University Library: Design

Group 9

Roberto La Brocca, Angelo Palladino, Pasquale Santoriello, Chiara Stabile

1. Project Reference Architecture..... 2

1.2 Package Diagram.....4

2. Class Diagrams..... 5

2.1 Class Diagram - Package Model.....5

2.2 Class Diagram - Package Controller.....8

3. Sequence Diagrams.....10

3.1 Use case 1.1..... 10

3.2 Use case 1.2..... 12

3.3 Use case 1.3..... 14

3.4 Use case 1.4..... 16

3.5 Use case 2.1..... 18

3.5 Use case 2.2..... 20

3.6 Use case 2.3..... 22

3.7 Use case 2.4..... 24

3.8 Use case 3.1..... 26

3.9 Use case 3.2..... 28

4. Interface Mock-up(s).....30

4.1 Home Page..... 31

4.2 Users Section.....31

4.3 Books Section..... 34

4.4 Loans Section.....36

5. Traceability Matrix Update.....38

1. Project Reference Architecture

In questa fase viene definito il design architetturale del sistema utilizzato, il cui obiettivo è stabilire una struttura organizzata basata sulla separazione dei ruoli.

L'architettura di riferimento del progetto è basata sul pattern di design **Model-View-Controller** (MVC), utilizzato per organizzare e strutturare il codice in tre componenti logici interconnessi.
Essi, nonostante lavorino insieme, hanno ruoli distinti:

- Model è il modello dell'applicazione, contiene i dati e i business flow, e non si occupa di come i dati vengono mostrati.
  Il nostro progetto presenta dunque le entità Libro, Utente e Prestito e le entità che gestiscono cosa è possibile fare con esse: GestioneUtente, GestioneLibro e GestionePrestito.
  Quando il bibliotecario vuole effettuare un'operazione (come la registrazione di un nuovo utente), il Controller invia la richiesta al Model, il quale si occupa di gestire l'operazione e assicurarsi che lo stato dell'applicazione rimanga valido e coerente.
- View è la vista dell'applicazione, rappresenta ciò che viene mostrato e con cui si ha interazione.
  La view è dunque il componente più passivo in quanto non prevede alcuna logica applicativa, si limita a mostrare i dati, quando ad esempio viene premuto un pulsante o viene digitato qualcosa in un campo di testo, esso raccoglie l'input e lo passa al Controller.
- Controller è il controllo dell'applicazione, fondamentale in quanto traduce le azioni umane in istruzioni per la logica di business.
  Esso riceve l'input (l'azione) dalla View e decide cosa deve esser fatto; in base a ciò comunica con il Model per poi decidere cosa mostrare in risposta (come una pagina di errore o di conferma).

La nostra scelta sull'utilizzo di tale pattern architetturale ha le seguenti motivazioni:

- Uno dei principali principi di buona progettazione è la separazione delle preoccupazioni (Separation of Concerns), secondo cui aspetti diversi del sistema sono gestiti da moduli distinti e non sovrapposti. Esso è il principio su cui si basa l'architettura MVC, la quale garantisce una chiara divisione dei compiti tra i suoi componenti.
- L'architettura Model-View-Controller facilita il testing unitario, i Model possono infatti essere testati in modo indipendente dai Controller e dalle View, permettendo di definire test unitari mirati migliorando l'affidabilità del codice.
- L'applicazione è sviluppata utilizzando JavaFX, la quale aderisce nativamente ai principi dell'architettura MVC.
- La separazione tra i tre componenti rende il sistema più flessibile e adattabile ad eventuali cambiamenti, in quanto permette di modificare la View (interfaccia utente) senza alterare il Model, cioè la logica di business, il che è fondamentale per la scalabilità del progetto.

Ad esso è stato affiancato il **Design Pattern Singleton**, allo scopo di garantire che una determinata classe abbia una sola istanza e di fornire un'interfaccia pubblica per ottenerla.
L'implementazione del pattern Singleton prevede che nel creare una classe si dichiari:

1. un costruttore privato; in questo modo il costruttore che è necessario chiamare per istanziare un oggetto da questa classe, è utilizzabile solo all'interno della classe stessa;
2. una variabile privata e statica dello stesso tipo della classe, chiamata "*instance*";
3. un metodo statico pubblico , chiamato *getInstance,* che definisce una semplice logica per restituire sempre la stessa istanza (unica) della classe stessa.

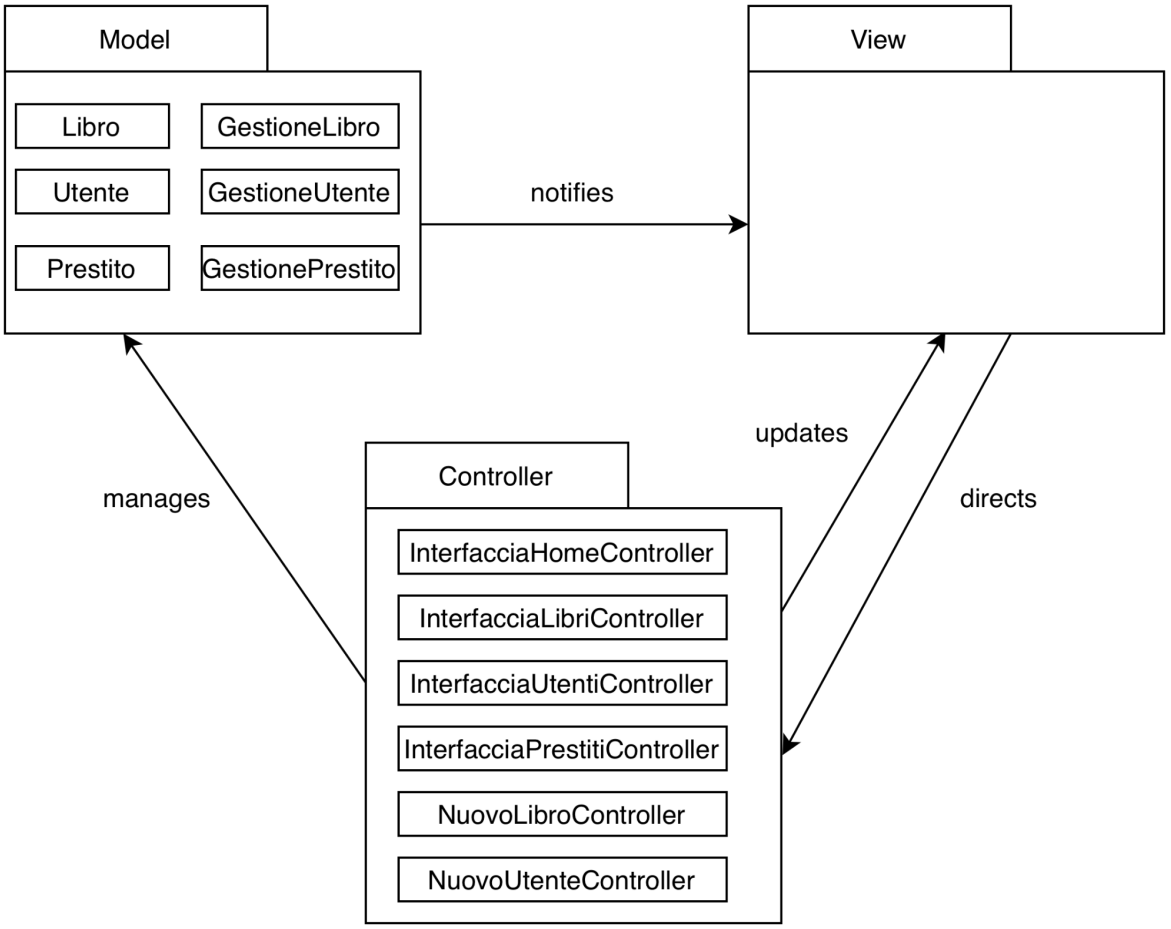
La nostra scelta sull'utilizzo di tale pattern ha le seguenti motivazioni:

- garantire alle classi Controller che utilizzano le classi di Gestione di lavorare sulle stesse strutture dati in memoria, evitando di istanziarne di non correlate;

- ottimizzare l’allocazione di memoria all’avvio dell’applicazione: nel momento in cui accede alle varie sezioni di Gestione, viene creata in memoria la rispettiva area dati.
- gestire la logica della persistenza dei dati in modo indipendente, ovvero ogni istanza delle classi di Gestione gestisce l’input/output sul file in relazione ai propri dati.

1.2 Package Diagram

In questa fase viene definito il diagramma dei package, esso rappresenta l'organizzazione logica del progetto ed è pertanto suddiviso in tre pacchetti secondo l'architettura MVC.

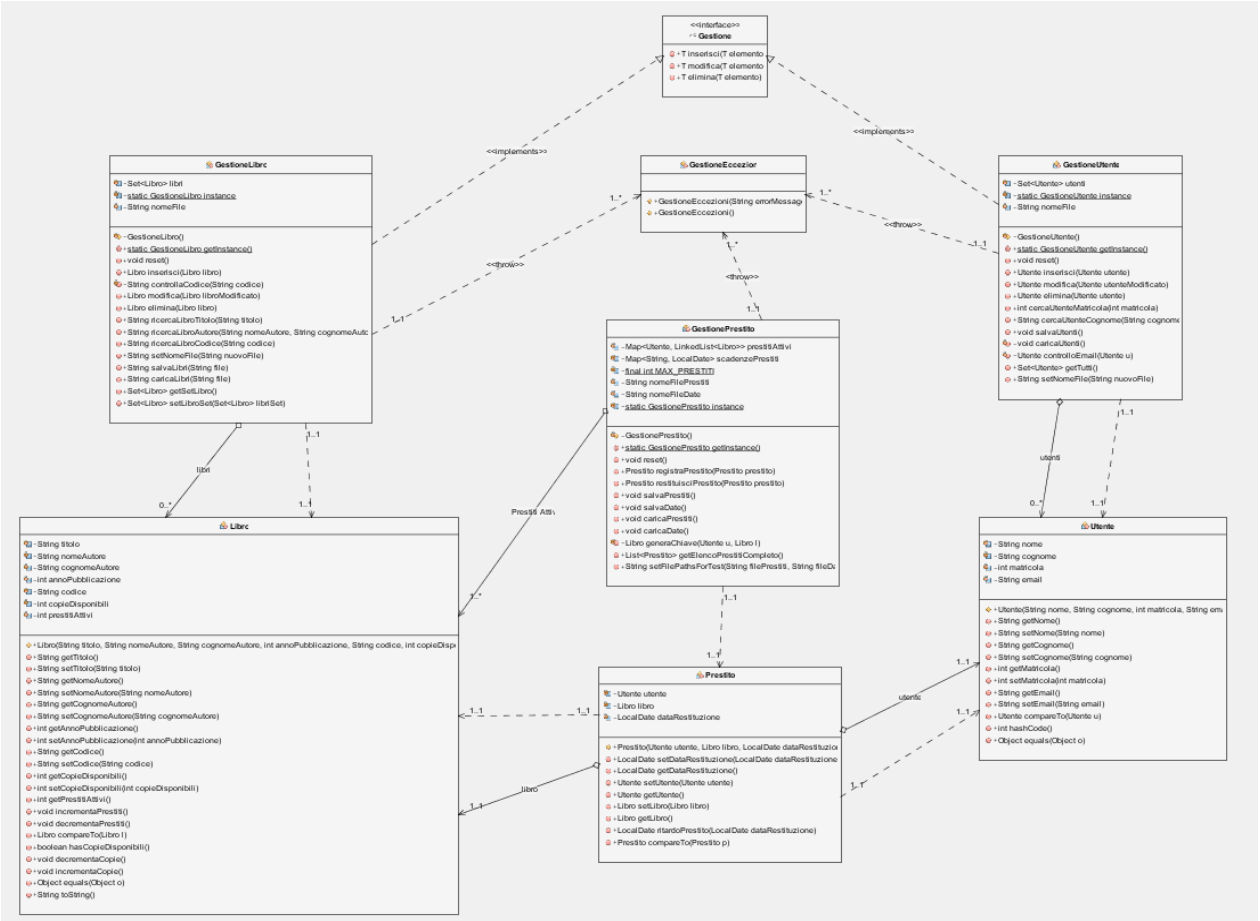


- **View -> Controller:** la View inoltra eventi al Controller ("directs").
- **Controller -> View:** il Controller aggiorna la View ("updates").
- **Controller -> Model:** il Controller manipola i dati del Model ("manages").
- **Model -> View:** JavaFX aggiorna la View quando le proprietà del Model cambiano ("notifies").

2. Class Diagrams

In questa fase, vengono presentati i diagrammi di classe, strumenti grafici che forniscono una panoramica di un sistema mostrando le sue classi, i loro attributi e operazioni e le relazioni tra di esse.

2.1 Class Diagram - Package Model



*Nota: le classi Entità (Libro, Utente e Prestito) implementano le interfacce {Serializable, Comparable} !

La progettazione del package Model ha cercato di ottenere il livello più alto possibile di coesione e il più basso possibile di accoppiamento, in oltre sono stati applicati alcuni dei principali principi di buona progettazione dell'Ingegneria del Software:

- **COESIONE**

Si è cercato di puntare al massimo livello di coesione (**Coesione Funzionale**), per realizzarlo abbiamo deciso di separare le classi Entità (Libro, Utente, Prestito) dalle classi Gestione (GestioneLibro, GestioneUtente, GestionePrestito).

- Le classi entità hanno lo scopo di mantenere i dati e fornire i metodi pubblici (getter e setter) per permettere un accesso controllato, garantendo un'alta coesione sui dati (**Coesione Comunicazionale**);
- Le classi Gestione si occupano principalmente della logica di elaborazione dei dati e della manipolazione delle relative collezioni, le classi GestioneLibro e GestioneUtente lavorano solo

con logiche relative ai propri dati/classi entità, questo rispetta il principio **Single Responsibility Principle** (SRP) secondo il quale un componente deve svolgere un singolo compito ben definito.

- **ACCOPPIAMENTO**

Si è cercato (per quanto possibile) di puntare a un livello accettabile di accoppiamento, l'**Accoppiamento per Dati**, questo è stato possibile garantendo che le interazioni tra le diverse classi avvengano tramite il passaggio di oggetti o dati primitivi necessari, evitando di passare l'intera struttura delle classi quando non necessaria.

- Le classi di gestione non accedono direttamente ai campi privati degli oggetti entità ma utilizzano l'interfaccia pubblica messa a disposizione tramite i metodi getter e setter, questo permette di rispettare il principio dell'incapsulamento e di **evitare** un alto livello di accoppiamento (**Accoppiamento per contenuti**);
- L'uso dell'interfaccia generica Gestione aiuta a mantenere un basso livello di accoppiamento.

