Giorgia Rispoli [0612709246]
Giorgio Lupo [0612709260]
Margherita Gerarda Morriello [0612704256]
Emanuele Troisi [0612709575]

DESIGN

Biblioteca universitaria - GRUPPO 25

INDICE

1 **INTRODUZIONE**

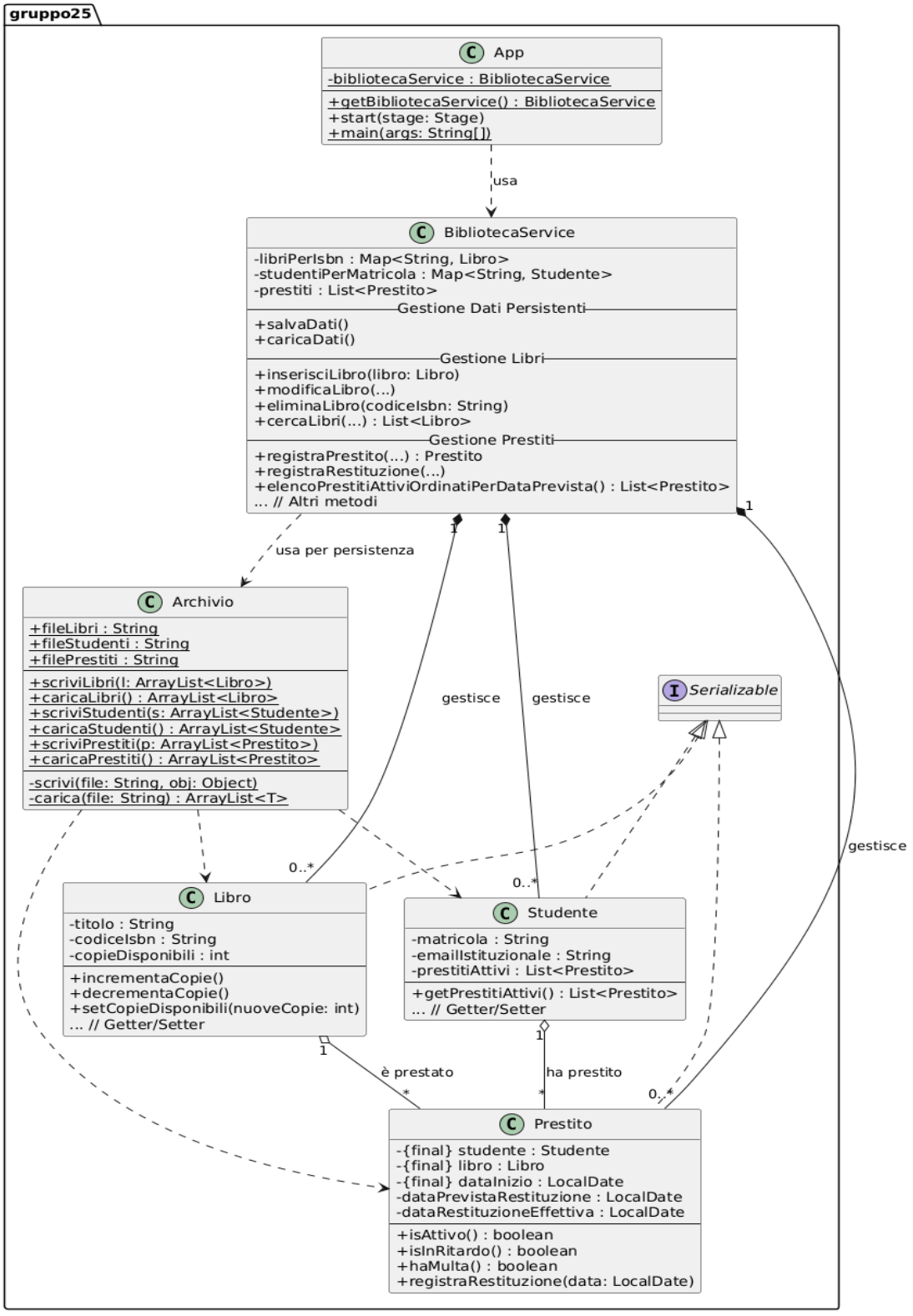
Il documento in seguito fornisce una visione d'insieme estetica e dettagliata del sistema che si sta sviluppando in modo da agevolare la comprensione di quest'ultimo, rendendo più facile il lavoro del team. Tale documento mostra i vari diagrammi di rappresentazione del sistema (per i quali vi si aggiunge un'opportuna descrizione dettagliata): vi si riconoscono i *class diagram*, capaci di descrivere il sistema nel suo aspetto **statico** attraverso classi e metodi, e *diagrammi di interazione* che invece ne rappresentano la componente **dinamica**. In particolar modo per i diagrammi di interazione faremo affidamento ai diagrammi di sequenza, realizzati grazie alla piattaforma di supporto *PlantUML*. In tale documento di design, verranno mostrate le interfacce che l'utente avrà a disposizione, realizzate mediante l'ausilio dello strumento di sviluppo visivo *SceneBuilder* e descritte nelle componenti essenziali che le costituiscono suddivise poi per funzionalità.

