ARCHITETTURA DEL SISTEMA

Gestione libro

Questo modulo si occupa della gestione della biblioteca per quanto riguarda l'archiviazione dei libri.
In questo modulo troviamo diverse classi: quella Libro che ha lo scopo di identificare una singola entità libro,la classe Autore e la classe per la gestione della biblioteca.

Gestione utente

Anche questo modulo presenta una divisione in classi simile al modulo per la gestione dei libri e dei prestiti quindi anche qui troviamo una classe che identifica la singola entità in questo caso la classe Utente ed una classe che gestisce l'archiviazione di più utenti.

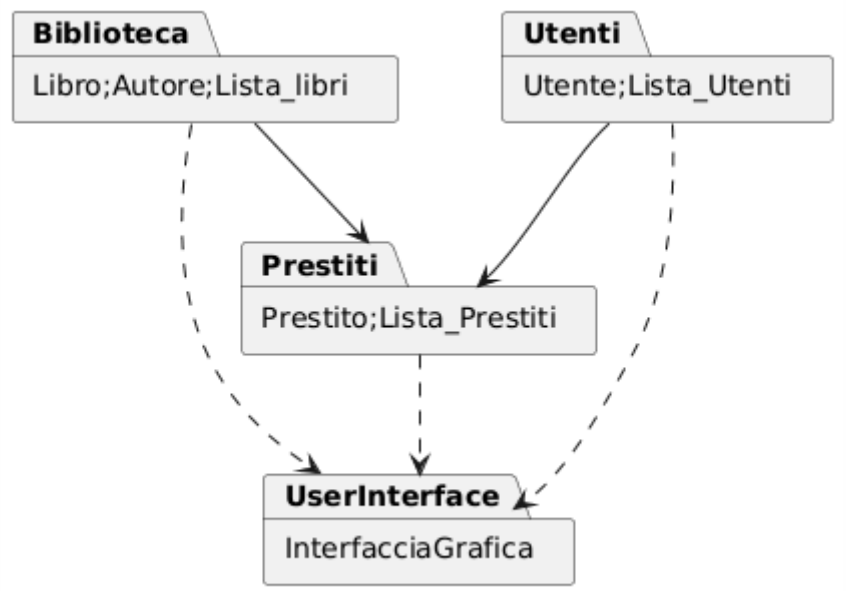
Gestione prestiti

Il modulo ha lo scopo di gestire la parte che riguarda i prestiti effettuati.
Segue una divisione in 2 classi quella che gestisce l'entità prestito e quella che gestisce la collection di Prestito.

Interfaccia GUI

Il modulo gestisce tutte le interazioni con il Bibliotecario tramite interfaccia grafica.
Il modulo viene diviso in diverse classi per mantenere la divisione in moduli descritta in precedenza.

Dipendenze



Come mostra il grafico tutte le classi comunicano con la User Interface che effettua le chiamate dei metodi e acquisisce i dati inseriti dal Bibliotecario.
Il modulo Utenti e Biblioteca no comunicano direttamente ma passano dati al modulo Prestiti.

MODELLO STATICO

Classe Libro

La classe libro come detto in precedenza gestisce l'entità libro memorizzando il titolo del libro l'anno di pubblicazione, l'ISBN, una lista di oggetti Autore, il numero di copie disponibili e una flag che permette ad un libro di non essere eliminato nonostante tutte le copie siano state prestate.
Inoltre ci sono i tre comparatori per gestire i diversi ordinamenti della lista dei libri

Attributi

private String titolo
private List<Autore> autori
private String ISBN
private int anno
private boolean disponibile
private int copie
public static final Comparator<Libro> TITOLO
public static final Comparator<Libro> ISBN
public static final Comparator<Libro> PRIMO_AUTORE

Metodi

Tralasciando i metodi funzionali la classe mette a disposizione i metodi per la visualizzazione e la modifica di tutti gli attributi e inoltre permette la creazione di una collection di Autore per la gestione di più autori e tutti i metodi che permettono la modifica l'aggiunta e la rimozione di oggetti da suddetta collection.
public Libro(String titolo,List<Autore> autori,String ISBN,int anno,int copie)
public String getTitolo()
public void setTitolo(String titolo)
public List<Autore> getAutori()
public void setAutori(List<Autore> autori)
public void modificaAutore(Autore autore)
public String getISBN()
public void setISBN(String ISBN)
public int getAnno()
public void setAnno(int anno)
public int getCopie()
public void setCopie(int copie)
public void aggiungiAutore(Autore a)
public void rimuovi Autore(Autore a)
public boolean getDisponibile()
public void setDisponibile(boolean disponibile)
public boolean equals(Object o)
public void OrdinaAutori()

Classe Autore

La classe Autore è una classe creata come scelta progettuale per semplificare la presenza di diversi autori di un solo libro.
In questa classe sono presenti solo il nome, il cognome e un comparatore per permettere l'ordinamento sul cognome nella classe Libro

Attributi

private String nome
private String cognome
public static final Comparator<Autore> COGNOME

Metodi

prevede i metodi per la visualizzazione e la modifica di un Autore
public Autore (String nome,String cognome)
public String getNome()
public String setNome(String nome)
public String getCognome()
public String setCognome(String cognome)
public boolean equals(Object o)

Classe Lista_Libri

In questa classe viene gestita la vera e propria struttura della biblioteca dal punto di vista dello storage dei libri.
Per fare questo è sufficiente un solo attributo la collection che gestisce l'aggregazione di libri

Attributi

private List<Libro> libri

Metodi

In questa classe sono presenti tutti i metodi per offrire i servizi espressi nei casi d'uso
public Libro aggiungiLibro(Libro libro)
public Libro rimuoviLibro(Libro libro)
public Libro modificaLibro(Libro libro)
public List<Libro> cercaPerTitolo(String str)
public List<Libro> cercaPerISBN(String str)
public List<Libro> cercaPerAutore(String str)
public boolean AutoreTrovato(List<Autore> autori, String str)
public void ordina(Comparator<Libro> cmp)
public boolean checkISBN(String ISBN)
public void salvataggioLibri(String nomefile)
public Lista_Libri letturaLibri(String nomefile)

Classe Utente

Questa classe gestisce la singola entità utente, non comprendendo il Bibliotecario.
Come da requisiti l'utente viene identificato da 4 attributi:il nome,il cognome,la matricola e la mail universitaria

Attributi

private String nome
private String cognome
private int matricola
private String mail

Metodi

Comprende i metodi funzionali
public Utente(String nome,String cognome,int matricola,private String mail)
public String getNome()
public void setNome(String nome)
public String getCognome()
public void setCognome(String cognome)
public int getMatricola()
public void setMatricola(int matricola)
public String getMail()
public void setMail(String mail)
public boolean equals(Object o)

Classe Lista_Utenti

La Lista_Utenti gestisce tutte le operazioni da effettuare sulla collection di Utente

Attributi

private List<Utente> utenti

Metodi

public void aggiungiUtente(Utente utente)
public void rimuoviUtente(Utente utente)
public Utente modificaUtente(Utente utente)
public List<Utente> getUtenti()
public void setUtenti(List<Utente> utenti)
public List<Utente> cercaPerMatricola(String str)
public List<Utente> cercaPerEmail(String str)public List<Utente> cercaPerNome(String str)
public List<Utente> cercaPerCognome(String str)
public boolean checkMatricola(Int matricola)
public void salvataggioUtenti(String nomefile)
public Lista_Utenti letturaUtenti(String nomefile)

