utente salva l'operazione.
- **Flusso di eventi alternativo:**
  - 4a. Utente annulla l'operazione;
  - 4a.1 L'esecuzione riprende dal passo 1.

**C9 – Cercare utente**

**Attore partecipante:** Utente

**Precondizioni:** Utente ha la schermata di gestione utenti aperta.

**Postcondizioni:** Gli utenti corrispondenti ai criteri di ricerca vengono visualizzati.

- **Flusso di eventi normale:**
  1. Utente inserisce nome, cognome o codice utente nella casella di ricerca;
- **Flusso di eventi alternativo:**
  - 1a Utente applica filtri aggiuntivi come stato iscrizione o numero prestiti.

**C10 – Visualizzare elenco utenti**

**Attore partecipante:** Utente

**Precondizioni:** Esiste almeno un utente nell'archivio.

**Postcondizioni:** L'utente visualizza tutti gli utenti registrati.

- **Flusso di eventi normale:**
  1. Utente apre la schermata "elenco utenti";
  2. Sistema mostra tutti gli utenti registrati.
- **Flusso di eventi alternativo:**
  - 2a. Utente applica filtri o ordinamenti (alfabetico, per numero prestiti).

**C11 – Registrare prestito**

**Attore partecipante:** Utente

**Precondizioni:** Esistono libri disponibili e utente registrato.

**Postcondizioni:** Il prestito viene registrato nel sistema.

- **Flusso di eventi normale:**
  1. Utente seleziona il libro da prestare;
  2. Utente seleziona l'utente che riceve il prestito;
  3. Sistema verifica copie disponibili (C12);
  4. Sistema verifica limite prestiti utente (C13);
  5. Utente conferma il prestito;
  6. Prestito registrato con data di scadenza.
- **Flusso di eventi alternativo:**
  - 3a. Nessuna copia disponibile → operazione bloccata;
  - 4a. Utente ha raggiunto il limite prestiti → operazione bloccata;
  - 5a. Utente annulla il prestito.

**C12 – Controllare copie disponibili (include C11)**

**Attore partecipante:** Sistema

**Precondizioni:** Libro selezionato per il prestito.

**Postcondizioni:** Sistema restituisce numero di copie disponibili.

- **Flusso di eventi normale:**
  1. Sistema conta le copie disponibili;
  2. Sistema restituisce il risultato a C11.

**C13 – Limite prestiti utente (include C11)**

**Attore partecipante:** Sistema

**Precondizioni:** Utente selezionato per il prestito.

**Postcondizioni:** Sistema verifica se utente ha superato il limite prestiti.

- **Flusso di eventi normale:**
  1. Sistema conta i prestiti attivi dell'utente;
  3. Sistema restituisce il risultato a C11.

**C14 – Registrare restituzione**

**Attore partecipante:** Utente

**Precondizioni:** Prestito registrato e libro in possesso dell'utente.

**Postcondizioni:** Prestito chiuso e copia restituita.

- **Flusso di eventi normale:**
  1. Utente seleziona il prestito da chiudere;
  2. Utente conferma la restituzione;
  3. Sistema aggiorna disponibilità libro.
- **Flusso di eventi alternativo:**
  - 3a. Utente annulla operazione;
  - 3a.1 L'esecuzione riprende dal passo 1.

**C15– Visualizzare prestiti attivi**

**Attore partecipante:** Utente

**Precondizioni:** Esistono prestiti registrati.

**Postcondizioni:** Vengono visualizzati tutti i prestiti attivi.

- **Flusso di eventi normale:**
  1. Utente apre la schermata "Prestiti attivi";
  2. Sistema mostra prestiti con informazioni su libro, utente e scadenza.

**C16– Evidenziare ritardi (include C15)**

**Attore partecipante:** Utente

**Precondizioni:** Prestiti attivi presenti.

**Postcondizioni:** Prestiti scaduti evidenziati.

- **Flusso di eventi normale:**
  1. Sistema confronta data scadenza prestiti con data odierna;
  2. Prestiti scaduti evidenziati nella schermata C15.



**C17 – Salvare archivio**

**Attore partecipante:** Sistema

**Precondizioni:** Esistono modifiche da salvare.

**Postcondizioni:** Archivio salvato su file.

- **Flusso di eventi normale:**
  1. Sistema scrive dati di libri, utenti e prestiti su file locale.
- **Flusso di eventi alternativo:**
  - 1a. Utente sceglie percorso personalizzato;
  - 1b. Operazione non riuscita → C19.

**C18 – Caricare archivio**

**Attore partecipante:** Utente

**Precondizioni:** Esiste un file di archivio valido.

**Postcondizioni:** Archivio caricato nel sistema.

- **Flusso di eventi normale:**
  1. Utente selezione file di archivio;
  2. Sistema carica dati di libri, utenti e prestiti.
- **Flusso di eventi alternativo:**
  - 1a. File non valido → C19.

**C19– Segnalare errori di lettura/scrittura**

**Attore partecipante:** Sistema

**Precondizioni:** Operazione di salvataggio o caricamento fallita.

**Postcondizioni:** Utente viene avvisato dell'errore

- **Flusso di eventi normale:**
  1. Sistema visualizza messaggio di errore;
  2. Utente prende decisione su retry o annulla.

**C20 – Chiusura applicazione**

**Attore partecipante:** Utente

**Precondizioni:** L'applicazione è in esecuzione.

**Postcondizioni:** L'applicazione viene chiusa.

- **Flusso di eventi normale:**
  1. Utente clicca sul pulsante di chiusura dell'applicazione;
  2. L'applicazione si chiude.
- **Flusso di eventi alternativo:**
  - 1a. Sono presenti modifiche non salvate;
    - 1a.1 Il sistema mostra una finestra di conferma chiedendo se si vuole chiudere senza salvare;
    - 1a.2 Utente conferma → l'applicazione si chiude;
    - 1a.3 Utente annulla → l'esecuzione riprende dalla schermata corrente.