- **PRINCIPI DI BUONA PROGETTAZIONE**

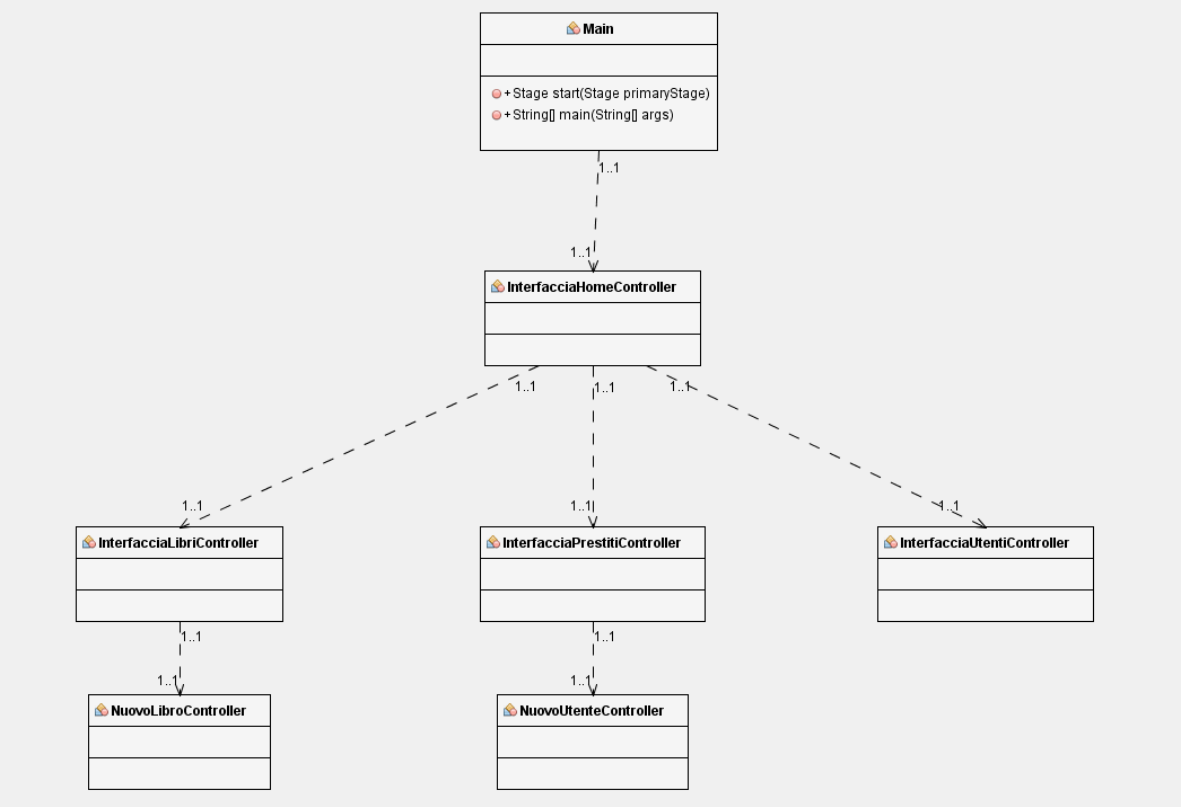
Nella progettazione delle classi sono stati applicati alcuni principi di buona progettazione:

1. **Separazione delle Preoccupazioni** (Separation of Concerns):
   - L'utilizzo del pattern MVC garantisce una separazione tra la logica della gestione dei dati (Model), la logica dell'interfaccia grafica (view) e quella del controllo (Controller);
   - La gestione delle eccezioni è concentrata esclusivamente nella classe GestioneEccezioni, separando la logica per la gestione dei casi di errore da quella del flusso principale;
2. **Principio di segregazione dell'Interfaccia** (ISP):
   - A differenza di GestioneLibro e GestioneUtente che implementano l'interfaccia Gestione, la classe GestionePrestito non la implementa, è una scelta progettuale dovuta al fatto che la gestione di un prestito non prevede l'operazione di modifica (presente nell'interfaccia), di conseguenza invece di implementare un metodo modifica() vuoto o inutile si è preferito non implementare l'interfaccia per questa classe in modo da rispettare il principio ISP ("un client non dovrebbe dipendere da metodi che non utilizza").
3. **Don't Repeat Yourself** (DRY):
   - La scelta di implementare l'interfaccia generica Gestione<T> permette di evitare riscritture di codice molto simili fra loro per le classi GestioneUtente e GestioneLibro (che implementano i metodi inserisci, elimina, modifica).
4. **Privilegiare l'associazione rispetto all'ereditarietà**:
   - Invece di utilizzare la relazione di ereditarietà tra le classi Entità e Gestione si è preferito utilizzare relazioni di aggregazione (has-a), aiuta a ridurre al minimo l'accoppiamento.
5. **Principio di Robustezza**:
   - Grazie all'aggiunta di una classe apposita per la gestione delle eccezioni (GestioneEccezioni) e le relazioni <<throws>> il sistema garantisce la gestione di errori (tramite messaggio a schermo) che porterebbero il sistema in uno stato inconsistente.
6. **Keep It Simple, Stupid** (KISS):
   - La struttura delle classi entità (in particolare Libro e Utente) è molto semplice dato che presenta principalmente attributi privati e i relativi getter e setter.
   - Le classi gestione hanno metodi semplici utili alla manipolazione della collezione su cui lavorano.
7. **Principio Aperto/Chiuso**:
   - In generale la maggior parte degli attributi delle classi entità hanno visibilità privata e sono messe a disposizioni interfacce pubbliche per accedere e modificare questi

attributi, in caso di modifiche / aggiunte future a queste classi non sarà necessario modificare anche il codice dei client (es: i controller) che utilizzano queste classi.

8. **You Aren't Going to Need It** (YAGNI):
    - La progettazione include soltanto le funzioni indispensabili per rispettare i requisiti attuali, non sono presenti metodi per funzionalità future o non necessarie.
9. **Principio della minima sorpresa**:
    - Abbiamo utilizzato sempre la stessa convenzione per la denominazione delle classi (es: per le classi gestione seguono il seguente pattern "Gestione[Entità]"), non sono presenti strutture o nomi contorti che potrebbero confondere altri sviluppatori.

2.2 Class Diagram - Package Controller



Analogamente al package Model, la progettazione del package Controller ha cercato di ottenere un alto livello di coesione (quanto le varie parti all'interno delle classi sono legate tra di loro) e un basso livello di accoppiamento (quanto le varie classi del package dipendono tra di loro).

- **COESIONE**

Dal punto di vista della coesione , la strutturazione delle classi InterfacciaLibriController, InterfacciaUtentiController e InterfacciaPrestitController aderisce ad una **coesione comunicazionale**, cioè ogni classe include funzionalità che lavorano sugli stessi dati: la classe InterfacciaLibriController, che gestisce le interazioni che avvengono tra il Bibliotecario e l'interfaccia dedicata alla gestione dei libri, manipola esclusivamente informazioni e dati relativi a libri; la classe InterfacciaUtentiController, che gestisce le interazioni che avvengono tra il bibliotecario e l'interfaccia dedicata alla gestione degli utenti del sistema, manipola esclusivamente informazioni e dati relativi a utenti; la classe InterfacciaPrestitiController, che gestisce le interazioni che avvengono tra il Bibliotecario e l'interfaccia dedicata alla gestione dei prestiti, manipola informazioni e dati relativi a prestiti; sebbene la logica dei prestiti richieda comunque l'utilizzo di informazioni relative a libri e utenti, il focus della classe è strettamente limitato alla gestione dell'associazione tra le entità Prestito.

- **ACCOPPIAMENTO**

In termini di accoppiamento, le varie relazioni di dipendenza tra le classi fanno sì che si abbia un **accoppiamento per controllo**: la classe InterfacciaHomeController, che rappresenta la schermata principale del sistema da cui si può accedere alle sezioni dedicate alla gestione dei vari dati , deve necessariamente utilizzare oggetti di tipo InterfacciaLibriController, InterfacciaUtentiController e InterfacciaPrestitiController per permettere l'interazione tra il bibliotecario e le diverse sezioni dell'interfaccia e gestire il flusso di navigazione dell'applicazione.

- **PRINCIPI DI BUONA PROGETTAZIONE**

Nella progettazione delle classi sono stati applicati alcuni principi di buona progettazione:

1. **Keep it Simple, Stupid** (KISS):
   - Sono state realizzate classe semplici, con poche operazioni, metodi brevi ; inoltre la struttura delle classi è chiara, facile da comprendere;

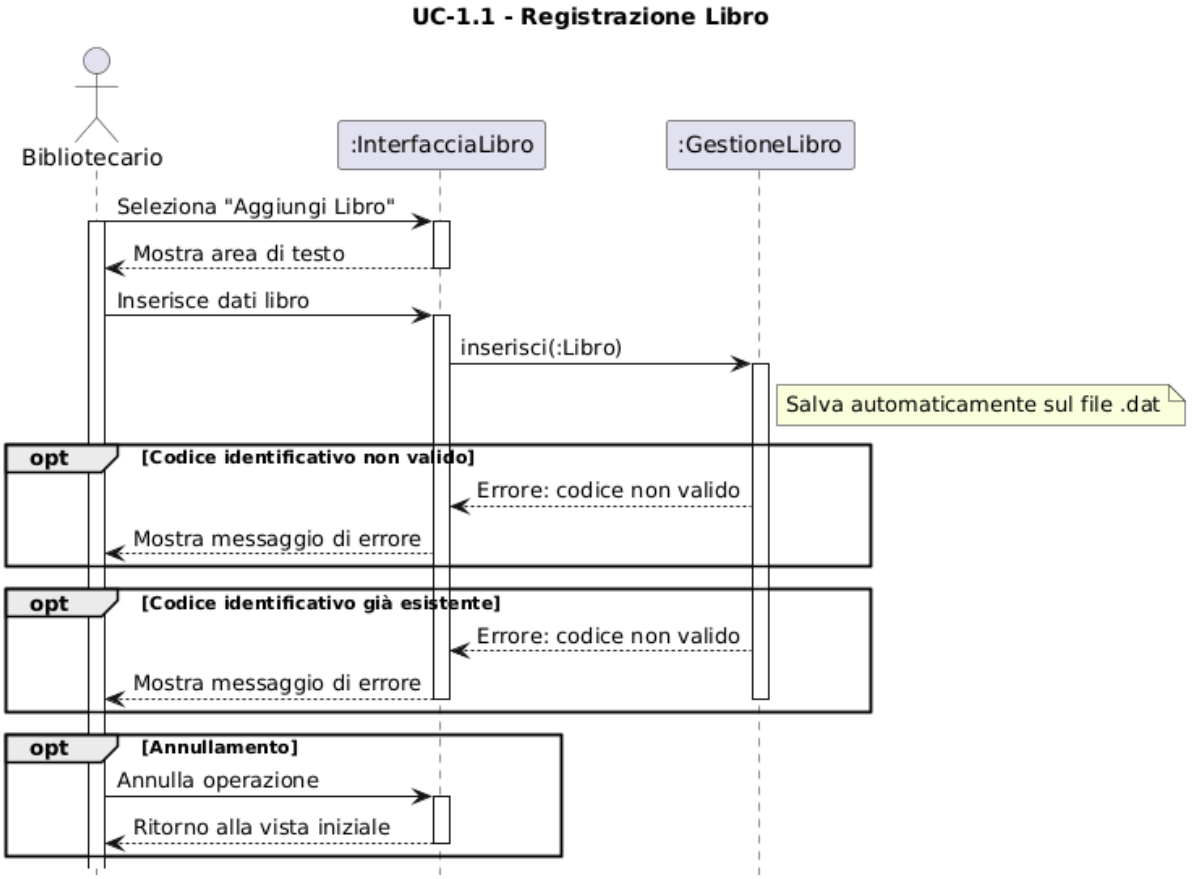
2. **Principio della singola responsabilità** (Single Responsibility Principle):
   - Ogni classe svolge un compito ben definito: la classe InterfacciaLibriController gestisce le interazioni tra il Bibliotecario e l'interfaccia di gestione dei libri, la classe InterfacciaUtentiController gestisce le interazioni tra il Bibliotecario e l'interfaccia di gestione degli utenti, la classe InterfacciaPrestitiController gestisce le interazioni tra il Bibliotecario e l'interfaccia di gestione dei prestiti, la classe InterfacciaHomeController gestisce le interazioni tra il Bibliotecario e la schermata principale dell'interfaccia da cui accedere ad altre sezioni; in questo modo, eventuali modifiche saranno necessarie solo in una classe.
3. **Principio di separazione delle preoccupazioni** (Separation of Concerns):
   - Aspetti diversi del sistema sono gestiti da classi distinte e non sovrapposte: ad esempio la classe InterfacciaLibriController si occupa delle interazioni tra il Bibliotecario e l'interfaccia dedicata esclusivamente alla gestione dei libri. Ciò semplifica in generale lo sviluppo e la manutenzione dell'applicazione software; inoltre, essendo i problemi ben separati le singole classi possono essere riutilizzate oltre che sviluppate e aggiornate in modo indipendente;
4. **Principio di segregazione delle interfacce**:
   - Piuttosto che creare un'unica classe generica Interfaccia che sarebbe stata molto grande, sono state realizzate classi specifiche e piccole che si occupano di gestire particolari aspetti dell'interfaccia del sistema;
5. **Principio di inversione della dipendenza**:
   - La classe Main non dipende direttamente dalle classi InterfacciaLibriController , InterfacciaUtentiController e InterfacciaPrestitiController, bensì dipende dalla classe InterfacciaHomeController; con questa astrazione, sia le classi di livello alto che quelle di livello basso riducono le loro dipendenze.
6. **Principio della minima sorpresa:**
   - Le denominazioni delle classi dedicate all'interfaccia seguono tutte lo stesso pattern "Interfaccia[Entità]Controller" ; in questo modo il codice è strutturato in modo da non sorprendere o confondere chi dovrà leggerlo o occuparsi della manutenzione.

3. Sequence Diagrams

In questa fase, vengono presentati i sequence diagrams, strumenti grafici che illustrano il comportamento di uno specifico scenario in modo dettagliato.

3.1 Use case 1.1

- Registrazione Libro



Questo diagramma di sequenza illustra il comportamento dello scenario “Registrazione Libro”, in cui il bibliotecario registra le informazioni relative ad un nuovo libro. I partecipanti a questo scenario sono l’attore Bibliotecario, un oggetto di tipo InterfacciaLibro e un oggetto di tipo GestioneLibro. La sequenza di interazioni e messaggi scambiati tra le diverse entità è la seguente:

- Bibliotecario -> InterfacciaLibro: l’attore “Bibliotecario” interagisce con l’oggetto InterfacciaLibro inviando a quest’ultimo il messaggio <<Seleziona “Aggiungi Libro”>>: questo messaggio rappresenta la situazione in cui il bibliotecario seleziona la voce “+ Nuovo Libro” presente nell’interfaccia grafica del sistema per aggiungere un nuovo libro. Il libro da registrare deve ovviamente essere disponibile. In risposta, l’interfaccia mostra al bibliotecario un’area con i vari campi di testo da compilare.

- Bibliotecario -> InterfacciaLibro: l’attore “Bibliotecario” interagisce con l’oggetto InterfacciaLibro inviando a quest’ultimo il messaggio <<Inserisci dati libro>>: questo messaggio rappresenta la situazione in cui il bibliotecario riempie i vari campi di testo con le informazioni del libro che vuole registrare, ossia titolo, nome e cognome degli autori, anno di pubblicazione, codice identificativo e numero di copie.