Classe Prestito

Questa classe identifica un singolo prestito associando un oggetto di tipo Utente ad un ISBN di un libro inoltre aggiunge la data di restituzione del prestito e altri 2 attributi inseriti come scelta progettuale:L'ID univoco che identifica un prestito e il comparator usato da Lista_Prestiti per ordinare in base alla scadenza

Attributi

private String ISBN
private Utente utente
private LocalDate dataScadenza
private final int ID
private static int counter=0
private final static
Comparator<Prestito>SCADENZA=Comparator.comparing(Prestito::getDataScadenza)

Metodi

I metodi implementati in questa classe prevedono la visione e la modifica degli attributi e i metodi funzionali
public Prestito(String ISBN,int Matricola,LocalDate dataScadenza)
public Comparator<Prestito> getComp()
public Libro getISBN()
public void setISBN(String ISBN)
public int getMatricola()
public void setMatricola(int Matricola)
public int getID()
public LocalDate getDataScadenza()
public void setDataScadenza(LocalDate dataScadenza)
public boolean equals(Object o)

Classe Lista_Prestiti

Questa classe gestisce la collection di Prestito e tutte le funzioni ad esso collegato

Attributi

private List<Prestito> prestiti

Metodi

Tutti i metodi presenti in questa classe si occupano di gestire l'aggiunta la rimozione e l'ordinamento dei prestiti e dei ritardi
public List<Prestito> getPrestiti()
public void setPrestiti(List<Prestito> prestiti)
public String aggiungiPrestito(Prestito p)
public String rimuoviPrestito(Prestito p)
public List<Prestito> aggiornaRitardi()
public List<Prestito> cercaPerISBN(String ISBN)
public List<Prestito> cercaPerID(int ID)

public List<Prestito> cercaPerMatricola(int Matricola)
public List<Prestito> cercaPerData(LocalDate dataScadenza)
public void OrdinaLibriPerScadenza(Comparstor<Prestito> cmp)
public String ModificaPrestito(Prestito p, Prestito l)
public void salvataggioPrestiti(String nomefile)
public void letturaPrestiti(String nomefile)
public boolean checkUtente(Prestito p)

ControllerPortale

Attributi

private Button btnEntra

Metodi

public void initialize()
public void handleEntra(ActionEvent event)

ControllerHome

Attributi

private Button GestioneLibri
private Button GestionePrestiti
private Button GestioneUtenti
private Button btnPortale
private AnchorPane contentPane

Metodi

public void initialize()
public void onBackToPortale()
public void onGestioneLibri()
public void onGestionePrestiti()
public void onGestioneUtenti()
public void onGestioneRitardi()

ControllerDashboard

Attributi

private Button btnPortale

Metodi

public void onBackToPortale()

ControllerGestioneLibri

Attributi

private Button btnModificaLibro;
private Button btnRimuoviLibro;
private Button btnVisualizzaLibri;

private Button btnAggiungiLibro;
private TextField txtRicercaLibro;
private TableView<?> tabellaLibri;
private Button btnCercaLibro;
private Button btnResetFiltroLibri;
private Label lblConteggioLibri;

Metodi

---

public void initialize()
private void onBack()
private void handleVisualizzaLista(ActionEvent event)
private void handleAggiungiLibro(ActionEvent event)
private void onModificaLibro(ActionEvent event)
private void onRimuoviLibro(ActionEvent event)
private void handleCercaLibro(ActionEvent event)
private void handleResetFiltroLibri(ActionEvent event)

ControllerListaLibri

Attributi

---

private TableView<Libro> tabellaLibri
private TableColumn<Libro,String> colTitolo
private TableColumn<Libro,String> colAutore
private TableColumn<Libro,Integer> colAnno
private TableColumn<Libro,String> colISBN
private TableColumn<Libro,Integer> colCopieTotali
private TableColumn<Libro,Integer> colCopieDisponibili
private Button btnModificaLibro;
private Button btnRimuoviLibro;
private Button btnChiudi;

Metodi

---

public void initialize()
private void onModificaLibro(ActionEvent event)
private void onRimuoviLibro(ActionEvent event)
private void handleChiudi()

ControllerModificaLibro

Attributi

---

private Button btnModificaLibro;
private TextField txtTitolo;
private TextField txtAutore;
private TextField txtAnno;
private TextField txtISBN;

Metodi

---

public void initialize()
private void onModificaLibro(ActionEvent event)

private void onCancella()
private void chiudiFinestra()

ControllerPopup

Attributi

private Label lblIcona;
private Label lblTitolo;
private Label lblMessaggio;
private Button btnOk;

Metodi

public void setSuccesso(String messaggio)
public void setErrore(String messaggio)
private void onOk()

ControllerAggiungiLibro

Attributi

private TextField txtTitolo;
private TextField txtAutore;
private TextField txtAnno;
private TextField txtISBN;
private Button btnAggiungi;
private ListView ListViewLibri;

Metodi

public void initialize()
private void onAggiungi()
private void onCancella()
private void chiudiFinestra()
private void mostraPopupSuccesso(String messaggio)
private void mostraPopupErrore(String messaggi

ControllerGestioneUtenti

Attributi

private TextField txtRicercaUtente;
private Button btnCercaUtente;
private Button btnResetFiltroUtenti;
private Button btnVisualizzaUtenti;
private Button btnAggiungiUtente;
private Label lblConteggioUtenti;
private TableView<?> tabellaUtenti;

Metodi

public void init()
private void onBack()
private void handleAggiungiUtente(ActionEvent event)
private void handleVisualizzaUtenti(ActionEvent event)
private void handleCercaUtente(ActionEvent event)
private void handleResetFiltroUtenti(ActionEvent event)

ControllerListaUtenti

Attributi

private TableView<Utente> tabellaUtenti
private TableColumn<Utente,String> colNome
private TableColumn<Utente,String> colCognome
private TableColumn<Utente,Integer> colMatricola
private TableColumn<Utente,String> colEmail
private Button btnRimuoviUtente;
private Button btnModificaUtente;
private Button btnChiudi;

Metodi

public void initialize()
private void onModificaUtente(ActionEvent event)
private void onRimuoviUtente(ActionEvent event)
private void handleChiudi()

ControllerModificaUtente

Attributi

private TextField txtNome;
private TextField txtCognome;
private TextField txtMatricola;
private TextField txtEmail;
private Button btnModificaUtente;

Metodi

public void init()
private void onModificaUtente(ActionEvent event)
private void onCancella()
private void chiudiFinestra()

ControllerAggiungiUtente

Attributi

private TextField txtNome;
private TextField txtCognome;
private TextField txtMatricola;
private TextField txtEmail;

private Button btnAggiungi;
private Button btnCancella;

Metodi

public void init()
private void onAggiungi()
private void onCancella()
private void mostraPopupSuccesso(String messaggio)
private void mostraPopupErrore(String messaggio)
private void chiudiFinestra()