## 2 Design

### 2.1 Diagramma delle classi

I diagrammi delle classi rappresentano il sistema dell'applicazione per la gestione della biblioteca con le funzionalità di gestire gli utenti, gestire i libri e gestire i prestiti.

**Utente:** la classe Utente definisce l'oggetto utente con nome, cognome, matricola, e-mail e lista dei prestiti attivi; contiene i metodi `get()` e `set()` e i metodi per il controllo della validità della matricola e dell'e-mail.

**Libro:** la classe Libro definisce l'oggetto libro con titolo, lista degli autori, anno di pubblicazione, codice ISBN e numero di copie disponibili; contiene i metodi `get()` e `set()` e i metodi necessari per gestire il numero di copie disponibili;

**Prestito:** la classe Prestito contiene tutte le informazioni necessarie per identificare un prestito, libro e utente di riferimento, data di inizio prestito, data di scadenza del prestito, data di restituzione effettiva, i metodi `get()` e `set()` e possiede un attributo di tipo Boolean che identifica la condizione di ritardo del prestito;

**GestoreUtenti:** la classe GestoreUtenti gestisce gli oggetti di tipo Utente; funge da intermediario tra l'interfaccia utente e i dati, e si occupa delle operazioni di inserimento, modifica o rimozione degli Utenti della biblioteca;

**GestoreLibri:** la classe GestoreLibri gestisce gli oggetti di tipo Libro; si occupa delle operazioni di inserimento, modifica o rimozione di un Libro dal catalogo mantenendolo aggiornato;

**GestorePrestiti:** la classe GestorePrestiti si occupa di registrare, tracciare e storicizzare le associazioni tra Utenti e Libri sotto forma di oggetti di tipo Prestito;

**ManagerGenerale<T>:**

l'interfaccia ManagerGenerale<T> definisce il comportamento di tutte le classi di gestione del sistema; garantisce l'astrazione e l'uniformità delle operazioni utilizzando il tipo generico <T>.

Figure 5: Diagramma classi completo



## 2.2 Diagramma dei package

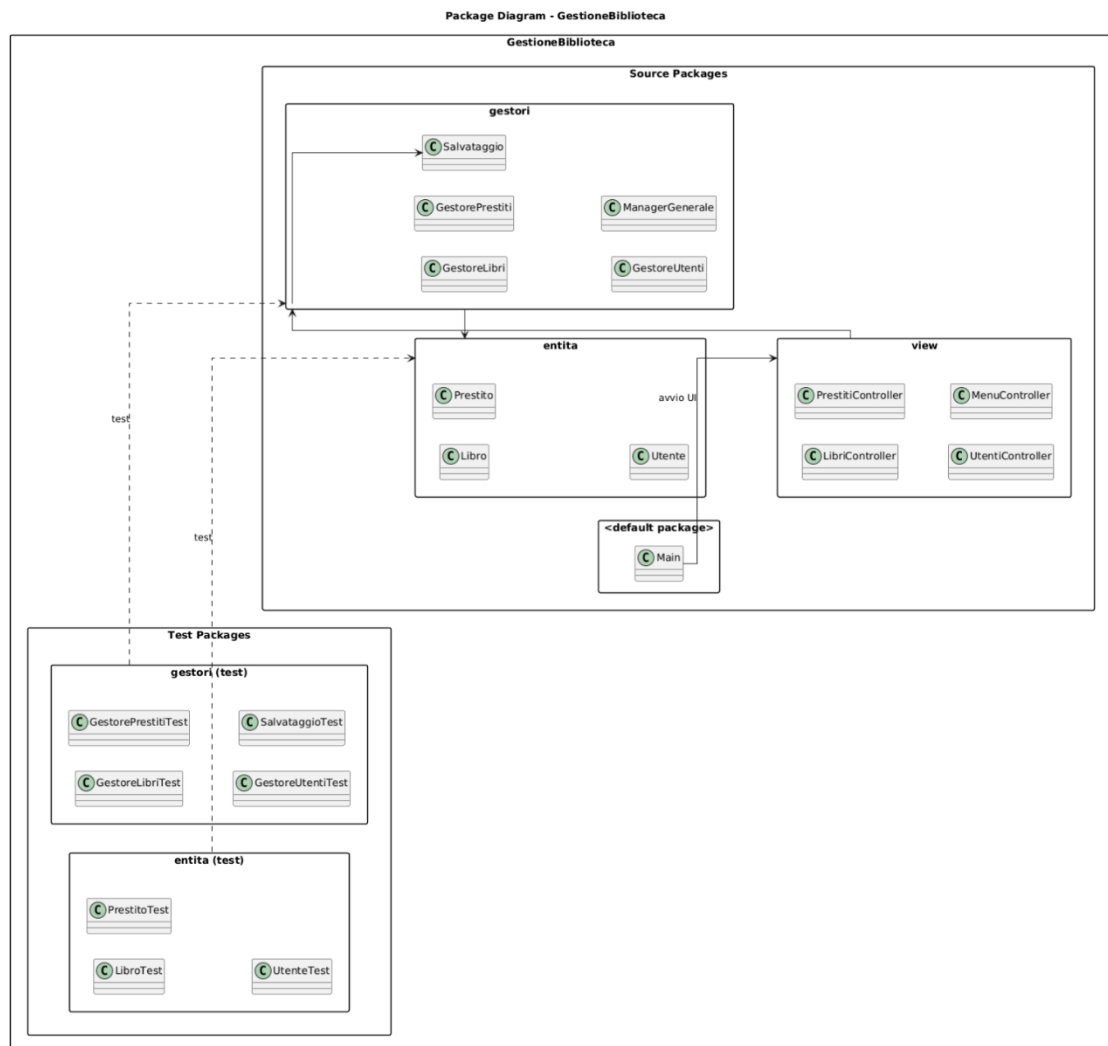


Figure 6: Diagramma package

Il diagramma dei package rappresenta l'organizzazione strutturale del sistema, mettendo in evidenza le principali funzionalità e il modo in cui esse sono suddivise secondo criteri di buona progettazione software.