- InterfacciaLibro -> GestioneLibro: l'oggetto InterfacciaLibro interagisce con l'oggetto GestioneLibro inviando a quest'ultimo il messaggio <<inserisci(:Libro) >> che corrisponde al metodo omonimo presente nella classe GestioneLibro che permette di aggiungere un nuovo libro alla collezione di libri e salvarlo automaticamente su un file binario .dat

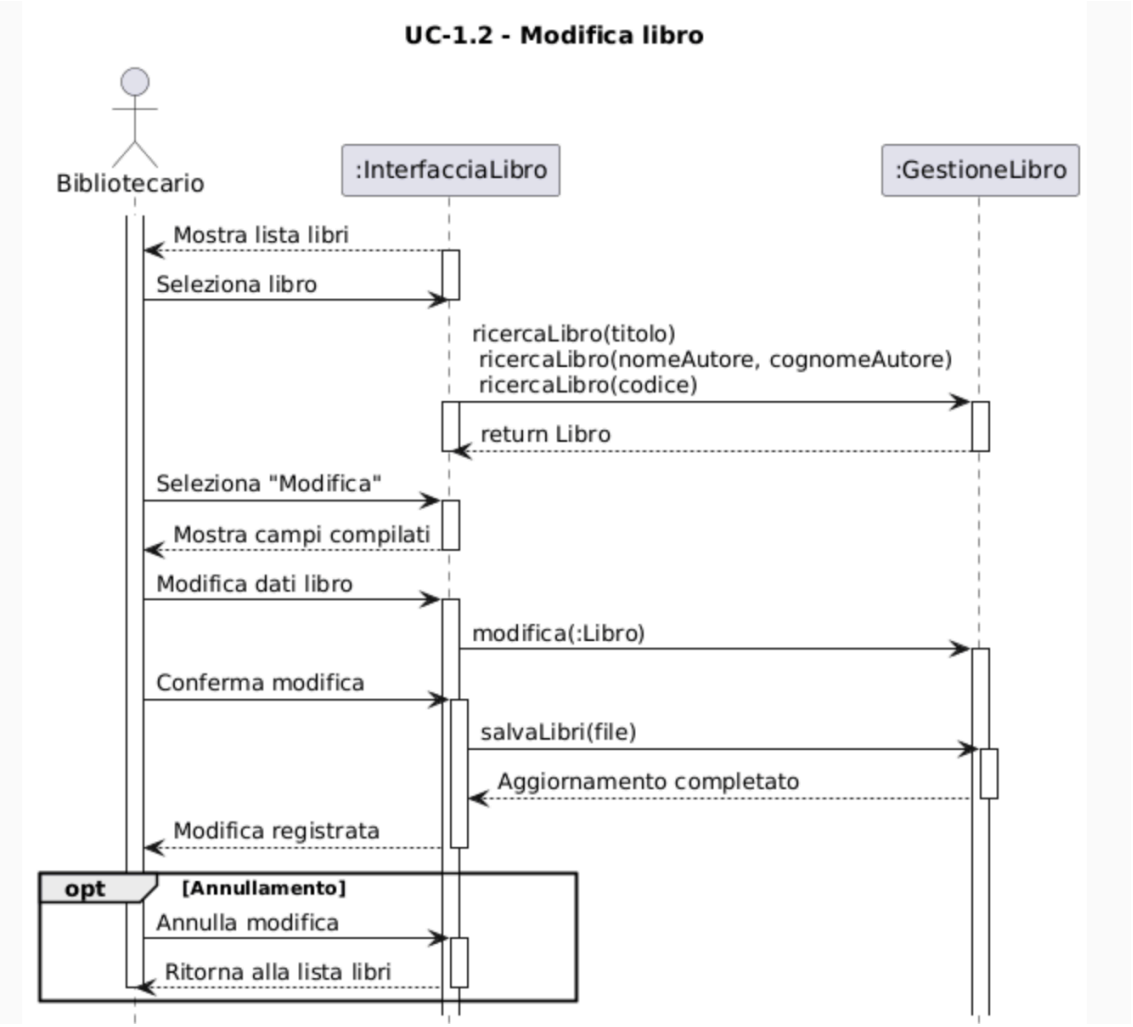
- **Frame di interazione opt:**
  - [Codice identificativo non valido] – Se il codice identificativo del libro che si sta inserendo non è valido, l'InterfacciaLibro mostra al Bibliotecario un messaggio di errore;

- **Frame di interazione opt:**
  - [Codice identificativo già esistente] – Se il codice identificativo del libro che si sta inserendo è già esistente , l'InterfacciaLibro mostra al Bibliotecario un messaggio di errore;

- **Frame di interazione opt:**
  - [Annullamento] : se l'operazione di inserimento di un nuovo libro viene annullata, l'interfaccia mostra nuovamente al Bibliotecario l'interfaccia dedicata alla gestione dei libri

3.2 Use case 1.2

- Modifica Libro



Questo diagramma di sequenza illustra il comportamento dello scenario “Modifica Libro”, in cui il bibliotecario aggiorna le informazioni relative ad un libro esistente. I partecipanti a questo scenario sono l’attore Bibliotecario, un oggetto di tipo InterfacciaLibro e un oggetto di tipo GestioneLibro. La sequenza di interazioni e messaggi scambiati tra le diverse entità è la seguente:

- InterfacciaLibro -> Bibliotecario: l’InterfacciaLibro mostra al bibliotecario la lista dei libri.
- Bibliotecario -> InterfacciaLibro: l’attore “Bibliotecario” interagisce con l’oggetto InterfacciaLibro inviando a quest’ultimo il messaggio <<Seleziona libro>>: esso rappresenta la situazione in cui il bibliotecario seleziona dalla lista mostrata dall’interfaccia il libro da modificare.
- InterfacciaLibro -> GestioneLibro: l’oggetto InterfacciaLibro interagisce con l’oggetto GestioneLibro inviando messaggi di ricerca (per ISBN, autore o titolo) per individuare il libro desiderato nel sistema.
- GestioneLibro -> InterfacciaLibro: in risposta l’oggetto GestioneLibro restituisce il libro cercato.

- Bibliotecario -> InterfacciaLibro: il bibliotecario interagisce con l'oggetto InterfacciaLibro inviando il messaggio <<Seleziona "Modifica">>: esso rappresenta la situazione in cui il bibliotecario seleziona il pulsante di modifica mostrato dall'interfaccia.

- InterfacciaLibro -> Bibliotecario: in risposta, l'InterfacciaLibro mostra al bibliotecario i campi di inserimento.

- Bibliotecario -> InterfacciaLibro: il bibliotecario invia il messaggio <<Modifica dati libro>>: esso rappresenta la situazione in cui il bibliotecario modifica mediante l'interfaccia i dati del libro selezionato.

- InterfacciaLibro -> GestioneLibro: l'oggetto InterfacciaLibro interagisce con l'oggetto GestioneLibro inviando ad esso i dati modificati mediante il messaggio <<modifica(:Libro))>>, che corrisponde al metodo omonimo presente nella classe GestioneLibro.

- Bibliotecario -> InterfacciaLibro: l'attore (bibliotecario) interagisce con l'oggetto InterfacciaLibro inviando il messaggio <<Conferma modifica>> attraverso cui il bibliotecario conferma mediante l'interfaccia l'operazione di modifica.

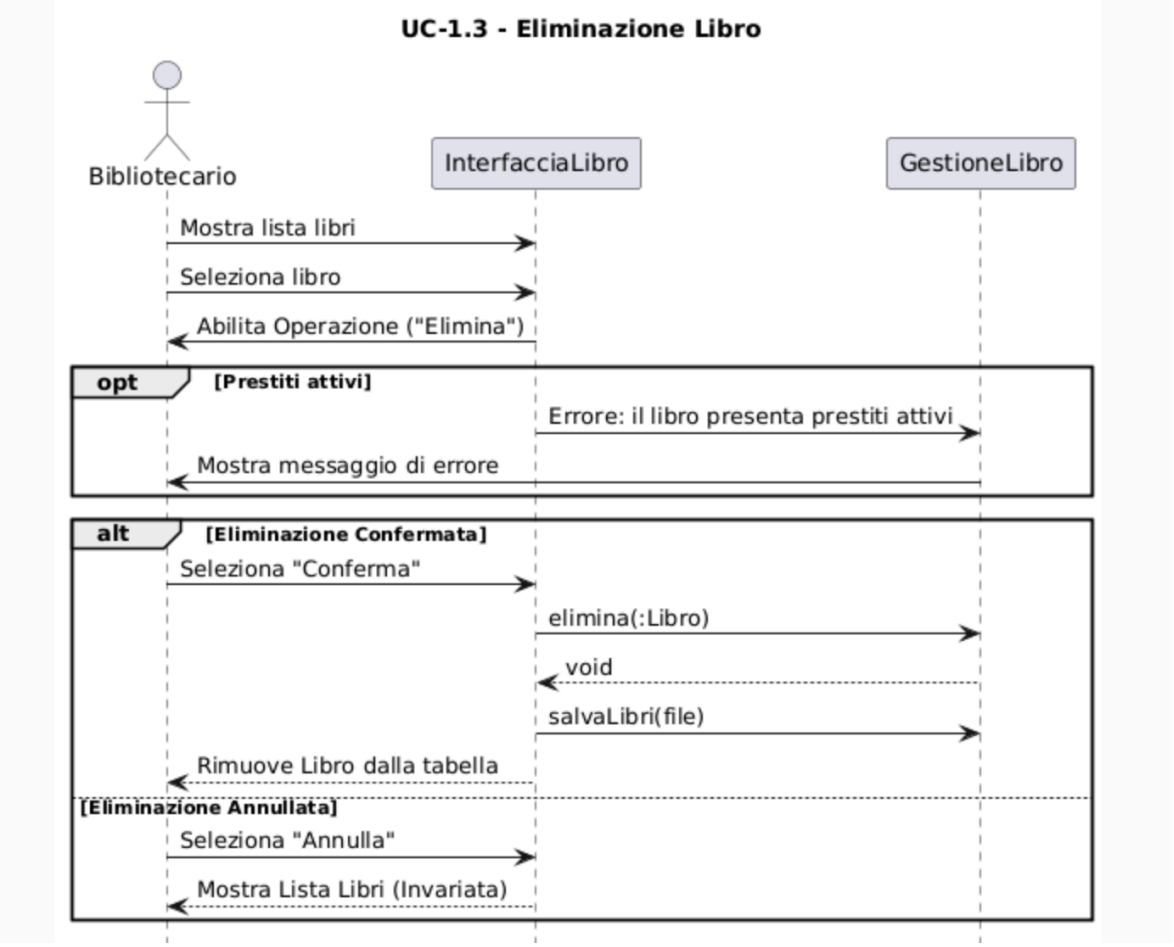
- InterfacciaLibro -> GestioneLibro: l'oggetto InterfacciaLibro delega l'azione di salvataggio dei dati modificati all'oggetto GestioneLibro tramite il messaggio <<salvaLibroi(file)>>, che corrisponde al metodo omonimo presente nella classe GestioneLibro, il quale permette di salvare su un file le informazioni relative all'utente.

- InterfacciaLibro -> Bibliotecario: in risposta l'oggetto InterfacciaLibro notifica al Bibliotecario che i dati relativi al libro sono stati modificati.

- **Frame di interazione opt:**
  - [Annullamento] : se l'operazione di modifica dei dati di un utente viene annullata, l'InterfacciaUtente

3.3 Use case 1.3

- Eliminazione Libro



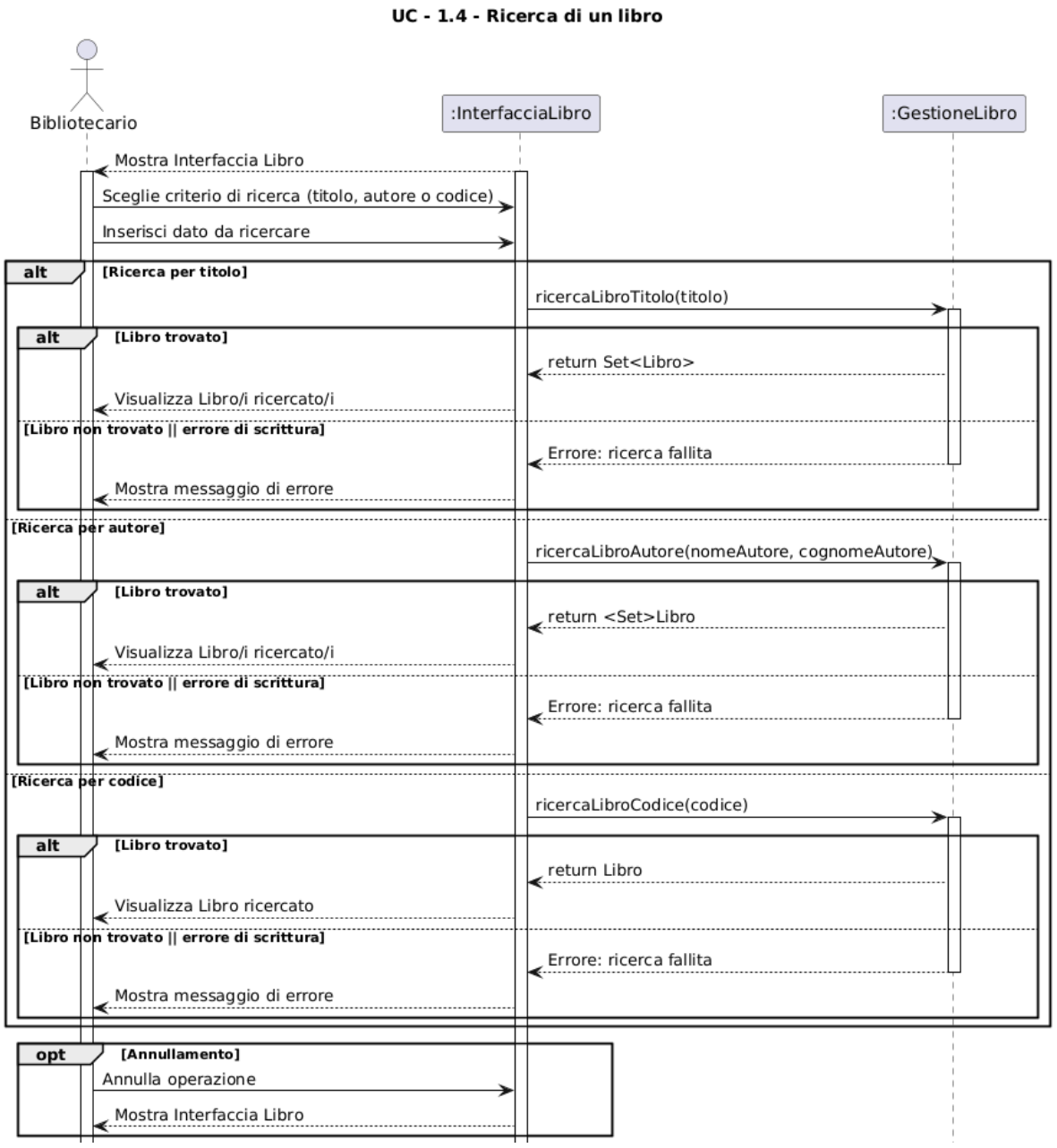
Questo diagramma di sequenza illustra il comportamento dello scenario “Eliminazione Libro”, in cui il bibliotecario interagisce con l’interfaccia relativa alla gestione dei libri al fine di rimuovere un libro dalla tabella.
I partecipanti di questo scenario sono l’attore Bibliotecario, un oggetto del tipo InterfacciaLibro (Controller) che interagisce con oggetto del tipo GestioneLibro (Model).
Precondizione: Il bibliotecario si trova nell'interfaccia per la gestione dei libri.
I messaggi scambiati tra le diverse unità sono i seguenti:

- InterfacciaLibro –> Bibliotecario: Mostra listra libri - mostra una TableView con le informazioni riguardanti i libri che sono stati aggiunti;
- Bibliotecario -> InterfacciaLibro: - Seleziona Libro - il bibliotecario seleziona a schermo il libro da eliminare;
- InterfacciaLibro -> Bibliotecario: - Abilita l’operazione - l’interfaccia permette di utilizzare (non lancia errore) il tasto “Elimina”;
- **Frame di interazione opt:**
  - [Prestiti attivi] – Se il libro risulta in prestito, GestioneLibro restituisce un errore all'interfaccia e l'operazione viene interrotta per preservare l'integrità dei dati.