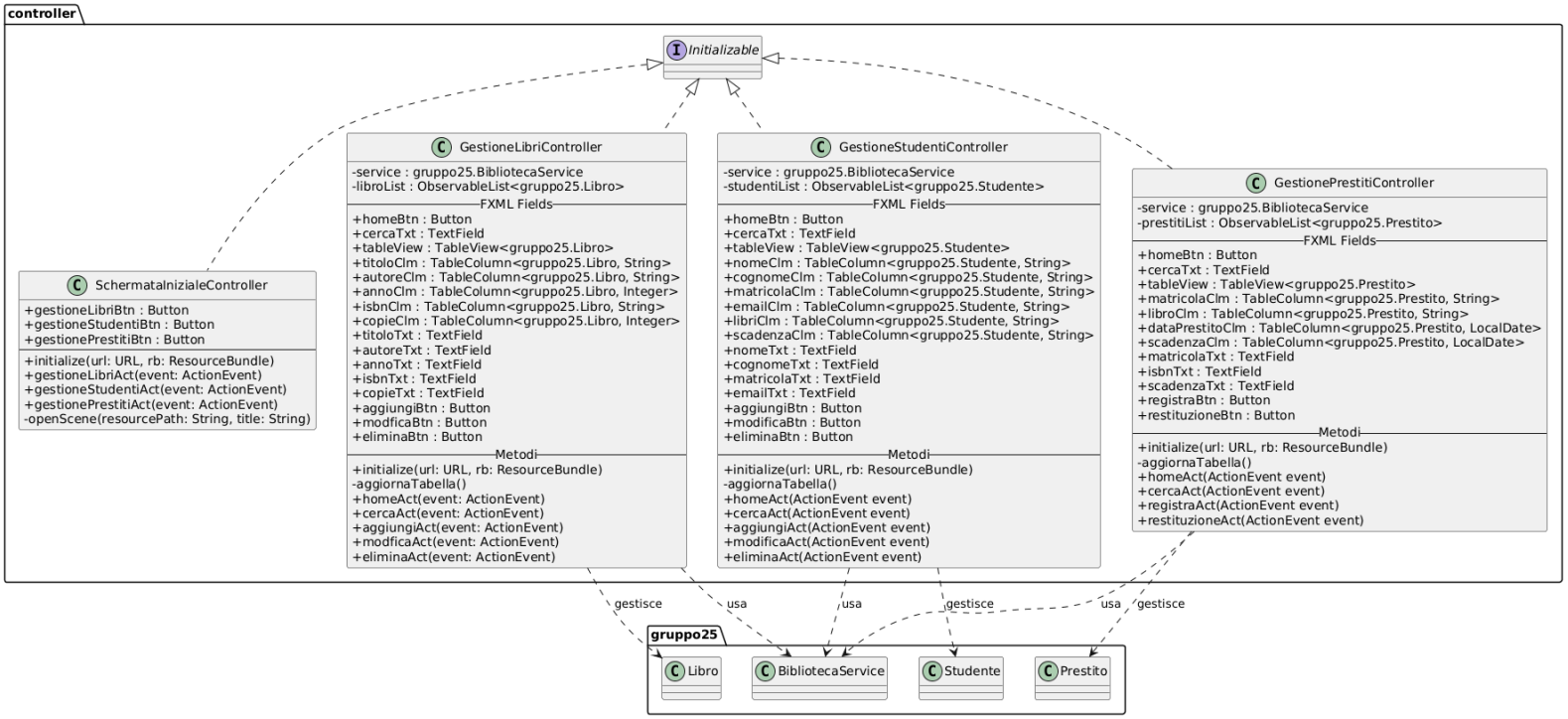
1.1 PATTERN PROGETTUALE

Per la realizzazione del sistema, l'architettura utilizzata è quella MVC. In un design orientato agli oggetti, sarà dunque possibile garantire la manutenibilità facilitando eventuali modifiche future. L'architettura MVC si basa su aspetti fondamentali, sui quali si basa lo sviluppo del software

Model: fornisce i metodi che interagiscono con i dati
View: consente di presentare i dati all'utente
Controller: è il tramite del model e della view

2 DIAGRAMMA DELLE CLASSI





2.1 Descrizione del diagramma

2.1.1 Classe Libro

Rappresenta l'entità "Libro" all'interno del sistema della biblioteca. Contiene i dati identificativi del volume (come titolo, autore, ISBN) e gestisce il conteggio delle copie disponibili per il prestito. La classe è serializzabile.

Attributi:

- **serialVersionUID**: Campo statico e privato richiesto per la serializzazione.
- **titolo**: Stringa che rappresenta il titolo del libro.
- **autore**: Stringa che rappresenta l'autore del libro.
- **annoPubblicazione**: Intero che indica l'anno in cui il libro è stato pubblicato.
- **codiceIsbn**: Stringa univoca (ISBN) che funge da identificatore del libro.
- **copieDisponibili**: Intero che indica il numero di copie del libro attualmente disponibili per il prestito.
-

Metodi:

- **Libro(...)**: Costruttore della classe. Inizializza l'oggetto con tutti i suoi attributi, eseguendo controlli di validazione (es. titoli non vuoti, anno non futuro, copie non negative).
- **getTitolo():** Restituisce il titolo.
- **setTitolo(String titolo)**: Imposta il titolo, con validazione.
- **getAutore()**: Restituisce l'autore.
- **setAutore(String autore)**: Imposta l'autore, con validazione.
- **getAnnoPubblicazione():** Restituisce l'anno di pubblicazione.

- **setAnnoPubblicazione(int annoPubblicazione)**: Imposta l'anno di pubblicazione, con validazione.
- **getCodiceIsbn():** Restituisce il codice ISBN.
- **getCopieDisponibili()**: Restituisce il numero di copie disponibili.
- **setCopieDisponibili(int nuoveCopie):** Imposta il numero di copie disponibili, con validazione.
- **incrementaCopie():** Aumenta il conteggio delle copie disponibili di 1 (usato alla restituzione).
- **decrementaCopie()**: Diminuisce il conteggio delle copie disponibili di 1 (usato al prestito), lanciando un'eccezione se le copie sono zero.
- **equals(Object o)**: Sovrascritto; due libri sono uguali se hanno lo stesso ISBN.
- **hashCode()**: Sovrascritto; basato sul codice ISBN.

2.1.2 Classe Studente

Rappresenta l'entità "Studente" iscritto alla biblioteca. Memorizza i dati anagrafici, la matricola, l'email istituzionale e tiene traccia dei prestiti che lo studente ha attualmente in corso. La classe è serializzabile.

Attributi:

- **serialVersionUID:** Campo statico e privato richiesto per la serializzazione.
- **nome**: Stringa che rappresenta il nome dello studente.
- **cognome**: Stringa che rappresenta il cognome dello studente.
- **matricola**: Stringa univoca che funge da identificatore dello studente.
- **emailIstituzionale**: Stringa che rappresenta l'email dello studente.
- **prestitiAttivi:** Lista (List<Prestito>) dei prestiti attualmente in corso per questo studente.

Metodi:

- **Studente(...):** Costruttore per inizializzare l'oggetto Studente.
- **getNome():** Restituisce il nome.
- **setNome(String nome):** Imposta il nome.
- **getCognome()**: Restituisce il cognome.
- **setCognome(String cognome):** Imposta il cognome.
- **getMatricola()**: Restituisce la matricola.
- **getEmailIstituzionale()**: Restituisce l'email istituzionale.
- **setEmailIstituzionale(String email):** Imposta l'email istituzionale.
- **getPrestitiAttivi():** Restituisce la lista dei prestiti attivi.
- **aggiungiPrestito(Prestito p)**: Aggiunge un prestito alla lista dei prestiti attivi, lanciando un'eccezione se si supera il limite di 3 prestiti.

- **chiudiPrestito(Prestito p)**: Rimuove un prestito dalla lista dei prestiti attivi, segnandolo come concluso.
- **equals(Object o)**: Sovrascritto; due studenti sono uguali se hanno la stessa matricola.
- **hashCode():** Sovrascritto; basato sulla matricola.

2.1.3 Classe Prestito

Rappresenta una transazione di prestito tra uno studente e un libro. Contiene i riferimenti all'oggetto Studente e Libro coinvolti, le date di inizio e di prevista restituzione, e la data effettiva di restituzione (se avvenuta). La classe è serializzabile.

Attributi:

- **serialVersionUID:** Campo statico e privato richiesto per la serializzazione.
- **studente:** Riferimento all'oggetto Studente a cui è stato fatto il prestito (finale).
- **libro:** Riferimento all'oggetto Libro prestato (finale).
- **dataInizio:** Data di inizio del prestito (impostata alla creazione).
- **dataPrevistaRestituzione:** Data entro cui il libro dovrebbe essere restituito.
- **dataRestituzioneEffettiva:** Data in cui il libro è stato effettivamente restituito (null se il prestito è attivo).

Metodi:

- **Prestito(Studente studente, Libro libro, LocalDate dataPrevistaRestituzione):** Costruttore che crea un nuovo prestito, impostando la data di inizio a quella corrente.
- **getStudente():** Restituisce l'oggetto Studente.
- **getLibro():** Restituisce l'oggetto Libro.
- **getDataInizio():** Restituisce la data di inizio del prestito.
- **getDataPrevistaRestituzione():** Restituisce la data prevista di restituzione.
- **getDataRestituzioneEffettiva():** Restituisce la data effettiva di restituzione.
- **isAttivo():** Restituisce true se il prestito non è stato ancora chiuso (dataRestituzioneEffettiva è null).
- **isInRitardo():** Restituisce true se il prestito è attivo e la data odierna è successiva a quella prevista.
- **haMulta():** Restituisce true se il prestito è stato chiuso con un ritardo di oltre due settimane.
- **registraRestituzione(LocalDate dataRestituzione):** Imposta la data effettiva di restituzione, chiudendo di fatto il prestito.
- **equals(Object o):** Sovrascritto; basato su matricola, ISBN e data di inizio.

- **hashCode():** Sovrascritto; basato su matricola, ISBN e data di inizio.