L'architettura adottata segue il paradigma MVC (Model-View-Controller), consentendo una netta distinzione tra la gestione dell'interfaccia utente, la logica applicativa e la rappresentazione dei dati. Il principio di Single Responsibility è rispettato attribuendo a ciascun package un compito ben definito:

- Il package di *view* e i relativi controller si occupano della gestione dell'interazione con l'utente;
- Il package di *gestori* contiene la logica applicativa e le varie coordinano le operazioni principali del sistema;

- Il package delle *entità* modella il dominio applicativo, rappresentando gli oggetti fondamentali della biblioteca;
- Il package dedicato al *salvataggio* gestisce la persistenza dei dati.

Le componenti grafiche, definite tramite file FXML e fogli di stile CSS, permettono di separare la presentazione dalla logica sottostante.

In accordo con il principio di Separation of Concerns (SoC), le diverse responsabilità sono mantenute in package distinti, migliorando la leggibilità del sistema, la modularità e la facilità di manutenzione.

Le dipendenze tra i package seguono un flusso ordinato e unidirezionale: i controller comunicano con i gestori, i quali operano sulle entità e sui meccanismi di persistenza. Le view dipendono esclusivamente dai controller tramite i riferimenti dichiarati nei file FXML.

Questa organizzazione evita la presenza di dipendenze cicliche e limita le interazioni tra livelli adiacenti, contribuendo a ridurre la complessità del sistema e a favorire un basso accoppiamento e un'elevata coesione interna.

## 2.3 Diagrammi di sequenza

Vengono presentati i diagrammi di sequenza relativi ai principali casi d'uso del sistema. Ogni diagramma mostra come l'Utente interagisce con le schermate e con i Gestori che gestiscono i dati di libri, utenti e prestiti.

### 2.3.1 C1 – Inserire libro

Il diagramma descrive l'inserimento di un nuovo libro. L'Utente inserisce i dati richiesti nella **SchermataLibri**, che li invia al **GestoreLibri**. Il gestore crea il nuovo oggetto Libro e lo salva nel sistema. È prevista anche la possibilità che l'Utente annulli l'operazione.

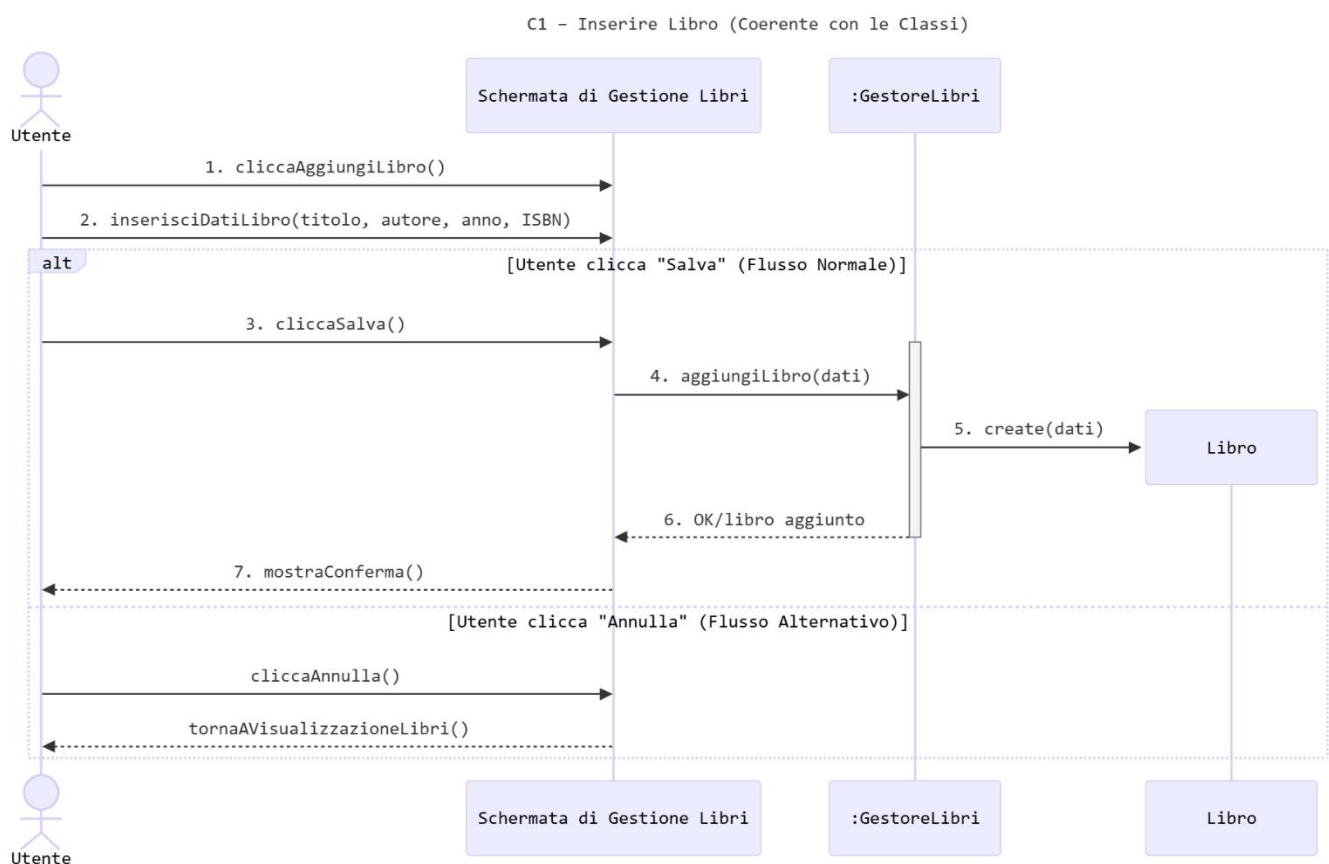


Figure 6: Diagramma sequenza C1 – Inserire libro

### 2.3.2 C4 – Cercare libro

Il seguente diagramma di sequenza mostra la ricerca dei libri. L'Utente può scegliere tra ricerca semplice o avanzata. La richiesta viene inviata al **GestoreLibri**, che esegue la ricerca tramite **cercaLibro(criteri)** e restituisce l'elenco dei risultati.

