

DOCUMENTO DI PROGETTAZIONE

GESTIONE DI UNA BIBLIOTECA UNIVERSITARIA

GRUPPO 20

Componenti:

- **Angelo Casella**
- **Matteo Adinolfi**
- **Luisa Dong**

Indice

1. Decisioni di Design	2
1.1 Modularità	2
1.2 Persistenza dei Dati	2
1.3 Principi SOLID applicati	2
1.4 Manutenibilità (Architettura MVC)	3
1.5 Ricerca e Ordinamento :	3
1.6 Gestione Errori	3
1.7 Strumenti e Tecnologie	4
2. Diagrammi UML	5
2.1 Diagramma delle classi - basso livello di dettaglio	5
2.1 Diagramma delle classi - alto livello di dettaglio	6
2.2 Diagrammi di sequenza	7
2.2.1 Caso UC-1: Registrazione Libro	7
2.2.2 Caso UC-2: Modifica Libro	8
2.2.3 Caso UC-3: Eliminazione Libro	9
2.2.4 Caso UC-4: Ricerca Libro	10
2.2.5 Caso UC-7: Elimina Utente	11
2.2.6 Caso UC-9: Richiesta di un prestito	12
2.2.7 Caso UC-10: Visualizzazione lista prestiti	13
2.2.8 Caso UC-11: Restituzione di un libro	14
2.2.9 Caso UC-12: Salvataggio file	15
2.2.10 Caso UC-15: Ricerca di un prestito	16
2.3 Diagramma dei package	17
2.3.1 Tabella Dei Packages	18
3. Tabella di Coesione	19
4. Tabella Di Accoppiamento	20

1. Decisioni di Design

1.1 Modularità

Il sistema è stato suddiviso in moduli funzionali distinti (pacchetti Entità ,Controllers, Tipologie Archivi) per garantire un'alta modularità. Questa scelta architetturale permette di:

- **Isolare le dipendenze:** Ogni parte del sistema comunica con le altre solo attraverso interfacce pubbliche ben definite.
- **Facilitare l'evoluzione:** L'aggiunta di nuove funzionalità (es. una nuova tipologia di risorsa) può avvenire aggiungendo nuove classi senza dover modificare pesantemente il codice già esistente.

1.2 Persistenza dei Dati

L'applicazione gestisce la persistenza mediante archivi serializzati (Serializable). I dati vengono salvati su file esterni con un meccanismo di aggiornamento automatico gestito dalla classe Archivio . Questo garantisce che lo stato del sistema (Libri, Utenti e Prestiti) sia sempre consistente e recuperabile tra una sessione e l'altra, supportando l'affidabilità del sistema.

1.3 Principi SOLID applicati

1.3.1 Single Responsibility Principle (SRP)

Il progetto rispetta il principio assegnando a ogni classe un'unica responsabilità ben definita:

- Entità (Libro, Utente, Prestito): Gestiscono esclusivamente la struttura dei dati e il loro stato interno.

- Archivi dati (Archivio Libri, Archivio Utenti, Archivio Prestiti): Si occupano solo della gestione delle liste (operazioni di aggiunta, rimozione, ricerca).
- Controller (Main Controller, Gestione...Controller): Gestiscono unicamente l'interazione con l'utente e l'aggiornamento dell'interfaccia grafica
- Archivio: Gestisce solo l'aggregazione dei dati e le operazioni di salvataggio/caricamento su file.

1.4 Manutenibilità (Architettura MVC)

L'elevata manutenibilità è garantita dalla separazione netta delle responsabilità tramite il pattern **MVC (Model-View-Controller)**:

- **Model (Dati):** Le modifiche alla struttura dei dati o alle regole di business non impattano sull'interfaccia grafica.
- **View (Grafica):** È possibile aggiornare il layout (FXML) e lo stile senza dover ricompilare o alterare la logica applicativa Java.
- **Controller (Logica):** I flussi di interazione possono essere modificati nel codice senza rischio di corrompere la consistenza dei dati sottostanti.

1.5 Ricerca e Ordinamento :

- **Ordinamento:** Gestito tramite Comparator, per mantenere le classi entità prive di logica di ordinamento fissa e consentire ordinamenti multipli (es. alfabetico o per codice).
- **Ricerca:** Implementata in modalità case-insensitive per garantire all'utente un'esperienza più fluida

1.6 Gestione Errori

L'applicazione effettua controlli automatici per evitare errori durante l'uso:

- **Controllo inserimento dati:** Verifica che l'utente abbia compilato tutti i campi necessari e che i dati siano inseriti correttamente.
- **Controllo scadenze:** Rileva automaticamente se un libro non è stato restituito entro i tempi previsti.
- **Coerenza dell'archivio:** Impedisce l'inserimento di un libro o studente già esistente e gestisce i casi in cui si tenta di ricercare un oggetto che non esiste.

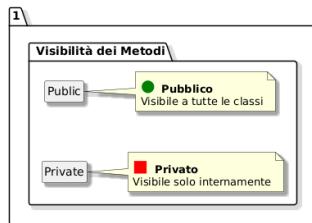
1.7 Strumenti e Tecnologie

La scelta degli strumenti è stata orientata a garantire l'efficienza dello sviluppo e la manutenibilità del progetto:

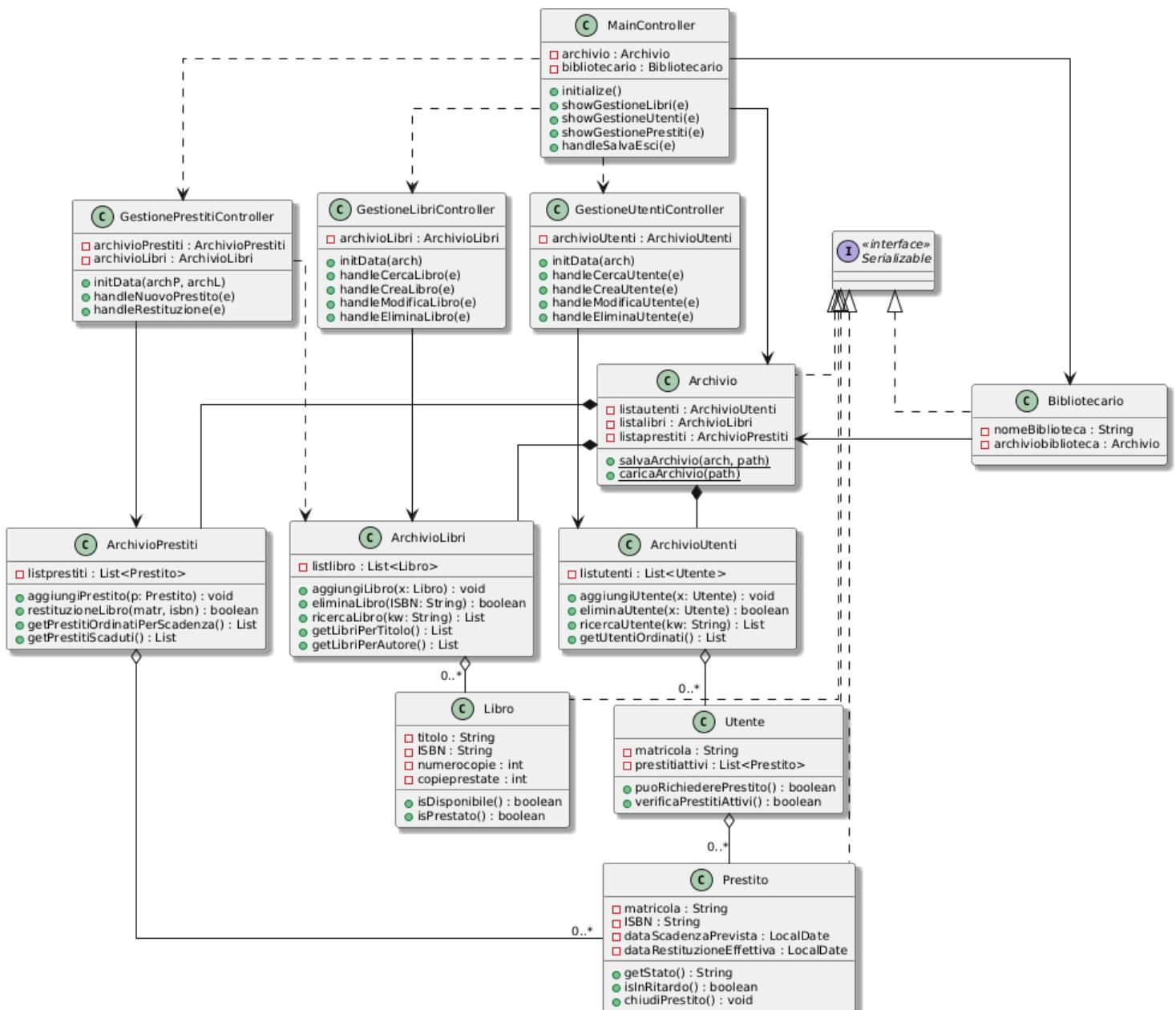
- **Java:** Linguaggio di programmazione principale utilizzato per l'intera logica applicativa.
- **Maven:** Strumento per la gestione automatica delle librerie esterne e per l'automazione del processo di compilazione (build).
- **JUnit:** Framework dedicato alla scrittura ed esecuzione dei test automatizzati per verificare il corretto funzionamento del codice.
- **JavaFX** (con FXML e SceneBuilder): Tecnologia per l'interfaccia utente che permette di separare la grafica (View) dalla logica di programmazione, facilitando la gestione del codice.

