Documento di Progettazione: Gestione Biblioteca Universitaria

Autori: Sabrina Soriano, Mirko Montella, Achille Romano, Ciro Senese

Gruppo: 02

Data: 8 novembre 2025

---

1. Architettura del sistema

Descrizione Generale:

L'applicazione è un software per la gestione di una biblioteca universitaria sviluppata in **Java** e con l'utilizzo di **JavaFX** per creare l'interfaccia grafica.

Per la sua implementazione abbiamo utilizzato il pattern architetturale MVC (Model-View-Controller).

Lo utilizziamo perché sono disponibili più modi per visualizzare e interagire con i dati:

Es.1

- **View 1 (Studente/Guest):** Lo studente sfoglia la lista dei libri senza visualizzare le opzioni di gestione di tale lista.
- **View 2 (Bibliotecario):** Il bibliotecario visualizza la *stessa* lista di libri, ma arricchita con le opzioni per gestirla. Una riga con pulsanti per **"Modifica"** e **"Elimina"** e vede dettagli operativi (es. storico dei prestiti di quel libro).

Grazie a questo modello possiamo fare cio non dovendo creare due database diversi (perche il model è uno solo) ma semplicemente attaccandoci due View diverse in base al login.

Es. 2

- **View 1 (Lista):** Mostra i risultati come un elenco testuale formato da titolo, autore e isbn.
- **View 2 (Griglia):** Mostra i risultati come una griglia di immagini (copertine dei libri).

Grazie a questo modello possiamo anche cambiare il modo in cui appaiono i risultati (griglia o lista) sempre aggiungendo semplicemente due view senza dover cambiare la ricerca che viene effettuata nel model.

Moduli Principali (Packages):

Model: (It.unisa.diem.ing.biblioteca.model)

- **Responsabilità:** Rappresenta il nucleo dell'applicazione perché contiene le classi principali che rappresentano e gestiscono i dati.
- **Classi principali:** Libro, Studente, Prestito, Bibliotecario.
- **Dipendenze:** N/A

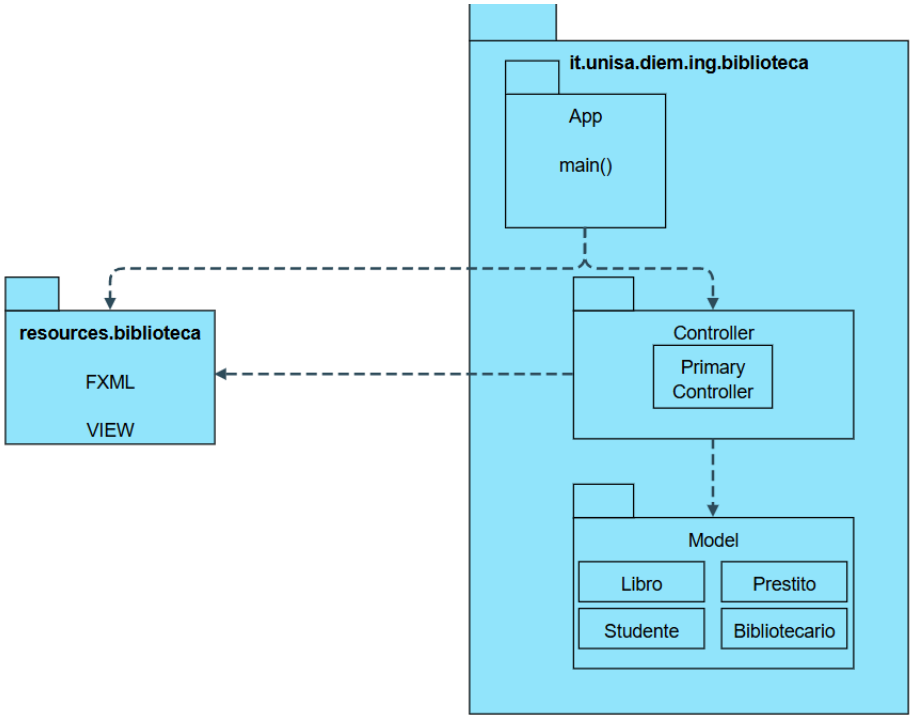
View: (it.unisa.diem.ing.biblioteca.app e resources.Biblioteca)

- **Responsabilità**: Contiene tutte le classi che rappresentano l'interfaccia grafica (UI). Questo modulo si occupa solo di mostrare i dati e catturare gli input dell'utente
- **Componenti principali**: File primary.fxml, secondary.fxml, classe App.java.
- **Dipendenze**: Utilizza i Controller per delegare la gestione degli eventi utente.

Controller: (it.unisa.diem.ing.biblioteca.controller)

- **Responsabilità:** Fa da tramite per il model e la view gestendo eventi generati dall'utente (click pulsante) e invoca i metodi delle classi nel model.
- **Classi principali:** Controller.
- **Dipendenze:** Dipende dal Model (per accedere ai dati) e dalla View (per cambiare l'interfaccia utente).

Diagramma dei Package:



App -> View/Controller: Il modulo App è il punto di ingresso e dipende dagli altri perché deve caricare i file FXML.

Controller -> Model: I Controller hanno una dipendenza diretta verso i Model perché chiamano i metodi delle classi (es. Libro.ricercaLibro()) per eseguire le operazioni.