ControllerAggiungiUtente

Attributi

private TextField txtNome;
private TextField txtCognome;
private TextField txtMatricola;
private TextField txtEmail;
private Button btnAggiungi;
private Button btnCancella;

Metodi

public void init()
private void onAggiungi()
private void onCancella()
private void mostraPopupSuccesso(String messaggio)
private void mostraPopupErrore(String messaggio)
private void chiudiFinestra()

ControllerAggiungiUtente

Attributi

private TextField txtNome;
private TextField txtCognome;
private TextField txtMatricola;
private TextField txtEmail;
private Button btnAggiungi;
private Button btnCancella;

Metodi

public void init()
private void onAggiungi()
private void onCancella()
private void mostraPopupSuccesso(String messaggio)
private void mostraPopupErrore(String messaggio)
private void chiudiFinestra()

ControllerGestionePrestiti

Attributi

private Button btnVisualizzaPrestiti;
private Button btnAggiungiPrestito;
private TextField txtRicercaPrestito;
private Button btnVisualizzaRitardi;

private Button btnCercaPrestito;
private Button btnResetFiltroPrestiti;

Metodi

public void init()
private void onBack()
private void handleVisualizzaListaPrestiti(ActionEvent event)
private void handleAggiungiPrestito(ActionEvent event)
private void handleVisualizzaRitardi(ActionEvent event)
private void handleCercaPrestito(ActionEvent event)
private void handleResetFiltroPrestiti(ActionEvent event)

ControllerListaPrestiti

Attributi

private Button btnModificaPrestito;
private ListView<Utente> utenti
private TextField txtISBN;
private TextField txtMatricola;
private TextField txtEmail;
private DatePicker dateScadenza;

Metodi

public void init()
private void onModificaPrestito(ActionEvent event)
private void onRimuoviPrestito(ActionEvent event)
private void handleChiudi()

ControllerModificaPrestito

Attributi

private Button btnModificaPrestito;
private ListView<Utente> utenti
private TextField txtISBN;
private TextField txtMatricola;
private TextField txtEmail;
private DatePicker dateScadenza;

Metodi

public void init()
private void onAggiungi()
private void onCancella()
private void mostraPopupSuccesso(String messaggio)
private void mostraPopupErrore(String messaggio)
private void chiudiFinestra()

ControllerAggiungiPrestito

Attributi

private TextField txtISBN;
private ListView<Utente> utenti
private TextField txtMatricola;

private TextField txtEmail;
private TextField txtID;
private DatePicker dateScadenza;
private Button btnAggiungi;

Metodi

public void init()
private void onAggiungi()
private void onCancella()
private void mostraPopupSuccesso(String messaggio)
private void mostraPopupErrore(String messaggio)
private void chiudiFinestra()

ControllerListaRitardi

Attributi

private TableView<Prestito> tabellaRitardi
private TableColumn<Prestito,String> colISBN
private TableColumn<Prestito,Utente> colUtente
private TableColumn<Prestito,Integer> colID
private TableColumn<Prestito,LocalDate> colDataScadenza
private TableColumn<Prestito,LocalDate> colDataPrestito
private Button btnChiudi;
private TextField txtRicercaRitardo;
private Button btnCercaRitardo;
private Button btnResetRitardi;
private Button btnModificaRitardo;
private Button btnRimuoviRitardo;

Metodi

public void init()
private void handleChiudi()
private void handleCercaRitardo(ActionEvent event)
private void handleResetFiltroRitardi(ActionEvent event)
private void onModificaRitardo(ActionEvent event)
private void onRimuoviRitardo(ActionEvent event)

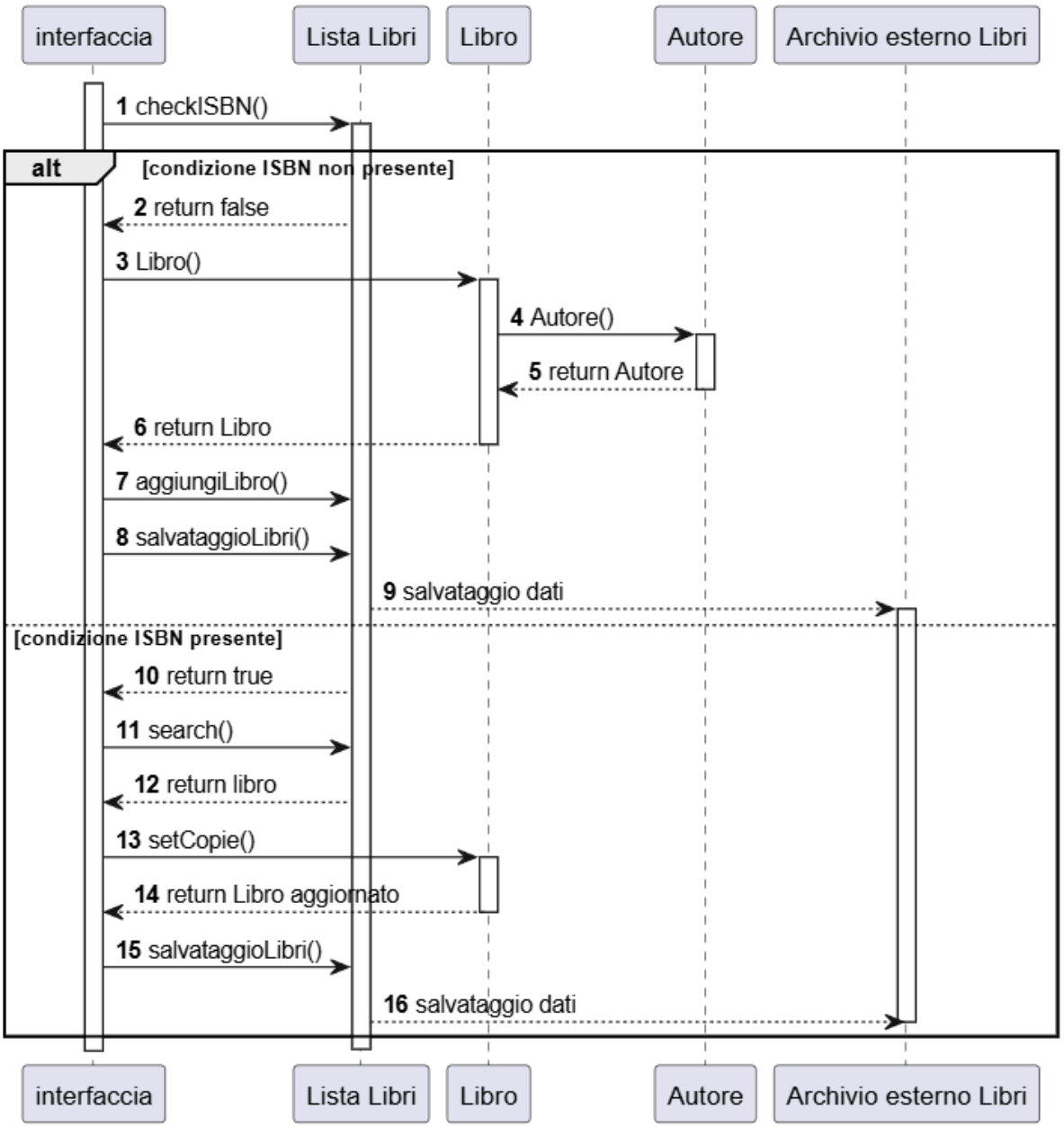
COESIONE E ACCOPPIAMENTO

- I moduli **Biblioteca, Utenti e Prestiti** presentano una coesione in linea di massima **Comunicazionale** dato che ogni modulo lavora sulla sua struttura anche se alcune funzionalità interne presentano una coesione sequenziale fra di loro.

