

ARCHITETTURA DEL SISTEMA

L'architettura del sistema “**Gestione Libreria**” è basata sulla logica introdotta dal pattern Model-View-Controller (**MVC**), in modo da garantire una totale separazione tra:

- Logica di business → Model
- Logica di presentazione → View
- Logica di controllo → Controller

Questa scelta garantisce:

- Testabilità → poiché il model può essere sottoposto a casi di test indipendentemente dall'interfaccia grafica
- Modificabilità → poiché è possibile effettuare modifica all'interfaccia utente senza intaccare la logica di business
- Manutenibilità → La separazione logica sopracitata permette di individuare facilmente eventuali errori

DESCRIZIONE DEI MODULI

Il sistema in coerenza con il pattern MVC è suddiviso in tre package principali.

Package `it.unisa.biblioteca.model`

Contiene le classi Entità (Libro, Utente, Prestito) implementando le regole di business fondamentali. **NON Dipende** da altri package, poiché non ha alcuna conoscenza del Controller o delle varie view.

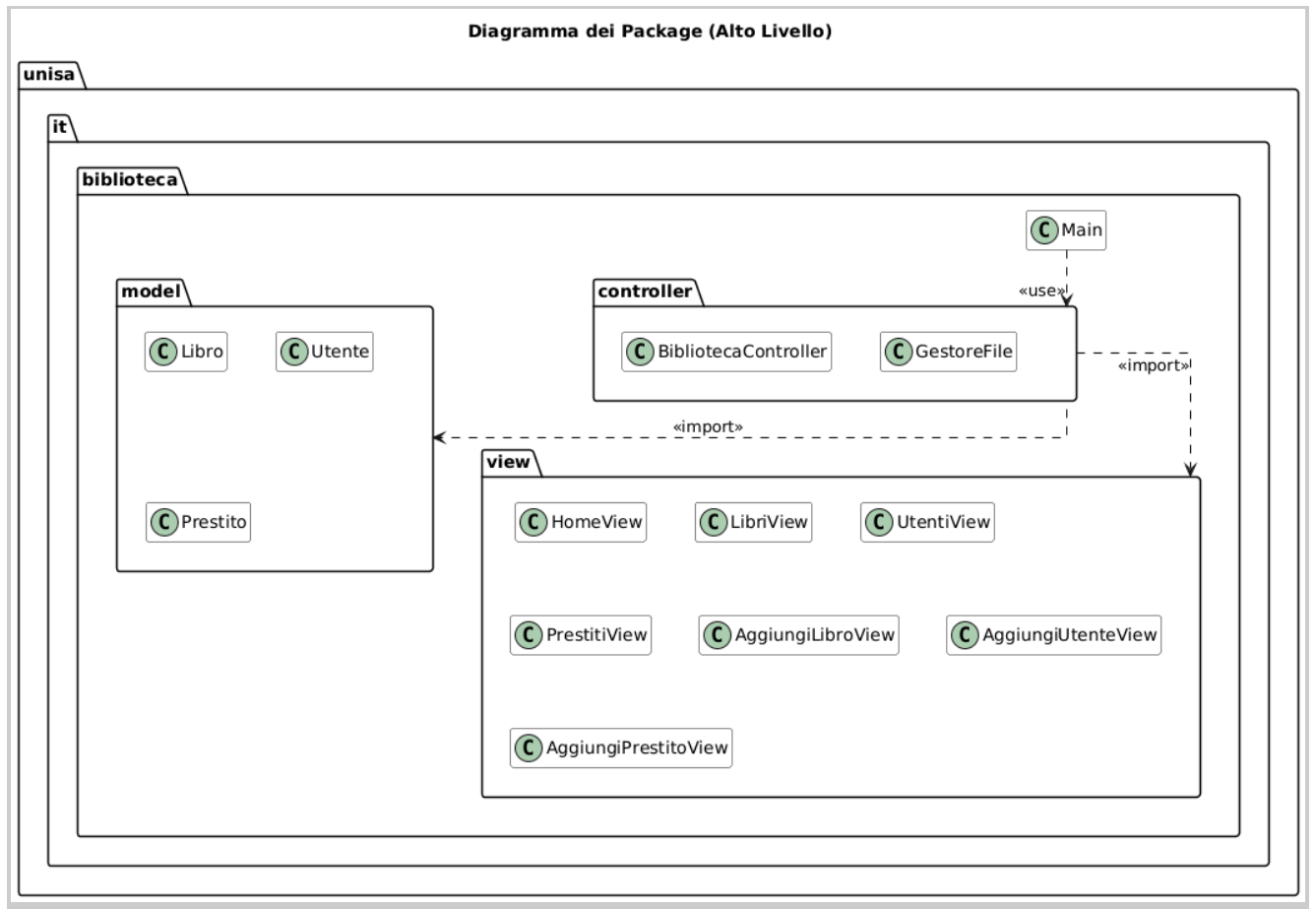
Package `it.unisa.biblioteca.controller`

Come da definizione agisce come collante tra View e Model: gestendo l'intero flusso dell'applicazione, la navigazione tra le varie schermate e la risposta agli eventi utente (click di un Button). **Dipende** sia dal Model che dalla View

Package `it.unisa.biblioteca.view`

Contiene le classi View che sono le dirette responsabili delle interfacce utente utilizzando la libreria messa a disposizione da Java → JavaFX.

DIAGRAMMA DEI PACKAGE



DESCRIZIONE DELLE CLASSI

Package **Model:**

- Libro:
 - **Attributi principali** : Titolo, autori, dataPubblicazione, disponibilità, isbn
 - **Metodi principali**: incrementaDisponibilità(), decrementaDisponibilità()
 - **Descrizione generale**: Rappresenta l'entità omonima, gestisce la validazione dei dati e gestisce lo stato della disponibilità
- Utente:
 - **Attributi principali** : matricola, nome, cognome, email, prestitiAttivi
 - **Metodi principali**: aggiungiPrestito(), rimuovi Prestito()
 - **Descrizione generale**: Rappresenta l'entità omonima, ovvero colui che usufruisce dei servizi della biblioteca. Gestisce i relativi dati anagrafici e mantiene traccia dei prestiti attivi
- Prestito
 - **Attributi principali** : Utente, Libro
 - **Metodi principali**: isScaduto(), giorniAllaScadenza()
 - **Descrizione generale**: Classe associativa che gestisce la relazione tra Utente e Libro

Package **View:**

- Viste di Visualizzazione (LibriView, UtentiView, PrestitiView):
 - **Responsabilità**: Permettono la visualizzazione dei dati della biblioteca tramite un formato tabellare (TableView), implementando logiche di filtraggio dinamico (nella ricerca) e l'evidenziazione condizionale (per mettere in risalto i prestiti scaduti)
 - **Descrizione generale**: Gestiscono il modo in cui l'utente può accedere, consultare e selezionare i dati, non contengono logica di business.