2. Diagrammi UML

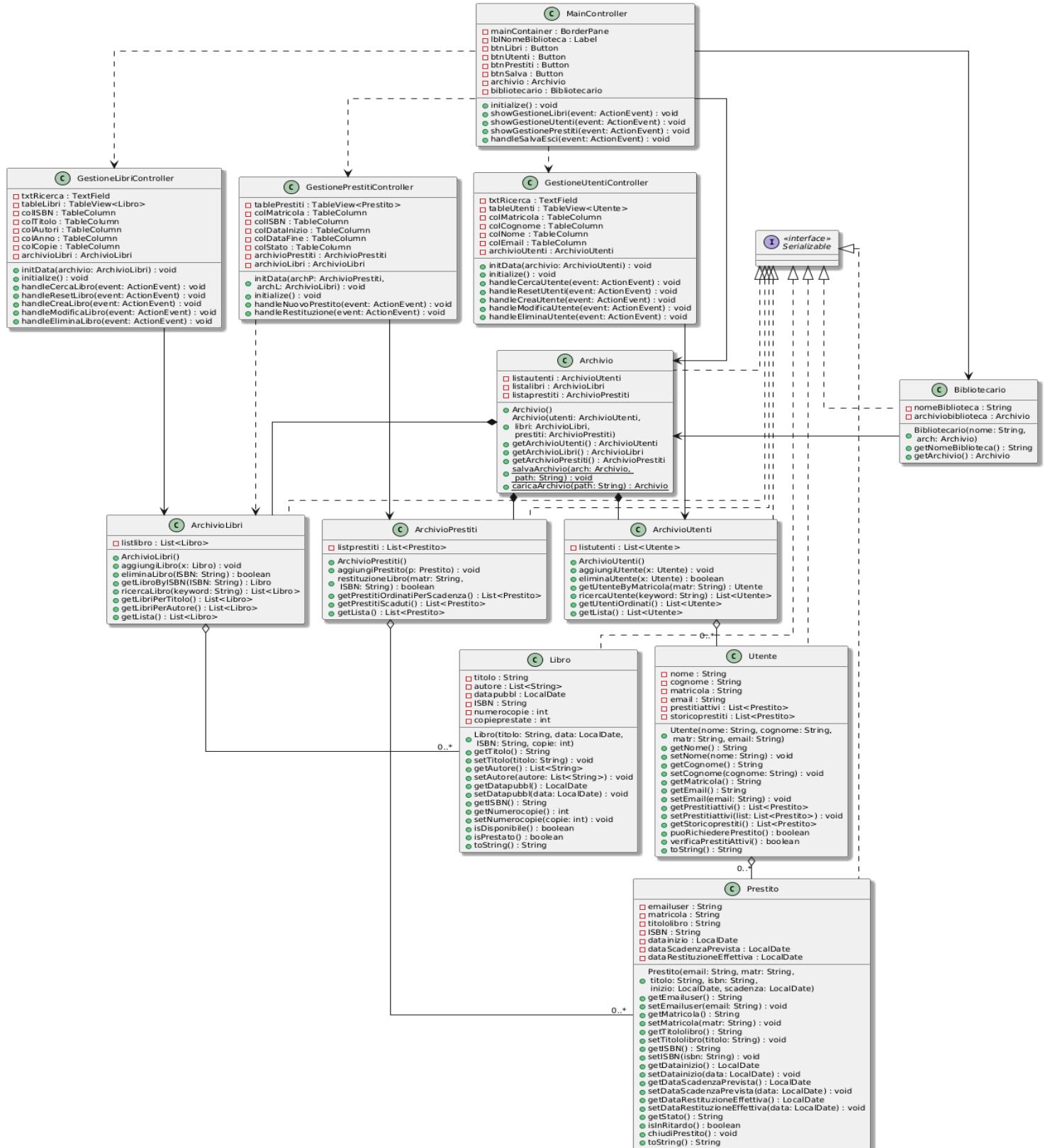
- Legenda



2.1 Diagramma delle classi - livello di dettaglio basso

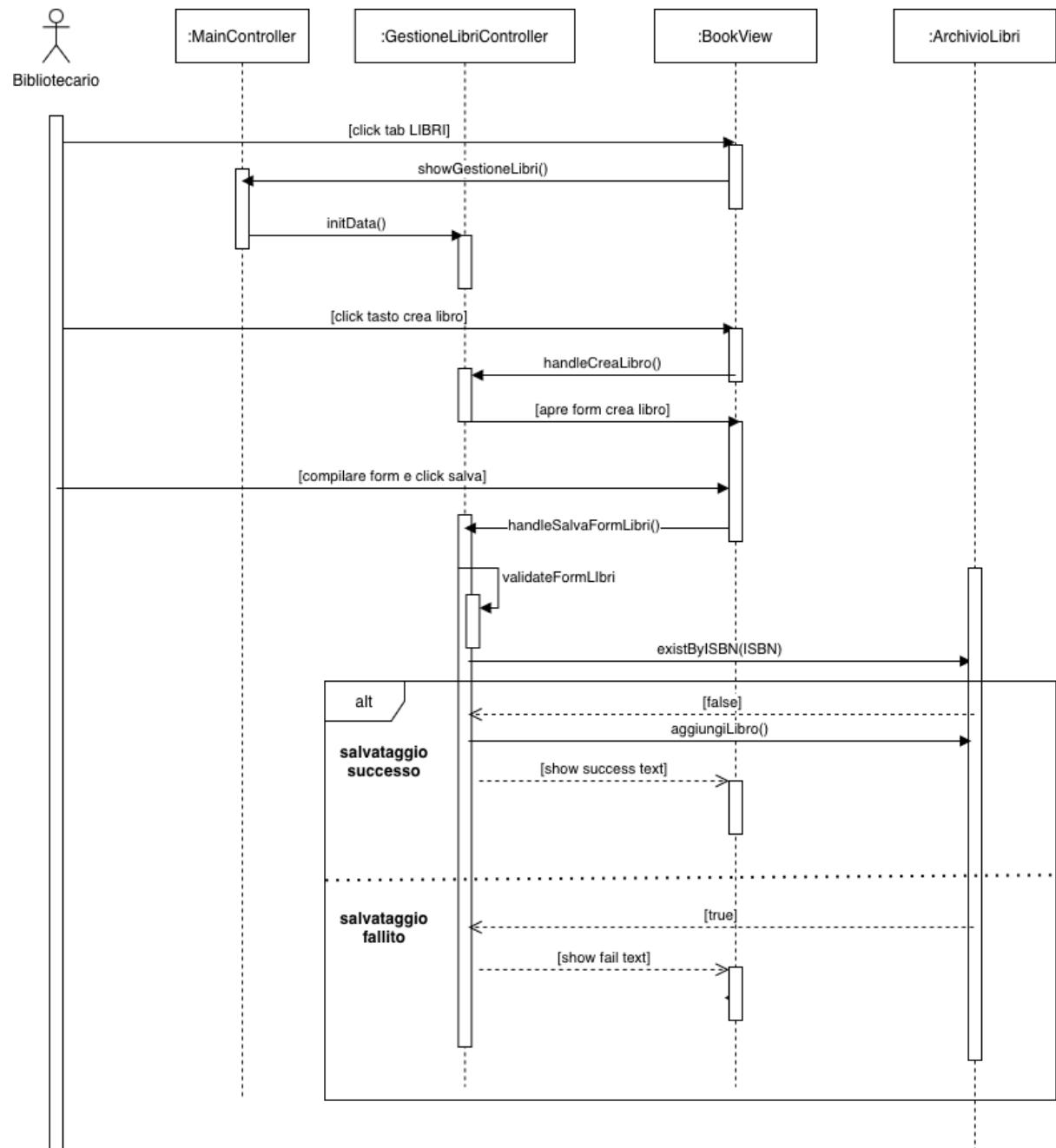