- La User Interface ha una coesione **Sequenziale** per la maggior parte delle funzionalità mentre altre funzionalità prevedono una coesione temporale.
- I moduli Biblioteca Utenti non presentano nessun accoppiamento diretto ma entrambi hanno una accoppiamento **per dati** con Prestit.
- La User Interface sfruttando le funzionalità intrinseche di java prevedono solo un passaggio di dati quindi presentano un **accoppiamento per dati** per l'appunto.

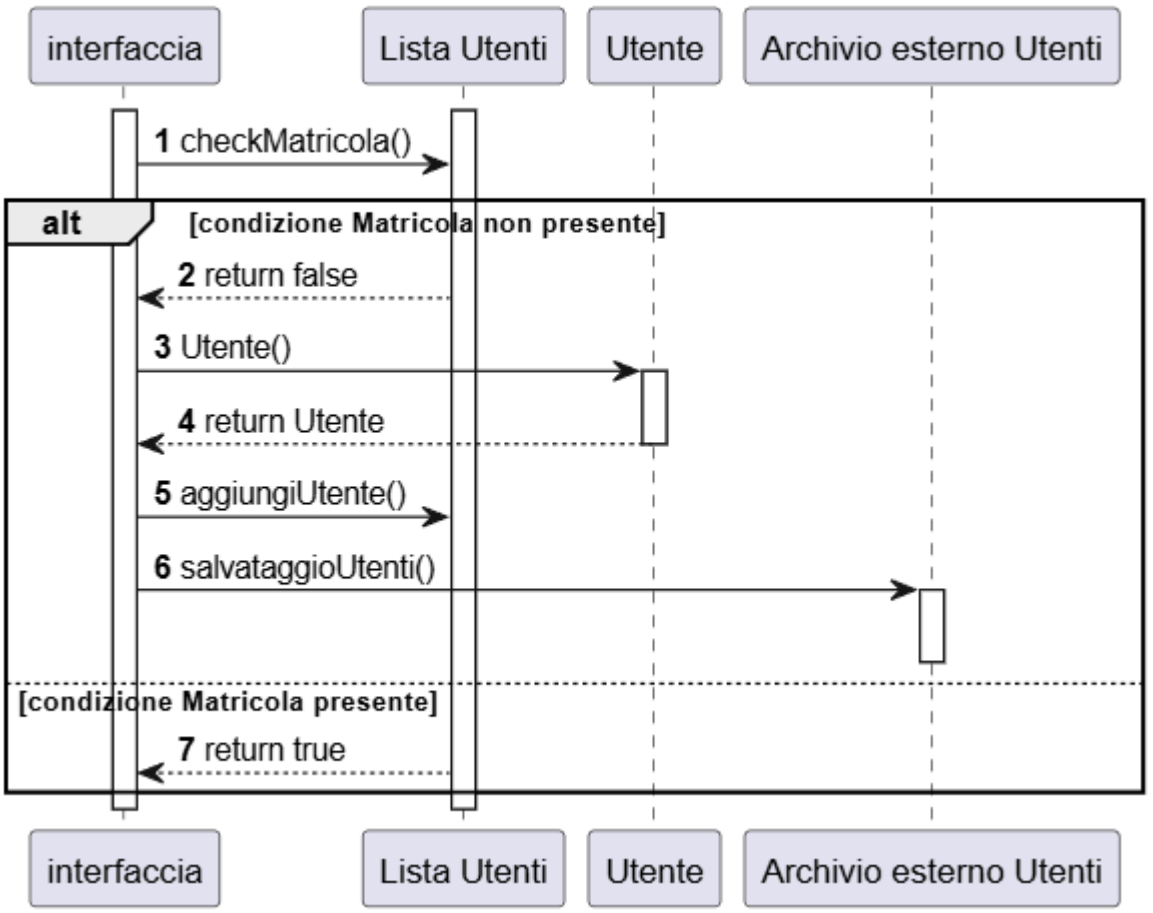
MODELLO DINAMICO

Aggiunta Libro



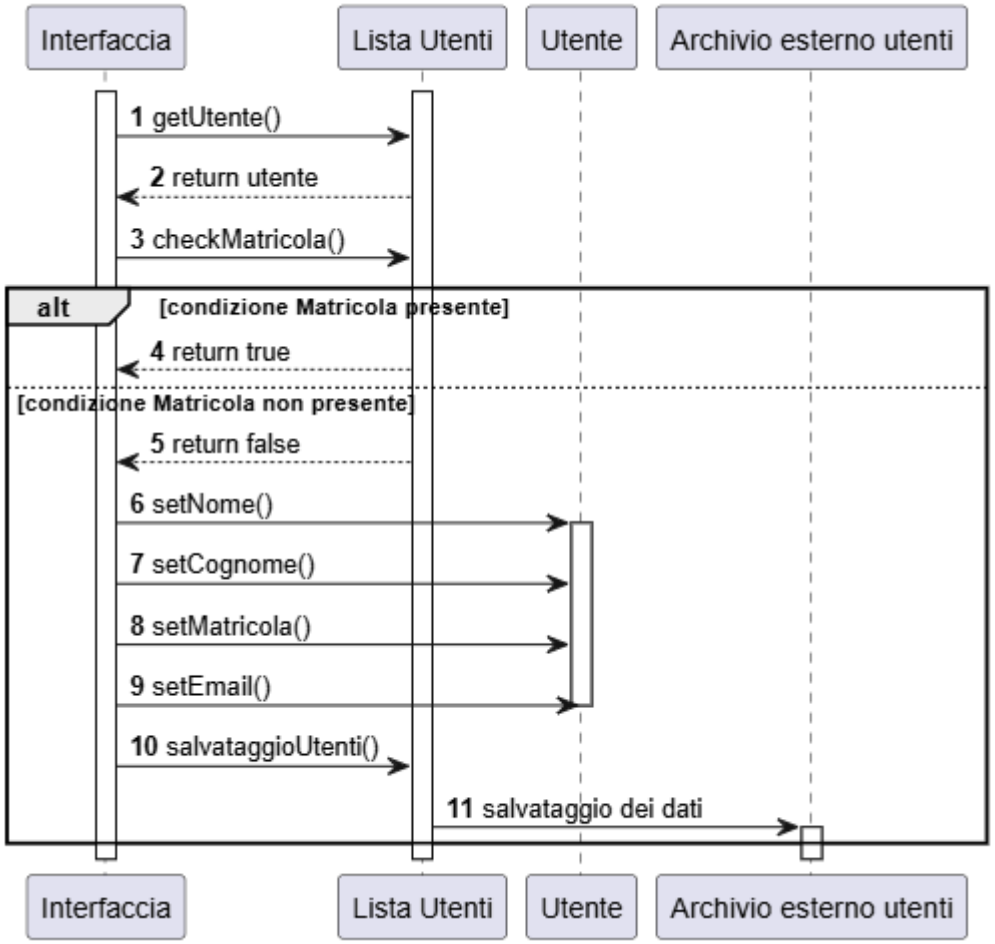
Il grafico rappresenta una delle funzionalità principali del modulo Biblioteca l'inserimento di un libro in archivio locale con relativa copia nell'archivio esterno. Il diagramma esegue un controllo sull' ISBN, in caso di ISBN presente viene fatta una modifica sul numero di copie mentre se l'ISBN non è presente viene creato un nuovo oggetto Libro e aggiunto alla lista.