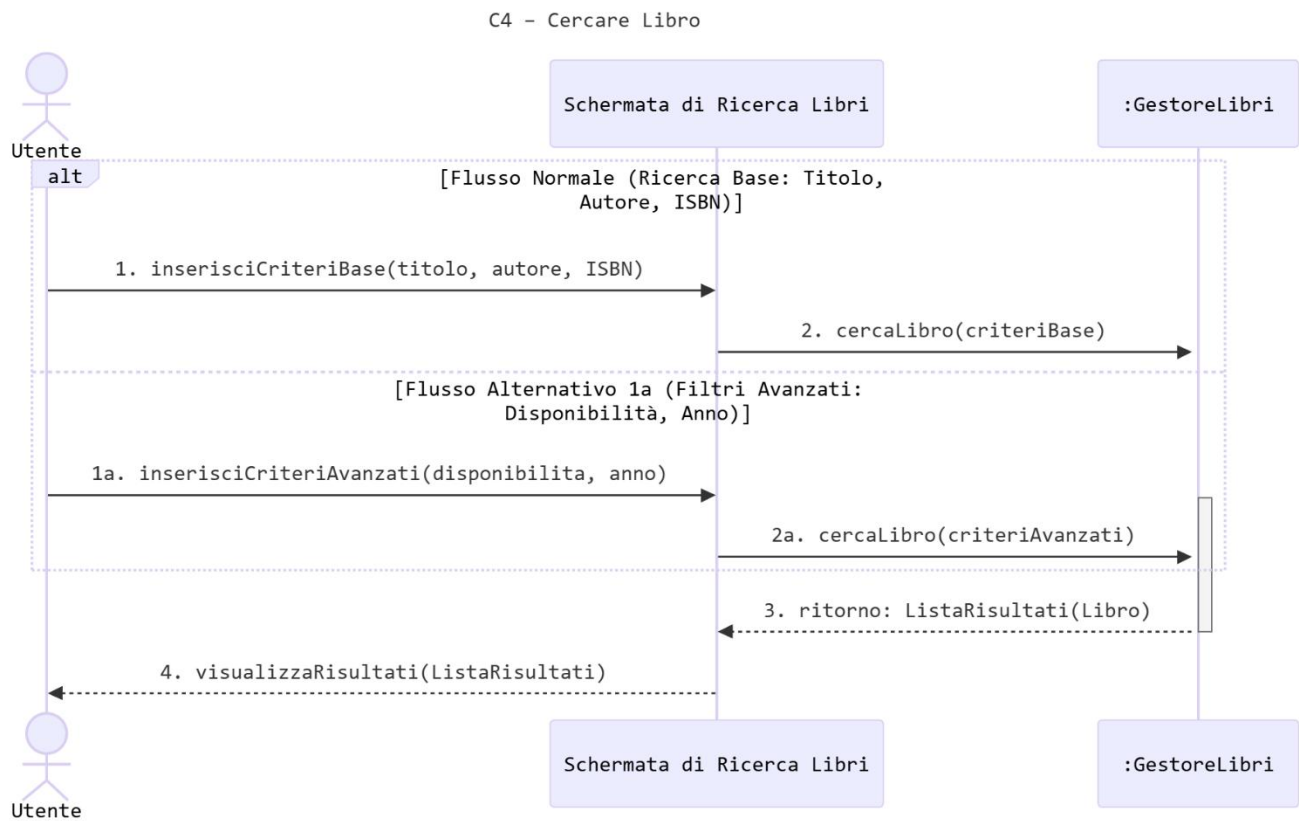


Figure 7: Diagramma sequenza C4 – Cercare libro

### 2.3.3 C5 – Visualizzare elenco libri

Il seguente diagramma di sequenza illustra la visualizzazione del catalogo libri. La **SchermataElenco** richiede al **GestoreLibri** l'intera lista (**getList()**). Se l'Utente applica filtri o ordinamenti, il gestore li applica prima di restituire i risultati.