2.1.4. Classe Archivio

Classe di utilità statica (public static) che gestisce la persistenza dei dati (Libri, Studenti, Prestiti) su file binari locali utilizzando la serializzazione di Java I/O.

Attributi:

- **fileLibri:** Stringa che contiene il nome del file per i dati dei libri ("archivioLibri.bin").
- **fileStudenti:** Stringa che contiene il nome del file per i dati degli studenti ("archivioStudenti.bin").
- **filePrestiti:** Stringa che contiene il nome del file per i dati dei prestiti ("archivioPrestiti.bin").

Metodi:

- **scriviLibri(ArrayList<Libro> l):** Salva la lista di libri nel file binario dedicato.
- **caricaLibri():** Carica e restituisce la lista di libri dal file binario.
- **scriviStudenti(ArrayList<Studente> s):** Salva la lista di studenti nel file binario dedicato.
- **caricaStudenti():** Carica e restituisce la lista di studenti dal file binario.
- **scriviPrestiti(ArrayList<Prestito> p):** Salva la lista di prestiti nel file binario dedicato.
- **caricaPrestiti():** Carica e restituisce la lista di prestiti dal file binario.
- **scrivi(String file, Object obj):** Metodo privato generico che esegue la serializzazione di un oggetto su un file specificato, gestendo eccezioni I/O.
- **carica(String file):** Metodo privato generico che esegue la deserializzazione di un oggetto (lista) da un file specificato, gestendo eccezioni I/O.

2.1.5. Classe BibliotecaService

È il service layer centrale che incapsula tutta la logica di business della biblioteca. Gestisce le collezioni di dati in memoria (Libri, Studenti, Prestiti) e offre metodi per interagire con esse (CRUD, gestione prestiti, ricerca, salvataggio).

Attributi:

- **libriPerIsbn:** Mappa (Map<String, Libro>) che memorizza i libri con l'ISBN come chiave, per ricerche veloci.

- **studentiPerMatricola:** Mappa (Map<String, Studente>) che memorizza gli studenti con la matricola come chiave.
- **prestiti:** Lista (List<Prestito>) di tutti i prestiti registrati, sia attivi che conclusi.

Metodi:

- **salvaDati():** Salva tutte le collezioni di dati persistenti utilizzando la classe Archivio.
- **caricaDati():** Carica i dati persistenti da file binario utilizzando Archivio e ricostruisce le strutture dati in memoria, ricollegando correttamente i prestiti attivi agli studenti.
- **aggiungiLibro(Libro l):** Aggiunge un libro se il suo ISBN non è già presente.
- **modificaLibro(String isbn, Libro nuovo):** Modifica i dati di un libro esistente.
- **eliminaLibro(String isbn):** Rimuove un libro, solo se non ha prestiti attivi.
- **getLibroByIsbn(String isbn):** Ottiene un libro tramite ISBN, o lancia un'eccezione se non trovato.
- **cercaLibri(String titolo, String autore, String isbn):** Cerca e filtra i libri in base ai parametri forniti.
- **elencoLibri():** Restituisce una lista di tutti i libri.
- **aggiungiStudente(Studente s):** Aggiunge uno studente se la matricola non è già presente.
- **modificaStudente(String matricola, Studente nuovo):** Modifica i dati di uno studente esistente.
- **eliminaStudente(String matricola):** Rimuove uno studente, solo se non ha prestiti attivi.
- **getStudenteByMatricola(String matricola):** Ottiene uno studente tramite matricola, o lancia un'eccezione se non trovato.
- **cercaStudente(String cognome, String matricola):** Cerca e filtra gli studenti per cognome o matricola.
- **elencoStudenti():** Restituisce una lista di tutti gli studenti.
- **registraPrestito(String matr, String isbn, LocalDate dataRest):** Crea e registra un nuovo prestito, con controlli su disponibilità copie e limite prestiti studente. Decrementa le copie del libro.
- **registraRestituzione(Prestito p, LocalDate dataRest):** Chiude un prestito registrando la data di restituzione effettiva. Incrementa le copie del libro e aggiorna lo studente.
- **elencoPrestitiAttiviOrdinatiPerDataPrevista():** Restituisce una lista dei prestiti attivi ordinati per data prevista di restituzione.
- **cercaPrestiti(String matr, String isbn):** Cerca e filtra i prestiti attivi per matricola e/o ISBN.

2.1.6 Classe App

È la classe di avvio dell'applicazione JavaFX, estendendo Application. Inizializza l'istanza di BibliotecaService, carica i dati persistenti all'avvio e assicura che i dati vengano salvati alla chiusura della finestra principale.

Attributi:

- **bibliotecaService:** Istanza statica e privata di BibliotecaService che gestisce tutti i dati e la logica.

Metodi:

- **static {}:** Blocco di inizializzazione statico. Viene eseguito una sola volta, crea l'oggetto bibliotecaService e chiama caricaDati().
- **getBibliotecaService():** Metodo statico per accedere all'unica istanza di BibliotecaService (pattern singleton).
- **start(Stage stage):** Metodo principale di JavaFX. Carica la scena iniziale, imposta il titolo della finestra e, crucialmente, imposta l'azione di chiusura (setOnCloseRequest) per chiamare bibliotecaService.salvaDati().
- **main(String[] args):** Punto di ingresso dell'applicazione, chiama launch() di JavaFX.

2.1.7 Classe SchermataInizialeController (Home):

Controller per la schermata di navigazione principale dell'applicazione. Fornisce i pulsanti per accedere alle tre sezioni di gestione: Libri, Studenti e Prestiti.

Attributi (FXML):

- **gestioneLibriBtn:** Pulsante per accedere alla sezione di gestione dei libri.
- **gestioneStudentiBtn:** Pulsante per accedere alla sezione di gestione degli studenti.
- **gestionePrestitiBtn:** Pulsante per accedere alla sezione di gestione dei prestiti.

Metodi:

- **initialize(URL url, ResourceBundle rb):** Metodo standard di inizializzazione del controller.
- **gestioneLibriAct(ActionEvent event):** Azione richiamata al clic su gestioneLibriBtn per aprire la schermata di Gestione Libri.

- **gestioneStudentiAct(ActionEvent event):** Azione richiamata al clic su gestioneStudentiBtn per aprire la schermata di Gestione Studenti.
- **gestionePrestitiAct(ActionEvent event):** Azione richiamata al clic su gestionePrestitiBtn per aprire la schermata di Gestione Prestiti.
- **openScene(String resourcePath, String title):** Metodo privato di utilità per caricare e visualizzare un nuovo file FXML.

2.1.8 Classe GestioneLibriController

Controller per l'interfaccia di gestione dei libri. Permette agli utenti di visualizzare, cercare, aggiungere, modificare ed eliminare i libri all'interno dell'archivio.

Attributi (FXML):

- **homeBtn:** Pulsante per tornare alla schermata iniziale.
- **cercaTxt:** Campo di testo per l'inserimento dei parametri di ricerca.
- **cercaBtn:** Pulsante per avviare la ricerca dei libri.
- **tableView:** Tabella per la visualizzazione della lista dei libri.
- **titoloClm:** Colonna della tabella per il titolo del libro.
- **autoreClm:** Colonna della tabella per l'autore del libro.
- **annoClm:** Colonna della tabella per l'anno di pubblicazione.
- **isbnClm:** Colonna della tabella per il codice ISBN.
- **copieClm:** Colonna della tabella per il numero di copie disponibili.
- **titoloTxt:** Campo di testo per l'inserimento/modifica del titolo.
- **autoreTxt:** Campo di testo per l'inserimento/modifica dell'autore.
- **annoTxt:** Campo di testo per l'inserimento/modifica dell'anno.
- **isbnTxt:** Campo di testo per l'inserimento/modifica dell'ISBN.
- **copieTxt:** Campo di testo per l'inserimento/modifica delle copie.
- **aggiungiBtn:** Pulsante per registrare un nuovo libro.
- **modficaBtn:** Pulsante per salvare le modifiche al libro selezionato.
- **eliminaBtn:** Pulsante per eliminare il libro selezionato.
- **libroList:** Lista osservabile (non FXML, ma interna) collegata a tableView.
- **service:** Riferimento all'istanza del BibliotecaService (logica di business).