2.1 Diagramma delle classi - livello di dettaglio alto

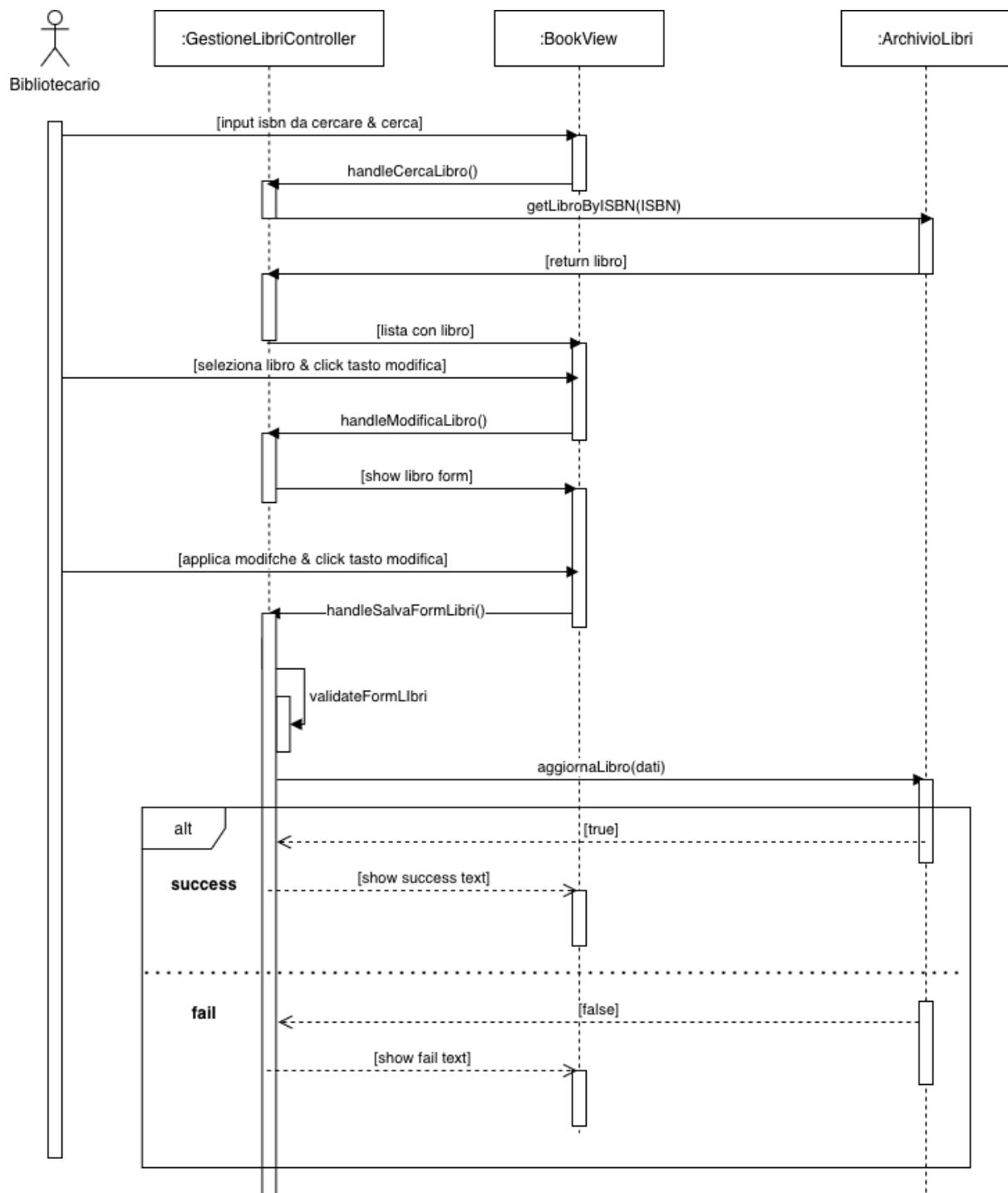


2.2 Diagrammi di sequenza

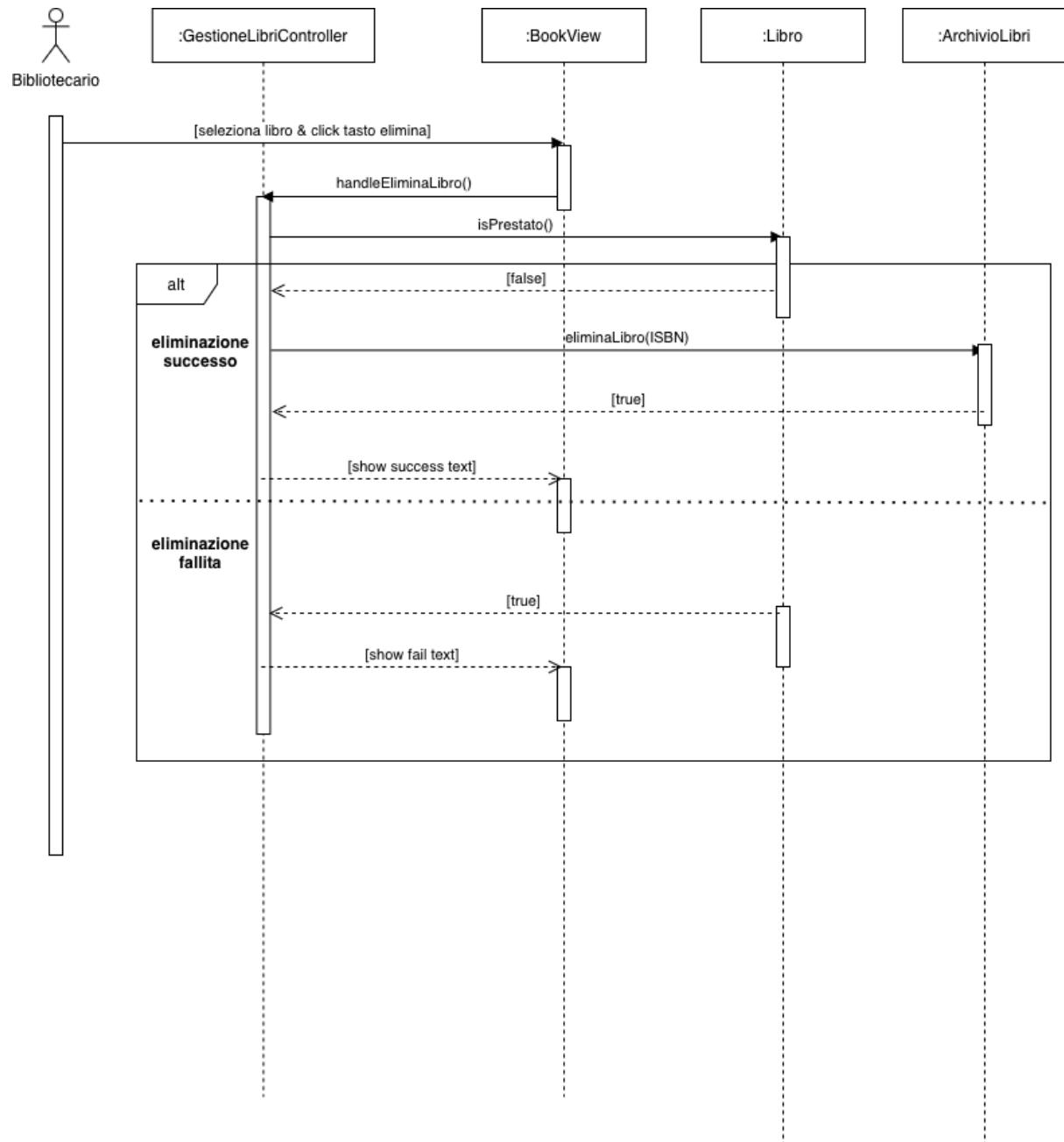