Aggiungi Utente



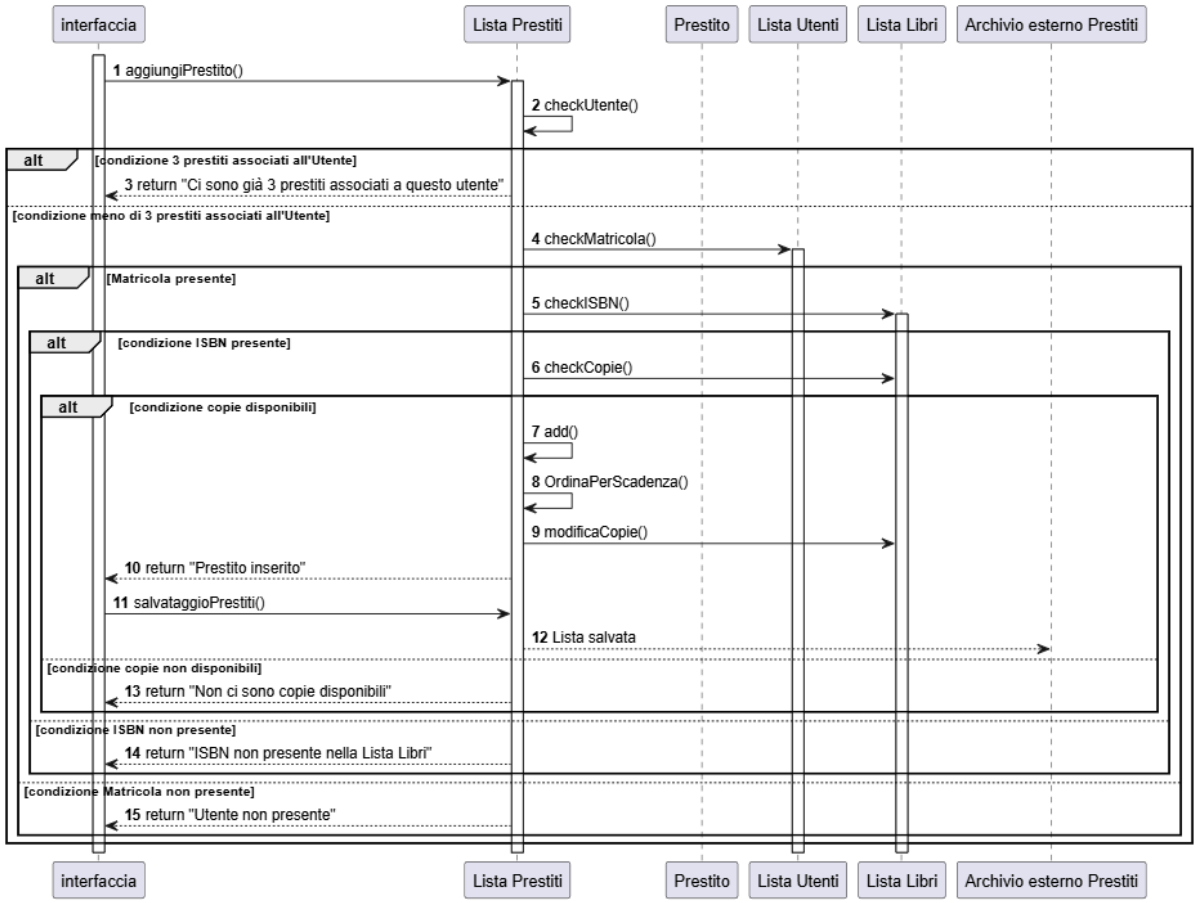
L'aggiunta di un utente avviene in maniera similare all'aggiunta di un libro.
Il check in questo caso avviene sulla matricola ed in caso di matricola presente viene notificato che l'utente è già presente in archivio.

Modifica utente



La modifica di un utente si divide in 2 parti principali la ricerca la ricerca dell'utente e i relativi metodi setter per modificare i dati inseriti