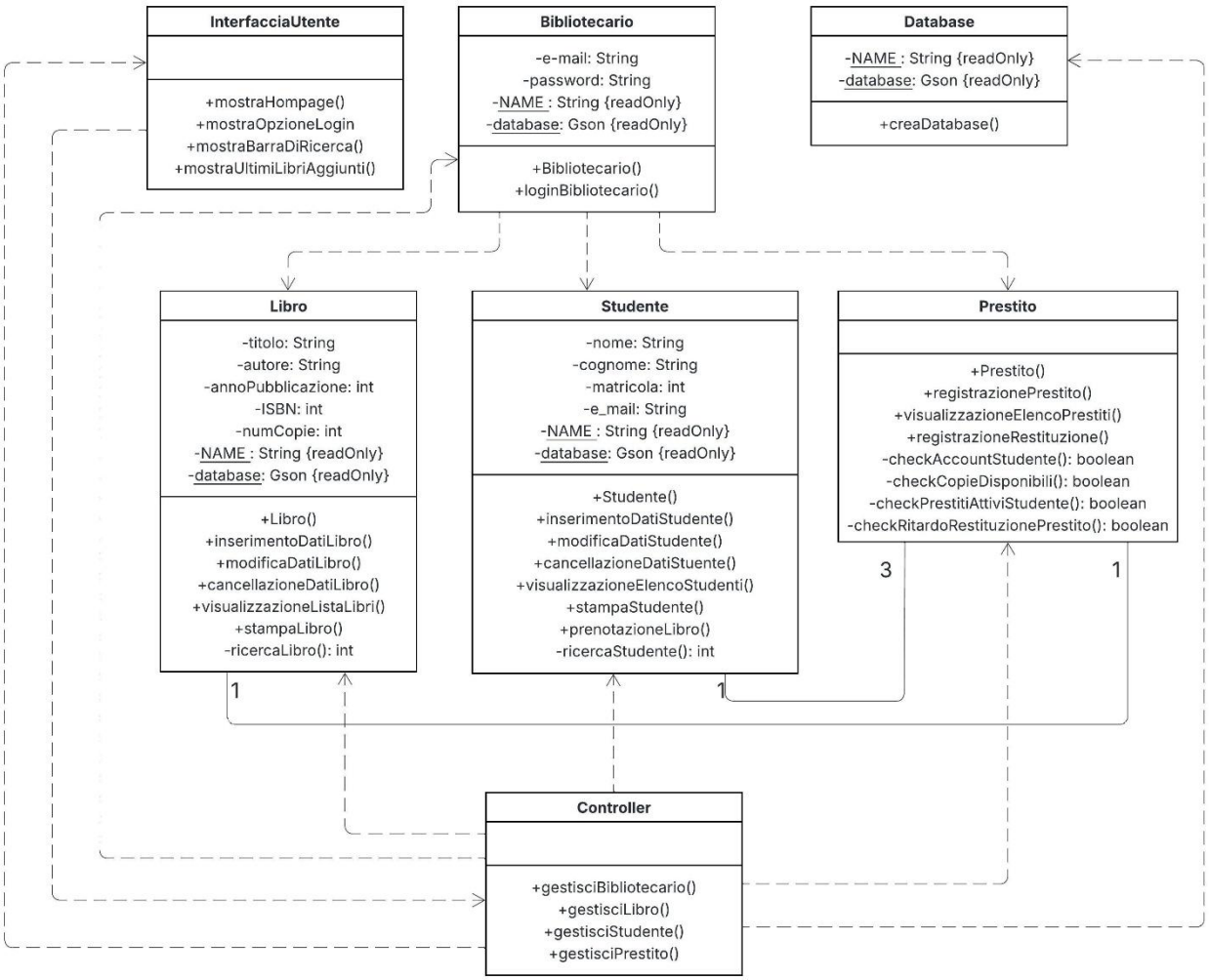
Controller -> View: I Controller sono collegati alle View (tramite le annotazioni @FXML) perché devono rispondere ai click dei pulsanti.

Model: È il pacchetto alla base della gerarchia e non ha frecce in uscita verso gli altri perché è indipendente dall'interfaccia grafica.

---

2. Modello statico

Diagramma delle Classi:



Descrizione delle Classi Principali:

Package it.unisa.diem.ing.biblioteca.model

- **Libro**
    - **Responsabilità:** Gestisce la creazione, modifica, cancellazione, visualizzazione e ricerca legata al catalogo di libri
    - **Attributi Principali:** titolo (String), autore (String), anno (LocalDate), ISBN (String), copieDisponibili (int).
    - **Metodi Pubblici:**
        - inserimentoDatiLibro(): Crea e salva una nuovo libro.
        - modificaDatiLibro(): Aggiorna gli attributi di un libro esistente.
        - cancellazioneDatiLibro(): Rimuove il libro dal database e catalogo.

        - visualizzazioneListaLibri(): Restituisce una lista della collezione completa dei libri.
        - ricercaLibro(): Restituisce i libri corrispondenti alla query di ricerca.
    - **Relazioni Rilevanti:** È associato alla classe Prestito (un prestito riguarda un libro).

- **Studente**
    - **Responsabilità:** Gestisce la creazione, modifica, cancellazione, visualizzazione e ricerca dell' elenco degli studente
    - **Attributi Principali:** nome (String), cognome (String), matricola (String), email (String).
    - **Metodi Pubblici:**
        - inserimentoDatiStudente(): Registra un nuovo studente.
        - modificaDatiStudente(): Aggiorna i dati anagrafici.
        - cancellazioneDatiStudente(): Rimuove lo studente.
        - visualizzazioneElencoStudenti(): Restituisce la lista degli studenti iscritti.
        - prenotazioneLibro(): Idealmente permette allo studente di prenotare libro per un presito che completera di persona.
        - ricercaStudente(): Restituisce gli studenti corrispondenti alla query di ricerca.
    - **Relazioni Rilevanti:** È associato alla classe Prestito (uno studente effettua un prestito).

- **Prestito**
    - **Responsabilità:** Classe che gestisce il prestito tra Studente e Libro.
    - **Attributi Principali:** dataInizio (LocalDate), dataFinePrevista (LocalDate), studente (Studente), libro (Libro).
    - **Metodi Pubblici:**
        - registrazionePrestito(): Avvia un nuovo prestito e decrementa il numero di copie del libro.
        - visualizzazioneElencoPrestiti(): Mostra i prestiti attivi.
        - registrazioneRestituzione(): Chiude il prestito e incrementa il numero di copie del libro.