- **Frame d'Interazione ALT [Eliminazione Confermata]:**
    - Bibliotecario -> InterfacciaLibro: - Seleziona Conferma - il bibliotecario seleziona il tasto “Elimina” per confermare l’eliminazione del libro selezionato;
    - InterfacciaLibro -> GestioneLibro: - elimina(:Libro) - l’oggetto InterfacciaLibro invoca il metodo “elimina(:Libro)” di GestioneLibro per rimuove il libro selezionato dalla collezione e salvare la collezione aggiornata su file;
    - InterfacciaLibro -> GestioneLibro: l’oggetto InterfacciaLibro invoca il metodo “salvaLibri(file)” di GestioneLibro per salvare i dati aggiornati sul file.
    - InterfacciaLibro -> Bibliotecario: - Rimuove libro dalla tabella - l’InterfacciaLibro mostra la TableView aggiornata senza il libro rimosso;
- **Frame d'Interazione ALT [Eliminazione Annullata]**:
    - Bibliotecario -> InterfacciaLibro: - Seleziona “Annulla” - il bibliotecario annulla l’operazione di eliminazione (non clicca sul pulsante “Elimina”);
    - InterfacciaLibro –> Bibliotecario: - Mostra Lista libri - mostra la TableView con le informazioni riguardanti i libri che sono stati aggiunti;

3.4 Use case 1.4

- Ricerca Libro



Questo diagramma di sequenza illustra il comportamento dello scenario “Ricerca di un libro ”, in cui il bibliotecario ricerca un libro all’interno della lista ordinata per titolo (alfabeticamente).

I partecipanti a questo scenario sono l’attore Bibliotecario, un oggetto di tipo InterfacciaLibro e un oggetto di tipo GestioneLibro.

La sequenza di interazioni e messaggi scambiati tra le diverse entità è la seguente:

- InterfacciaLibro -> Bibliotecario: Mostra Interfaccia Libro - L’oggetto InterfacciaLibro interagisce con l’attore Bibliotecario inviando a quest’ultimo il messaggio <<Mostra Interfaccia Libro>>: questo messaggio rappresenta la situazione in cui l’interfaccia presenta al Bibliotecario la sezione dedicata alla gestione dei libri.

- Bibliotecario -> InterfacciaLibro: Sceglie criterio di ricerca (titolo, autore, codice) - L'attore "Bibliotecario" interagisce con l'oggetto InterfacciaLibro inviando a quest'ultimo il messaggio <<Sceglie criterio di ricerca (titolo, autore o codice)>>: questo messaggio rappresenta la situazione in cui il bibliotecario sceglie mediante l'interfaccia il criterio di ricerca da utilizzare (per titolo, per autore o per codice identificativo).

- Bibliotecario -> InterfacciaLibro: Inserisci dato da ricercare - L'attore "Bibliotecario" interagisce con l'oggetto InterfacciaLibro inviando a quest'ultimo il messaggio <<Inserisce dato da ricercare>>: questo messaggio rappresenta la situazione in cui il bibliotecario, a seconda del criterio di ricerca utilizzato, inserisce mediante l'interfaccia il dato appropriato da ricercare, ossia un titolo, un autore o un codice identificativo.

- Frame di interazione alt:

[**Ricerca per titolo**] - Se il criterio di ricerca utilizzato è per titolo, allora:
  - InterfacciaLibro -> GestioneLibro: ricercaLibro(titolo) **-** L'oggetto InterfacciaLibro interagisce con l'oggetto GestioneLibro inviando a quest'ultimo il messaggio <<ricercaLibro(titolo) >> che corrisponde al metodo omonimo presente nella classe GestioneLibro; questo metodo permette di ricercare uno o più libri libri in base al titolo (possono esserci più libri con lo stesso titolo).
- Frame di interazione alt annidato:
  - [Libro trovato] – Se il libro è stato trovato, l'interfaccia riceve mediante GestioneLibro la collezione di libri in questione e la visualizza al Bibliotecario;
  - [Libro non trovato] – Se il libro non è stato trovato perché il titolo inserito non corrisponde a nessun libro attualmente registrato oppure perché si è verificato un errore di scrittura, la ricerca fallisce e l'interfaccia mostra al Bibliotecario un messaggio di errore.

[**Ricerca per autore**] - Se il criterio di ricerca utilizzato è per autore, allora:
  - InterfacciaLibro -> GestioneLibro : ricercaLibro(nomeAutore, cognomeAutore) - L'oggetto InterfacciaLibro interagisce con l'oggetto GestioneLibro inviando a quest'ultimo il messaggio <<ricercaLibro(nomeAutore, cognomeAutore) >> che corrisponde al metodo omonimo presente nella classe GestioneLibro; questo metodo permette di ricercare uno o più libri in base al nome e al cognome dell'autore del libro (possono esserci più libri con lo stesso autore).

- **Frame di interazione alt annidato:**
  - [Libro trovato] – Se il libro è stato trovato, l'interfaccia riceve mediante GestioneLibro la collezione (set) di libri in questione e la visualizza al Bibliotecario;
  - [Libro non trovato] – Se il libro non è stato trovato perché il nome e il cognome inseriti non corrispondono a nessun nome e cognome dell'autore di uno dei libri attualmente registrati oppure perché si è verificato un errore di scrittura, la ricerca fallisce e l'interfaccia mostra al Bibliotecario un messaggio di errore.

[Ricerca per codice] - Se il criterio di ricerca utilizzato è per codice identificativo, allora:
  - InterfacciaLibro -> GestioneLibro: ricercaLibro(codice) **-** L'oggetto InterfacciaLibro interagisce con l'oggetto GestioneLibro inviando a quest'ultimo il messaggio <<ricercaLibro(codice) >> che corrisponde al metodo omonimo presente nella classe

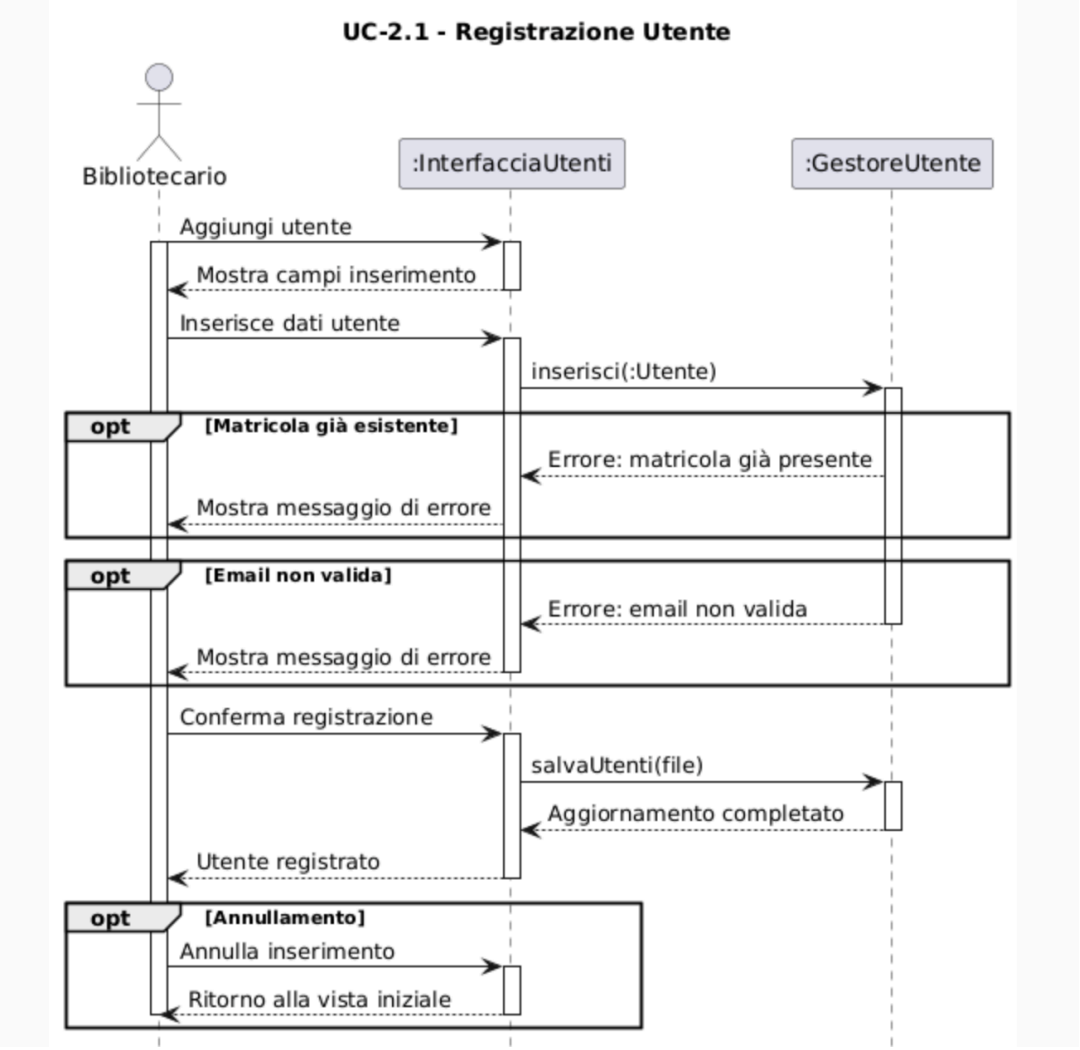
GestioneLibro; questo metodo permette di ricercare un libro in base al codice identificativo.

- **Frame di interazione alt annidato:**
  - [Libro trovato] – Se il libro è stato trovato, l'interfaccia riceve mediante GestioneLibro il libro in questione e lo visualizza al Bibliotecario;
  - [Libro non trovato] – Se il libro non è stato trovato perché il codice inserito non corrisponde a nessun libro attualmente registrato oppure perché si è verificato un errore di scrittura, la ricerca fallisce e l'interfaccia mostra al Bibliotecario un messaggio di errore

- **Frame di interazione opt:**
  - [Annullamento]: L'operazione di ricerca di un nuovo libro può essere annullata mentre si digita il dato da ricercare

3.5 Use case 2.1

- Registrazione Utente



Il diagramma descrive le interazioni necessarie per la registrazione di un nuovo utente, coinvolgendo diversi partecipanti: l'attore Bibliotecario, un oggetto di tipo InterfacciaUtenti e un oggetto di tipo GestioneUtente.
Precondizione: il bibliotecario si trova nell'interfaccia per la gestione degli utenti.

- Bibliotecario -> InterfacciaUtenti: l'attore (bibliotecario) inizia l'interazione con l'oggetto InterfacciaUtenti inviando il messaggio <<Aggiungi Utente>>: esso rappresenta la situazione in cui il bibliotecario seleziona la voce "Aggiungi utente" presente nell'interfaccia grafica del sistema per aggiungere un utente all'interno della collezione di utenti.

- InterfacciaUtenti -> Bibliotecario: in risposta, l'InterfacciaUtenti mostra al bibliotecario i campi di inserimento.

- Bibliotecario -> InterfacciaUtenti: l'attore (bibliotecario) interagisce con l'oggetto InterfacciaUtenti inviando a quest'ultimo il messaggio <<Inserisce dati utenti>>: esso rappresenta la situazione in cui il bibliotecario inserisce mediante l'interfaccia i dati dell'utente da registrare.

- InterfacciaUtenti -> GestioneUtente: l'oggetto InterfacciaUtenti interagisce con l'oggetto GestioneUtente inviando ad esso i dati mediante il messaggio <<inserisci(:Utente)>>, che corrisponde al metodo omonimo presente nella classe GestioneUtente.

- **Frame di interazione opt:**
  - [Matricola già esistente] : poiché la matricola identifica univocamente un utente, nel caso in cui la matricola inserita è già presente (già associata ad un altro utente), l'InterfacciaUtenti mostra al Bibliotecario un messaggio di errore.

- Bibliotecario -> InterfacciaUtenti: l'attore (bibliotecario) interagisce con l'oggetto InterfacciaUtenti inviando il messaggio <<Conferma registrazione>> attraverso cui il bibliotecario conferma mediante l'interfaccia l'operazione di registrazione.

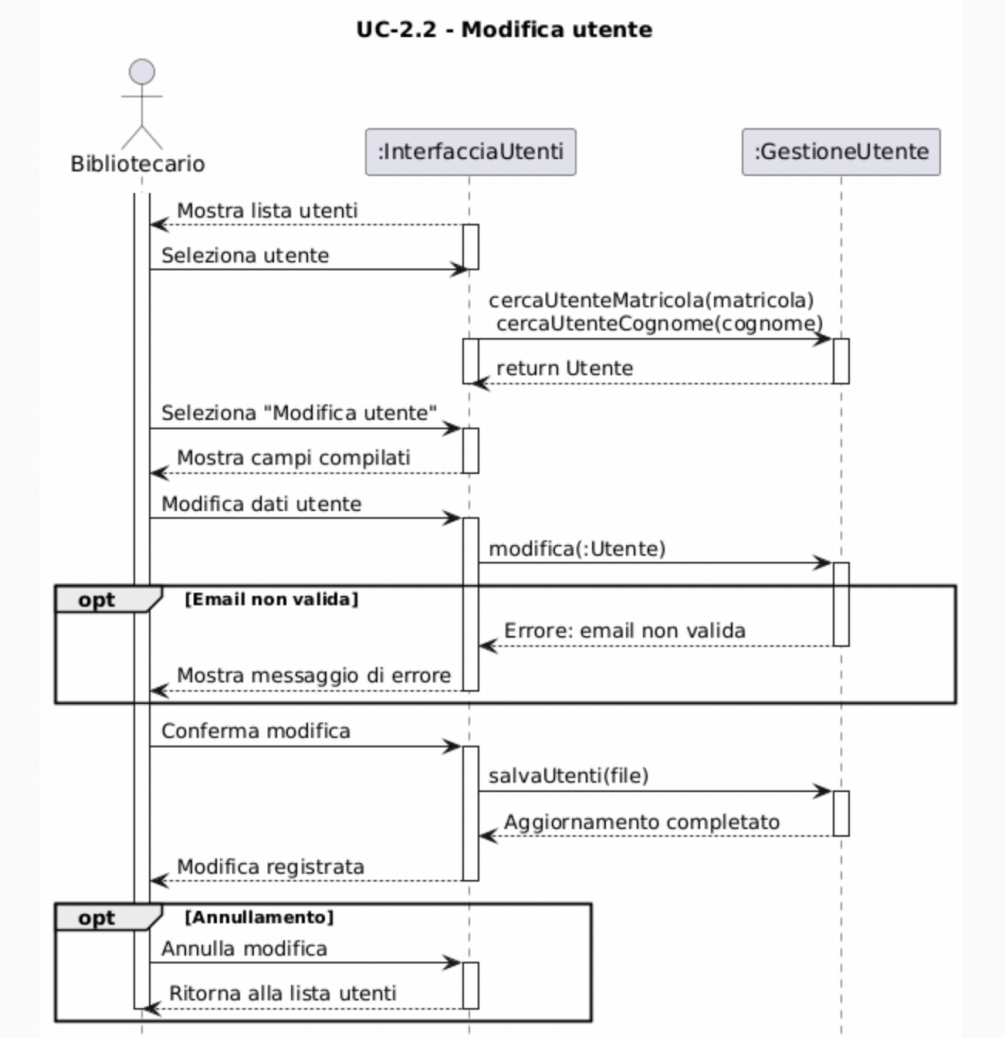
- InterfacciaUtenti -> GestioneUtente: l'oggetto InterfacciaUtenti delega l'azione di salvataggio dei dati al Gestore Utente tramite il messaggio <<salvaUtenti(file)>>, che corrisponde al metodo omonimo presente nella classe GestioneUtente, il quale permette di salvare su un file le informazioni relative all'utente.