Package **Controller:**

- BibliotecaController:
 - **Metodi principali**: mostraHome(), mostraLibri(), mostraAggiungiLibro(), mostraUtenti(), mostraAggiungiUtente(), mostraPrestiti(), mostraAggiungiPrestito().
 - **Descrizione generale**: Gestore della logica d'esecuzione, inizializza il sistema e mantiene attive le interazioni tra model e view. Gestisce le eccezioni e coordina il salvataggio dei dati
- GestoreFile:
 - **Metodi principali**: salvaTutto(), caricaTutto()
 - **Descrizione generale**: Gestore della persistenza su dei dati tramite serializzazione binaria

SCELTE PROGETTUALI E PRINCIPI ADOTTATI

1) *Alta Coesione (High Cohesion)*

Ogni classe è stata progettata per avere un'unica responsabilità ben definita (Single Responsibility Principle)

2) *Basso Accoppiamento (Low Coupling)*

Le dipendenze tra le classi sono state minimizzate in favore della modularità.

Ad esempio il pacchetto model non ha nessuna dipendenza verso il pacchetto view o controller, questo permette di riutilizzare le entità in contesti diversi oltre alla possibilità di cambiare o gestire l'interfaccia grafica.

3) *Information Hiding e Incapsulamento*

L'accesso diretto agli attributi delle classi è impostato su "private", in modo da avere un corretto information hiding e non permettere alle altre classi la possibilità di violare gli attributi di un determinato oggetto come per esempio "utente".

Per accedere a questi dati e queste informazioni abbiamo implementato dei metodi "getter/setter" che permettono di richiedere o inserire (dove possibile) i dati negli attributi.

Abbiamo inoltre implementato il meccanismo delle "Defensive Copy" nelle classi Libro e Utente, in modo che liste passate o restituite vengano copiate avvolte in Collection.unmodifiableList, per impedire che riferimenti esterni possano corrompere lo stato interno dell'oggetto.

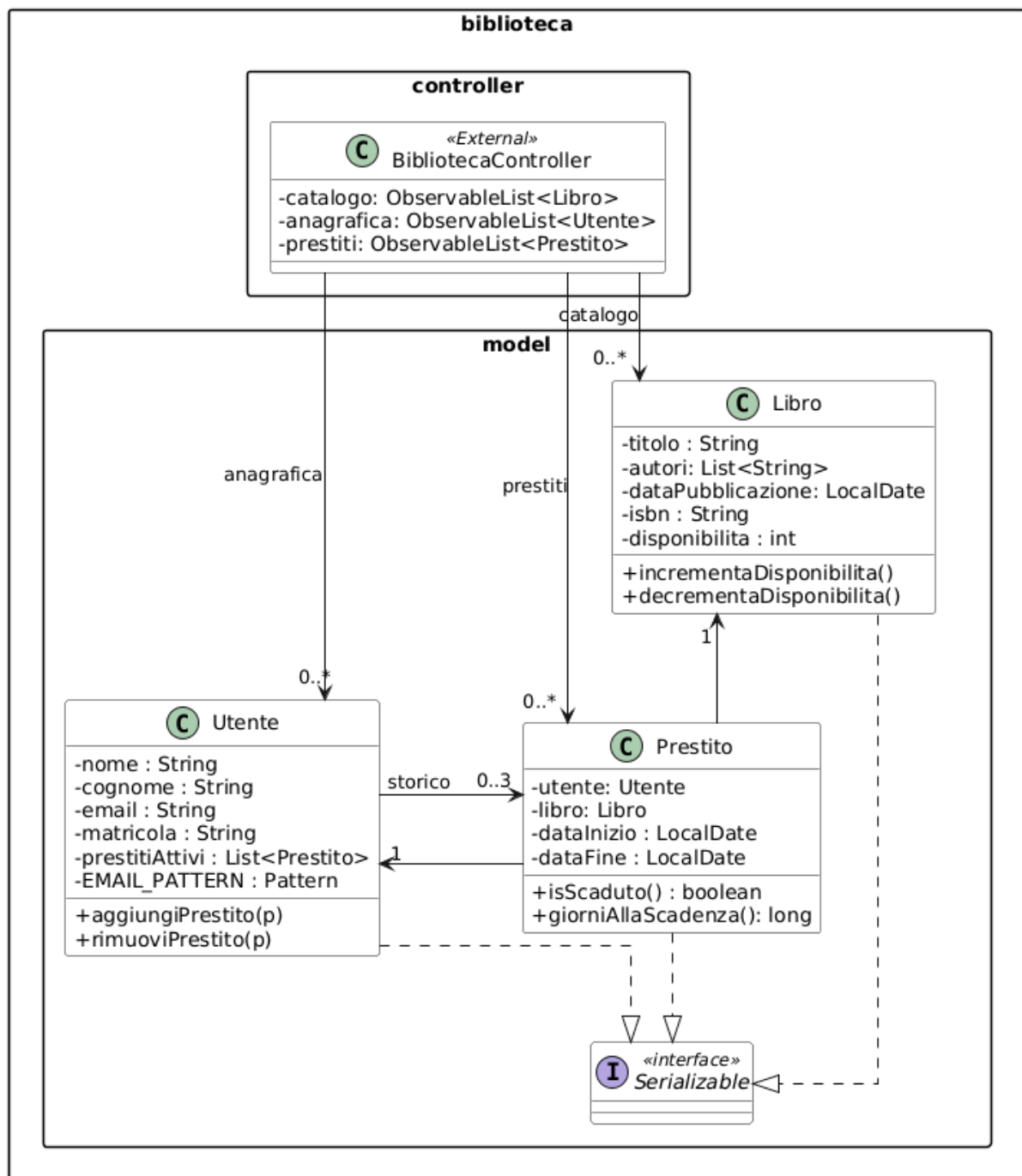
4) *Integrità dei Dati*

Inserito nella logica una priorità alla coerenza del sistema, i campi identificativi sono marcati final quindi immutabili dopo la costruzione, in modo da garantire l'identità di una entità prevenendo errori nelle relative strutture dati.