        - checkAccountStudente(): Controlla che lo Studente sia registrato nel sistema.
        - checkCopieDisponibili(): Controlla che ci siano ancora copie disponibili del libro.
        - checkPrestitiAttiviStudente(): Controlla che lo studente non abbia più di 3 prestiti attivi.
        - checkRitardoRestituzionePrestito(): Controlla che lo studente non abbia prestiti attivi ma in ritardo di restituzione.
    - **Relazioni Rilevanti:** Dipende da Libro e Studente.
- **Bibliotecario**
    - **Responsabilità:** Gestisce l'accesso alle funzionalità amministrative.
    - **Attributi Principali:** email (String), password (String).
    - **Metodi Pubblici:**
        - loginBibliotecario(): Verifica le credenziali per concedere l'accesso alla dashboard amministrativa.

Package it.unisa.diem.ing.biblioteca.controller

- **Controller**
    - **Responsabilità:** Gestisce gli eventi generati dall'utente nell'interfaccia grafica.
    - **Metodi Pubblici:**
        - *Vari Event Handlers:* Metodi annotati con @FXML che invocano a loro volta i metodi delle classi nel package Model (es. chiamando Libro.inserimentoDatiLibro() al click di un bottone).
    - **Relazioni Rilevanti:** Dipendono dalle classi del package model e dalla classe App.

Package it.unisa.diem.ing.biblioteca.app

- **App**
    - **Responsabilità:** Gestisce lo Stage principale e il caricamento delle risorse.
    - **Metodi Pubblici:** start(), loadFXML().

Scelte Progettuali:

- **Coesione:** Ogni classe ha una singola responsabilità ben definita. Le classi del Model si occupano solo dei dati, i Controller solo della gestione delle azioni e le View solo dell'interfaccia grafica. Questo rispetta il **Single Responsibility Principle (SRP)**.

- **Accoppiamento:** L'uso del pattern MVC riduce le dipendenze perché le classi del Model funziona a discapito di come è implementata l'interfaccia.
- **Incapsulamento:** Tutti gli attributi delle classi sono private e accessibili solo tramite metodi pubblici controllati.

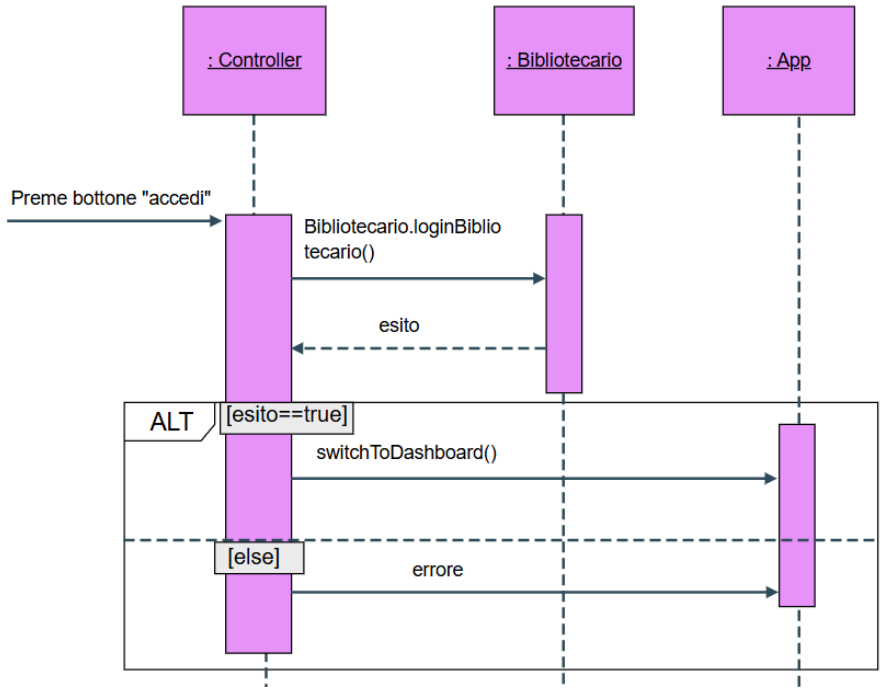
---

3. Modello dinamico

Descrizione delle Interazioni:

IF-01: Login (Bibliotecario)

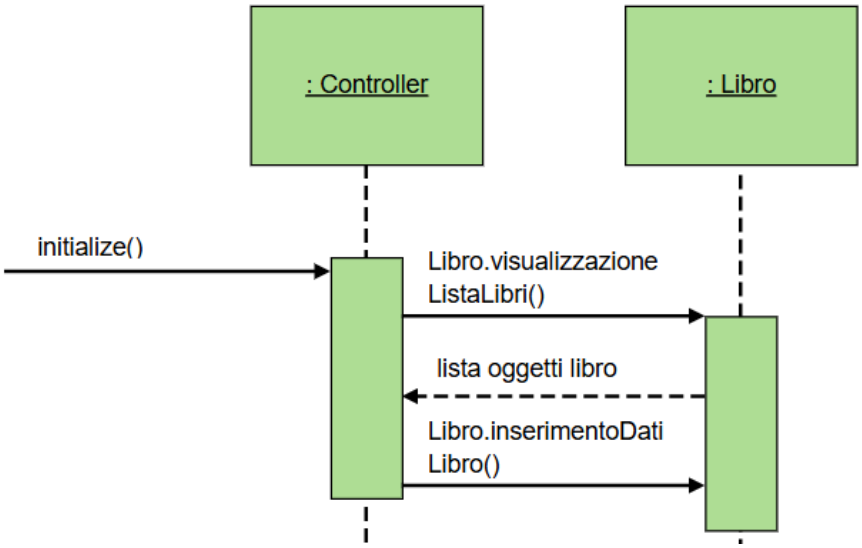
- **Descrizione:** L'utente accede alle funzionalità amministrative.
- **Diagramma di Sequenza**



- **Commento al diagramma:**
    1. L'utente inserisce le credenziali nella schermata di login e preme "Accedi".
    2. Il Controller recupera i dati dai campi di testo.
    3. Il controller invoca il metodo Bibliotecario.loginBibliotecario().
    4. Il metodo Bibliotecario.loginBibliotecario() verifica le credenziali e restituisce un esito (booleano).
    5. Se positivo, il controller chiama switchToDashboard() (in App?) per mostrare la Dashboard. Altrimenti, mostra un errore.