- InterfacciaUtenti -> Bibliotecario: in risposta l'oggetto InterfacciaUtenti notifica al Bibliotecario che l'utente è stato registrato.

- **Frame di interazione opt:**
  - [Annullamento] : se l'operazione di registrazione di un nuovo utente viene annullata, l'InterfacciaUtente mostra nuovamente al Bibliotecario la vista iniziale.

3.5 Use case 2.2

- Modifica Utente



Il diagramma descrive le interazioni necessarie per la modifica di un utente, coinvolgendo diversi partecipanti: l'attore Bibliotecario, un oggetto di tipo InterfacciaUtenti e un oggetto di tipo GestioneUtente.
Precondizione: il bibliotecario si trova nell'interfaccia per la gestione degli utenti.

- InterfacciaUtenti -> Bibliotecario: l'InterfacciaUtenti mostra al bibliotecario la lista degli utenti.
- Bibliotecario -> InterfacciaUtenti: l'attore (bibliotecario) inizia l'interazione con l'oggetto InterfacciaUtenti inviando il messaggio <<Seleziona Utente>>: esso rappresenta la situazione in cui il bibliotecario seleziona dalla lista mostrata dall'interfaccia l'utente da modificare.
- InterfacciaUtenti -> GestioneUtente: l'oggetto InterfacciaUtenti interagisce con l'oggetto GestioneUtente inviando il messaggio <<cercaUtenteMatricola(matricola) \n cercaUtenteCognome(cognome)>> , che corrisponde ai metodi omonimi presenti nella classe GestioneUtente, e sono necessari per effettuare una ricerca dell'utente selezionato.
- GestioneUtente -> InterfacciaUtenti: in risposta l'oggetto GestioneUtente restituisce l'utente cercato.

- Bibliotecario -> InterfacciaUtenti: il bibliotecario interagisce con l'oggetto InterfacciaUtenti inviando il messaggio <<Seleziona Modifica Utente>>: esso rappresenta la situazione in cui il bibliotecario seleziona il pulsante di modifica utente mostrato dall'interfaccia.

- InterfacciaUtenti -> Bibliotecario: in risposta, l'InterfacciaUtenti mostra al bibliotecario i campi di inserimento.

- Bibliotecario -> InterfacciaUtenti: il bibliotecario invia all'oggetto InterfacciaUtenti il messaggio <<Modifica dati utenti>>: esso rappresenta la situazione in cui il bibliotecario modifica mediante l'interfaccia i dati dell'utente selezionato.

- InterfacciaUtenti -> GestioneUtente: l'oggetto InterfacciaUtenti interagisce con l'oggetto GestioneUtente inviando ad esso i dati modificati mediante il messaggio <<modifica(:Utente)>>, che corrisponde al metodo omonimo presente nella classe GestioneUtente.

- **Frame di interazione opt:**
  - [Email non valida] : nel caso in cui l'email inserita non rispetta il formato stabilito nome.cognome@universita.it, l'InterfacciaUtenti mostra al Bibliotecario un messaggio di errore.

- Bibliotecario -> InterfacciaUtenti: l'attore (bibliotecario) interagisce con l'oggetto InterfacciaUtenti inviando il messaggio <<Conferma modifica>> attraverso cui il bibliotecario conferma mediante l'interfaccia l'operazione di modifica.

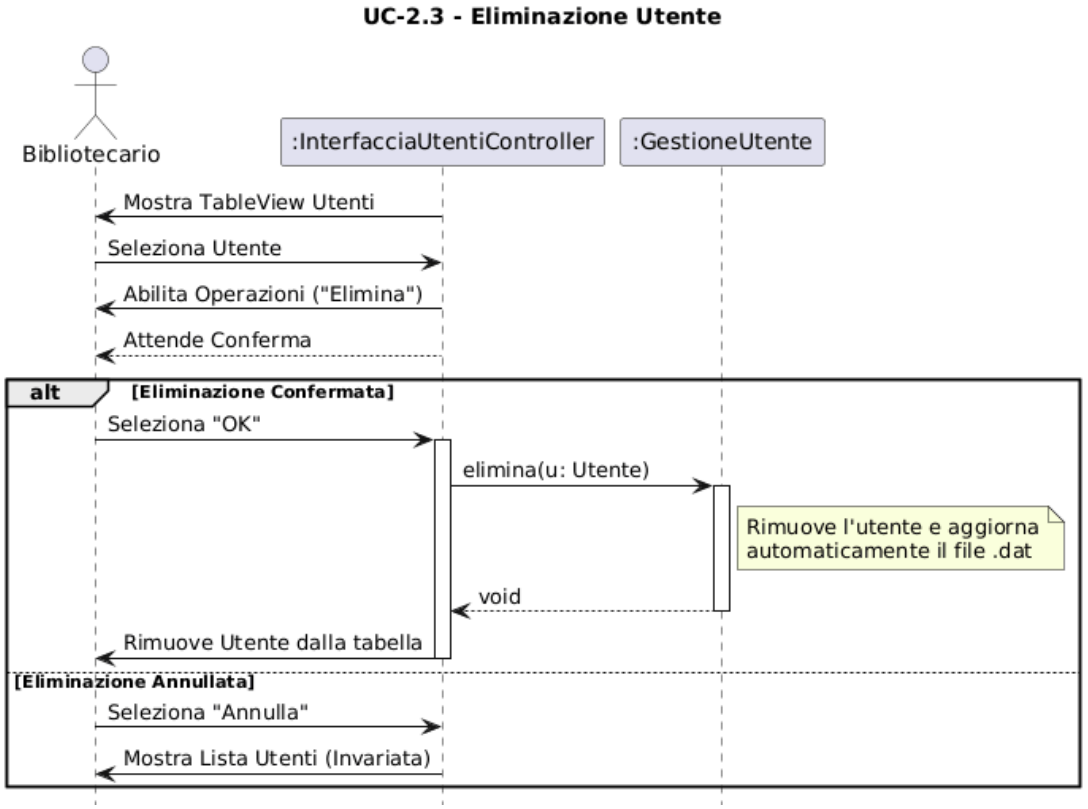
- InterfacciaUtenti -> GestioneUtente: l'oggetto InterfacciaUtenti delega l'azione di salvataggio dei dati modificati all'oggetto GestioneUtente tramite il messaggio <<salvaUtenti(file)>>, che corrisponde al metodo omonimo presente nella classe GestioneUtente, il quale permette di salvare su un file le informazioni relative all'utente.

- InterfacciaUtenti -> Bibliotecario: in risposta l'oggetto InterfacciaUtenti notifica al Bibliotecario che i dati relativi all'utente sono stati modificati.

- **Frame di interazione opt:**
  - [Annullamento] : se l'operazione di modifica dei dati di un utente viene annullata, l'InterfacciaUtente mostra nuovamente al Bibliotecario la vista iniziale dove è presente la lista degli utenti.

3.6 Use case 2.3

- Eliminazione Utente



Questo diagramma di sequenza illustra il comportamento dello scenario “Eliminazione Utente”, in cui il bibliotecario interagisce con l’interfaccia relativa alla gestione degli Utenti al fine di rimuovere un utente dalla tabella.
I partecipanti di questo scenario sono l’attore Bibliotecario, un oggetto del tipo InterfacciaUtenti (Controller) che interagisce con oggetto del tipo GestioneUtenti (Model).
I messaggi scambiati tra le diverse unità sono i seguenti:
Precondizione: Il bibliotecario si trova nell'interfaccia per la gestione degli Utenti.

- InterfacciaUtenti –> Bibliotecario: - Mostra TableView Utenti - mostra una TableView con le informazioni riguardanti gli utenti che sono stati aggiunti;
- Bibliotecario -> InterfacciaUtenti: - Seleziona Utente - il bibliotecario seleziona a schermo l’utente da eliminare;
- InterfacciaUtenti -> Bibliotecario: - Abilita Operazioni - l’interfaccia permette di utilizzare (non lancia errore) il tasto “Elimina Utente”;
- Interfaccia Utenti -> Bibliotecario: - Attende Conferma - il bibliotecario dopo aver selezionato l’utente nella tabella deve confermare l’eliminazione cliccando “Elimina”;
- **Frame d'Interazione ALT [Eliminazione Confermata]:**
  - Bibliotecario -> InterfacciaUtenti: - Seleziona “OK” - il bibliotecario seleziona il tasto “Elimina” per confermare l’eliminazione dell'utente selezionato;

- InterfacciaUtenti -> GestioneUtente: - elimina(:Utente) - l'oggetto InterfacciaUtenti invoca il metodo "elimina(Utente)" di GestioneUtente per rimuove l'utente selezionato dalla collezione e salvare la collezione aggiornata su file;

- InterfacciaUtenti -> Bibliotecario: - Rimuove Utente dalla tabella - l'interfacciaUtenti mostra la TableView aggiornata senza l'utente rimosso;

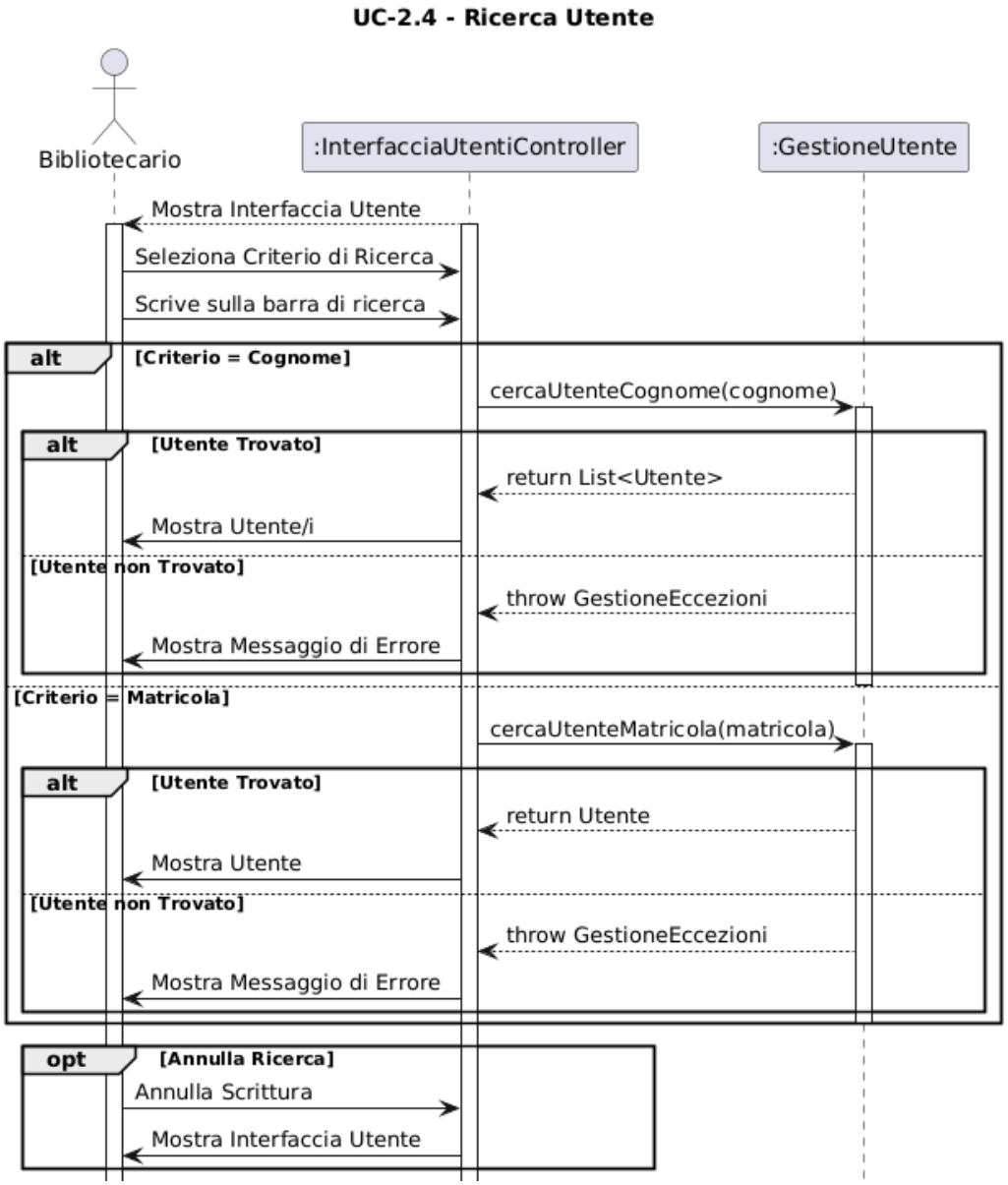
- **Frame d'Interazione ALT [Eliminazione Annullata]**:

  - Bibliotecario -> InterfacciaUtenti: - Seleziona "Annulla" - il bibliotecario annulla l'operazione di eliminazione (non clicca sul pulsante "Elimina");

  - InterfacciaUtenti –> Bibliotecario: - Mostra Lista Utenti - mostra una TableView con le informazioni riguardanti gli utenti che sono stati aggiunti;

3.7 Use case 2.4

- Ricerca Utente



Questo diagramma di sequenza illustra il comportamento dello scenario “Ricerca Utente”, in cui il bibliotecario interagisce con l’interfaccia relativa alla gestione degli Utenti al fine di ricercare un utente dalla tabella.
I partecipanti di questo scenario sono l’attore Bibliotecario, un oggetto del tipo InterfacciaUtenti (Controller) che interagisce con oggetto del tipo GestioneUtenti (Model).
I messaggi scambiati tra le diverse unità sono i seguenti:
Precondizione: Il bibliotecario si trova nell'interfaccia per la gestione degli Utenti.