2.2.1 Caso UC-1: Registrazione Libro



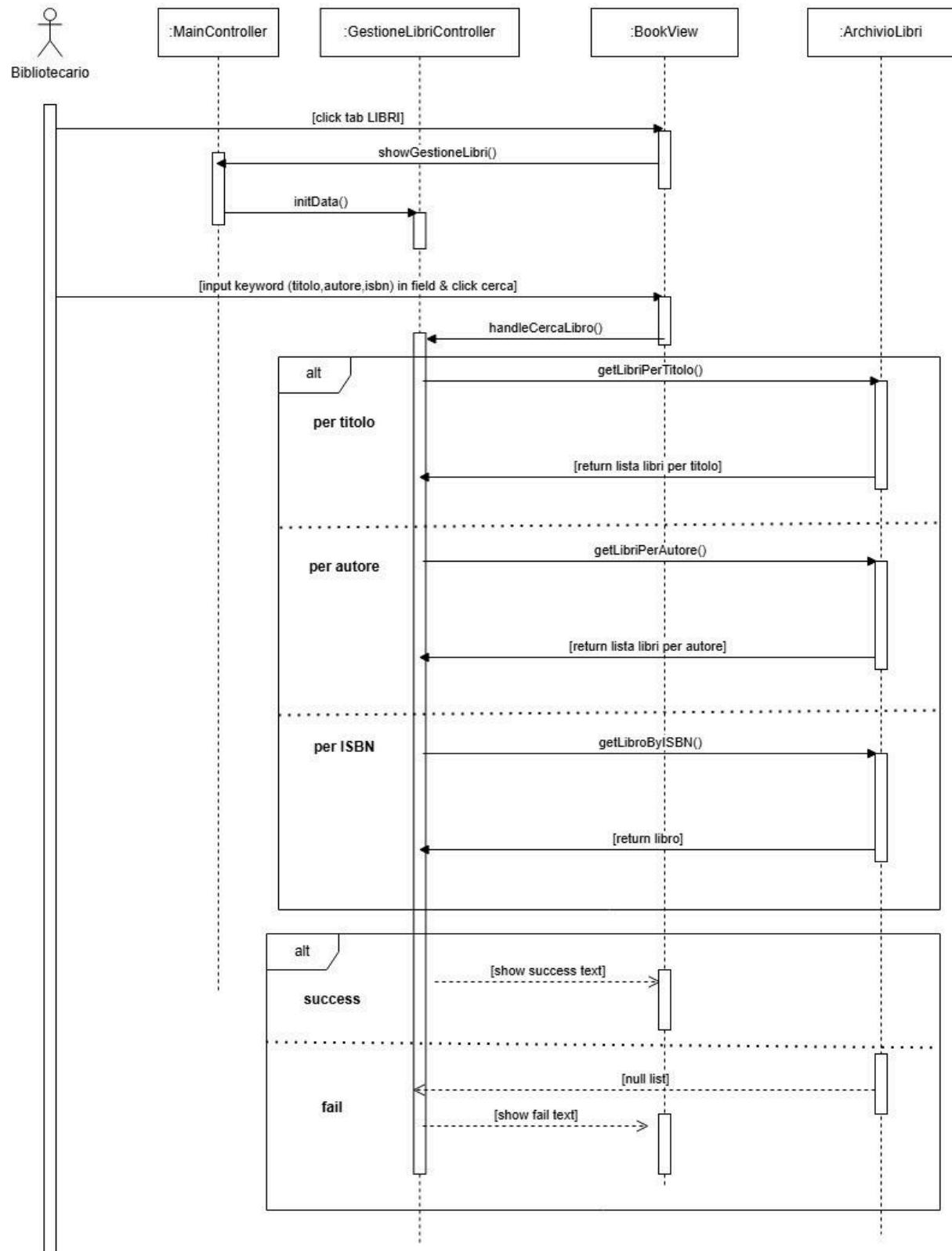
2.2.2 Caso UC-2: Modifica Libro



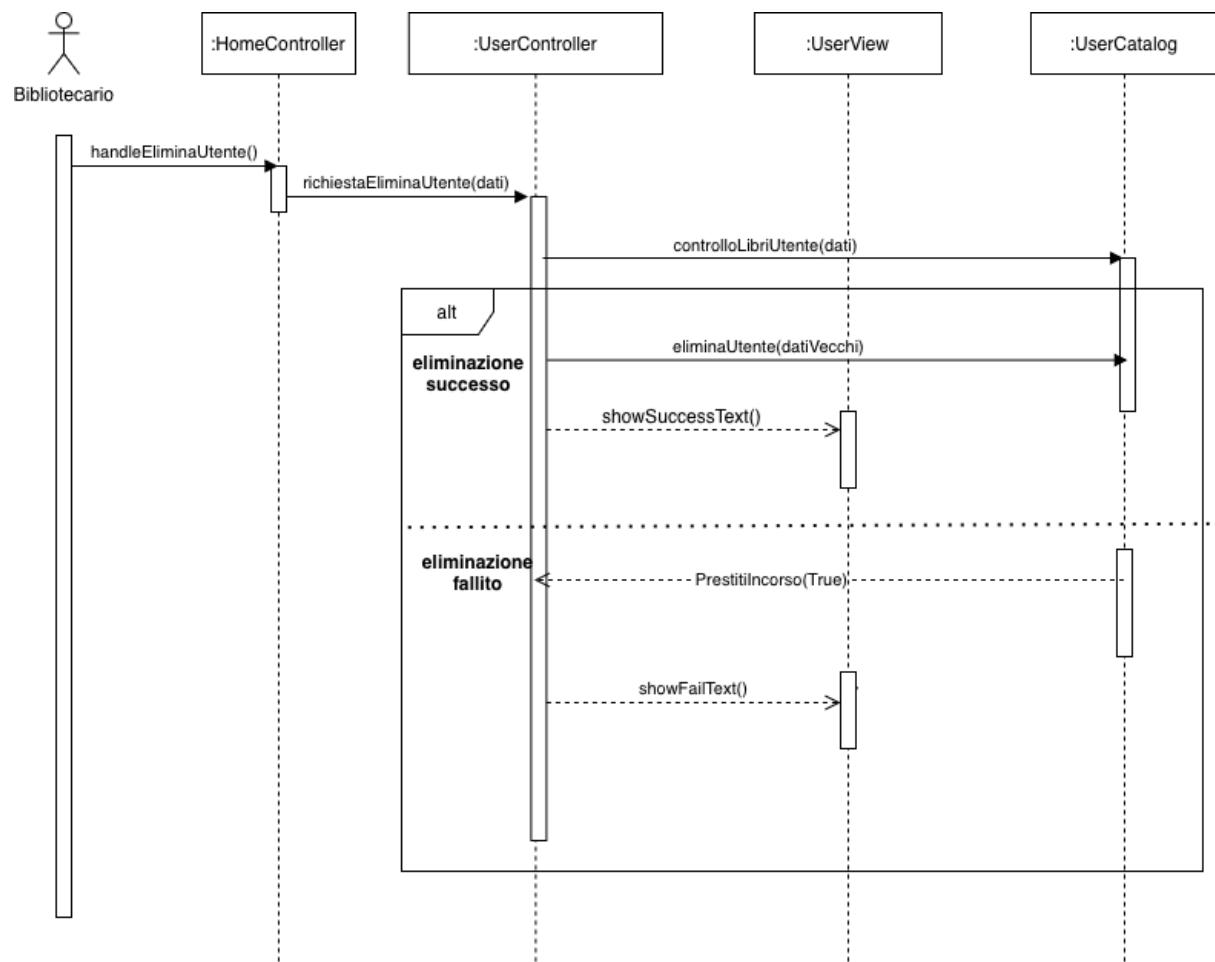
2.2.3 Caso UC-3: Eliminazione Libro