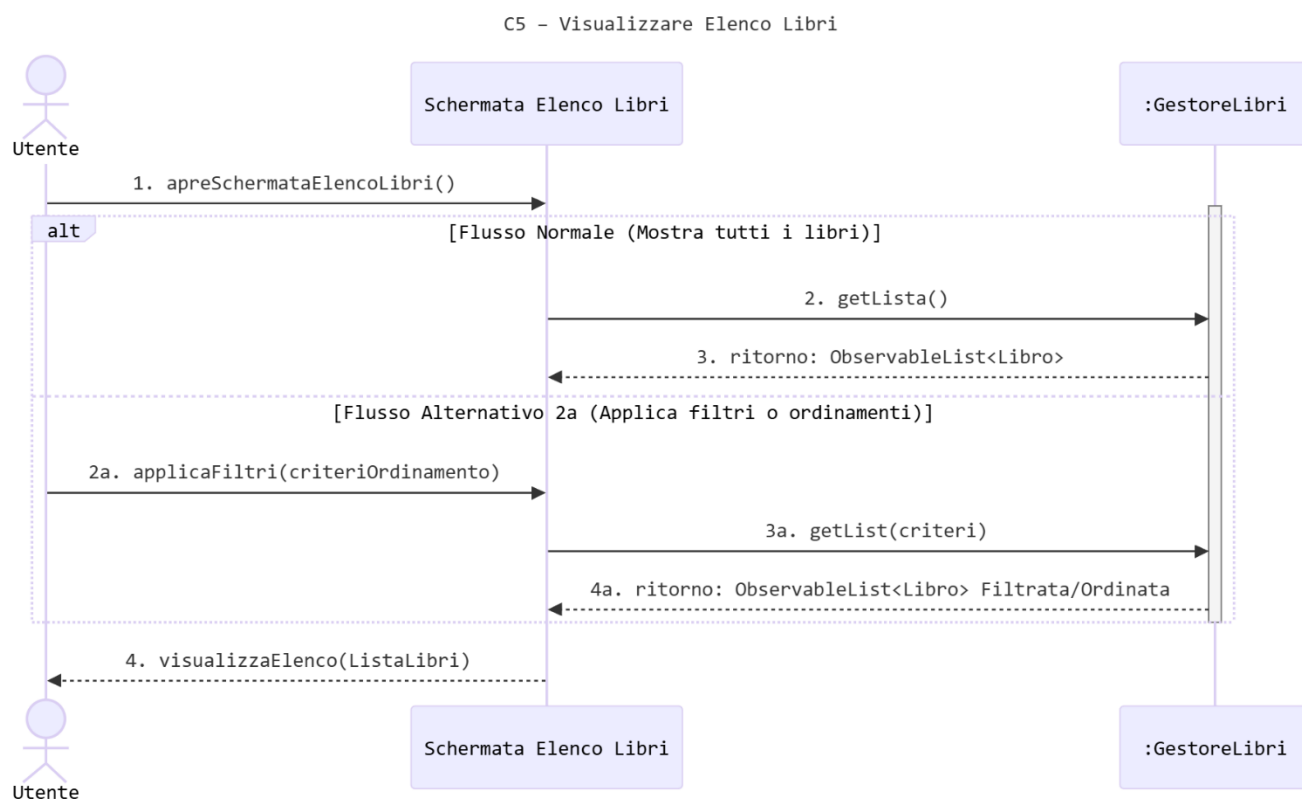


Figure 8: Diagramma sequenza C5 – Visualizzare elenco libri



### 2.3.4 C6 – Inserire Utente

Qui viene mostrata la registrazione di un nuovo utente. La **SchermataUtenti** raccoglie i dati anagrafici e li invia al **GestoreUtenti**, che crea la nuova entità **Utente** tramite **aggiungi(dati)**. Il diagramma prevede anche l'annullamento.

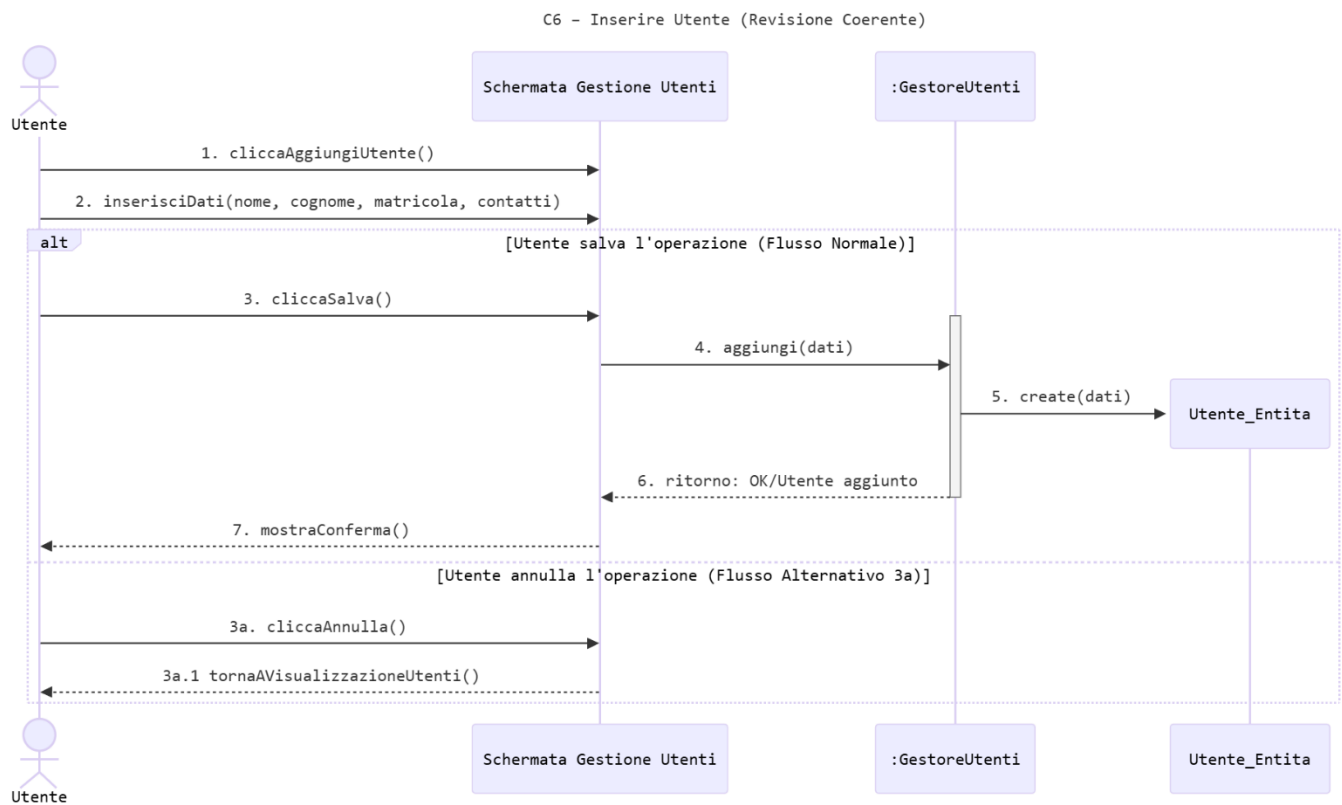


Figure 9: Diagramma sequenza C6 – Inserire Utente

### 2.3.5 C9 – Cercare Utente

Questo diagramma descrive la ricerca degli utenti. L'Utente può effettuare una ricerca semplice o avanzata. La **SchermataUtenti** invia la richiesta al **GestoreUtenti**, che esegue la ricerca con `cercaUtente(criteri)` e restituisce i risultati.