- InterfacciaUtenti –> Bibliotecario: - Mostra Interfaccia Utenti - mostra l’interfaccia relativa alla Gestione degli Utenti con la TableView e una barra di ricerca con un filtro per il criterio di ricerca;
- Bibliotecario -> InterfacciaUtente: - Seleziona Criterio di Ricerca - il Bibliotecario seleziona dal menù a tendina il criterio di ricerca da utilizzare (Cognome - Matricola);

- Bibliotecario -> InterfacciaUtente: - Scrive sulla barra di ricerca - Il Bibliotecario ricerca l'utente desiderato inserendo nella barra di ricerca la matricola o il cognome;

- **Frame d'Interazione ALT [Criterio = Cognome]**:

  - InterfacciaUtente -> GestioneUtente: - cercaUtenteCognome(cognome) - l'InterfacciaUtente invoca il metodo "cercaUtenteCognome(cognome)" di Gestione Utente per ricercare l'utente nella collezione in base al cognome;

- **Frame d'Interazione ALT annidato [Utente Trovato]**:

  - GestioneUtente -> InterfacciaUtente: - return List<Utente> - la GestioneUtente restrituisce all'interfaccia la lista con gli utenti col cognome ricercato;

  - InterfacciaUtente -> Bibliotecario: - Mostra Utente - l'interfaccia mostra a schermo una TableView con solo gli utenti ricercati;

- **Frame d'Interazione ALT annidato [Utente non Trovato]**:

  - GestioneUtente -> InterfacciaUtente: - Errore: Utente non Trovato - la GestioneUtente lancia un'eccezione con un messaggio di errore che viene riportata all'Interfaccia Utente;

  - InterfacciaUtente -> Bibliotecario: - Mostra Messaggio di Errore - l'InterfacciaUtente mostra a schermo un messaggio di errore con il relativo messaggio;

- **Frame d'Interazione ALT [Criterio = Matricola]**:

  - InterfacciaUtente -> GestioneUtente: - cercaUtenteMatricola(matricola) - l'InterfacciaUtente invoca il metodo "cercaUtenteMatricola(matricola)" di Gestione Utente per ricercare l'utente nella collezione in base alla matricola;

- **Frame d'Interazione ALT annidato [Utente Trovato]**:

  - GestioneUtente -> InterfacciaUtente: - return Utente - la GestioneUtente restrituisce all'interfaccia l'utente ricercato;

  - InterfacciaUtente -> Bibliotecario: - Mostra Utente - l'interfaccia mostra a schermo una TableView con solo l'utente ricercato;

- **Frame d'Interazione ALT annidato [Utente non Trovato]**:

  - GestioneUtente -> InterfacciaUtente: - Errore: Utente non Trovato - la GestioneUtente lancia un'eccezione con un messaggio di errore che viene riportata all'Interfaccia Utente;

  - InterfacciaUtente -> Bibliotecario: - Mostra Messaggio di Errore - l'InterfacciaUtente mostra a schermo un messaggio di errore con il relativo messaggio;

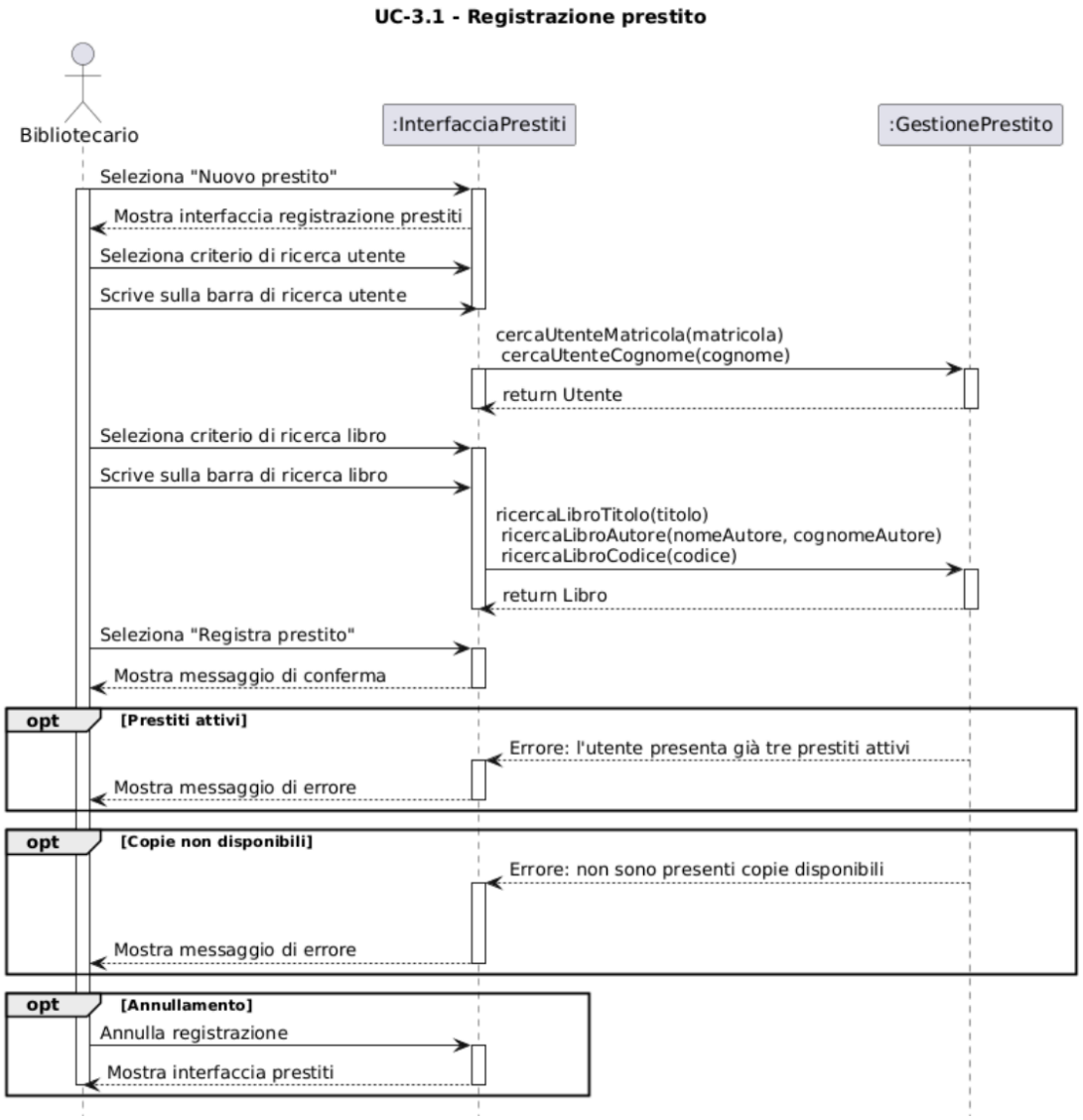
- **Frame d'Interazione OPT [Annulla ricerca]**:

  - Bibliotecario -> InterfacciaUtente: - Annulla Ricerca - il bibliotecario mentre stava scrivendo nella barra di ricerca annulla l'operazione;

  - InterfacciaUtenti –> Bibliotecario: - Mostra Interfaccia Utenti - mostra l'interfaccia relativa alla Gestione degli Utenti con la TableView e una barra di ricerca con un filtro per il criterio di ricerca;

3.8 Use case 3.1

- Registrazione prestito



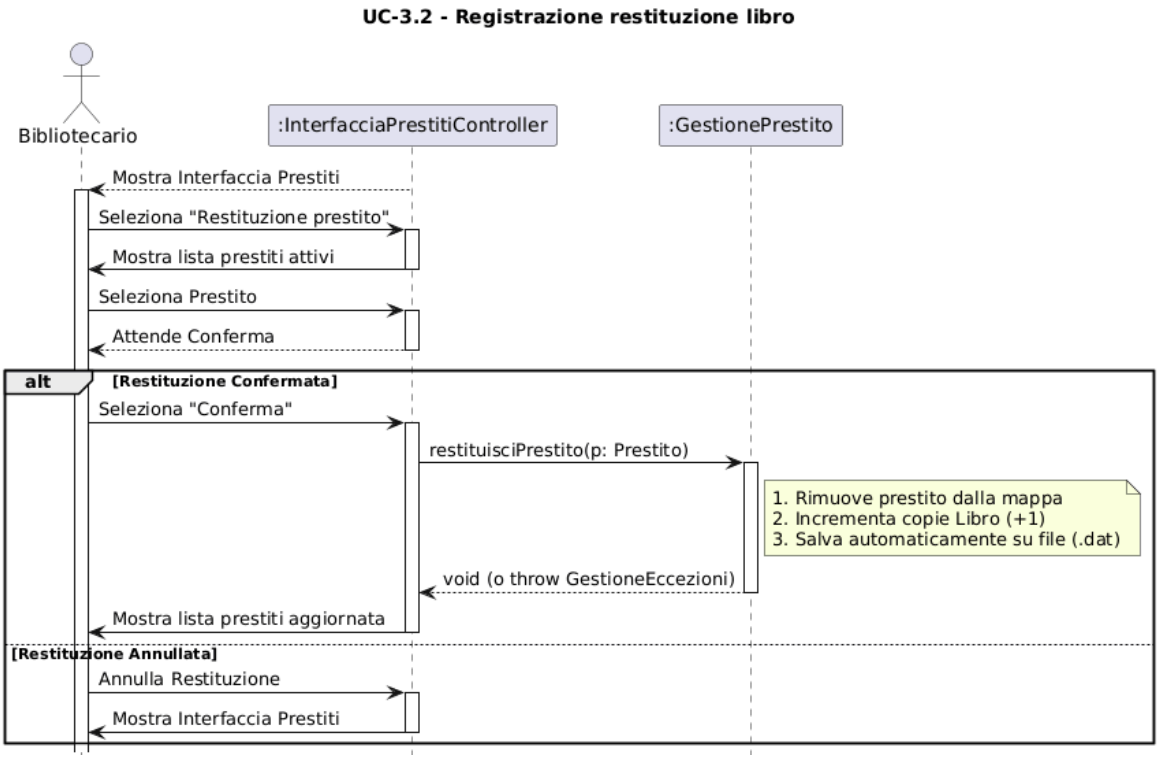
Questo diagramma di sequenza illustra il comportamento dello scenario “Registrazione Prestito”, in cui il bibliotecario assegna un libro ad un utente. I partecipanti a questo scenario sono l’attore Bibliotecario, l'Interfaccia Prestiti, e il controller GestionePrestiti.
La sequenza di interazioni è la seguente:

- Bibliotecario -> Interfaccia Prestito: Seleziona “Nuovo Prestito” - L’attore “Bibliotecario” interagisce con l’oggetto InterfacciaPrestiti per avviare una nuova registrazione di prestito.
- Interfaccia Prestiti -> GestionePrestiti: viene mostrata l’interfaccia di registrazione prestiti.
- Bibliotecario -> InterfacciaPrestiti: - Seleziona Criterio di Ricerca utente - il Bibliotecario seleziona dal menù a tendina il criterio di ricerca da utilizzare (Cognome - Matricola);
- InterfacciaPrestiti -> GestionePrestiti: l’InterfacciaPrestiti invoca il metodo "cercaUtenteMatricola(matricola) / cercaUtenteCognome(cognome)" per ricercare l’utente nella collezione;

- InterfacciaPrestiti -> Bibliotecario: l’interfaccia mostra un messaggio per confermare che l’utente è stato trovato.
- Bibliotecario -> InterfacciaPrestiti: - Seleziona Criterio di Ricerca libro - il Bibliotecario seleziona dal menù a tendina il criterio di ricerca da utilizzare (Titolo - Autore - Codice).
- InterfacciaPrestiti -> GestionePrestiti: l’InterfacciaPrestiti invoca il metodo "ricercaLibroCodice(codice) / ricercaLibroAutore(nomeAutore, cognomeAutore) / ricercaLibroTitolo(titolo)" per ricercare l’utente nella collezione;
- InterfacciaPrestiti -> Bibliotecario: l’interfaccia mostra un messaggio per confermare che il libro è stato trovato.
- Bibliotecario -> InterfacciaPrestiti: il bibliotecario seleziona “Registra prestito” per confermare la registrazione.
- InterfacciaPrestiti -> Bibliotecario: L’interfaccia mostra un messaggio di conferma di avvenuta registrazione con successo.
- **Frame di interazione opt [Prestiti Attivi]**
  - Nel caso in cui l’utente selezionato non presenta già tre prestiti attivi, l’interfaccia mostrerà un messaggio di errore.
- **Frame di interazione opt [Copie non disponibili]**
  - Nel caso in cui l’utente selezionato presenta più copie disponibili, l’interfaccia mostrerà un messaggio di errore.
- **Frame di interazione opt [Annullamento]**
  - Bibliotecario -> InterfacciaPrestiti: - Annulla Ricerca - il bibliotecario decide di annullare l’operazione;
  - InterfacciaPrestiti –> Bibliotecario: - Mostra Interfaccia Prestiti - mostra l’interfaccia relativa alla Gestione dei Prestiti.

3.9 Use case 3.2

- Registrazione restituzione libro



Il diagramma descrive le interazioni necessarie per la registrazione della restituzione di un libro, coinvolgendo diversi partecipanti: l'attore Bibliotecario, un oggetto di tipo InterfacciaPrestiti e un oggetto di tipo GestionePrestito.
Precondizione: il bibliotecario si trova nell'interfaccia per la gestione dei prestiti.

- InterfacciaPrestiti-> Bibliotecario: - Mostra interfaccia Prestiti - l'InterfacciaPrestiti mostra al bibliotecario l'interfaccia generale di Prestiti.
- Bibliotecario -> InterfacciaPrestiti: - Seleziona Restituzione prestito - rappresenta la situazione in cui il bibliotecario seleziona dall'interfaccia la sezione di restituzione prestito.
- InterfacciaPrestiti -> Bibliotecario: - mostra lista prestiti attivi - l'interfaccia prestiti risponde mostrando al bibliotecario la lista dei prestiti.
- Bibliotecario -> InterfacciaPrestiti: - Seleziona prestito - rappresenta la situazione in cui il bibliotecario seleziona dalla lista mostrata, il prestito da voler cancellare.
- InterfacciaPrestiti -> Bibliotecario: - Attende conferma - l'interfaccia attende la conferma della restituzione (il bibliotecario deve cliccare sul bottone "Conferma restituzione")

- **Frame di interazione alt:**

  - [Restituzione confermata] :

    - Bibliotecario -> InterfacciaPrestiti: - Seleziona conferma - rappresenta la situazione in cui il bibliotecario preme il pulsante di Conferma restituzione mediante l'interfaccia di restituzione del libro.

    - InterfacciaPrestiti -> GestionePrestito: - RestituisciPrestito(:Prestito)) -corrisponde al metodo omonimo presente nella classe GestionePrestito, il quale permette la registrazione della restituzione di un libro, per cui modifica la lista dei prestiti, incrementa le copie disponibili del libro e slava tutto sui rispettivi file.

    - InterfacciaPrestiti -> Bibliotecario: - Mostra Lista prestiti aggiornata - l'Interfaccia mostra al bibliotecario la lista dei prestiti aggiornata in seguito al salvataggio.

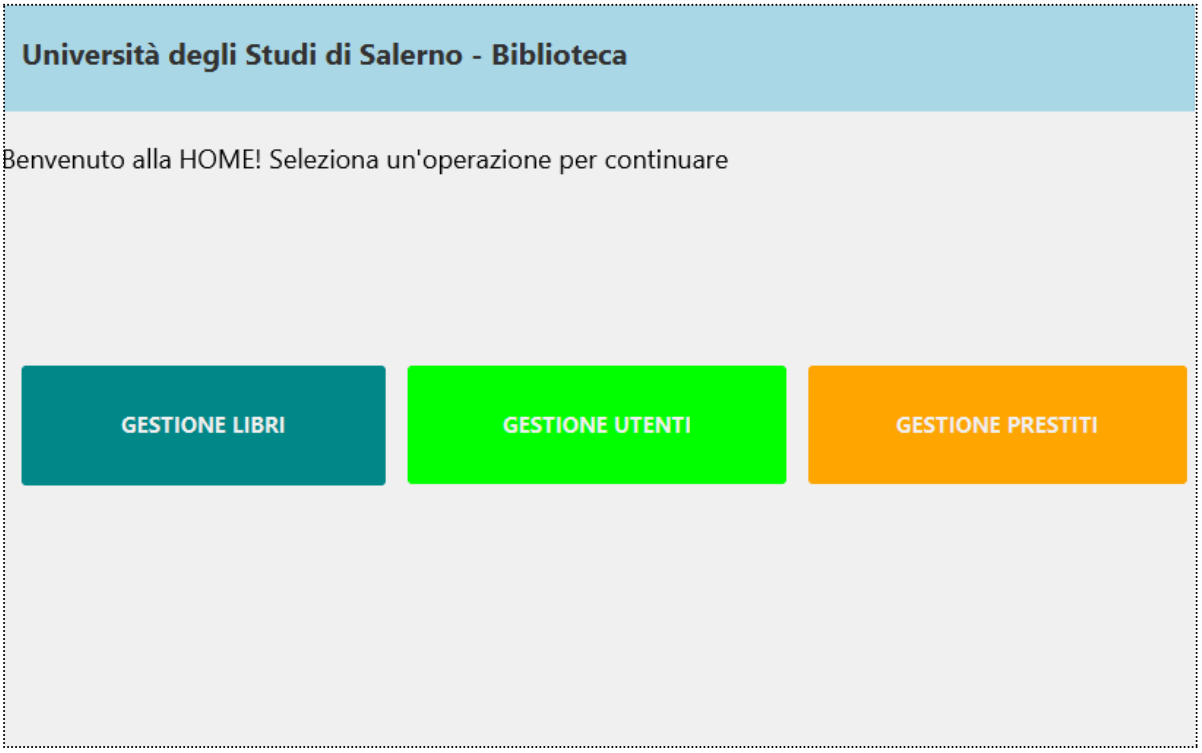
  - [Restituzione Annullata] :

- Se l'operazione di modifica dei dati di un utente viene annullata (il bibliotecario dopo aver selezionato il prestito non clicca il bottone di Conferma restituzione) , l'InterfacciaPrestiti mostra nuovamente al Bibliotecario la vista iniziale.