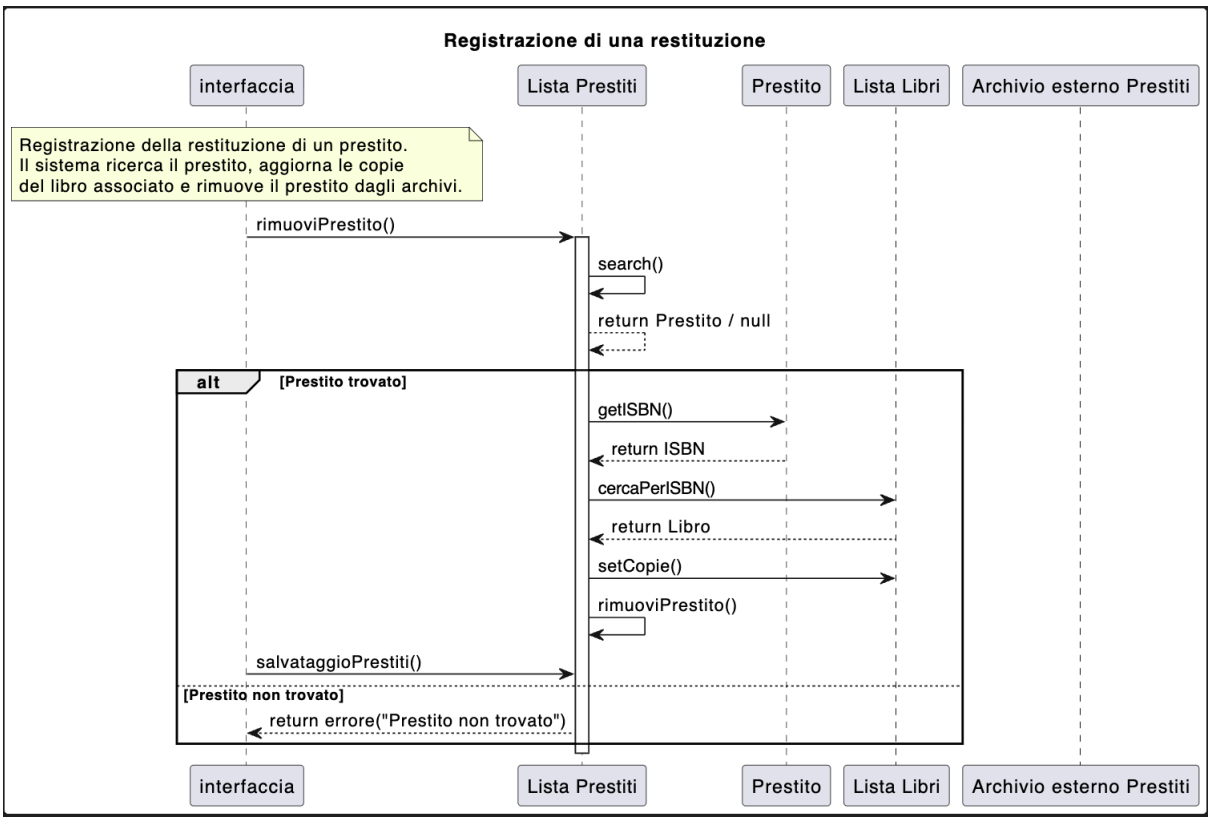
Aggiungi prestito



I prestiti uniscono un ISBN ad un utente ed una data quindi come prima cosa vengono eseguiti i controlli per avere la certezza che i dati inseriti siano presenti. Poi vengono eseguiti i controlli sull' Utente per assicurarsi che non abbia più di 3 prestiti attivi. Nel caso in cui tutti i controlli siano andati a buon fine viene aggiunto il prestito sia in locale che nel archivio esterno.

Elimina prestito

L'eliminazione di un prestito prevede la ricerca del prestito e poi la rimozione da entrambi gli archivi(locale e esterno).

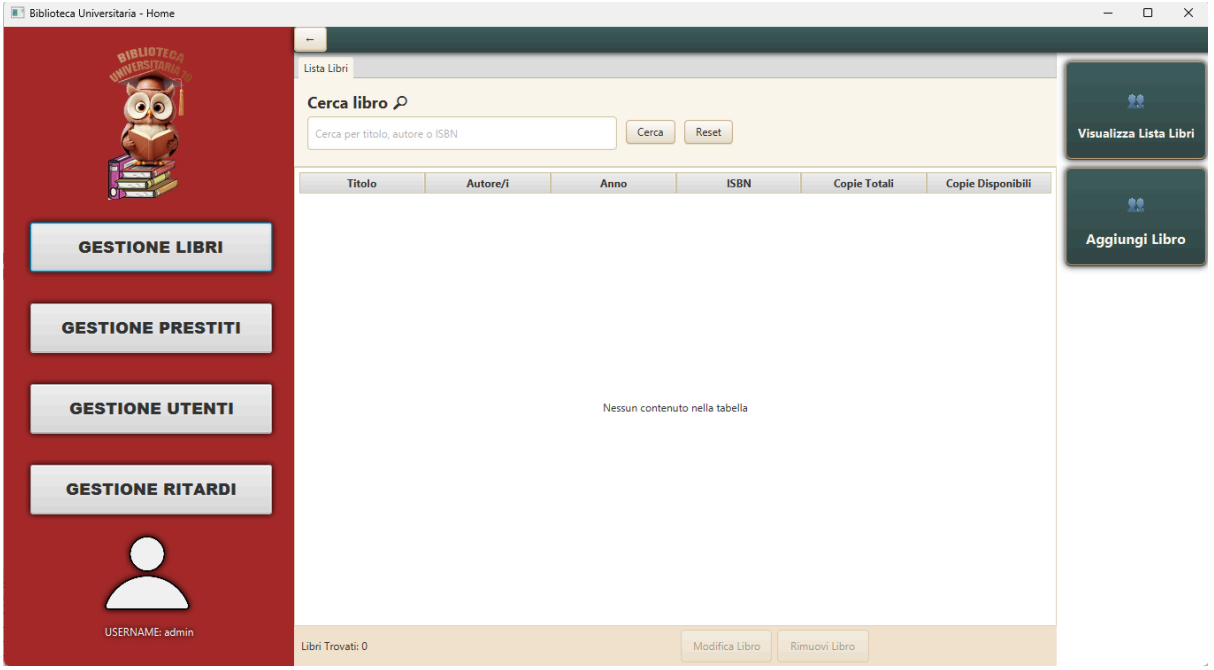

Registrazione di una restituzione
interfaccia
Lista Prestiti
Prestito
Lista Libri
Archivio esterno Prestiti
Registrazione della restituzione di un prestito.
Il sistema ricerca il prestito, aggiorna le copie
del libro associato e rimuove il prestito dagli archivi.
rimuoviPrestito()
search()
return Prestito / null
alt
[Prestito trovato]
getISBN()
return ISBN
cercaPerISBN()
return Libro
setCopie()
rimuoviPrestito()
salvataggioPrestiti()
[Prestito non trovato]
return errore("Prestito non trovato")
interfaccia
Lista Prestiti
Prestito
Lista Libri
Archivio esterno Prestiti