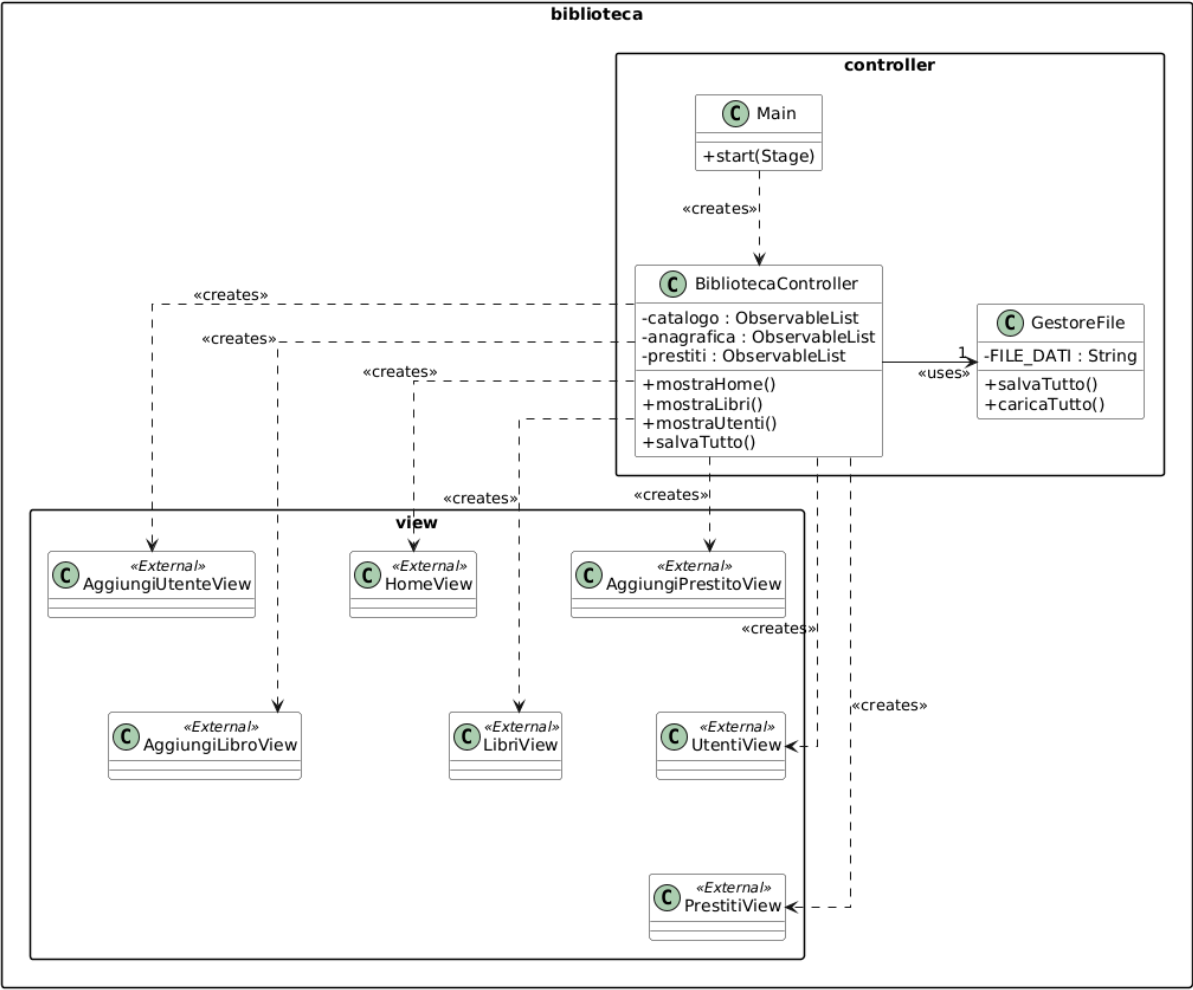
Le Relazioni bidirezionali come Utente<->Prestito sono gestite atomicamente, nei metodi del model, dove il metodo "aggiungiPrestito" aggiorna entrambi i lati della relazione

DIAGRAMMA DELLE CLASSI

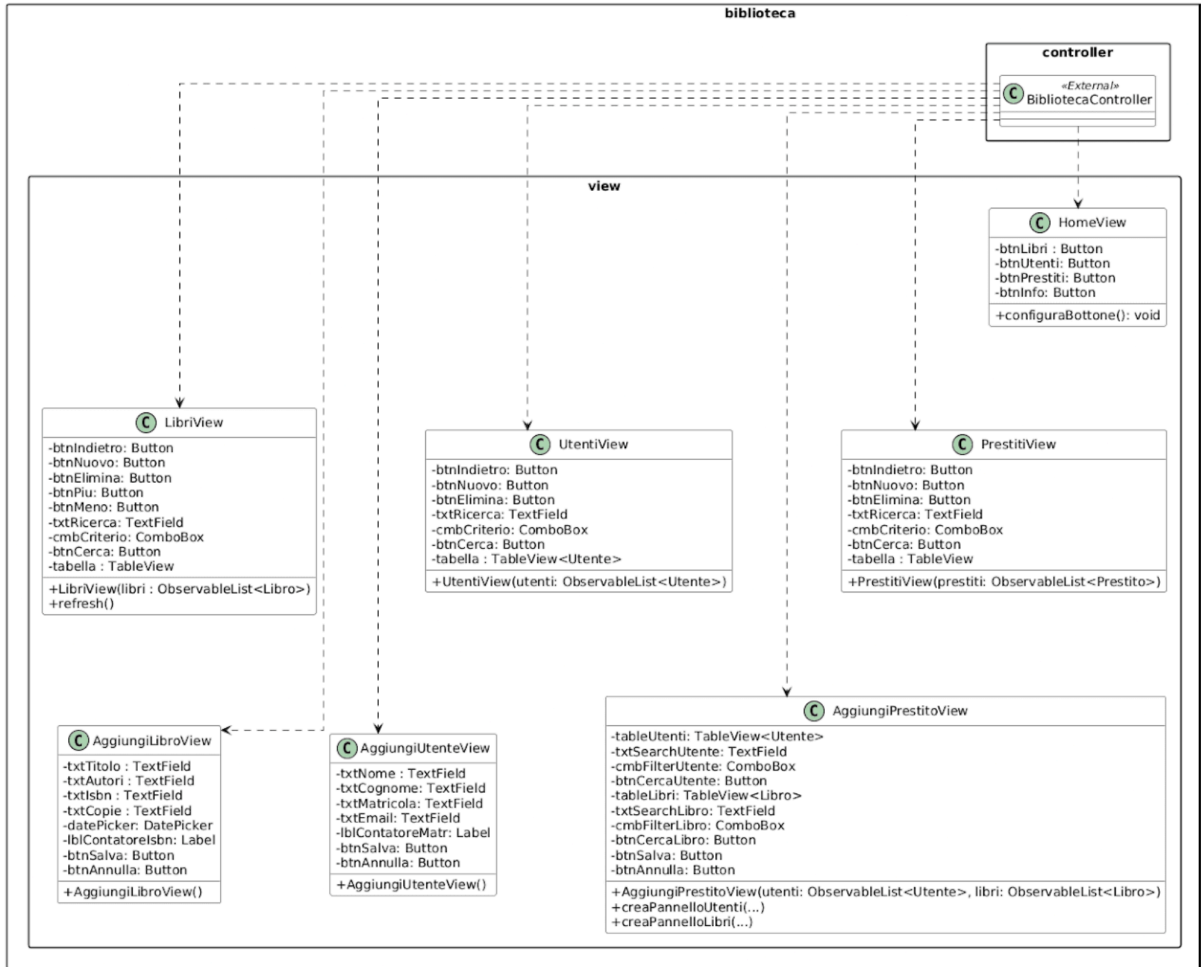
1. Focus Layer MODEL (Dati e Business Logic)