BF-04: Visualizzazione lista libri

- **Descrizione:** Recupero dell'intero catalogo.
- **Diagramma di sequenza:**

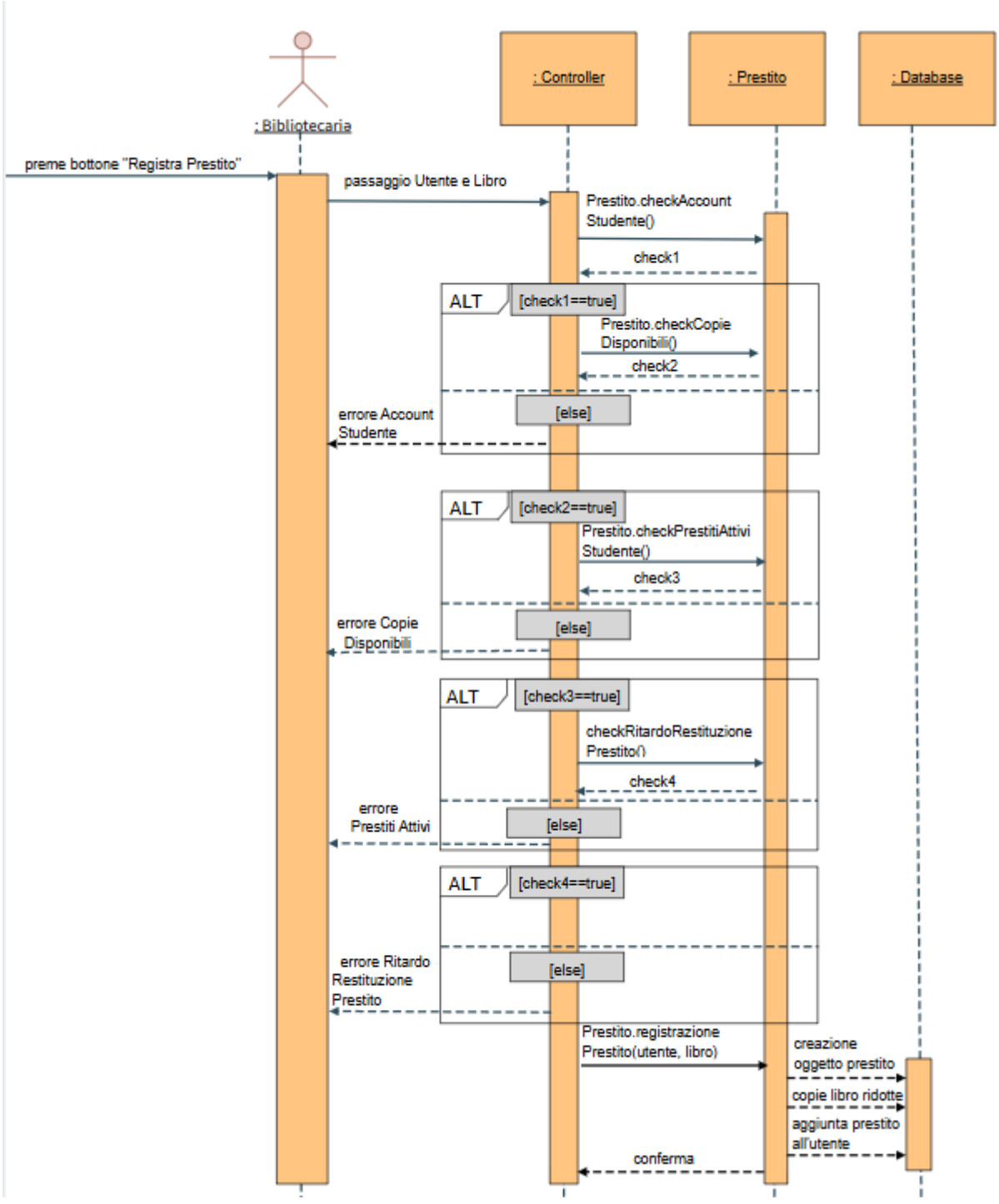


- **Commento al diagramma:**
    1. All'inizializzazione della vista (initialize()), il controller chiama Libro.visualizzazioneListaLibri().
    2. Il metodo restituisce una lista/array di oggetti Libro.
    3. Il controller popola la TableView o ListView.

BF-11: Registrazione prestito

- **Descrizione:** Assegnazione di un libro a uno studente.