4. Interface Mock-up(s)

In questa fase, vengono presentati i mock-up dell'interfaccia accompagnati dalla spiegazione di ciascun componente.

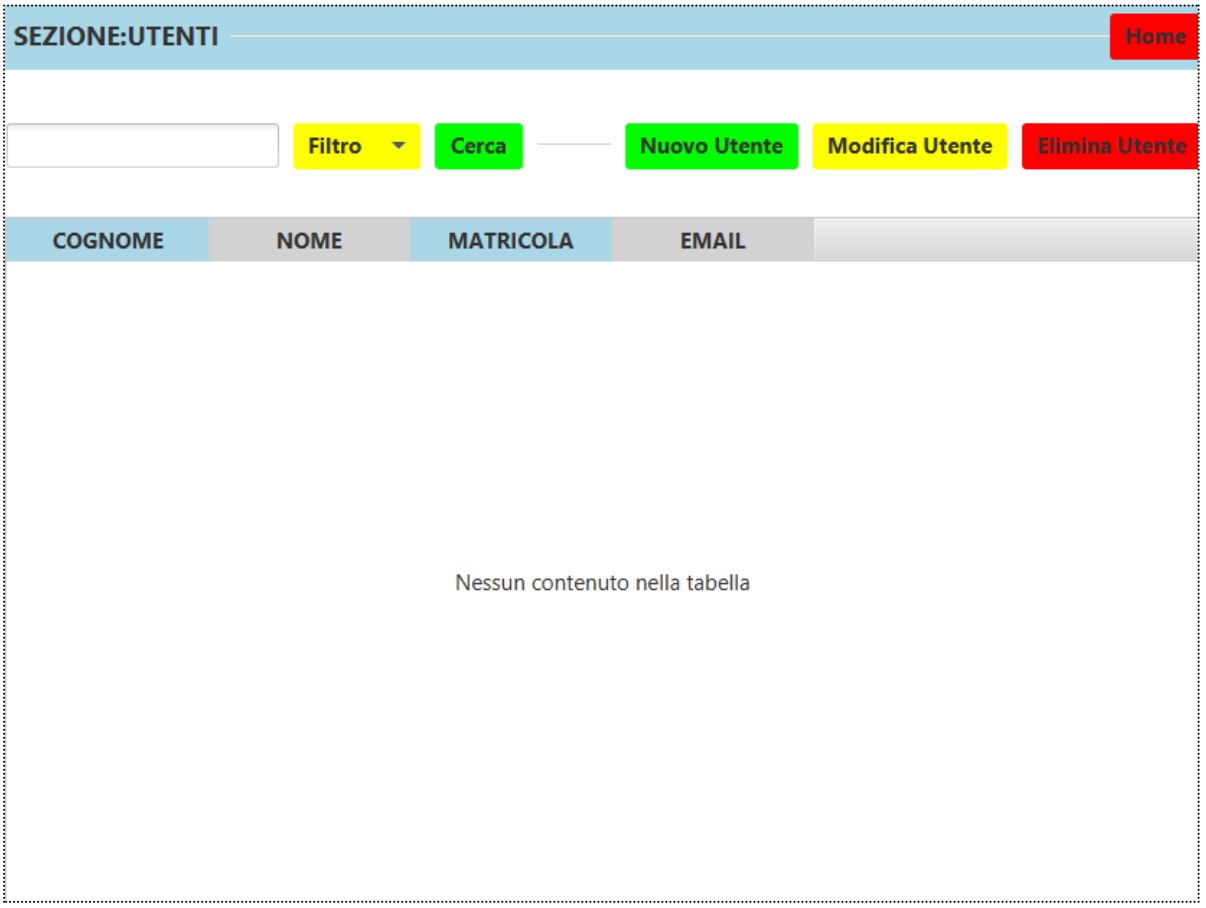
4.1 Home Page



Questa schermata rappresenta la HOME del sistema informatico della Biblioteca Universitaria. Il bibliotecario può interagire in diversi modi:

- **Gestione Libri**: se il bibliotecario preme il pulsante "Gestione Libri", accede alla sezione dedicata alla gestione dei libri;
- **Gestione Utenti**: se il bibliotecario preme il pulsante "Gestione Utenti", accede alla sezione dedicata alla gestione degli utenti;
- **Gestione Prestiti**: se il bibliotecario preme il pulsante "Gestione Prestiti", accede alla sezione dedicata alla gestione dei prestiti.

4.2 Users Section



Questa schermata rappresenta la "Sezione Utenti", ovvero la sezione dedicata alla gestione degli utenti. Il Bibliotecario può interagire con l'interfaccia in diversi modi:

- **Registrazione Utente** : se il bibliotecario vuole aggiungere un nuovo utente, preme il pulsante "Nuovo utente"; l'interfaccia quindi mostra un'area di testo con i vari campi da compilare. Il bibliotecario può inserire le informazioni relative all'utente, ossia cognome, nome, matricola ed email istituzionale. Se il bibliotecario annulla l'operazione di inserimento premendo l'apposito pulsante, si ritorna alla schermata "Sezione Utenti"; nel caso in cui, invece, la matricola inserita sia già esistente oppure il formato dell'email inserita non rispetti il formato standard, l'interfaccia mostra un messaggio di errore e, come prima, si ritorna alla schermata "Sezione Utenti". Una volta confermata la registrazione dell'utente, nella schermata "Sezione Utenti" compare una nuova riga nella tabella contenente le informazioni del nuovo utente;

- **Modifica Utente**: se il bibliotecario vuole modificare i dati di un utente precedentemente registrato, preme il pulsante "Modifica Utente"; l'interfaccia quindi mostra la lista degli utenti attualmente registrati ; successivamente , il bibliotecario seleziona dalla lista l'utente da modificare; l'interfaccia mostra un'area di testo con i campi compilati. Il bibliotecario modifica il dato desiderato . Se il bibliotecario annulla a questo punto l'operazione di modifica premendo l'apposito pulsante, l'interfaccia visualizza nuovamente la lista di utenti attualmente registrati. Nel caso in cui il bibliotecario abbia scelto di modificare la matricola, e la nuova matricola inserita sia già associata ad un altro utente, l'interfaccia mostra un messaggio di errore; nel caso in cui invece il bibliotecario abbia scelto di modificare l'email istituzionale e la nuova email inserita non rispetti il formato standard , l'interfaccia ,come

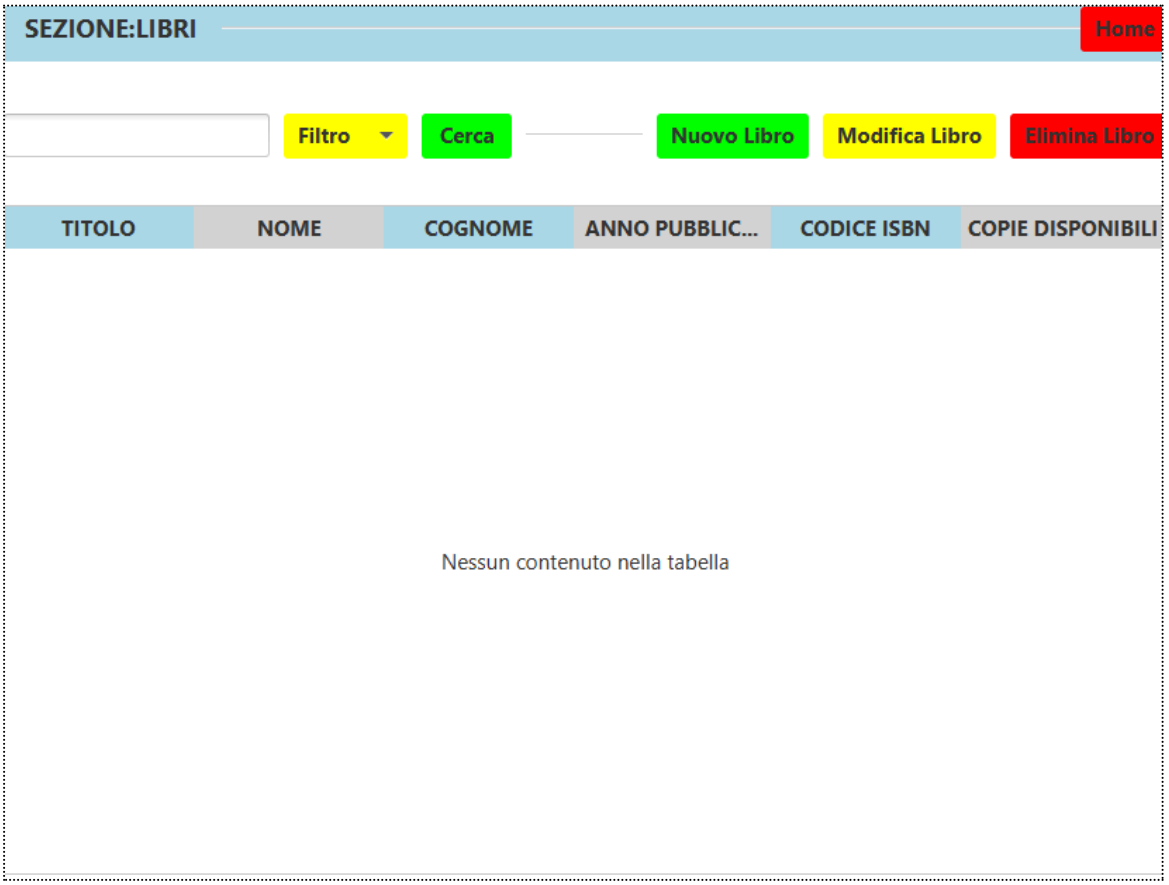
prima, mostra un messaggio di errore. Una volta confermata la modifica, i dati dell'utente modificato vengono aggiornati nella tabella.

- **Eliminazione utente**: l'interfaccia mostra la tabella con le informazioni riguardanti gli utenti che sono stati aggiunti. Se il bibliotecario vuole eliminare un utente precedentemente registrato, seleziona l'utente da eliminare: il pulsante "Elimina Utente" è così abilitato. Il bibliotecario preme il pulsante "Elimina Utente". Se il bibliotecario annulla l'operazione di eliminazione premendo l'apposito pulsante, l'interfaccia mostra nuovamente la tabella contenente la lista di utenti attualmente registrati. Una volta confermata l'operazione di eliminazione,invece, l'utente selezionato viene rimosso dalla tabella.

- **Ricerca utente**: Se il bibliotecario vuole ricercare un utente, specifica mediante un filtro il criterio di ricerca scelto (per cognome o per matricola) e inserisce nel campo di testo il dato da ricercare. Nel caso in cui il bibliotecario ricerchi per cognome, se l'utente viene trovato, l'interfaccia mostra al bibliotecario l'utente cercato, altrimenti se la ricerca fallisce viene mostrato un messaggio di errore. Analogamente , nel caso in cui il bibliotecario ricerchi per matricola, se l'utente viene trovato, l'interfaccia mostra al bibliotecario l'utente cercato, altrimenti se la ricerca fallisce viene mostrato un messaggio di errore. Il bibliotecario può in ogni momento annullare l'operazione di ricerca;

- **Ritorno alla schermata HOME:** se il bibliotecario preme il pulsante "Home", si ritorna alla schermata principale.

4.3 Books Section



Questa schermata rappresenta la "Sezione Libri", ovvero la sezione dedicata alla gestione dei libri. Il Bibliotecario può interagire con l'interfaccia in diversi modi:

- **Registrazione Libro**: se il bibliotecario vuole aggiungere un nuovo libro, preme il pulsante "Nuovo libro"; l'interfaccia quindi mostra un'area di testo con i vari campi da compilare. Il bibliotecario può inserire le informazioni relative al libro, ossia titolo, nome e cognome degli autori, anno di pubblicazione, codice identificativo e numero di copie disponibili. Se il bibliotecario annulla l'operazione di inserimento premendo l'apposito pulsante, si ritorna alla schermata "Sezione Libri"; se il codice identificativo inserito non rispetta il formato standard l'interfaccia mostra un messaggio di errore e si ritorna alla schermata "Sezione Libri"; analogamente, nel caso in cui il codice identificativo sia già esistente, l'interfaccia mostra un messaggio di errore e si ritorna alla schermata "Sezione Libri". Una volta confermata la registrazione del libro, nella schermata "Sezione Libri" compare una nuova riga nella tabella contenente le informazioni del nuovo libro;

- **Modifica Libro:** se il bibliotecario vuole modificare i dati di un libro precedentemente registrato, preme il pulsante "Modifica Libro"; l'interfaccia quindi mostra la lista dei libri attualmente registrati; successivamente , il bibliotecario seleziona dalla lista il libro da modificare; l'interfaccia mostra un'area di testo con i campi compilati. Il bibliotecario modifica il dato desiderato. Se il bibliotecario annulla a questo punto l'operazione di modifica premendo l'apposito pulsante, l'interfaccia visualizza nuovamente la lista di libri attualmente registrati. Nel caso in cui il bibliotecario abbia scelto di modificare il codice identificativo , e il nuovo codice identificativo inserito non rispetta il formato standard , l'interfaccia mostra un messaggio di errore; analogamente, se il codice identificativo inserito

è invece già esistente , l'interfaccia mostra un messaggio di errore. Una volta confermata la modifica, i dati del libro modificato vengono aggiornati nella tabella;

- **Eliminazione Libro**: l'interfaccia mostra la tabella con le informazioni riguardanti i libri che sono stati aggiunti. Se il bibliotecario vuole eliminare un libro precedentemente registrato, seleziona il libro da eliminare: il pulsante "Elimina Libro" è così abilitato. Il bibliotecario preme il pulsante "Elimina Libro". Se il bibliotecario annulla l'operazione di eliminazione premendo l'apposito pulsante, l'interfaccia mostra nuovamente la tabella contenente la lista di libri attualmente registrati. Se libro che si intende eliminare ha attualmente ancora prestiti attivi, l'interfaccia mostra un messaggio di errore e l'operazione di eliminazione viene automaticamente annullata; se invece il libro in questione non ha attualmente prestiti attivi, l'operazione ha successo e il libro selezionato viene rimosso dalla tabella;

- **Ricerca Libro:** Se il bibliotecario vuole ricercare un libro, specifica mediante un filtro il criterio di ricerca scelto (per titolo, per nome e cognome degli autori o per codice identificativo) e inserisce nel campo di testo il dato da ricercare. Nel caso in cui il bibliotecario ricerchi per titolo, se il libro viene trovato, l'interfaccia mostra al bibliotecario il libro cercato, altrimenti se la ricerca fallisce viene mostrato un messaggio di errore. Analogamente , nel caso in cui il bibliotecario ricerchi per nome e cognome degli autori, se il libro viene trovato, l'interfaccia mostra al bibliotecario il libro cercato, altrimenti se la ricerca fallisce viene mostrato un messaggio di errore. Infine, nel caso in cui il bibliotecario ricerchi per codice identificativo, se il libro viene trovato l'interfaccia mostra al bibliotecario il libro cercato altrimenti se la ricerca fallisce, viene mostrato un messaggio di errore. Il bibliotecario può in ogni momento annullare l'operazione di ricerca;

- **Ritorno alla schermata HOME:** se il bibliotecario preme il pulsante "Home", si ritorna alla schermata principale.