3. Focus Layer CONTROLLER (Logica Applicativa)



2. Focus Layer VIEW (Interfaccia Utente)



DESCRIZIONE DELLE RELAZIONI TRA LE CLASSI

1. Relazione: Associazione Unidirezionale

Dove si trova:

1. BibliotecaController → Libro / Utente / Prestito
2. Prestito → Libro
3. Prestito → Utente
4. BibliotecaController → GestoreFile
5. Utente → Prestito

Caratteristiche:

- **Descrizione:** In una relazione di associazione un oggetto di una classe mantiene un riferimento a un oggetto dell'altra per poter svolgere il proprio lavoro.
- **Perché la scelta dell'associazione unidirezionale?** Perché nello scenario di associazione unidirezionale una classe A (Da dove parte la freccia) mantiene un riferimento a un oggetto della classe B, ma B non mantiene un riferimento a un oggetto della classe A.
Ad esempio il controller mantiene per l'istanziamento del catalogo libri, l'anagrafica utenti e l'elenco prestiti, riferimenti alle rispettive classi Libro, Utente e Prestito. A loro volta Libro, Utente e Prestito non hanno riferimenti a controller, è il controller a "conoscere" loro, ma non il contrario.
- **Molteplicità tra controller e model → 0..*:**
 - **0:** La lista può essere vuota (all'inizio non ci sono libri).
 - ***** (Asterisco): Non c'è limite superiore. Il controller può gestire infiniti libri.
 - Quindi **0..*** si legge: *"Il Controller possiede da zero a infiniti Libri"*.
- **Molteplicità tra Prestito e Utente, e tra Prestito e Libro → 1:**
 - **1** si legge: "Un prestito è associato a uno e un solo utente/libro"
- **Molteplicità tra BibliotecaController e GestoreFile → 1:**
 - **1** si legge: "Ogni istanza di BibliotecaController possiede una ed una sola istanza di GestoreFile"
- **Molteplicità tra Utente e Prestito → 0..3:**
 - **0:** Un utente appena registrato non ha prestiti attivi
 - **3:** Un utente può avere più prestiti, nel nostro caso con limite superiore pari a tre
 - Quindi **0..3** si legge : "L'utente possiede da zero a tre Prestiti"

2. Relazione: Dipendenza

Dove si trova:

1. BibliotecaController → Tutte le view
2. Main → BibliotecaController

Caratteristiche:

- **Descrizione:** Una relazione di dipendenza indica che una classe dipende da un'altra, ad esempio una classe A usa una classe B in una variabile locale, in un parametro o nel valore di ritorno di un metodo. Ed è il caso della relazione tra il controller e le view ad esempio, poiché il controller esegue `LibriView view = new LibriView(...)` ma lo fa all'interno del metodo `mostraLibri()`. Quando il metodo finisce il controller non ha "ricordo" poiché non è presente un'istanziamento del tipo: `private LibriView viewLibriPermanente`
- **Perché "Dipendenza" e non "Associazione"?** Perché come descritto nell'esempio delle view, esse vengono gestite (create e distrutte) dinamicamente, la relazione è temporanea e non è parte dello stato persistente del controller, ciò esclude il tutto dalla definizione di associazione tra classi
-

3. Relazione: Realizzazione

Dove si trova:

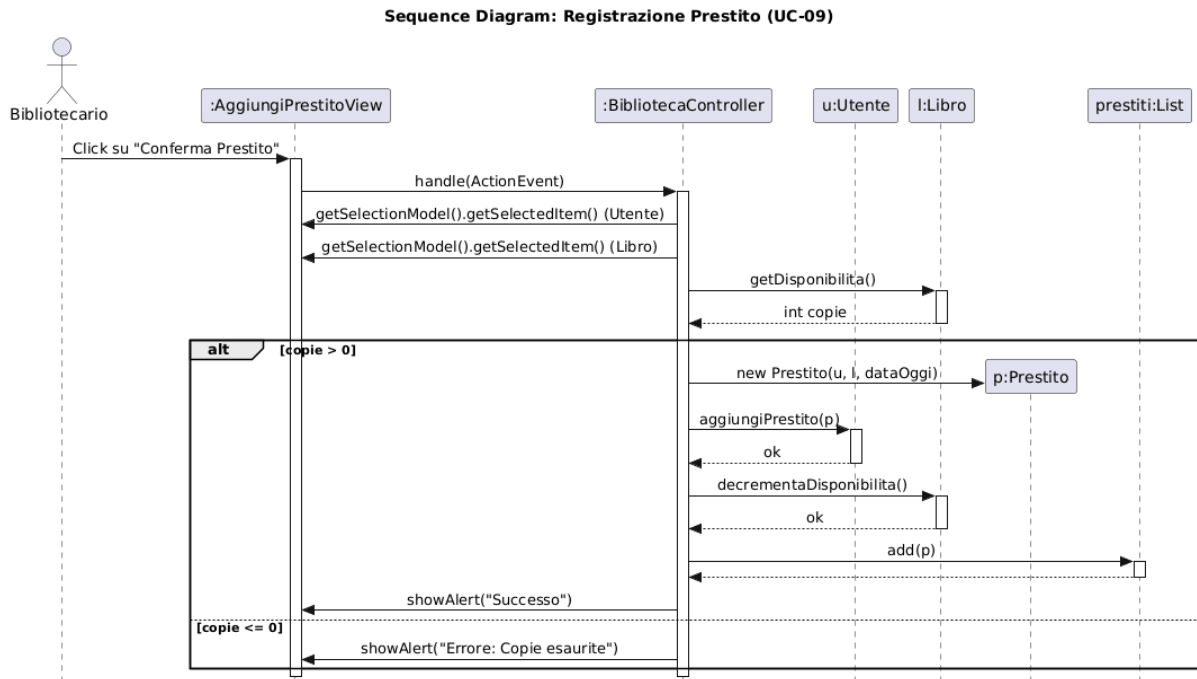
1. Libro/Utente/Prestito → Interfaccia Serializable

Caratteristiche:

- **Descrizione:** La relazione di realizzazione indica che le classi Libro, Utente, Prestito dichiarano esplicitamente di fornire l'implementazione richiesta dall'interfaccia, in questo caso l'interfaccia Serializable fornita da Java, per permettere alla JVM di trasformare gli oggetti in stream di Byte
- **Perché Realizzazione?** Le classi Libro, Utente, Prestito realizzano (implementano) l'interfaccia Serializable.