Metodi:

- **initialize(URL url, ResourceBundle rb):** Metodo di inizializzazione del controller.
- **inizializzaBinding():** Definisce le regole di disabilitazione/abilitazione dei componenti GUI..
- **inizializzaListenerSelezione():**i Listener che si attiva alla selezione di una riga in tabella per popolare i campi di input.

- **aggiornaTabella():** Aggiorna il contenuto della tableView recuperando i dati aggiornati dal *service*.
- **pulisciCampi():** Svuota tutti i campi di input.
- **mostraErrore(String messaggio):** Visualizza un pop-up di errore all'utente.
- **homeAct(ActionEvent event):** Torna alla schermata principale.
- **cercaAct(ActionEvent event):** Esegue l'azione di ricerca e filtra la lista dei libri visualizzati.
- **aggiungiAct(ActionEvent event)**: Esegue l'azione di aggiunta di un nuovo libro.
- **modficaAct(ActionEvent event):** Esegue l'azione di modifica del libro selezionato.
- **eliminaAct(ActionEvent event):** Esegue l'azione di eliminazione del libro selezionato, previa conferma.

2.1.9 Classe GestioneStudentiController

Controller per l'interfaccia di gestione degli studenti. Permette operazioni creazione, lettura, modifica, eliminazione sugli studenti e visualizza informazioni sui loro prestiti attivi.

Attributi (FXML):

- **homeBtn:** Pulsante per tornare alla schermata iniziale.
- **cercaTxt:** Campo di testo per la ricerca di studenti.
- **cercaBtn:** Pulsante per avviare la ricerca degli studenti.
- **tableView:** Tabella per la visualizzazione della lista degli studenti.
- **nomeClm:** Colonna per il nome dello studente.
- **cognomeClm:** Colonna per il cognome dello studente.
- **matricolaClm:** Colonna per la matricola.
- **emailClm:** Colonna per l'email istituzionale.
- **libriClm:** Colonna che mostra il numero di libri in prestito (derivata).
- **scadenzaClm:** Colonna che mostra la scadenza dei prestiti (derivata).
- **nomeTxt:** Campo di testo per l'inserimento/modifica del nome.
- **cognomeTxt:** Campo di testo per l'inserimento/modifica del cognome.
- **matricolaTxt:** Campo di testo per l'inserimento/modifica della matricola.
- **emailTxt:** Campo di testo per l'inserimento/modifica dell'email.
- **aggiungiBtn:** Pulsante per registrare un nuovo studente.
- **modificaBtn:** Pulsante per salvare le modifiche allo studente selezionato.
- **eliminaBtn:** Pulsante per eliminare lo studente selezionato.
- **studentiList:** Lista osservabile collegata a tableView.
- **service:** Riferimento all'istanza del BibliotecaService.

Metodi:

- **initialize(URL url, ResourceBundle rb):** Metodo di inizializzazione del controller.
- **inizializzaBinding():** Definisce le regole di disabilitazione/abilitazione dei pulsanti CRUD.
- **inizializzaListenerSelezione():** Popola i campi di testo quando uno studente viene selezionato.
- **aggiornaTabella():** Aggiorna il contenuto della tableView.
- **pulisciCampi():** Svuota i campi di input.
- **mostraErrore(String messaggio):** Visualizza un pop-up di errore.
- **homeAct(ActionEvent event):** Torna alla schermata principale.
- **cercaAct(ActionEvent event):** Esegue la ricerca e filtra gli studenti.
- **aggiungiAct(ActionEvent event):** Esegue l'azione di aggiunta di un nuovo studente.
- **modificaAct(ActionEvent event):** Esegue l'azione di modifica dello studente.
- **eliminaAct(ActionEvent event):** Esegue l'azione di eliminazione dello studente.

2.1.10 Classe **GestionePrestitiController**

Controller per l'interfaccia di gestione dei prestiti. Permette la visualizzazione dei prestiti attivi, la registrazione di nuovi prestiti e la registrazione delle restituzioni.

Attributi (FXML):

- **homeBtn**: Pulsante per tornare alla schermata iniziale.
- **cercaTxt:** Campo di testo per la ricerca di prestiti attivi (per matricola o ISBN).
- **cercaBtn:** Pulsante per avviare la ricerca dei prestiti.
- **tableView:** Tabella per la visualizzazione della lista dei prestiti attivi.
- **matricolaClm:** Colonna per la matricola dello studente.
- **libroClm:** Colonna per il titolo o l'ISBN del libro prestato.
- **dataPrestitoClm:** Colonna per la data di inizio del prestito.
- **scadenzaClm:** Colonna per la data prevista di restituzione.
- **matricolaTxt:** Campo di testo per l'inserimento della matricola dello studente nel nuovo prestito.
- **isbnTxt:** Campo di testo per l'inserimento dell'ISBN del libro nel nuovo prestito.
- **scadenzaTxt:** Campo di testo per l'inserimento della data di scadenza del nuovo prestito.
- **registraBtn:** Pulsante per registrare un nuovo prestito.
- **restituzioneBtn:** Pulsante per registrare la restituzione del prestito selezionato.
- **prestitiList:** Lista osservabile collegata a tableView.

- **service:** Riferimento all'istanza del BibliotecaService.

Metodi:

- **initialize(URL url, ResourceBundle rb):** Metodo di inizializzazione del controller.
- **inizializzaBinding():** Definisce le regole di disabilitazione/abilitazione dei pulsanti di registrazione e restituzione.
- **inizializzaRigaEvidenziata():** Implementa la logica per evidenziare le righe dei prestiti in ritardo.
- **inizializzaListenerSelezione():** Listener per la selezione delle righe in tabella.
- **aggiornaTabella():** Aggiorna il contenuto della tableView con l'elenco dei prestiti attivi.
- **mostraErrore(String messaggio):** Visualizza un pop-up di errore.
- **homeAct(ActionEvent event):** Torna alla schermata principale.
- **cercaAct(ActionEvent event):** Esegue la ricerca e filtra i prestiti attivi.
- **registraAct(ActionEvent event):** Esegue l'azione per registrare un nuovo prestito.
- **restituzioneAct(ActionEvent event):** Esegue l'azione per registrare la restituzione del prestito selezionato.