4.4 Loans Section

- Schermata per aggiungere un prestito

Questa schermata rappresenta la scheda “Aggiungi prestito” all’interno della “Sezione Prestiti” , che è la sezione dedicata alla gestione dei prestiti.

La scheda “Aggiungi prestito” permette al bibliotecario di registrare le informazioni relative ad un nuovo prestito. Per farlo , il bibliotecario deve indicare l’utente che vuole prendere un libro in prestito e il libro da prendere in prestito:

- **Scelta Utente**: per scegliere l’utente (che deve essere stato registrato) , il bibliotecario specifica mediante il primo filtro, posto al di sotto del pannello “Sezione Utente” il criterio di ricerca per l’utente (cognome o matricola). Se l’utente ha attualmente già tre prestiti attivi, l’operazione di registrazione del prestito fallisce e l’interfaccia mostra un messaggio di errore; in caso contrario, si procede alla scelta del libro;
- **Scelta Libro**: per scegliere il libro (che deve al pari dell’utente essere stato registrato), il bibliotecario specifica mediante il secondo filtro, posto al di sotto del pannello “Sezione Libro” il criterio di ricerca per il libro (titolo, nome e cognome degli autori e codice identificativo). Se il libro non ha copie disponibili, l’operazione di registrazione del prestito fallisce e l’interfaccia mostra un messaggio di errore; in caso contrario si può procedere a registrare il prestito.

Per registrare il prestito, il bibliotecario preme il pulsante “Registra prestito”; in tal modo il prestito effettuato viene inserito all’interno della lista dei prestiti attivi. Il bibliotecario può anche annullare l’operazione premendo il pulsante di annullamento.

Ritorno alla schermata HOME: se il bibliotecario preme il pulsante “Home”, si ritorna alla schermata principale

- Schermata per la restituzione di un prestito

Questa schermata rappresenta la scheda “Restituzione Prestito” all’interno della “Sezione Prestiti” , che è la sezione dedicata alla gestione dei prestiti.
La scheda “Restituzione Prestito” permette al bibliotecario di effettuare la restituzione di un libro in prestito.
La scheda mostra la lista dei prestiti attualmente attivi, ordinata per data di restituzione; per ogni prestito, sono indicati:

1. la matricola e il cognome dell’utente;
2. il titolo e il codice identificativo del libro;
3. la data di restituzione del prestito.

Il bibliotecario seleziona il prestito effettuato da un utente tra la lista dei prestiti attivi; per confermare l’operazione preme il pulsante “Conferma Restituzione”: in questo modo, la restituzione del prestito viene registrata e il prestito rimosso dalla lista. Il bibliotecario può anche annullare l’operazione premendo il pulsante di annullamento

Ritorno alla schermata HOME: se il bibliotecario preme il pulsante “Home”, si ritorna alla schermata principale.

5. Traceability Matrix Update

In questa fase viene aggiornata la matrice di tracciabilità, in modo da poter stabilire collegamenti tra i requisiti e le diverse fasi del progetto, permettendo di facilitare il monitoraggio dello stato di avanzamento di ciascuno di essi.

Requisiti	Design	Implementazione	Test
IF-1.1	Diagramma di Sequenza: UC-1.1 (registrazione libro)	Classe: GestioneLibro Metodo: inserisci(Libro libro)	Classe: GestioneLibroTest Metodi: testInserimentoCorretto(); testInserimentoDuplicato(); testInserimentoCodiceNonValido()
IF-1.2	Diagramma delle Classi (Model): Libro	Classe: Libro Metodo: equals(Object o)	Classe: LibroTest Metodo: testEquals()
IF-1.3	Diagramma di Sequenza: UC-1.4 (Ricerca di un libro)	Classe: GestioneLibro Metodi: ricercaLibroTitolo(Stringtitolo); ricercaLibroAutore(String nomeAutore, String cognomeAutore); ricercaLibroCodice(String codice)	Classe: GestioneLibroTest Metodi: testRicercaLibroTitolo(); testRicercaLibroAutore(); testRicercaLibroCodice(); testRicercaTitoloFallitaFirst(); testRicercaTitoloFallitaSecond(); testRicercaAutoreFallitaFirst(); testRicercaAutoreFallitaSecond(); testRicercaCodiceFallitaFirst(); testRicercaCodiceFallitaSecond()
IF-1.4	Interface Mock-up: 4.3 Books Section	Classi:Libro, InterfacciaLibriController Metodo classe Libro: compareTo(Libro l) Metodiclasse InterfacciaLibriController: inizializzaTabella(); aggiornaTabella()	Test GUI Manuale + Classe: LibroTest Metodo: testCompareTo()

IF-1.5	Diagramma di Sequenza: UC-1.2 (Modifica di un libro)	Classe: GestioneLibro Metodo: modifica(Libro libroModificato)	Classe: GestioneLibroTest Metodo: testModifica()
IF-1.6	Diagramma di Sequenza: UC-1.3 (Eliminazione di un libro)	Classe: GestioneLibro Metodo: elimina(Libro libro)	Classe: GestioneLibroTest Metodi: testEliminaFirst(); testEliminaSecond();
IF-2.1	Diagramma di Sequenza: UC-2.1 (Registrazione Utente)	Classe: GestioneUtente Metodo: inserisci(Utente utente)	Classe: GestioneUtenteTest Metodi: testInserimentoCorretto(); testInserimentoMatricola Duplicata();
IF-2.2	Diagramma delle Classi (Model): Utente	Classe: Utente Metodo: equals(Object o)	Classe: UtenteTest Metodo: testEquals()
IF-2.3	Diagramma di Sequenza: UC-2.4 (Ricerca Utente)	Classe: GestioneUtente Metodi: cercaUtenteCognome(String cognome); cercaUtenteMatricola(int matricola)	Classe: GestioneUtenteTest Metodi: testRicercaMatricolaSuccesso(); testRicercaMatricolaFallimento1(; testRicercaMatricolaFallimento2(; testRicercaCognomeFallimento() testRicercaCognomeMultiplo();
IF-2.4	Interface Mock-up: 4.2 Users Section	Classi:Utente, InterfacciaUtentiControlle Metodo classe Utente: compareTo(Utente u) Metodo classe InterfacciaUtentiController classeInterfacciaUtentiController: aggiornaTabella()	Test GUI Manuale + Classe: UtenteTest Metodi: testOrdinamento1(); testOrdinamento2(); testOrdinamento3(); testOrdinamento4();

IF-2.5	Diagramma delle Classi (Model): GestionePrestito	Classe: GestionePrestito Metodo: registraPrestito(Prestito prestito)	Classe: GestionePrestitoTest Metodi: testRegistraPrestitoSuccesso(); testRegistraPrestitoSenzaCopie()
IF-2.6	Diagramma di Sequenza: UC-2.2 (Modifica Utente)	Classe: GestioneUtente Metodo: modifica(Utente utenteModificato)	Classe: GestioneUtenteTest Metodo: testModificaUtente()
IF-2.7	Diagramma di Sequenza: UC-2.3 (Eliminazione Utente)	Classe: GestioneUtente Metodo: elimina(Utente utente)	Classe: GestioneUtenteTest Metodo: testEliminaUtente()
IF-3.1	Diagramma di Sequenza: UC-3.1 (Registrazione Prestito)	Classe: GestionePrestito Metodo: registraPrestito(Prestito prestito)	Classe: GestionePrestitoTest Metodi: testRegistraPrestitoSuccesso(); testRegistraPrestitoSenzaCopie()
IF-3.2	Diagramma di Sequenza: UC-3.1 (Registrazione restituzione libro)	Classe: GestionePrestito Metodo: restituisciPrestito(Prestito prestito)	Classe: GestionePrestitoTest Metodi: testRestituzionePrestito()
IF-3.3	Interface Mock-up: 4.4 Loans Section	Classi:Prestito, InterfacciaPrestitiController Metodo classe Prestito: compareTo(Prestito p) Metodi classeinterfacciaPrestitiController: initialize(); aggiornaTabella()	Test Manuale GUI + Classe: PrestitoTest Metodo: testCompareTo()
IF-3.4	Diagramma delle Classi (Model): GestionePrestito	Classe: GestionePrestito Metodo: registraPrestito(Prestito prestito)	Classe: GestionePrestitoTest Metodo: testRegistraPrestitoSenzaCopie()

IF-3.5	Diagramma delle Classi (Model): GestionePrestito	Classe: GestionePrestito Metodo: registraPrestito(Prestito prestito)	Classe: GestionePrestitoTest Metodo: testLimiteMassimoPrestiti()
IF-3.6	Diagramma delle Classi (Controller): InterfacciaPrestitiController	Classe: InterfacciaPrestitiController Metodo: updateItem(LocalDate data, boolean empty)	Test Manuale GUI + Ispezione Visiva
DF-4.1	Diagramma delle Classi (Model): Libro	Classe:Libro (attributi titolo, nomeAutore cognomeAutore,annoPubblicazione, codice, copieDisponibili)	Classe: LibroTest Metodo: testConstructor()
DF-4.2	Diagramma delle Classi (Model): Utente	Classe: Utente(attributi nome,cognome, matricola, email)	Classe: UtenteTest Metodo: testCostruttore()
DF-4.3	Diagramma delle Classi (Model): Prestito	Classe: Prestito (attributi utente, libro, dataRestituzione)	Classe: PrestitoTest Metodo: testCostruttore()

DF-4.4	Diagramma delle Classi (Model): Utente	Classe:
		Utente(attributi nome e cognome)
			Classe:
			UtenteTest
			Metodi:
			testCostruttore();
			testSetter();

DF-4.5	Diagramma delle Classi (Model):GestioneUtente	Classe: GestioneUtente
		Metodo: controlloEmail(Utente u)
			Classe:
			GestioneUtenteTest
			
			Metodi:
			testEmailFormatoSbagliato();
			testEmailNomeSbagliato();
			testEmailDuplicata();
			testEmailConNumeri()

DF-4.6	Diagramma delle Classi (Model): Libro	Classe: Libro (attributi nomeAutore e cognomeAutore)	Classe: LibroTest
			Metodi:
			testConstructor();
			testSetNomeAutore();
			testSetCognomeAutore()

DF-4.7	Diagramma delle Classi (Model):GestioneLibro	Classe: GestioneLibro Metodo: controllaCodice(String codice)	Classe: GestioneLibroTest Metodo: testInserimentoCodiceNonValido()
DF-4.8	Diagramma delle Classi (Model): GestioneLibro - GestioneUtente - GestionePrestito	Classi: GestioneLibro,GestioneUtenti, GestionePrestiti Metodo classe GestioneLibro: salvaLibri(String file) Metodo classe GestioneUtente: salvaUtenti() Metodo classe GestionePrestito: salvaPrestiti()	Classi: GestioneLibroTest; GestioneUtenteTest; GestionePrestitoTest; Metodi: testCaricaSalvaLibri(); testPersistenzaDati() testPersistenzaPrestiti()
UI-5.1	Diagramma dei Package (Controller - View)	Package: it.unisa.biblioteca.controller; it.unisa.biblioteca.view;	Ispezione Visiva
UI-5.2	Interface Mock-up: 4.1 Home Page	Classi: InterfacciaLibriController; InterfacciaUtentiController; InterfacciaPrestitiController;	Ispezione Visiva
UI-5.3	Interface Mock-up: 4.2 Users Section	File FXML (Package it.unisa.biblioteca.view): InterfacciaUtentiView.fxml Classe: InterfacciaUtentiiController	Test Manuale GUI + Ispezione Visiva
UI-5.4	Interface Mock-up: 4.3 Books Section	File FXML (Package it.unisa.biblioteca.view): InterfacciaLibriView.fxml Classe: InterfacciaLibriController	Test Manuale GUI + Ispezione Visiva
UI-5.5	Interface Mock-up: 4.4 Loans Section	File FXML (Package it.unisa.biblioteca.view): InterfacciaPrestitiView.fxml Classe: InterfacciaPrestitiController	Test Manuale GUI + Ispezione Visiva

UI-5.6	Interface Mock-up: 4.3 Books Section	File FXML (Package it.unisa.biblioteca.view): InterfacciaLibriView.fxml Classe: InterfacciaLibriController	Test Manuale GUI + Ispezione Visiva
UI-5.7	Interface Mock-up: 4.2 Users Section	File FXML (Package it.unisa.biblioteca.view): InterfacciaUtentiView.fxml Classe: InterfacciaUtentiController	Test Manuale GUI + Ispezione Visiva
UI-5.8	Interface Mock-up: 4.4 Loans Section	File FXML (Package it.unisa.biblioteca.view): InterfacciaPrestitiView.fxml Classe: InterfacciaPrestitiController	Test Manuale GUI + Ispezione Visiva
UI-5.9	Interface Mock-up: 4.4 Loans Section	File FXML (Package it.unisa.biblioteca.view): InterfacciaPrestitiView.fxml Classe: InterfacciaPrestitiController	Test Manuale GUI + Ispezione Visiva
UI-5.10	Interface Mock-up: 4.3 Books Section	File FXML (Package it.unisa.biblioteca.view): InterfacciaLibriView.fxml Classe: InterfacciaLibriController	Test Manuale GUI + Ispezione Visiva
UI-5.11	Interface Mock-up: 4.2 Users Section	File FXML (Package it.unisa.biblioteca.view): InterfacciaUtentiView.fxml Classe: InterfacciaUtentiController	Test Manuale GUI + Ispezione Visiva
UI-5.12	Interface Mock-up: 4.2 - 4.3 - 4.4 Users - Books - Loans Sectior	File FXML (Package it.unisa.biblioteca.view): NuovoLibroView.fxml NuovoUtenteView.fxml InterfacciaPrestitiView.fxml Classe: NuovoLibroController NuovoUtenteController InterfacciaPrestitiController	Test Manuale GUI + Ispezione Visiva
UI-5.13	Interface Mock-up: 4.2 - 4.3 - 4.4 Users - Books - Loans Sectior	File FXML (Package it.unisa.biblioteca.view): NuovoLibroView.fxml NuovoUtenteView.fxml InterfacciaPrestitiView.fxml Classe: NuovoLibroController NuovoUtenteController InterfacciaPrestitiController	Test Manuale GUI + Ispezione Visiva