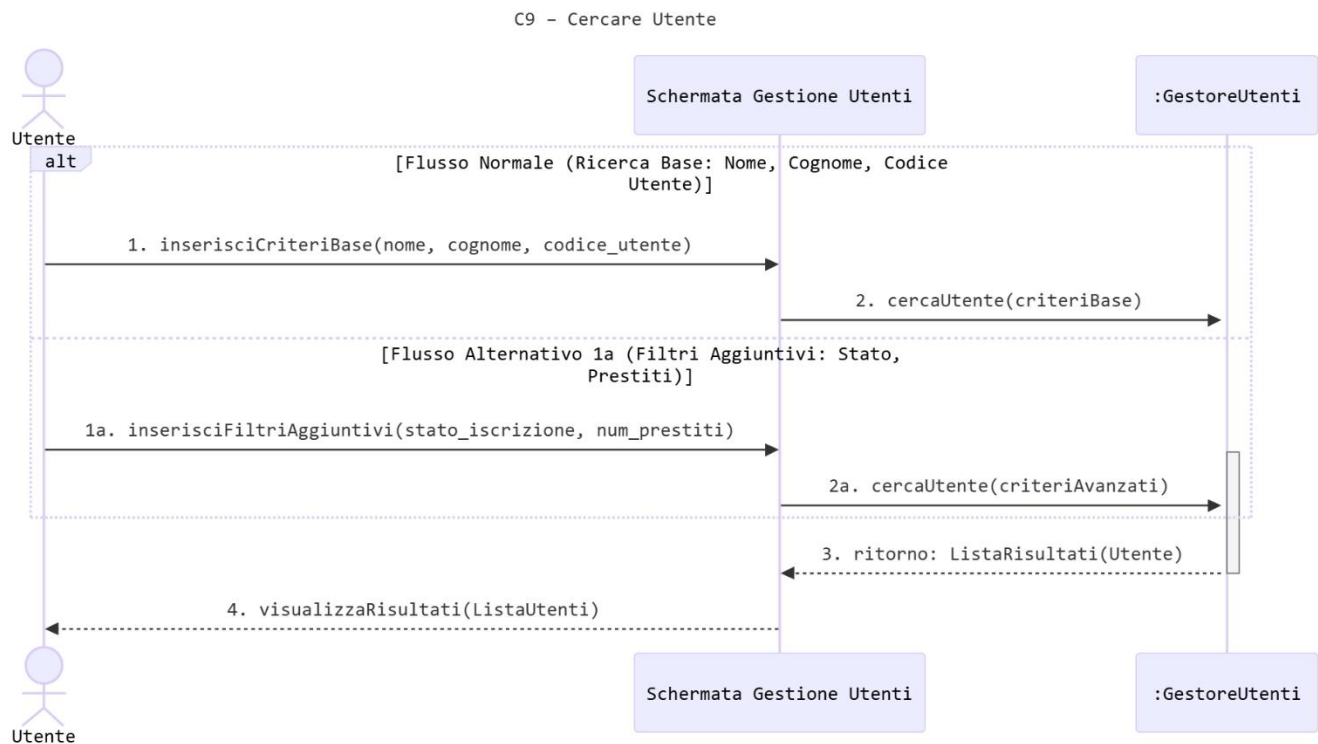


Figure 10: Diagramma sequenza C9 – Cercare Utente

### 2.3.6 C11 – Registrare Prestito

Il seguente diagramma è uno dei casi più importanti.

Il sistema deve:

1. Controllare che il libro sia disponibile.
2. Verificare che l'utente non abbia superato il limite di prestiti.

Se entrambi i controlli sono superati, il **GestorePrestiti**: aggiorna le copie disponibili del libro, e crea il nuovo Prestito con la data di scadenza.