3 USER INTERFACE

In questa schermata vengono descritte e mostrate le interfacce che l'utente troverà a disposizione. Con ognuna di esse, potrà interagire e visionare i dati desiderati in modo rapido, efficace ed intuitivo

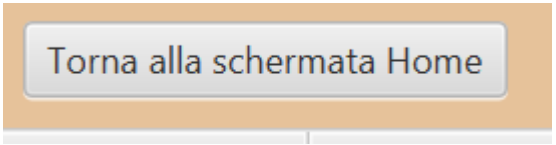
3.1 Interfaccia iniziale:

Quella mostrata in figura è la primissima schermata con la quale l'utente può interfacciarsi. In un design facile ed intuitivo, il bibliotecario vedrà in posizione alta la scritta *Biblioteca Universitaria* indicativa del software del quale sta usufruendo. Grazie alla presenza di uno sfondo a tema dai colori tenui, l'attenzione viene catturata nell'immediato, consentendo la scelta intuitiva degli unici pulsanti presenti in essa come *Gestione Libri*, *Gestione Studenti*, *Gestione Prestiti.* Ognuno di essi porta ad un collegamento con le interfacce descritte in seguito, raggiungibili in modo molto chiaro dal nome dei pulsanti

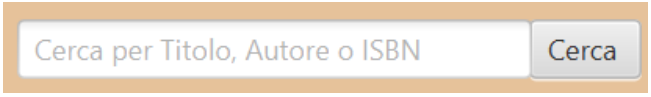
3.2 Interfaccia 1: Gestione Libri

Pulsante inerente che collega l'interfaccia

L'interfaccia mostrata, consente all'utente di gestire i vari libri della biblioteca nelle azioni che li riguardano come l'eliminazione,l'aggiunta o la modifica. Nella parte alta della pagina compare una scritta accogliente, utile a mostrare al bibliotecario la schermata nella quale si trova. Quando il bibliotecario accede, può interagire con i pulsanti presenti. Nello specifico:



Pulsante che consente di tornare alla pagina iniziale, già descritta precedentemente



Barra di ricerca: come appare nel prompt, permette di effettuare ricerche di libri nella lista per titolo,autore o ISBN

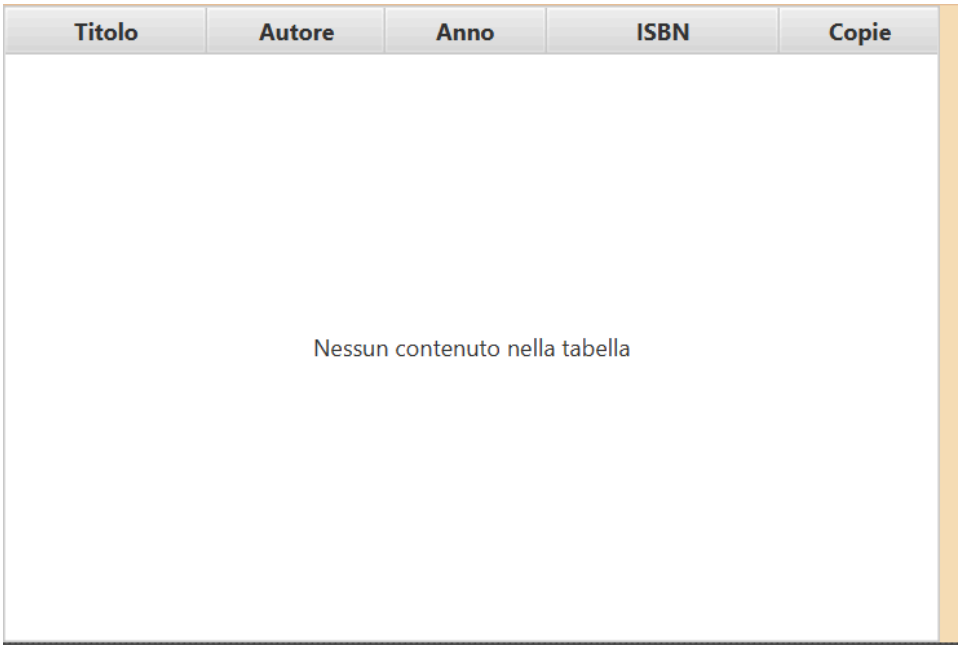
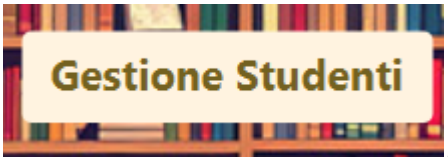


Tabella: è il contenitore della lista, le cui colonne mostrano i campi specifici del libro in modo efficace grazie alla suddivisione dei valori sottolineati in grassetto

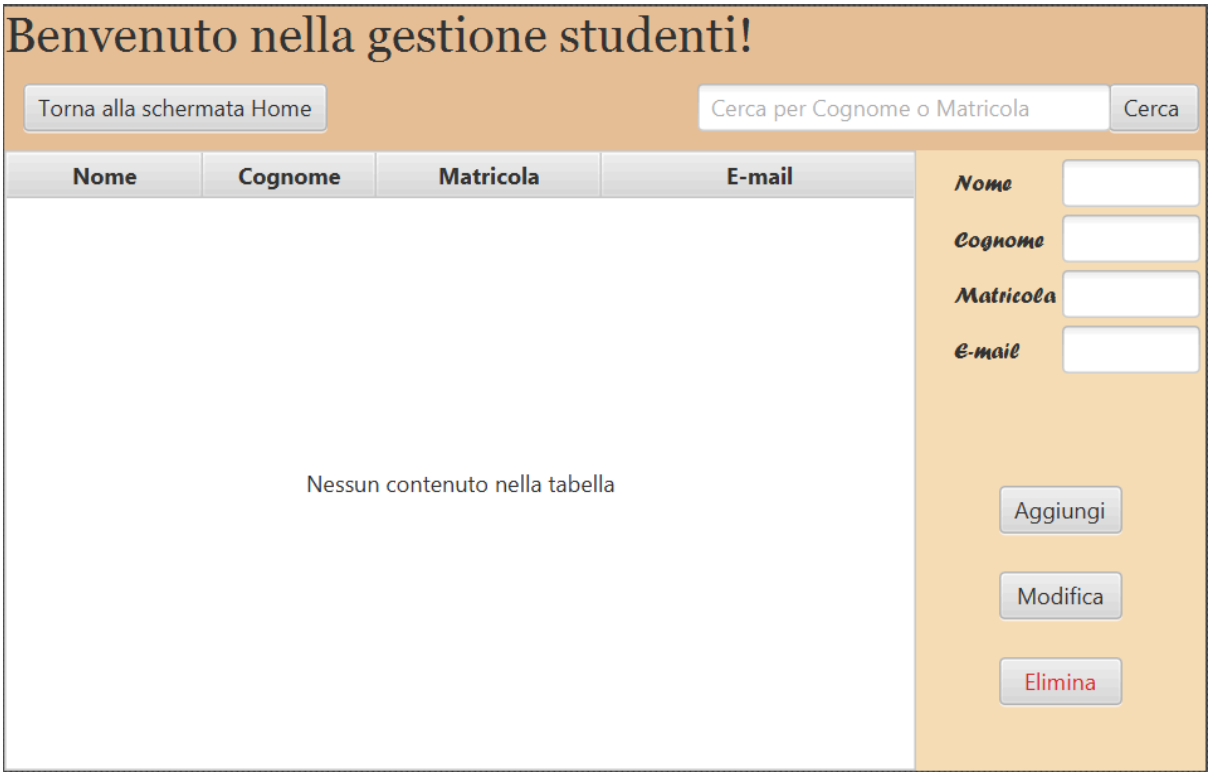
form di compilazione dei campi: Tale struttura, è posizionata nella schermata accanto alla tabella in modo da facilitare la visualizzazione dell'elemento subito dopo inserito

pulsanti di riferimento: "Aggiungi" consente di inserire nuovi elementi non appena viene compilato il form. "Modifica" consente la modifica dell'elemento selezionato in tabella e "Elimina" sottolineato in rosso per evidenziare l'importanza data la sua operazione non reversibile