- **Diagramma di sequenza:**



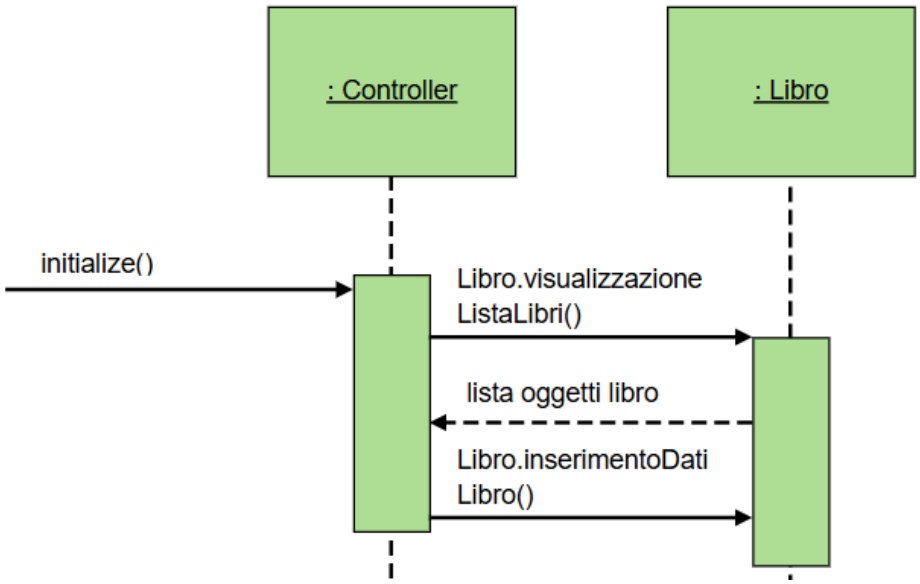
- **Commento al diagramma:**

1. Il Bibliotecario seleziona uno Studente e un Libro dalla sezione per registrare un prestito e clicca “Registra Prestito”
2. Il Controller intercetta l'evento e chiama Prestito.registrazionePrestito().
3. Il Controller chiama Studente.canBorrow(): verifica se lo studente ha meno di 3 libri.
4. Il Controller chiama Libro.isDisponibile(): verifica se le copie > 0.

5. Se i controlli passano, il Controller chiama BibliotecaManager.nuovoPrestito(studente, libro).
6. Viene creato un oggetto Prestito.
7. Libro decrementa le copie disponibili.
8. Studente aggiunge il prestito alla sua lista.

BF-12: Visualizzazione elenco prestiti

- **Descrizione:** Monitoraggio dei prestiti attivi.
- **Diagramma di sequenza:**

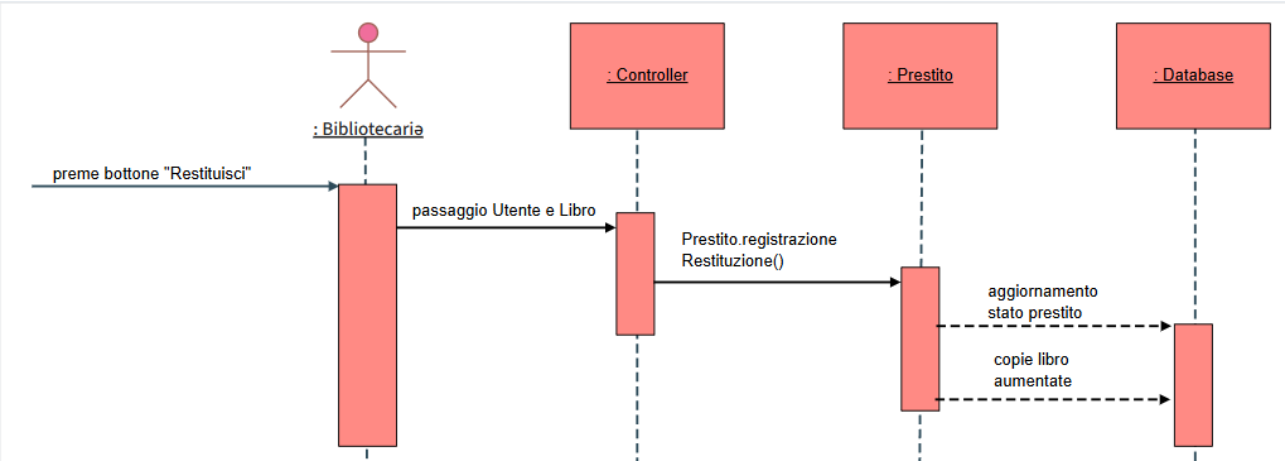


- **Commento al diagramma:**
  1. Il controller richiede la lista chiamando Prestito.visualizzazioneElencoPrestiti().
  2. Il metodo restituisce i dati, inclusi le date di scadenza per evidenziare eventuali ritardi nella UI.

BF-13: Registrazione restituzione

- **Descrizione:** Chiusura di un prestito e rientro del libro.

- **Diagramma di sequenza:**



- **Commento al diagramma:**

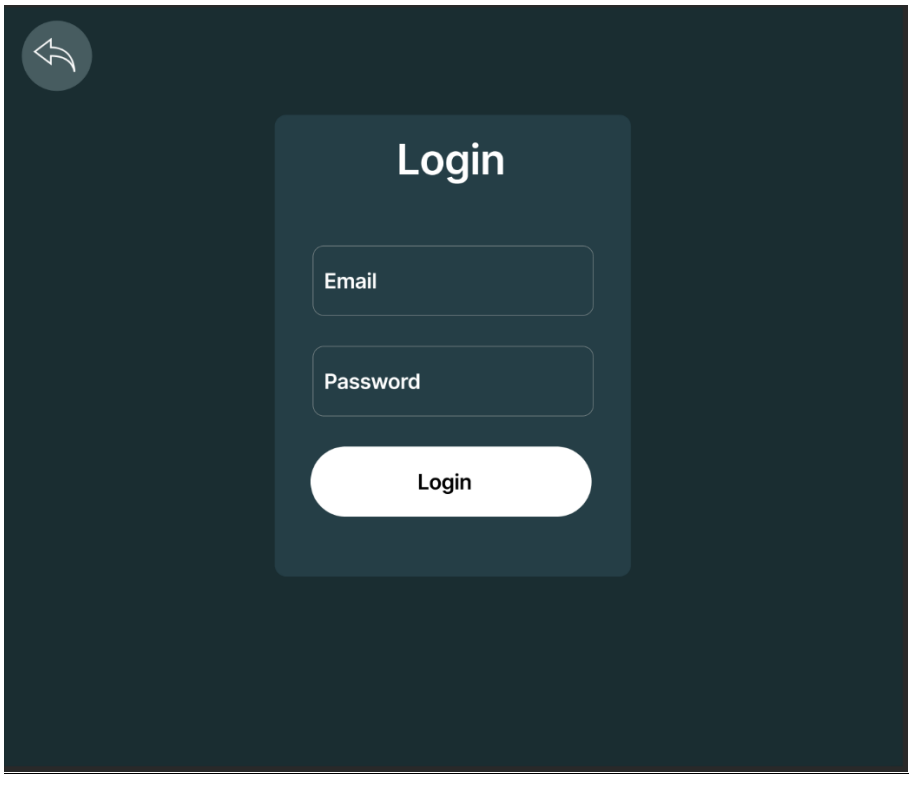
1. Il bibliotecario seleziona un prestito attivo e clicca "Restituisci".
2. Il controller invoca Prestito.registrazioneRestituzione()
3. Il metodo segna il prestito come chiuso e incrementa le copie disponibili del libro associato.