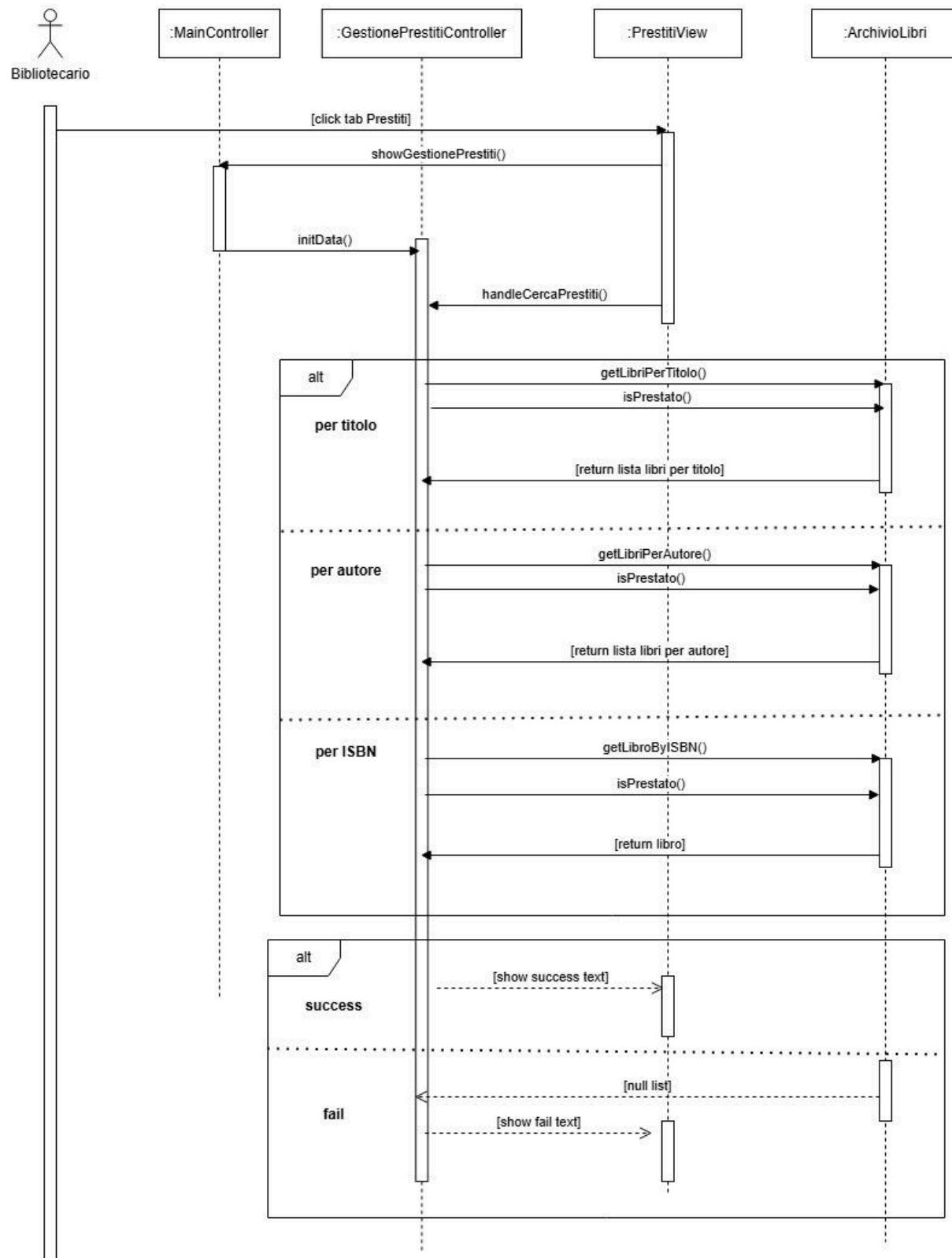
2.2.4 Caso UC-4: Ricerca Libro



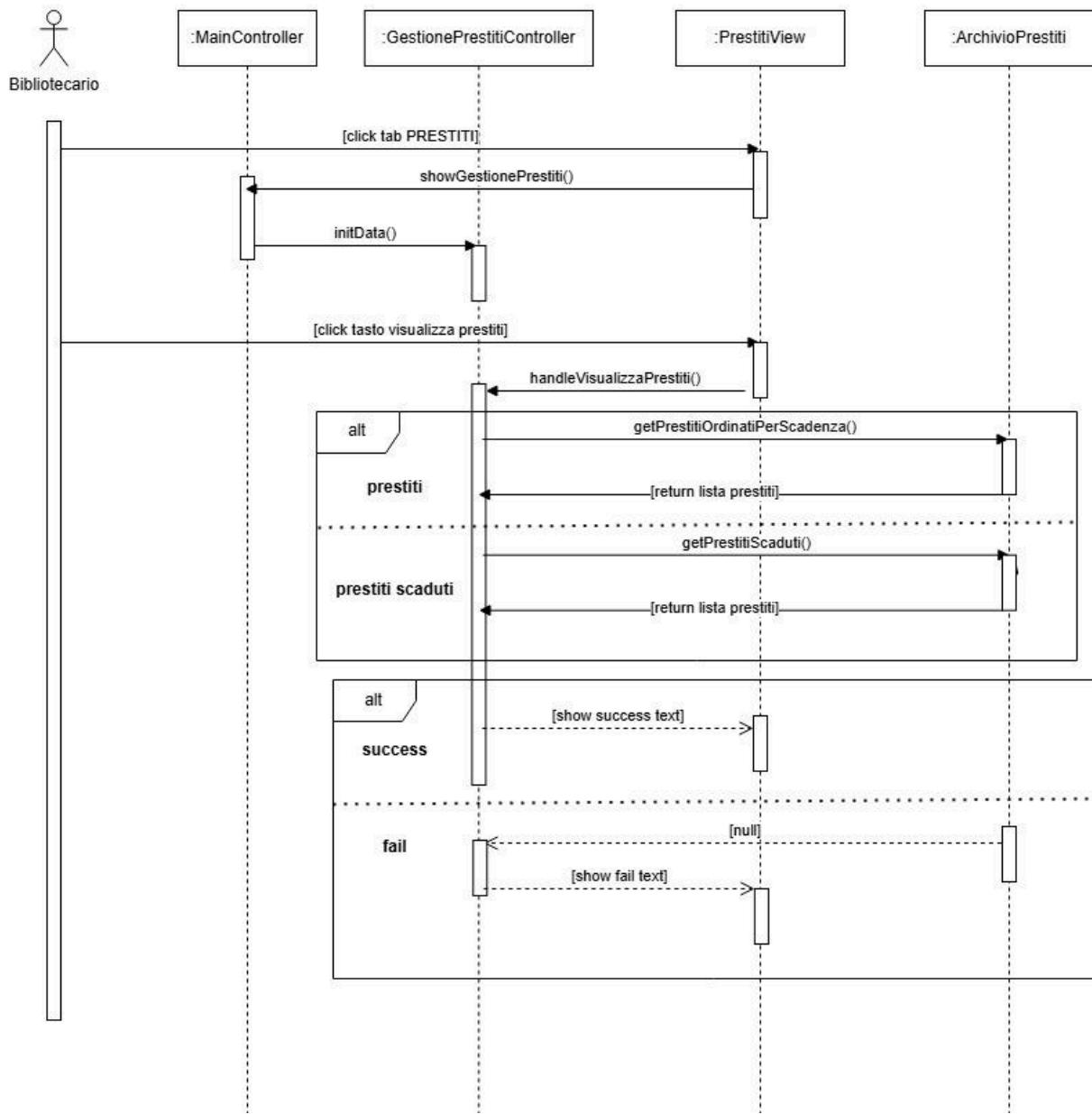
2.2.5 Caso UC-7: Elimina Utente