DESCRIZIONE DELLE INTERAZIONI PIU' IMPORTANTI TRA LE CLASSI

Scenario: Registrazione di un nuovo prestito



Questo scenario è definibile il più significativo sul piano di interazione tra le classi poiché coinvolge la validazione tra più entità, in questo caso Libro ed Utente, oltre che l'aggiornamento del sistema

Commento del diagramma:

Innesco: L'evento viene inizializzato all'interno dell'AggiungiPrestitoView al momento del click da parte del bibliotecario del tasto di conferma per la nuova registrazione, la logica verrà però eseguita nel controller (come da definizione del pattern MVC utilizzato per la realizzazione).

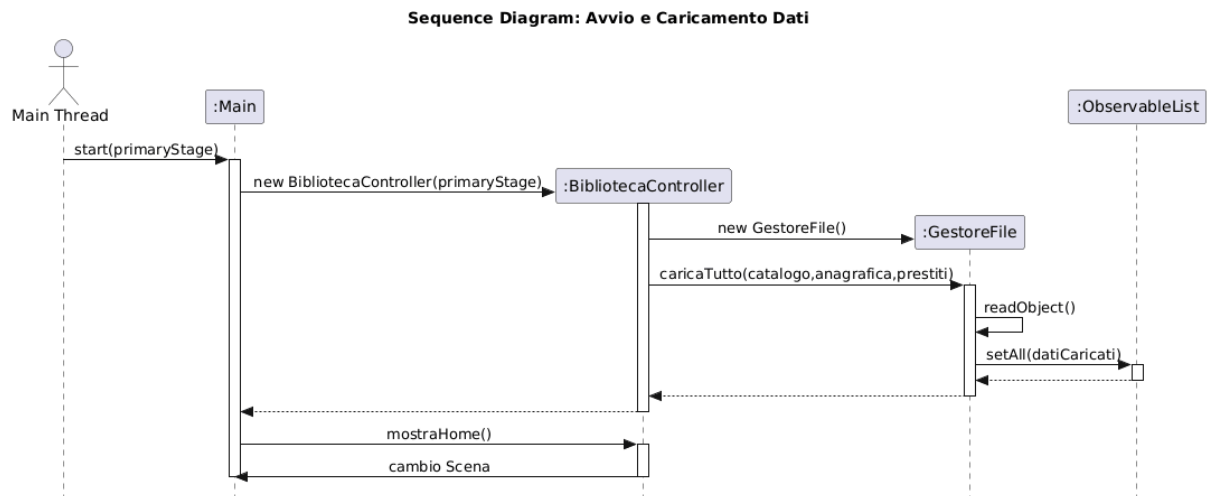
Controllo validità: Il controller come prima cosa verifica la disponibilità del Libro, interrogando l'omonima entità la quale restituisce il numero di copie disponibili, che dovranno essere >0.

Sviluppo: Se la verifica è superata Il controller esegue quattro operazioni in sequenza:

- 1) Istanza il nuovo Prestito
- 2) Il prestito creato viene aggiunto alla lista prestiti attivi dell'Utente
- 3) Decrementa la disponibilità del libro appena prestato
- 4) Aggiunge il prestito all'interno dell'elenco prestiti attivi della propria classe

Feedback: Alla fine, il controller invoca un metodo per mostrare un alert contenente l'esito dell'operazione. Positivo nel caso di copie > 0 , di warning in caso di copie ≤ 0 .

Scenario: Avvio e Caricamento dati



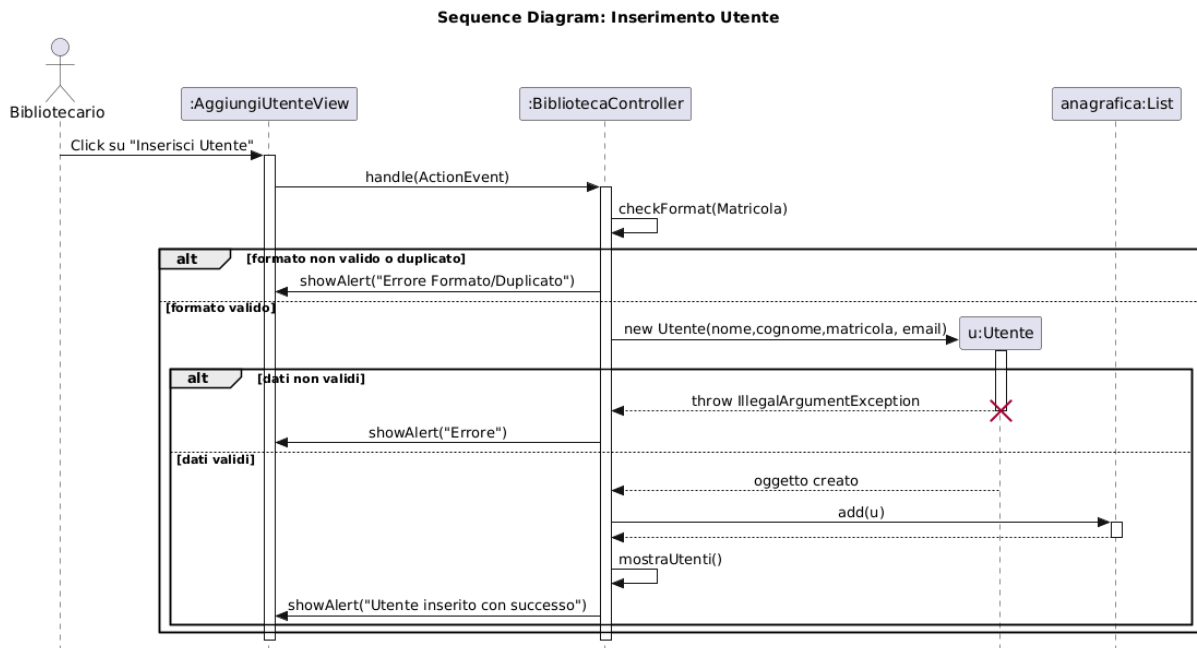
Commento del diagramma:

Innesco: All'avvio del programma il thread Main tramite il metodo start dà il via all'esecuzione della classe Main

Sviluppo: La classe main inizializza il controller passando come parametro primaryStage. Il controller a sua volta inizializza il gestoreFile e invoca il metodo caricaTutto, per effettuare la lettura da file degli elementi presenti sul file di stato, passando come parametri le ObservableList che conterranno rispettivamente: il catalogo libri, l'anagrafica utenti e la lista prestiti. Il gestore file al suo interno si occupa di leggere dal file di stato tramite il metodo readObject, e tramite il metodo setAll popola le liste.

Feedback: Il controller dopo di ciò, se non va incontro a errori, chiama il metodo mostraHome() per inizializzare la view principale dell'applicazione

Scenario: Inserimento Utente



Commento del diagramma:

Innesco: L'evento viene inizializzato all'interno dell'AggiungiUtenteView al momento del click da parte del bibliotecario del tasto di conferma per il nuovo inserimento, la logica verrà però eseguita nel controller (come da definizione del pattern MVC utilizzato per la realizzazione).

Controllo validità: Il controller come prima cosa effettua il controllo validità sul campo matricola, per assicurarsi che esso non sia un duplicato o che non sia valido ai fini della registrazione

Sviluppo: Se il controllo avrà successo il controller si occuperà di istanziare un nuovo oggetto utente con i relativi campi passati come parametro

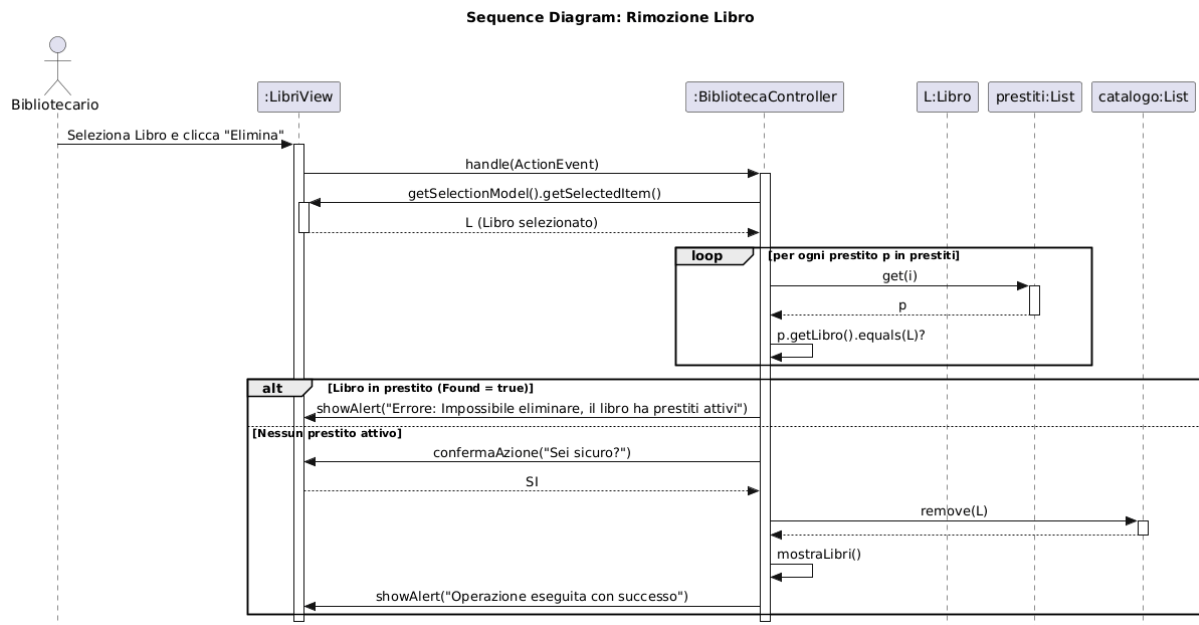
Secondo Controllo Validità: La classe utente in questo caso si occuperà di verificare che i dati inseriti (passati ad esso tramite parametri) siano corretti.

Sviluppo: Se il controllo non avrà successo sarà compito della classe Utente lanciare una IllegalArgumentException e di conseguenza a ciò, il controller mostrerà un alert di errore. In caso di esito positivo il nuovo Utente verrà creato e tramite il metodo add() il controller lo inserirà all'interno del listato "Anagrafica" utenti.

Feedback: Il controller invocherà il metodo mostraUtenti(), per mostrare il listato aggiornato in base alla nuova modifica e mostrerà un alert che confermerà il successo dell'operazione

NOTA: Analogo discorso può essere fatto per l'inserimento di nuovo libro

Scenario: Rimozione Libro



Commento del diagramma:

Innesco: Il bibliotecario seleziona il libro da eliminare all'interno del catalogo visibile nella view "Libriview", la logica verrà però eseguita nel controller (come da definizione del pattern MVC utilizzato per la realizzazione).

Controllo validità: Il controller recupera il selectedItem dal catalogo e successivamente scorre la lista prestiti per assicurarsi che il libro non sia attualmente in prestito

Sviluppo: In caso il libro selezionato risulti in prestito sarà compito del controller mostrare un alert segnalando che per tale motivo è impossibile eliminarlo. In caso il libro non sia attualmente in prestito, il controller mostrerà un alert di verifica ulteriore per far sì che non ci siano ambiguità, e in tal caso rimuove il libro dal catalogo.