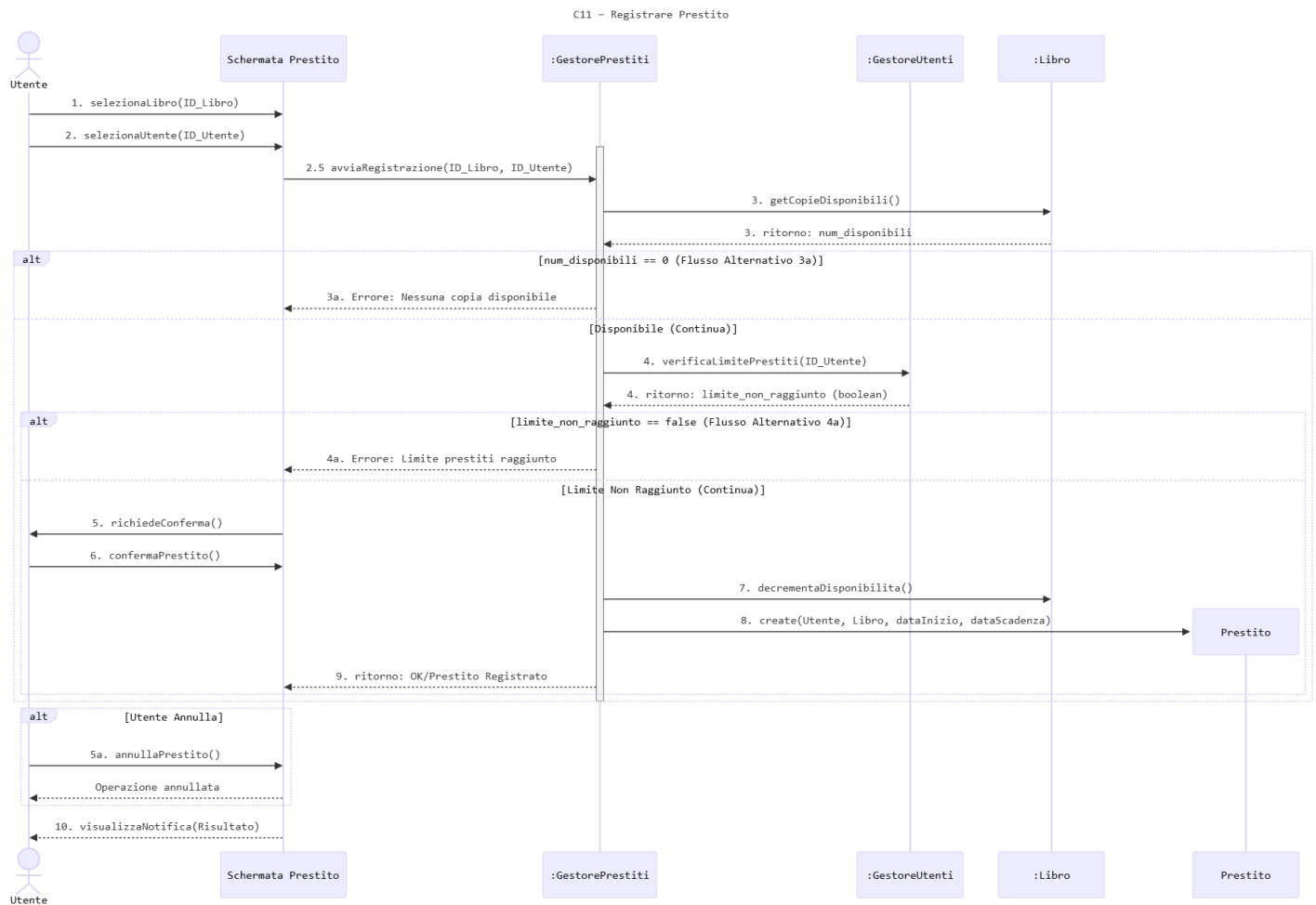


Figure 11: Diagramma sequenza C11 – Registrare Prestito

### 2.3.7 C15 – Visualizzare Prestiti Attivi

Il seguente diagramma di sequenza mostra come vengono visualizzati i prestiti attivi. La richiesta parte dall'Utente e viene gestita dal **GestorePrestiti**, che restituisce alla SchermataPrestiti l'elenco dei prestiti non ancora restituiti tramite `getPrestitiAttivi()`.

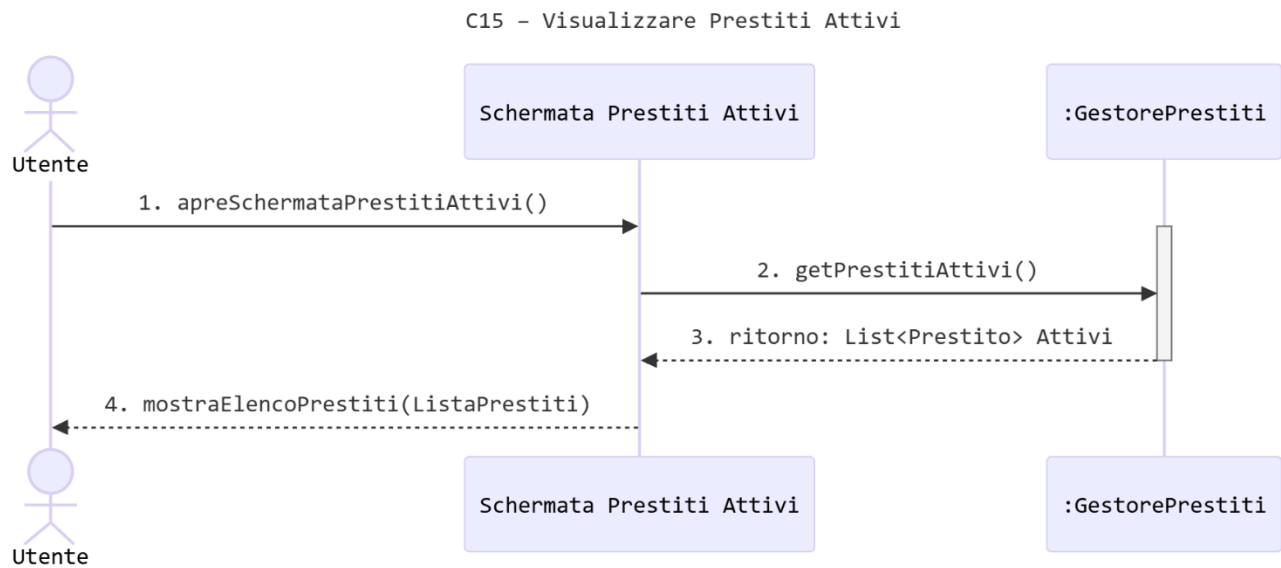


Figure 12: Diagramma sequenza C15– Visualizzare Prestiti Attivi

## 2.4 Principi di buona progettazione

Si garantisce l'aderenza ai principi di buona progettazione del software, migliorando modularità, manutenibilità e chiarezza del sistema.