---

4. Design dell'interfaccia utente

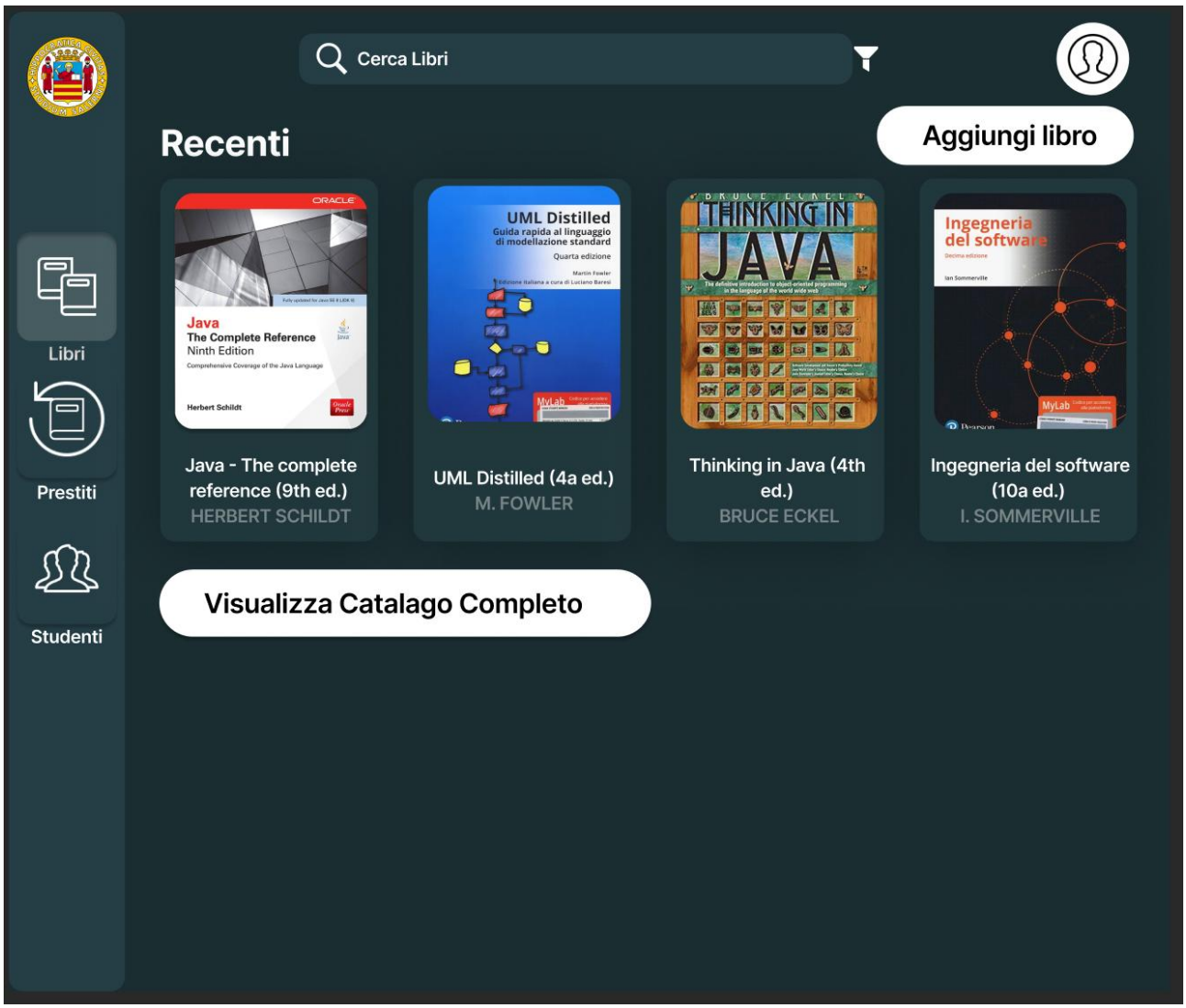
Mock-up delle Schermate Principali:

- **Schermata di Login (UI-01/UI-05)**



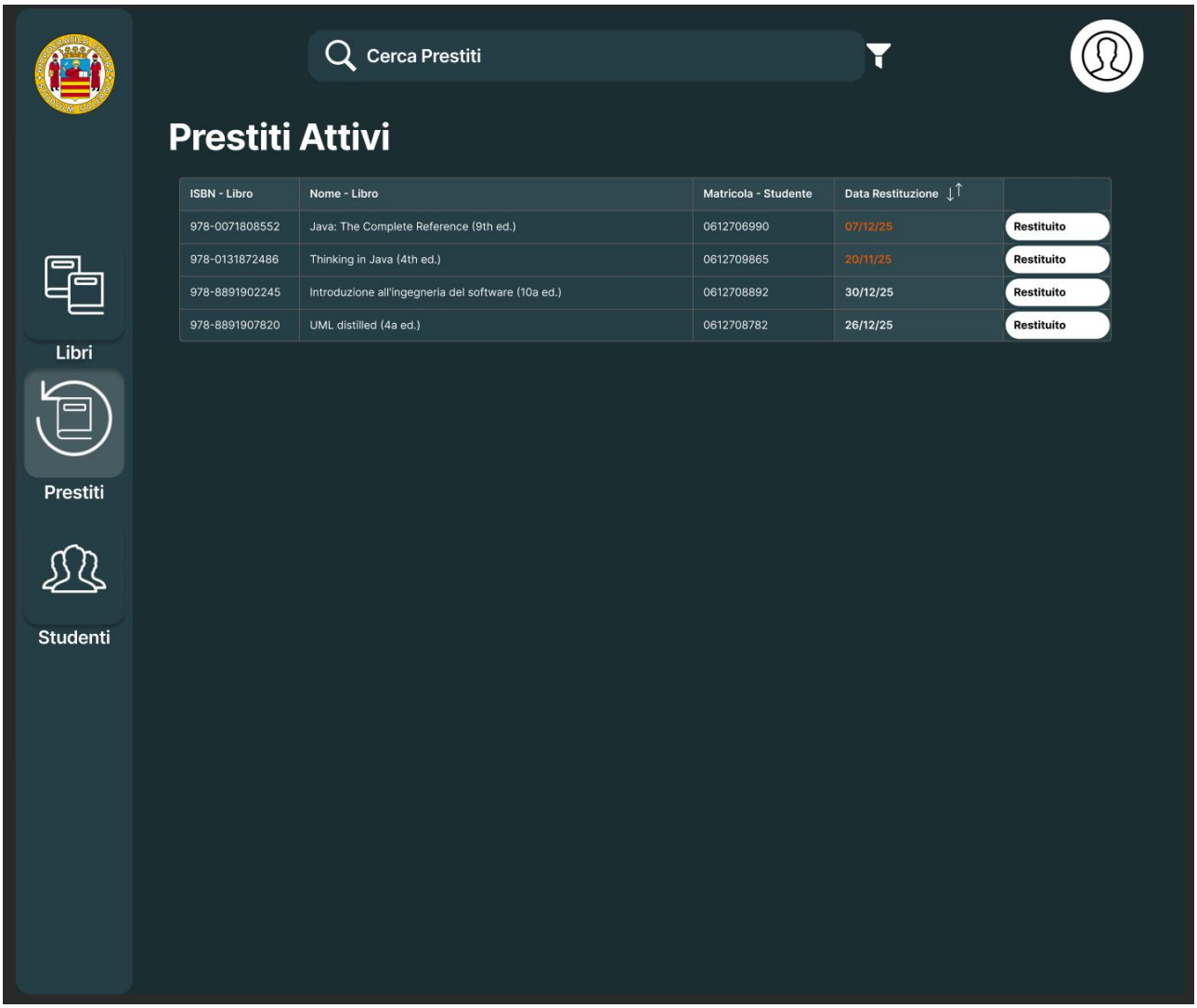
- **Interazione:** Al centro della finestra, due campi di testo ("Email" e "Password") e un pulsante "Login".
- **Feedback:** Inserendo dati errati, appare una label arancione sotto il pulsante "Credenziali non valide".

- **Dashboard Amministrativa - Libri (UI-02/UI-04)**



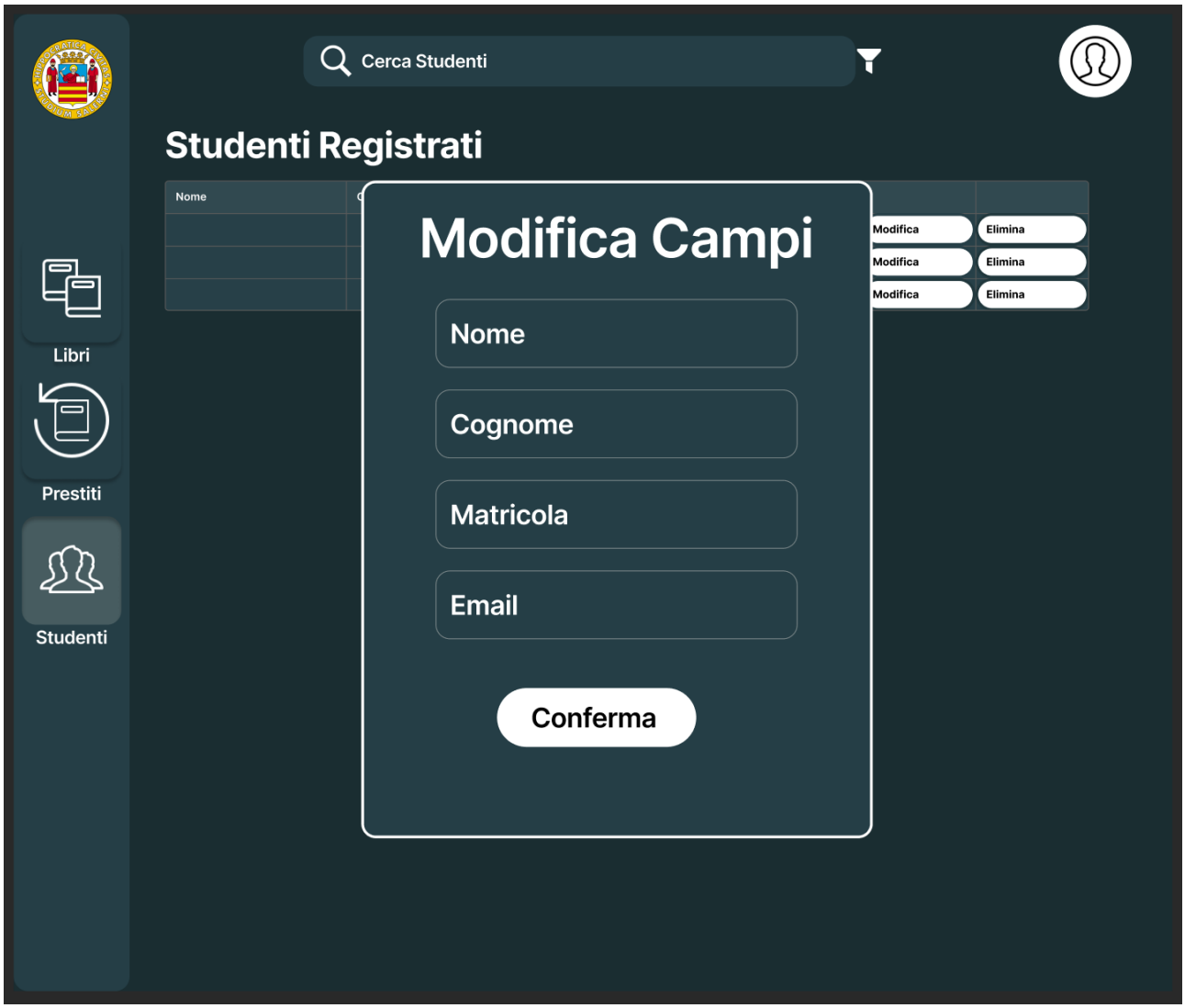
- **Layout:** In alto una barra degli strumenti con campo di ricerca e filtro. A destra una barra di navigazione tra le varie tab della dashboard amministrativa.
- **Contenuto:** Al centro una Griglia che mostra: Copertina, Titolo e Autore dei libri recentemente aggiungi. Un bottone per aggiungere nuovi libri e per visualizzare il catalogo completa