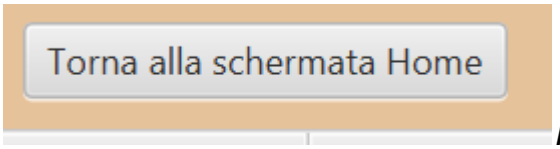
3.3 Interfaccia 2: Gestione Studenti



Pulsante inerente che collega l'interfaccia



L'interfaccia mostrata, consente all'utente di gestire i vari studenti della biblioteca nelle azioni che li riguardano come l'eliminazione,l'aggiunta o la modifica. Nella parte alta della pagina compare una scritta accogliente, utile a mostrare al bibliotecario la schermata nella quale si trova. Quando il bibliotecario accede, può interagire con i pulsanti presenti. Nello specifico:



Pulsante che consente di tornare alla pagina iniziale, già descritta precedentemente

Barra di ricerca: come appare nel prompt, permette di effettuare ricerche di studenti nella lista per Cognome o Matricola

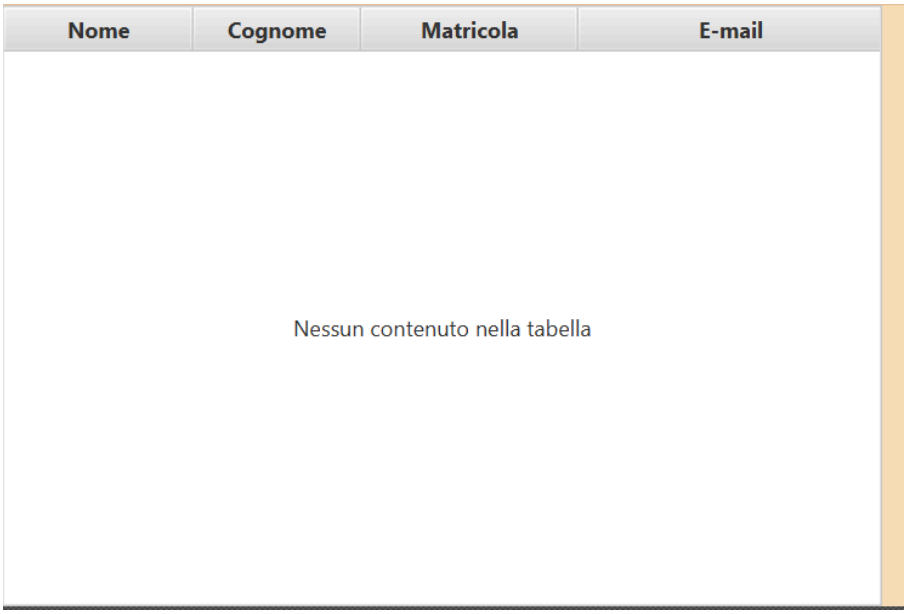


Tabella: è il contenitore della lista, le cui colonne mostrano i campi specifici dello studente in modo efficace grazie alla suddivisione dei valori sottolineati in grassetto

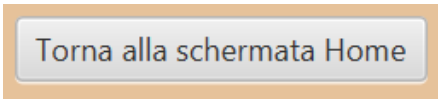
form di compilazione dei campi: Tale struttura, è posizionata nella schermata accanto alla tabella in modo da facilitare la visualizzazione dell'elemento subito dopo inserito

pulsanti di riferimento: "Aggiungi" consente di inserire nuovi elementi non appena viene compilato il form. "Modifica" consente la modifica dell'elemento selezionato in tabella e "Elimina" sottolineato in rosso per evidenziare l'importanza data la sua operazione non reversibile

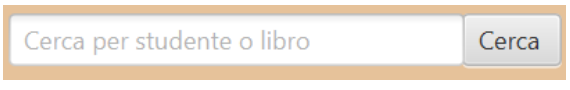
3.4 Interfaccia 3: Gestione Prestiti

Pulsante inerente che collega l'interfaccia

L'interfaccia mostrata, consente all'utente di gestire i prestiti dei libri della biblioteca nelle azioni che li riguardano come la registrazione di uno o la restituzione. Nella parte alta della pagina compare una scritta accogliente, utile a mostrare al bibliotecario la schermata nella quale si trova. Quando il bibliotecario accede, può interagire con i pulsanti presenti. Nello specifico:



Pulsante che consente di tornare alla pagina iniziale, già descritta precedentemente



Barra di ricerca: come appare nel prompt, permette di effettuare ricerche dei vari prestiti nella lista per Studente o Libro(ISBN)

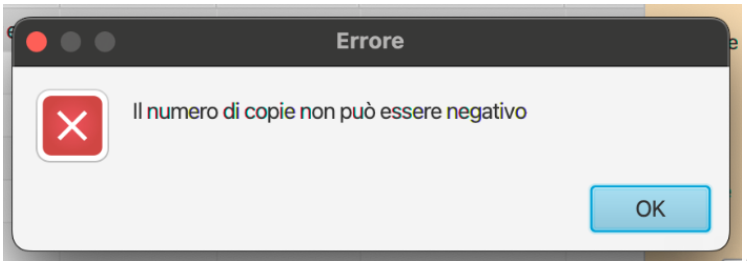
Tabella: è il contenitore della lista, le cui colonne mostrano i campi specifici del prestito in corso (o scaduto) dei libri da parte degli studenti in modo efficace grazie alla suddivisione dei valori sottolineati in grassetto

form di compilazione dei campi: Tale struttura, è posizionata nella schermata accanto alla tabella in modo da facilitare la visualizzazione dell'elemento subito dopo inserito

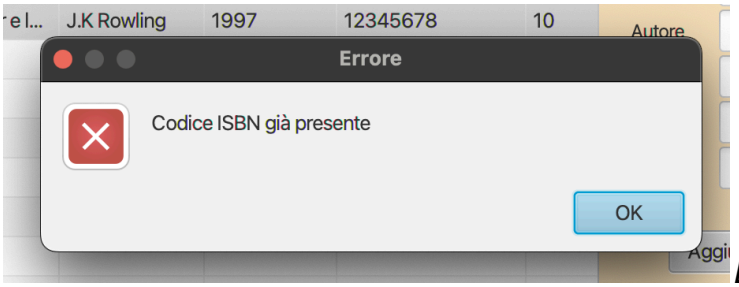
pulsanti di riferimento: "Registra Prestito" consente di inserire nuovi elementi non appena viene compilato il form. "Restituzione" consente la modifica dell'elemento selezionato in tabella, verificando il procedimento di consegna del libro preso precedentemente in prestito. E' sottolineato in verde per evidenziare l'importanza data la sua operazione non reversibile

3.5 Interfaccia 4: Casi di errore e conferma

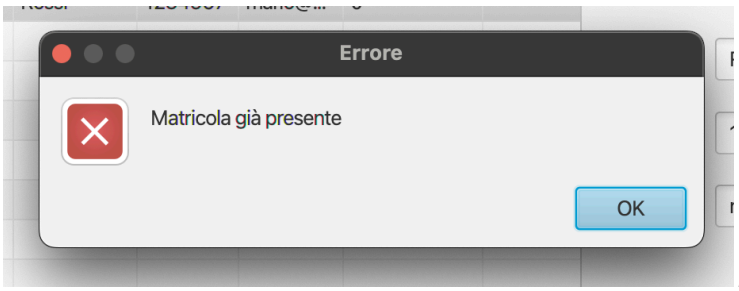
Errore: viene inserito un anno di pubblicazione in una data futura