DESIGN INTERFACCIA UTENTE

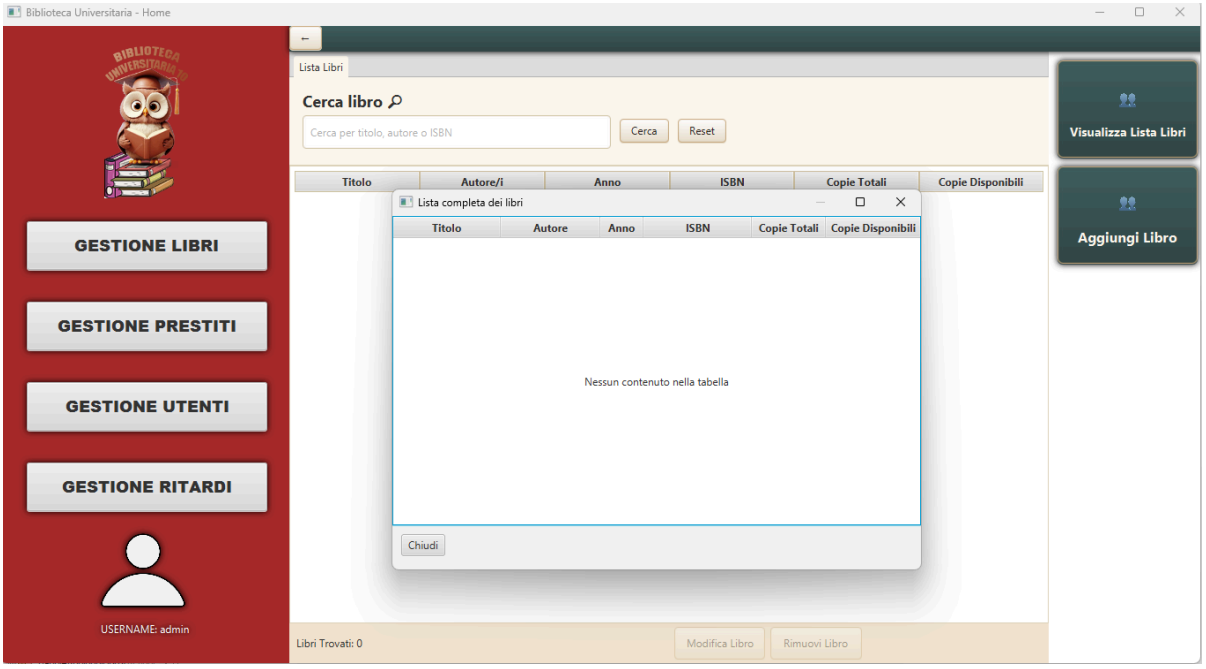
Accesso al sistema

Home

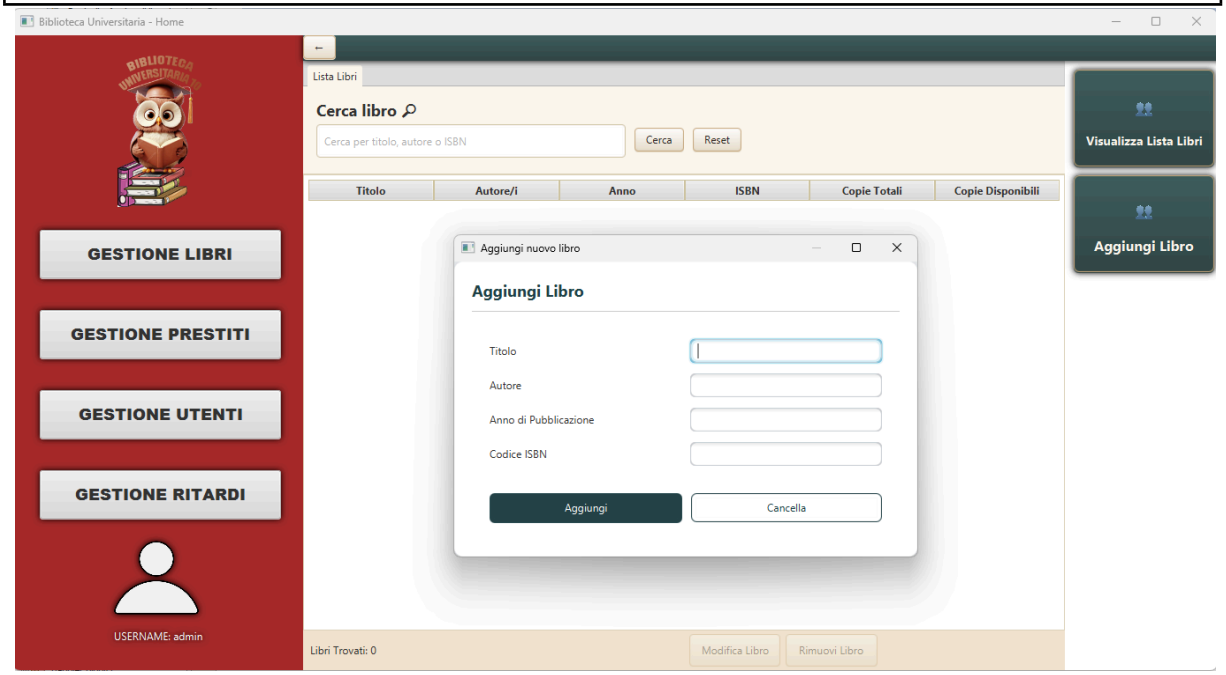
Interfaccia generale per la gestione libri