- **Dashboard Operatore - Prestiti (UI-05)**



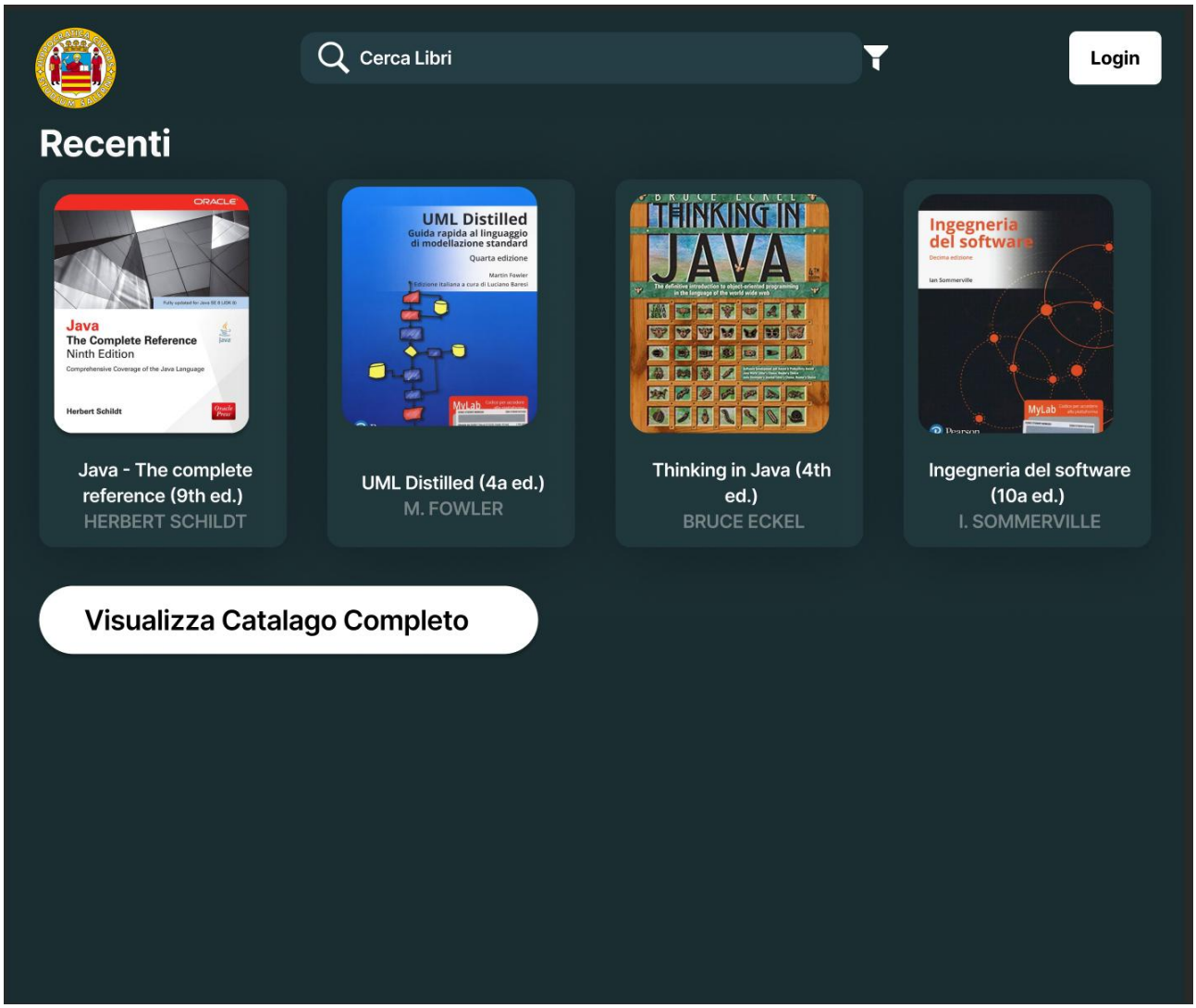
- **Contenuto:** Una tabella con elenco prestiti: matricola, titolo Libro e isbn , Data di restituzione e un bottone per confermare l'avvenuta restituzione
- **Interazione Visiva:** Le righe con "Data restituzione" antecedente a oggi sono evidenziate con un colore arancione (Ritardo).

- **Inserimento utente (BF-01)**



    - **Interazione:** Finestra modale (popup) che si apre sopra la dashboard. Contiene form con validazione (es. non permette di salvare se l'ISBN è vuoto) e un tasto di conferma.

- Homepage per tutti
-



- **Contenuto**: stesso schema di quella amministrativa ma senza alcuna possibilità di aggiungere nuovi libri o di navigare le altre dashboard amministrative.