2.2.6 Caso UC-9: Richiesta di un prestito

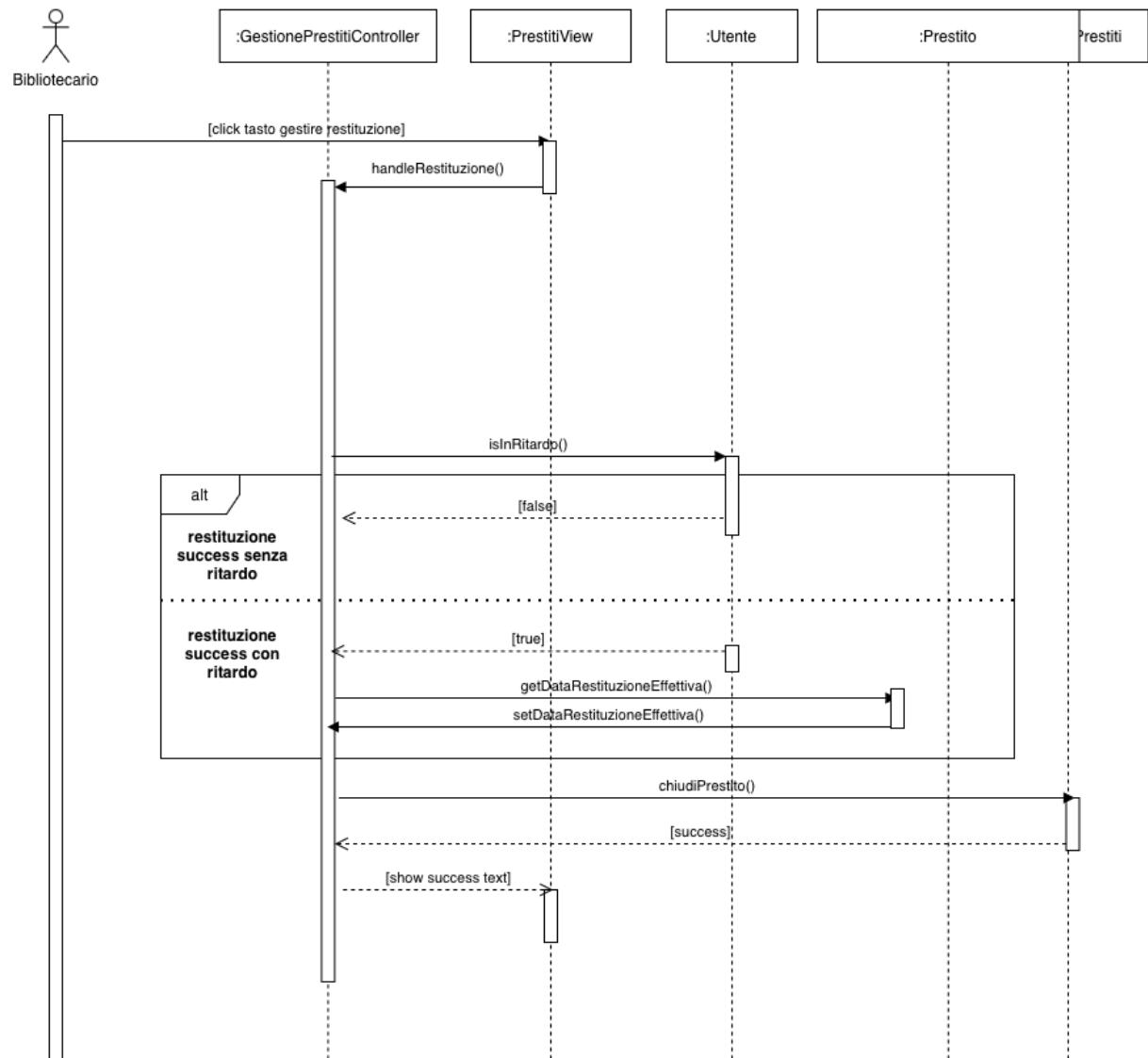


2.2.7 Caso UC-10: Visualizzazione lista prestiti

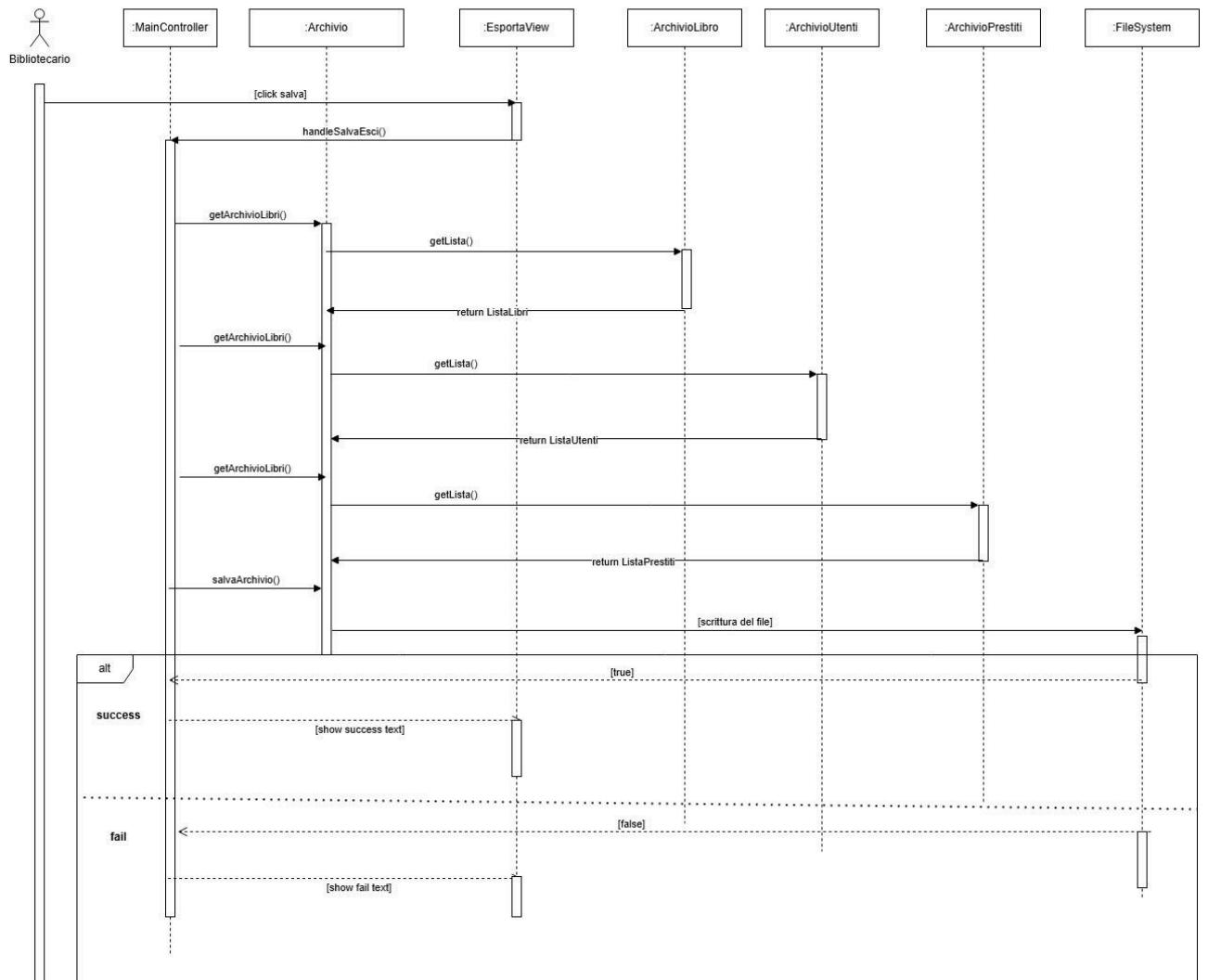