Feedback: Il libro viene correttamente rimosso dal catalogo, non solo graficamente ma anche logicamente

NOTA: Analogo discorso può essere fatto per la cancellazione di un Utente

MOCK-UP DELLE SCHERMATE PRINCIPALI

1) HOME



2) Gestione Libri

2.1) Gestione Libri - Ricerca

[illegible]

2.2) Gestione Libri - Cancellazione

< Indietro CATALOGO LIBRI

Cerca per: Titolo Testo da cercare... Cerca

Titolo	Autore/I	Anno	ISBN	Disp.
Libro di Ingegneria Vol. 1	Autore Generico 1, Co-Autore 1	2025	9780000000001	5
Libro di Ingegneria Vol. 2	Autore Generico 2, Co-Autore 2	2025	9780000000002	5
Libro di Ingegneria Vol. 3	Autore Generico 3, Co-Autore 3	2025	9780000000003	4
Libro di Ingegneria Vol. 4	Autore Generico 4, Co-Autore 4	2025	9780000000004	5
Libro di Ingegneria Vol. 5	Autore Generico 5, Co-Autore 5	2025	9780000000005	5
Libro di Ingegneria Vol. 6	Autore Generico 6, Co-Autore 6	2025	9780000000006	4
Libro di Ingegneria Vol. 7	Autore Generico 7, Co-Autore 7	2025	9780000000007	5
Libro di Ingegneria Vol. 8	Autore Generico 8, Co-Autore 8	2025	9780000000008	5
Libro di Ingegneria Vol. 9	Autore Generico 9, Co-Autore 9	2025	9780000000009	5
Libro di Ingegneria Vol. 10	Autore Generico 10, Co-Autore 10	2025	9780000000010	5
Libro di Ingegneria Vol. 11	Autore Generico 11, Co-Autore 11	2025	9780000000011	5
Libro di Ingegneria Vol. 12	Autore Generico 12, Co-Autore 12	2025	9780000000012	5
Libro di Ingegneria Vol. 13	Autore Generico 13, Co-Autore 13	2025	9780000000013	5
Libro di Ingegneria Vol. 14	Autore Generico 14, Co-Autore 14	2025	9780000000014	5
Libro di Ingegneria Vol. 15	Autore Generico 15, Co-Autore 15	2025	9780000000015	5
Libro di Ingegneria Vol. 16	Autore Generico 16, Co-Autore 16	2025	9780000000016	5
Libro di Ingegneria Vol. 17	Autore Generico 17, Co-Autore 17	2025	9780000000017	5
Libro di Ingegneria Vol. 18	Autore Generico 18, Co-Autore 18	2025	9780000000018	5
Libro di Ingegneria Vol. 19	Autore Generico 19, Co-Autore 19	2025	9780000000019	5
Libro di Ingegneria Vol. 20	Autore Generico 20, Co-Autore 20	2025	9780000000020	5
Libro di Ingegneria Vol. 21	Autore Generico 21, Co-Autore 21	2025	9780000000021	5
Libro di Ingegneria Vol. 22	Autore Generico 22, Co-Autore 22	2025	9780000000022	5
Libro di Ingegneria Vol. 23	Autore Generico 23, Co-Autore 23	2025	9780000000023	5
Libro di Ingegneria Vol. 24	Autore Generico 24, Co-Autore 24	2025	9780000000024	5
Libro di Ingegneria Vol. 25	Autore Generico 25, Co-Autore 25	2025	9780000000025	5
Libro di Ingegneria Vol. 26	Autore Generico 26, Co-Autore 26	2025	9780000000026	5
Libro di Ingegneria Vol. 27	Autore Generico 27, Co-Autore 27	2025	9780000000027	5
Libro di Ingegneria Vol. 28	Autore Generico 28, Co-Autore 28	2025	9780000000028	5
Libro di Ingegneria Vol. 29	Autore Generico 29, Co-Autore 29	2025	9780000000029	5
Libro di Ingegneria Vol. 30	Autore Generico 30, Co-Autore 30	2025	9780000000030	5
Libro di Ingegneria Vol. 31	Autore Generico 31, Co-Autore 31	2025	9780000000031	5
Libro di Ingegneria Vol. 32	Autore Generico 32, Co-Autore 32	2025	9780000000032	5

+ Aggiungi Libro **Elimina Selezionato** Gestione Copie: + Copie - Copie

2.3) Gestione Libri - Inserimento

Gestione Libri - Gruppo 11

AGGIUNGI NUOVO LIBRO

Titolo Opera: Ingegneria Del Software vol.11

Autori (virgola): Autore Generico

Codice ISBN: 1234567891011 13 su 13

Data Pubblicazione: 09/12/2025

Copie Iniziali: 10

Annulla Salva Libro

3) Gestione Utenti

< Indietro

ANAGRAFICA UTENTI

Filtra per:

Cognome

Testo da cercare...

Cerca

Matricola	Nome	Cognome	Email
0000000001	Studente	Numero 1	studente1@unisa.it
0000000002	Studente	Numero 2	studente2@unisa.it
0000000003	Studente	Numero 3	studente3@unisa.it
0000000004	Studente	Numero 4	studente4@unisa.it
0000000005	Studente	Numero 5	studente5@unisa.it
0000000006	Studente	Numero 6	studente6@unisa.it
0000000007	Studente	Numero 7	studente7@unisa.it
0000000008	Studente	Numero 8	studente8@unisa.it
0000000009	Studente	Numero 9	studente9@unisa.it
0000000010	Studente	Numero 10	studente10@unisa.it
0000000011	Studente	Numero 11	studente11@unisa.it
0000000012	Studente	Numero 12	studente12@unisa.it
0000000013	Studente	Numero 13	studente13@unisa.it
0000000014	Studente	Numero 14	studente14@unisa.it
0000000015	Studente	Numero 15	studente15@unisa.it
0000000016	Studente	Numero 16	studente16@unisa.it
0000000017	Studente	Numero 17	studente17@unisa.it
0000000018	Studente	Numero 18	studente18@unisa.it
0000000019	Studente	Numero 19	studente19@unisa.it
0000000020	Studente	Numero 20	studente20@unisa.it
0000000021	Studente	Numero 21	studente21@unisa.it
0000000022	Studente	Numero 22	studente22@unisa.it
0000000023	Studente	Numero 23	studente23@unisa.it
0000000024	Studente	Numero 24	studente24@unisa.it
0000000025	Studente	Numero 25	studente25@unisa.it
0000000026	Studente	Numero 26	studente26@unisa.it
0000000027	Studente	Numero 27	studente27@unisa.it
0000000028	Studente	Numero 28	studente28@unisa.it
0000000029	Studente	Numero 29	studente29@unisa.it
0000000030	Studente	Numero 30	studente30@unisa.it

3.1) Gestione Utenti - Ricerca

< Indietro

ANAGRAFICA UTENTI

Matricola	Nome	Cognome	Filtro per	Cognome	1	2	3
0000000001	Studiante	Numero 1	Matricola	studente1	1		
0000000010	Studiante	Numero 10	Email	studente10@unisa.it			
0000000011	Studiante	Numero 11	Matricola	studente11@unisa.it			
0000000012	Studiante	Numero 12	Email	studente12@unisa.it			
0000000013	Studiante	Numero 13	Matricola	studente13@unisa.it			
0000000014	Studiante	Numero 14	Email	studente14@unisa.it			
0000000015	Studiante	Numero 15	Matricola	studente15@unisa.it			
0000000016	Studiante	Numero 16	Email	studente16@unisa.it			
0000000017	Studiante	Numero 17	Matricola	studente17@unisa.it			
0000000018	Studiante	Numero 18	Email	studente18@unisa.it			
0000000019	Studiante	Numero 19	Matricola	studente19@unisa.it			
0000000021	Studiante	Numero 21	Email	studente21@unisa.it			