### 2.4.1 Livelli di coesione

Nome classe	Livello coesione	Nota
ManagerGenerale<T>	Funzionale	Tutti i metodi della classe sono dedicati alla gestione generica di una lista. Tutte le operazioni svolgono un singolo compito ben definito.
GestoreUtenti	Funzionale	I metodi della classe lavorano tutti sullo stesso obiettivo: la gestione degli utenti. Tutte le operazioni sono orientate allo stesso scopo.
GestoreLibri	Funzionale	Tutti i metodi gestiscono esclusivamente i libri, mantenendo alta coesione attorno alle operazioni di aggiunta, rimozione e ricerca nella lista dei libri.
GestorePrestiti	Funzionale	Tutti i metodi operano insieme per la gestione dei prestiti, dalla ricerca alla restituzione dei prestiti attivi o associati a un utente. Le operazioni lavorano allo stesso scopo, realizzando un'unica funzionalità centrale.
Utente	Funzionale	La classe concentra tutte le sue operazioni sulla gestione dei dati dell'utente e sulla restituzione dei suoi prestiti, tutte le funzioni collaborano per modellare un utente.
Libro	Funzionale	Tutti i metodi della classe sono adibiti alla gestione e modifica degli attributi del libro. Le operazioni lavorano sullo stesso insieme di dati.
Prestito	Funzionale	I metodi della classe riguardano tutti la gestione del singolo prestito: controllano se è attivo, permettono di modificarne lo stato e restituiscono le informazioni collegate.

Figure 13: Livelli di coesione classi

### 2.4.2 Livelli di accoppiamento

Nome classe 1	Nome classe 2	livello accoppiamento	Nota
GestoreUtenti	Utente	per dati	La classe GestoreUtenti contiene la lista degli Utenti e utilizza i loro riferimenti solo per gestire la creazione, la ricerca e la rimozione di ogni singolo utente.
GestoreLibri	Libro	per dati	Essendoci una relazione 1 a molti, la classe GestoreLibri mantiene i riferimenti ai libri e li usa solo per le operazioni di aggiunta, rimozione o ricerca.
GestorePrestiti	Prestito	per dati	La classe GestorePrestiti conserva la lista dei Prestiti e lavora sui loro dati per gestire le operazioni relative ai prestiti (creazione, rimozione, ricerca).
GestorePrestiti	Utente	per dati	La classe GestorePrestiti utilizza il riferimento all'Utente solo per filtrare i prestiti relativi a quel determinato utente, senza accedere ad altri dati non necessari.
GestorePrestiti	Libro	per dati	La classe GestorePrestiti usa il riferimento al Libro per verificare la disponibilità e collegarlo al prestito, accedendo solo alle informazioni strettamente utili.
Prestito	Utente	per dati	La classe Prestito contiene il riferimento all'Utente solo perché necessario a identificare chi ha effettuato il prestito.
Prestito	Libro	per dati	La classe Prestito mantiene il riferimento alla classe Libro esclusivamente per sapere quale libro è stato preso in prestito.
GestoreUtenti	ManagerGenerale<T>	per timbro	La classe GestoreUtenti eredita da ManagerGenerale<T>, ma utilizza soltanto alcuni dei metodi generici ereditati, ricevendo quindi più operazioni di quelle che realmente le servono.
GestoreLibri	ManagerGenerale<T>	per timbro	La classe GestoreLibri estende ManagerGenerale<T>, ma sfrutta solo una parte delle funzionalità offerte dalla superclasse, usando quindi un "pacchetto" più grande del necessario.
GestorePrestiti	ManagerGenerale<T>	per timbro	Anche GestorePrestiti eredita metodi da ManagerGenerale<T> che non utilizza completamente, mostrando un accoppiamento per timbro.

Figure 14: Livelli di accoppiamento classi

### 2.4.3 Relazioni tra le Classi

Nel progetto è stata privilegiata l'associazione rispetto alla specializzazione (ereditarietà).

In particolare:

Esiste una relazione di aggregazione 1 a molti tra:

- *GestoreUtente* e *Utente*;
- *GestoreLibri* e *Libro*;
- *GestorePrestiti* e *Prestito*.

È presente una relazione di composizione 1 a molti tra *Utente* e *Prestito*, dato che ogni prestito esiste solo se associato a uno specifico utente: eliminando l'utente, cessano di esistere anche tutti i suoi prestiti.

È presente una relazione di composizione 1 a molti tra *Libro* e *Prestito*, poiché un prestito riguarda necessariamente un libro specifico. Se un libro viene rimosso dal sistema, anche i prestiti associati non sono più validi.

*GestorePrestiti* è associato sia a *Utente* che a *Libro*, poiché per creare o rimuovere un prestito sono necessari entrambi gli oggetti.

### 2.4.4 Riduzione dell'Accoppiamento

Per ridurre il livello di accoppiamento tra le classi, il progetto prevede una separazione netta tra:

- Gestione degli utenti (*GestoreUtenti*);
- Gestione dei libri (*GestoreLibri*);
- Gestione dei prestiti (*GestorePrestiti*).

In questo modo ogni classe gestore dipende solo dai dati strettamente necessari per il proprio ruolo, riducendo sia la dipendenza diretta che la propagazione delle modifiche. Tuttavia:

- *GestorePrestiti* ha un accoppiamento per dati con *Utente* e *Libro*, poiché ha necessariamente bisogno dei loro riferimenti per costruire, ricercare o eliminare un prestito.
- Le classi *Utente* e *Libro* hanno un accoppiamento per dati con *Prestito*, in quanto devono conoscere i prestiti a loro associati.
- Le classi dei gestori hanno un accoppiamento per timbro con la classe generica *ManagerGenerale<T>*, perché ereditano un insieme di metodi più ampio rispetto a quelli che effettivamente utilizzano. Questo permette di riutilizzare del codice, ma introduce un livello di accoppiamento non completamente ottimale.