2.2.8 Caso UC-11: Restituzione di un libro

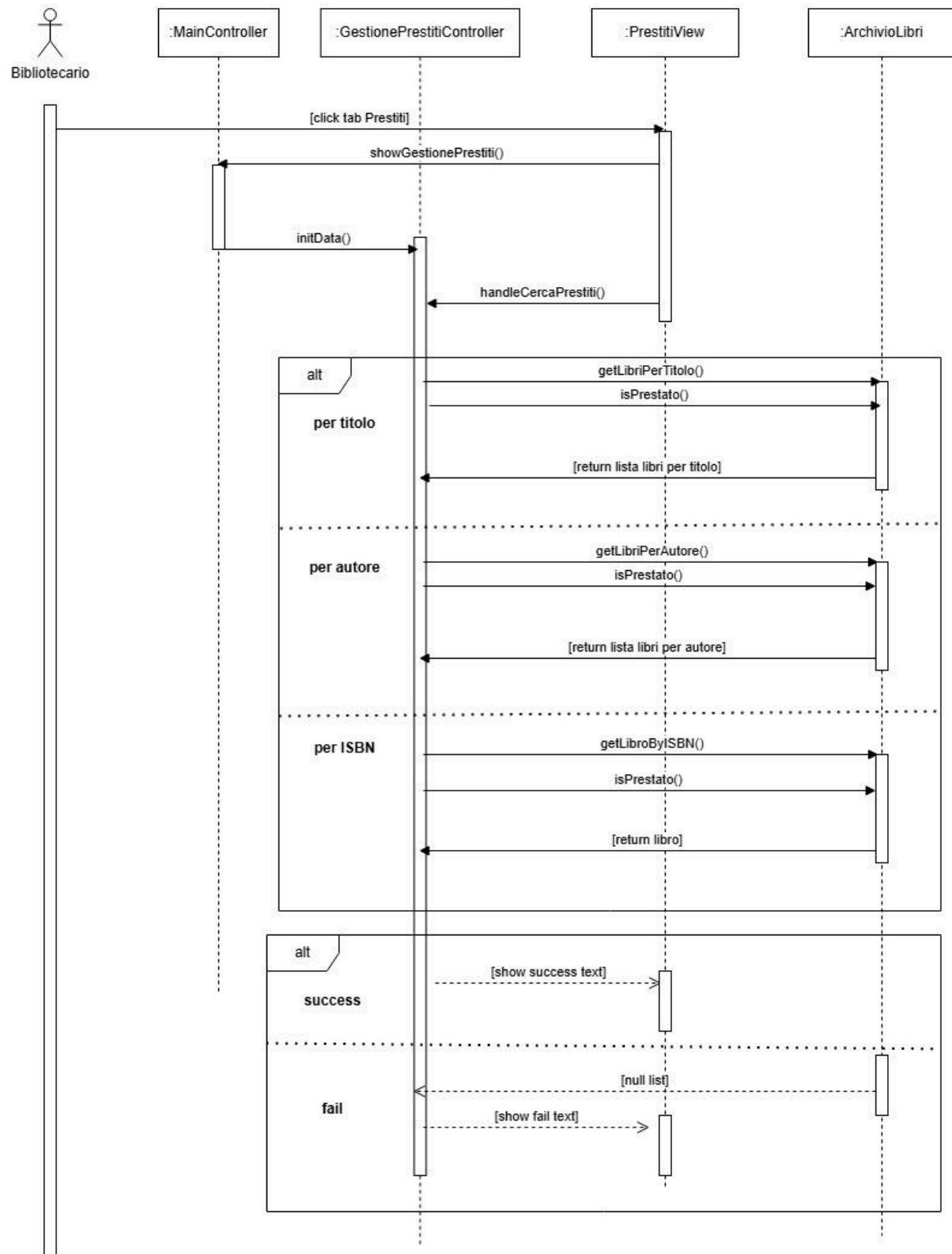
Precondizione: il bibliotecario ha effettuato la ricerca ed individuato il libro da restituire