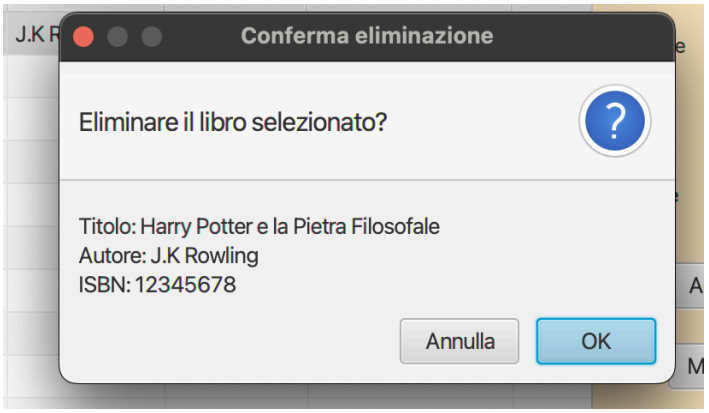
Errore: viene inserito un numero di copie negativo



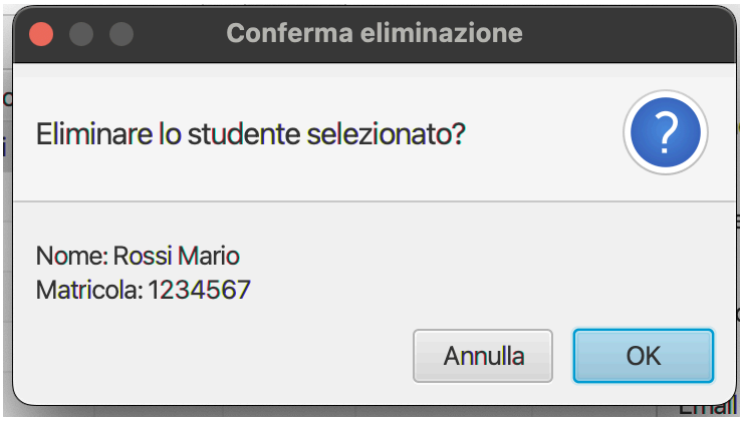
Errore: viene inserito un ISBN già presente nel sistema



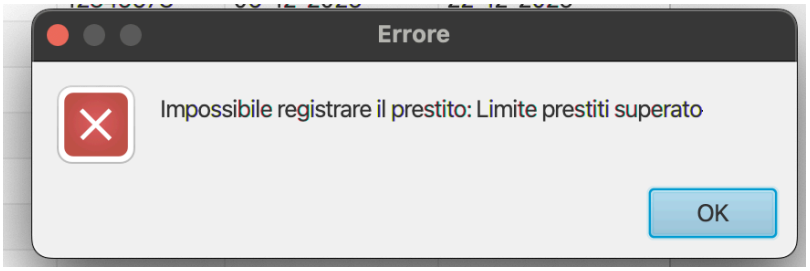
Errore: viene inserita una matricola già presente nel sistema



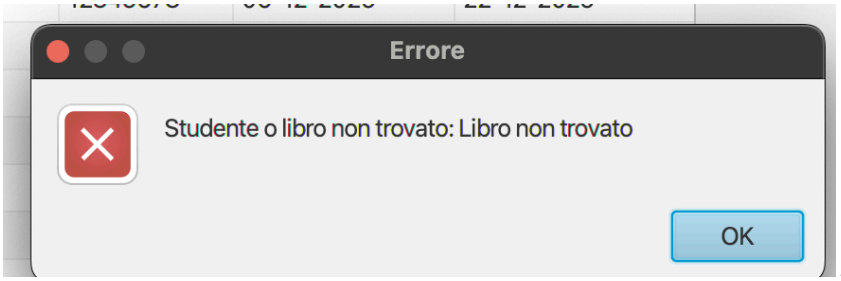
Conferma: conferma di eliminazione del libro



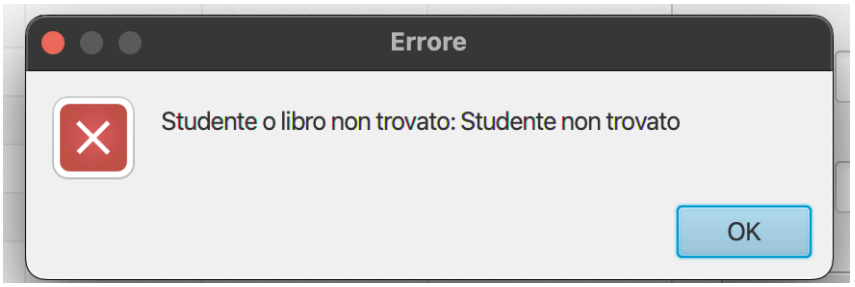
Conferma: conferma di eliminazione dello studente



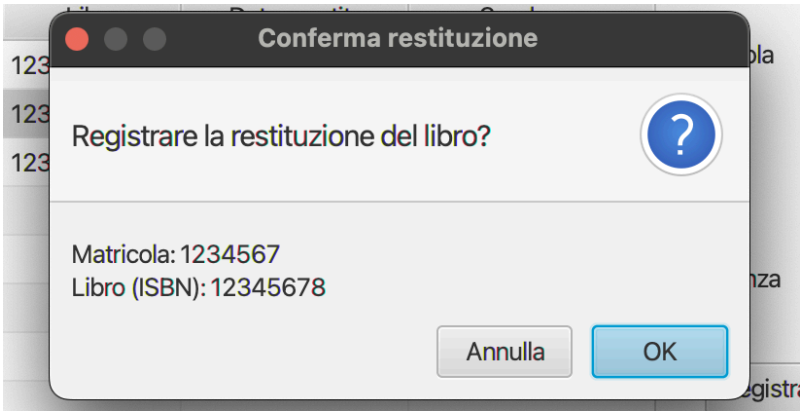
Errore: superato il limite di prestiti possibile (3)



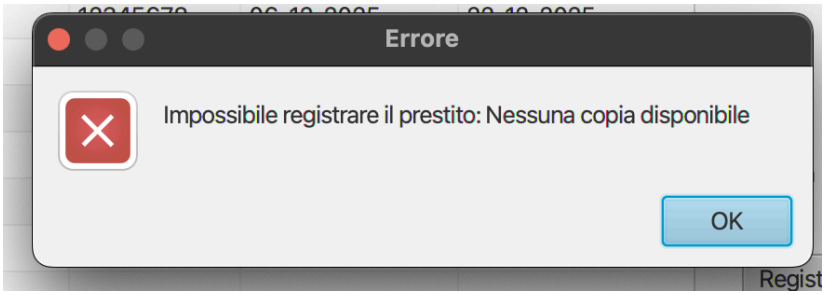
Errore: libro cercato ma non presente nel sistema