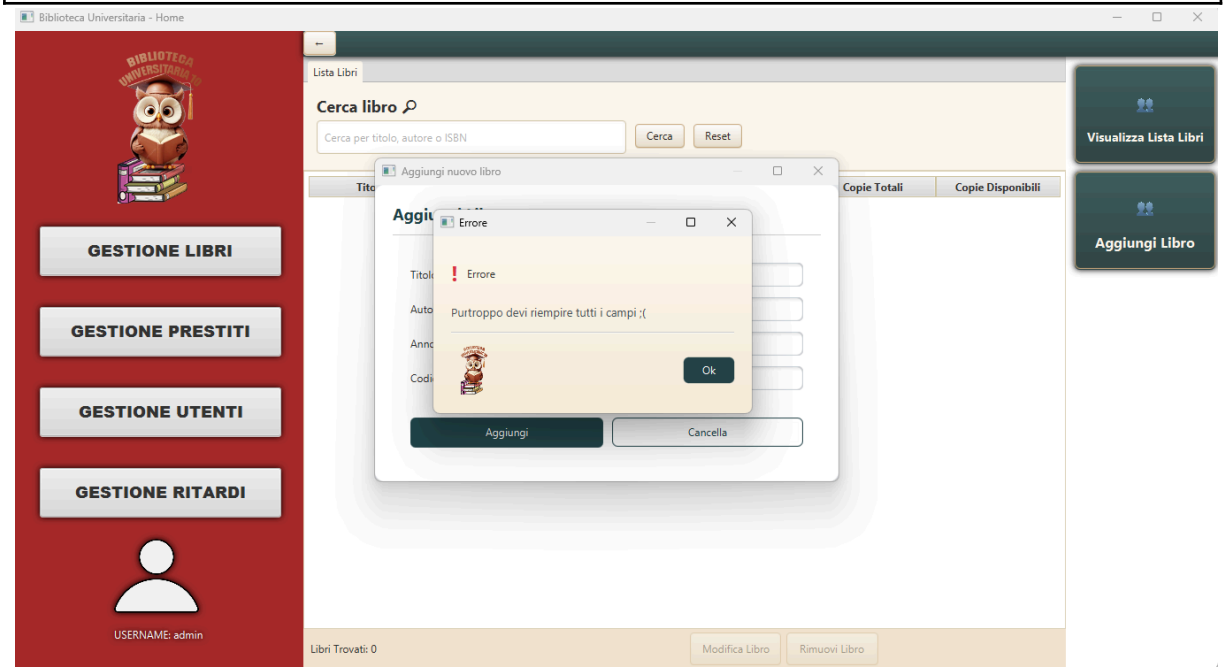
Visualizza libri



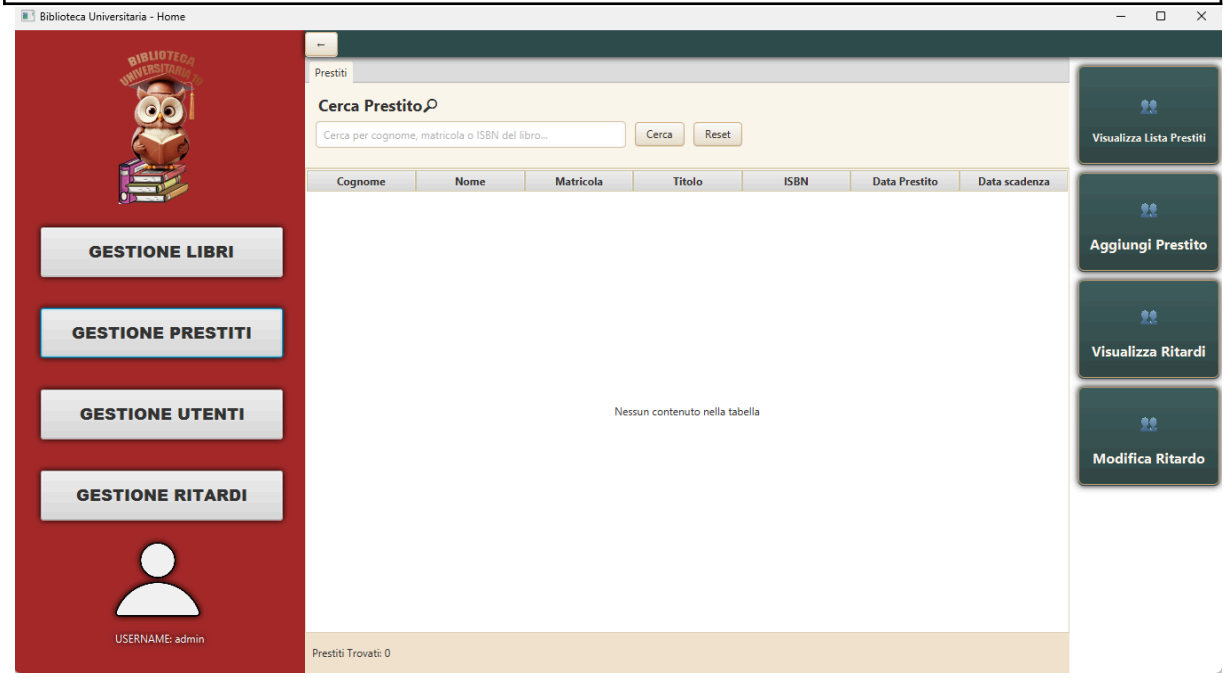
Aggiungi libro



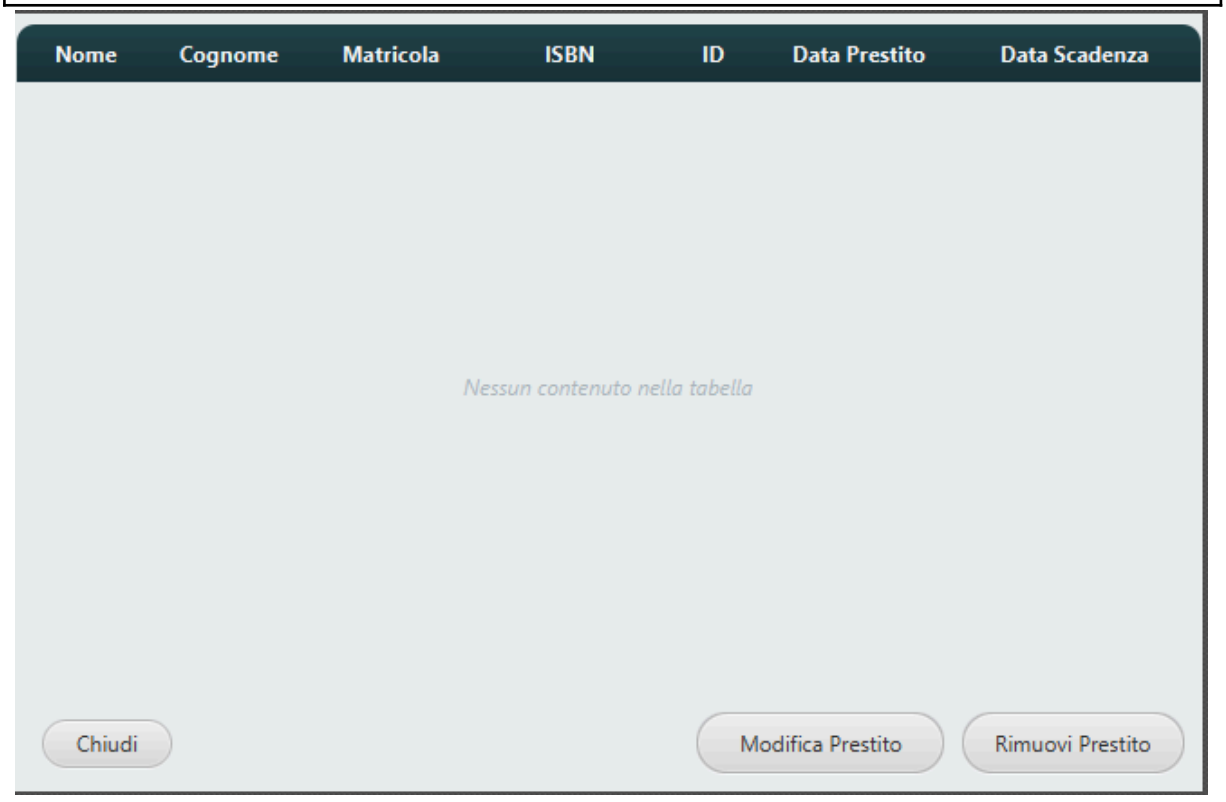
Pop-up di errore



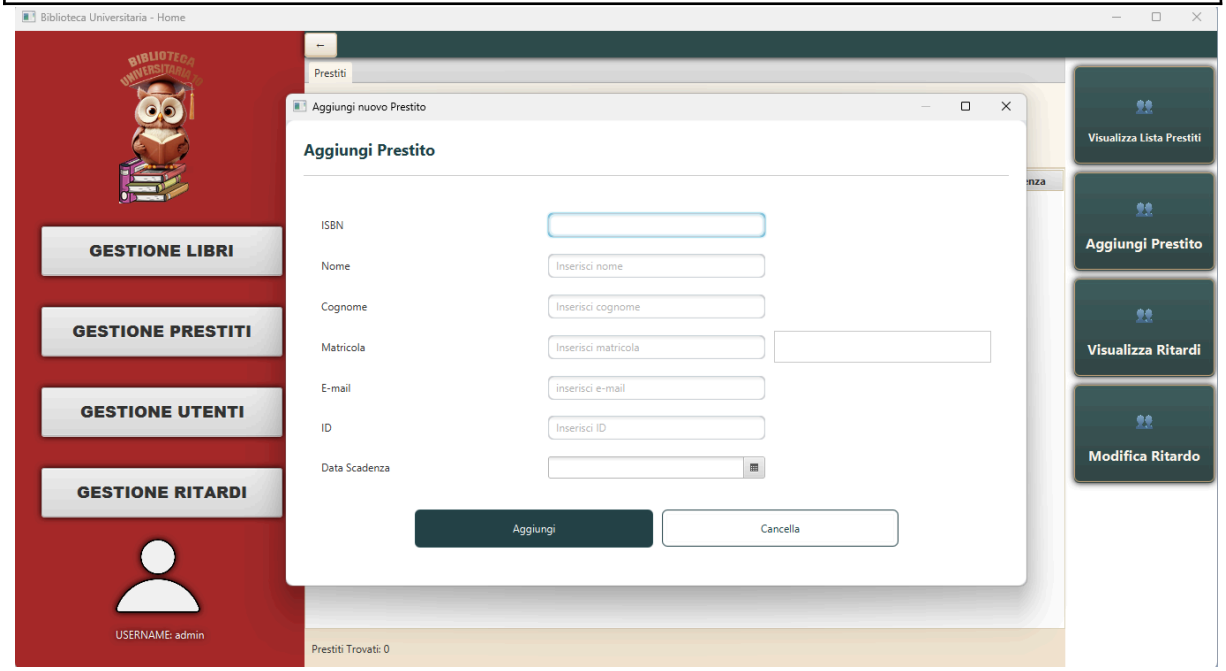
Interfaccia generale per la gestione prestiti



Visualizza prestiti



Aggiungi prestito



Visualizza ritardi