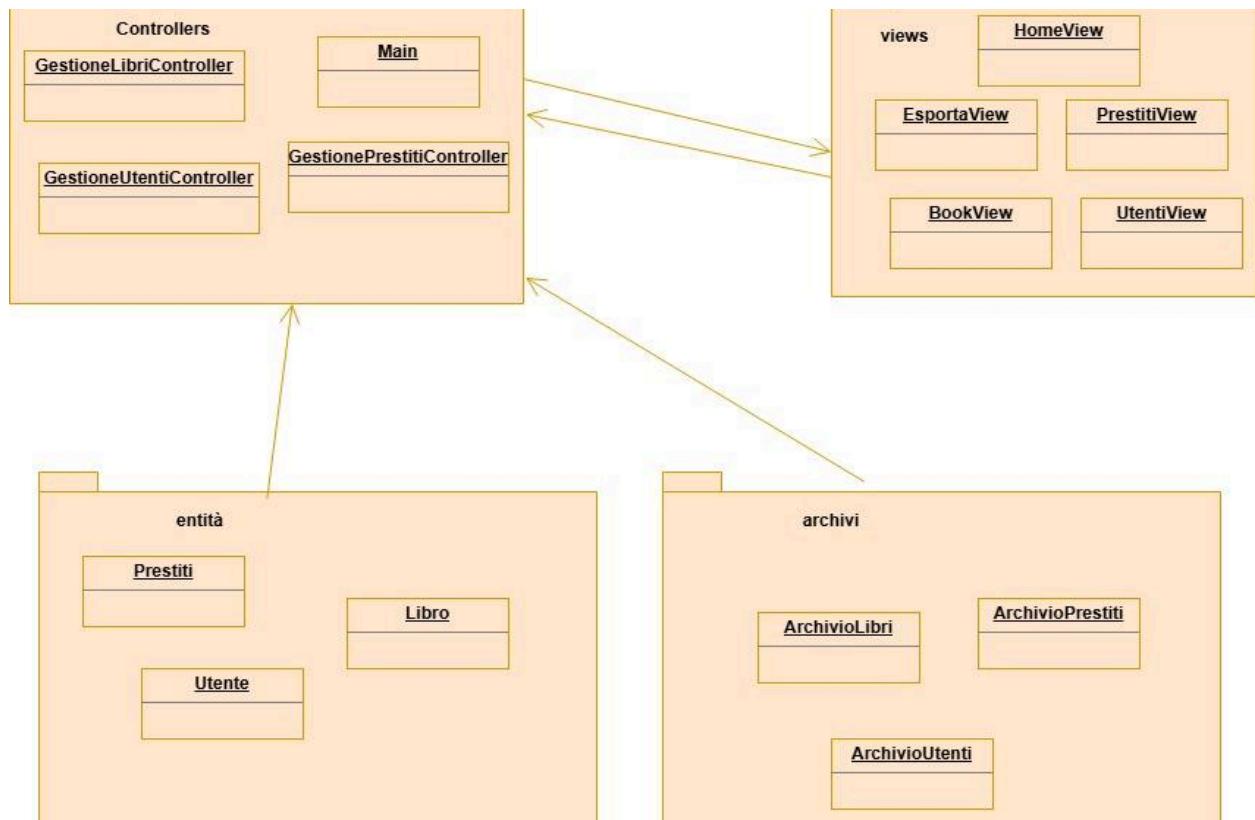
2.2.9 Caso UC-12: Salvataggio file



2.2.10 Caso UC-15: Ricerca di un prestito



2.3 Diagramma dei package



2.3.1 Tabella Dei Packages

Nome Package		Descrizione
Controllers	Main	Il pacchetto Controllers contiene le classi che gestiscono la logica di controllo e interazioni tra il bibliotecario e il sistema, quindi comunica con le views, ricava ed aggiorna dati dell'archivio ed opera sugli oggetti dell'entità.
	GestioneLibriController	
	GestioneUtentiController	
	GestionePrestitiController	
Views	HomeView	Il pacchetto Views contiene l'interfaccia utente UI del sistema. Si presentano le varie views che vengono mostrate al bibliotecario.
	UtenteView	
	BookView	
	PrestitiView	
	EsportaView	
Entità	Prestiti	Il pacchetto Entità contiene le classi che corrispondono agli oggetti. Si occupa della gestione e rappresentazione dei dati.
	Utente	
	Libro	
Archivi	ArchivioUtenti	Il pacchetto Archivi contiene gli archivi necessari per la persistenza dei dati, garantendo operazioni come l'aggiornamento, recupero e salvataggio.
	ArchivioLibri	
	ArchivioPrestiti	

3. Tabella di Coesione

Nome della classe	Livello Coesione	Descrizione
Libro / Utente / Prestito	Funzionale (Ottimo)	Le classi contengono solamente gli attributi e i metodi che servono alla definizione delle singole entità.
Archivio Libri / Utenti / Prestiti	Funzionale (Ottimo)	Ogni archivio si occupa di gestire la propria lista di oggetti attraverso i metodi dichiarati.
Bibliotecario	Funzionale (Ottimo)	La classe ha l'unica responsabilità di rappresentare la sessione dell'attore e mantenere il riferimento ai dati correnti.
Gestione Controller	Comunicazione(Buono)	I metodi all'interno del controller eseguono compiti funzionalmente diversi (es. Ricerca o Inserimento), ma sono raggruppati perché operano tutti sugli stessi dati di input (l'archivio spec.) e aggiornano la stessa struttura di output.
MainController	Comunicazione(Buono)	I metodi gestiscono attività diverse (navigazione verso viste differenti, salvataggio), ma sono coesi perché condividono e manipolano lo stesso contesto dati.
Archivio	Comunicazione (Buono)	La classe contiene tre elementi diversi ma che sono messi insieme perché hanno lo stesso ciclo di vita.

4. Tabella Di Accoppiamento

Nome Classe 1	Nome Classe 2	Livello Accoppiamento	Descrizione
MainController	Archivio	Per Dati (Ottimo)	Il controller principale possiede il riferimento all'Archivio solo per gestirne il ciclo di vita (caricamento/salvataggi) e passarlo ai sotto-controller.
Gestione Libri Controller	ArchivioLibri	Per Timbro (Buono)	Il controller riceve l'intera struttura dati ArchivioLibri tramite il metodo initData per aggiungere oggetti alla tabella, ma ne utilizza solo i metodi pubblici.
Gestione Prestiti Controller	ArchivioPrest iti	Per Timbro (Buono)	Il controller riceve l'istanza dell'archivio per invocare alcuni suoi metodi, senza accedere ai dettagli interni della lista.

Nome Classe 1	Nome Classe 2	Livello Accoppiamento	Descrizione
Gestione Prestiti Controller	ArchivioLibri	Per Timbro (Buono)	Il controller deve accedere all'Archivio Libri per decrementare/incrementare le copie disponibili durante le operazioni di prestito.
Bibliotecario	Archivio	Per Dati (Ottimo)	Il Bibliotecario mantiene un riferimento all'Archivio per lavorare.
Archivio	ArchivioLibri	Per Dati (Ottimo)	La classe Archivio istanzia e contiene i riferimenti ai tre sotto-archivi.
Utente	Prestito	Per Dati (Ottimo)	L'entità Utente contiene una lista di oggetti Prestito per mantenere lo storico, scambiando solo i dati necessari.