Errore: libro cercato ma non presente nel sistema



Conferma: conferma di restituzione del libro



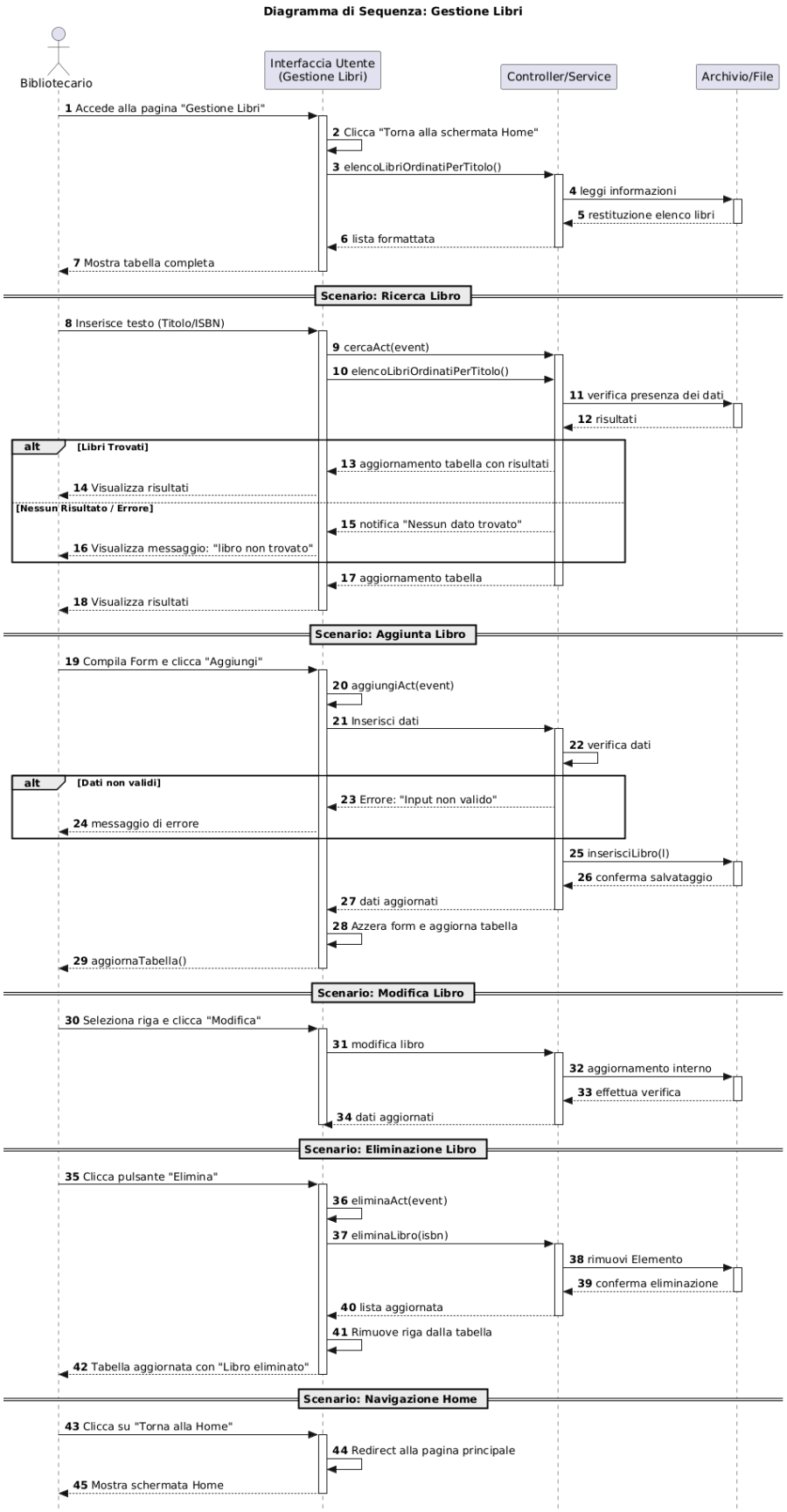
Errore: volontà di registrare un prestito per un libro il cui numero di copie disponibili è 0

4 DIAGRAMMI DI SEQUENZA

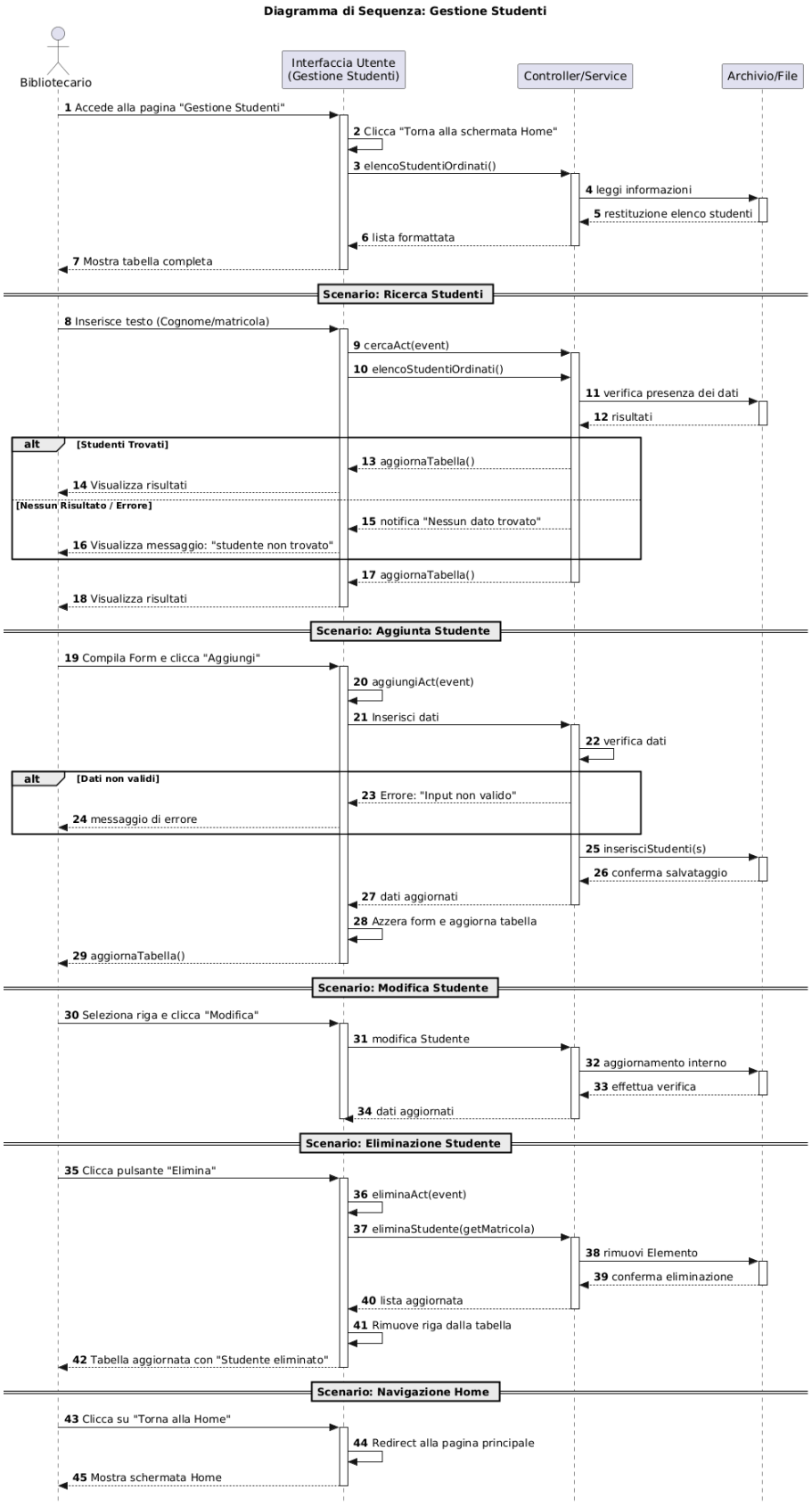
4.0 Introduzione

I diagrammi di interazione descrivono il modo in cui gruppi di oggetti collaborano tra loro. Nella realizzazione di tale sistema, è stato scelto come diagramma di riferimento il diagramma di sequenza in quanto capace di concentrarsi sulla sequenza dei messaggi tra gli oggetti, passaggio fondamentale ed opportuno per capire le corrette interazioni tra i partecipanti e le entità in gioco. Nello specifico la scelta dei partecipanti riguarda la stratificazione del sistema: per esso l'utente si interfaccia alla UI, la quale poi comunica con il controller che a sua volta si confronta con il file sul quale è salvato l'archivio, al pari di una struttura gerarchicamente suddivisa

4.1 Diagramma 1: caso -> Gestione Libri



4.2 Diagramma 2: caso -> Gestione Studenti



4.3 Diagramma 3: caso -> Gestione Prestiti