+ Nuovo Utente

Elimina Utente

Blocco Fm: SI

3.2) Gestione Utenti - Cancellazione

< Indietro ANAGRAFICA UTENTI Filtra per:

Matricola	Nome	Cognome	Email
0000000001	Studiante	Numero 1	studente1@unisa.it
0000000002	Studiante	Numero 2	studente2@unisa.it
0000000003	Studiante	Numero 3	studente3@unisa.it
0000000004	Studiante	Numero 4	studente4@unisa.it
0000000005	Studiante	Numero 5	studente5@unisa.it
0000000006	Studiante	Numero 6	studente6@unisa.it
0000000007	Studiante	Numero 7	studente7@unisa.it
0000000008	Studiante	Numero 8	studente8@unisa.it
0000000009	Studiante	Numero 9	studente9@unisa.it
0000000010	Studiante	Numero 10	studente10@unisa.it
0000000011	Studiante	Numero 11	studente11@unisa.it
0000000012	Studiante	Numero 12	studente12@unisa.it
0000000013	Studiante	Numero 13	studente13@unisa.it
0000000014	Studiante	Numero 14	studente14@unisa.it
0000000015	Studiante	Numero 15	studente15@unisa.it
0000000016	Studiante	Numero 16	studente16@unisa.it
0000000017	Studiante	Numero 17	studente17@unisa.it
0000000018	Studiante	Numero 18	studente18@unisa.it
0000000019	Studiante	Numero 19	studente19@unisa.it
0000000020	Studiante	Numero 20	studente20@unisa.it
0000000021	Studiante	Numero 21	studente21@unisa.it
0000000022	Studiante	Numero 22	studente22@unisa.it
0000000023	Studiante	Numero 23	studente23@unisa.it
0000000024	Studiante	Numero 24	studente24@unisa.it
0000000025	Studiante	Numero 25	studente25@unisa.it
0000000026	Studiante	Numero 26	studente26@unisa.it
0000000027	Studiante	Numero 27	studente27@unisa.it
0000000028	Studiante	Numero 28	studente28@unisa.it
0000000029	Studiante	Numero 29	studente29@unisa.it
0000000030	Studiante	Numero 30	studente30@unisa.it

+ Nuovo Utente **Elimina Utente**

3.3) Gestione Utenti - Inserimento

Gestione Utenti - Gruppo 11

REGISTRAZIONE NUOVO UTENTE

Nome:

Cognome:

Matricola: 10 su 10

Email Istituzionale:

4) Gestione Prestiti

4.1) Gestione Prestiti - Inserimento

1. Selezione Studente

Cognome

Cerca studente...

Mate.	Cognome	Nome
0000000001	Numero 1	Studiante
0000000002	Numero 2	Studiante
0000000003	Numero 3	Studiante
0000000004	Numero 4	Studiante
0000000005	Numero 5	Studiante
0000000006	Numero 6	Studiante
0000000007	Numero 7	Studiante
0000000008	Numero 8	Studiante
0000000009	Numero 9	Studiante
0000000010	Numero 10	Studiante
0000000011	Numero 11	Studiante
0000000012	Numero 12	Studiante
0000000013	Numero 13	Studiante
0000000014	Numero 14	Studiante
0000000015	Numero 15	Studiante
0000000016	Numero 16	Studiante
0000000017	Numero 17	Studiante
0000000018	Numero 18	Studiante
0000000019	Numero 19	Studiante
0000000020	Numero 20	Studiante
0000000021	Numero 21	Studiante
0000000022	Numero 22	Studiante
0000000023	Numero 23	Studiante
0000000024	Numero 24	Studiante
0000000025	Numero 25	Studiante
0000000026	Numero 26	Studiante
0000000027	Numero 27	Studiante

2. Selezione Libro

Titolo

Cerca libro...


Titolo	ISBN	Disp.
Libro di Ingegneria Vol. 1	9780000000001	5
Libro di Ingegneria Vol. 2	9780000000002	5
Libro di Ingegneria Vol. 3	9780000000003	4
Libro di Ingegneria Vol. 4	9780000000004	5
Libro di Ingegneria Vol. 5	9780000000005	5
Libro di Ingegneria Vol. 6	9780000000006	4
Libro di Ingegneria Vol. 7	9780000000007	5
Libro di Ingegneria Vol. 8	9780000000008	5
Libro di Ingegneria Vol. 9	9780000000009	5
Libro di Ingegneria Vol. 10	9780000000010	5
Libro di Ingegneria Vol. 11	9780000000011	5
Libro di Ingegneria Vol. 12	9780000000012	5
Libro di Ingegneria Vol. 13	9780000000013	5
Libro di Ingegneria Vol. 14	9780000000014	5
Libro di Ingegneria Vol. 15	9780000000015	5
Libro di Ingegneria Vol. 16	9780000000016	5
Libro di Ingegneria Vol. 17	9780000000017	5
Libro di Ingegneria Vol. 18	9780000000018	5
Libro di Ingegneria Vol. 19	9780000000019	5
Libro di Ingegneria Vol. 20	9780000000020	5
Libro di Ingegneria Vol. 21	9780000000021	5
Libro di Ingegneria Vol. 22	9780000000022	5
Libro di Ingegneria Vol. 23	9780000000023	5
Libro di Ingegneria Vol. 24	9780000000024	5
Libro di Ingegneria Vol. 25	9780000000025	5
Libro di Ingegneria Vol. 26	9780000000026	5
Libro di Ingegneria Vol. 27	9780000000027	5

Annulla


Conferma Prestito


3


4.2) Gestione Prestiti - Cancellazione

< Indietro REGISTRO PRESTITI Cerca in: Utente (Cognome) Testo da cercare... 

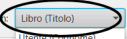


Utente	Libro	Scadenza
Studente Numero 10	Libro di Ingegneria Vol. 3	2026-02-02
Studente Numero 11	Libro di Ingegneria Vol. 6	2026-02-03
Studente Numero 1	Libro di Ingegneria Vol. 1	2026-02-03

1 

2 


+ Nuovo Prestito  Elimina / Concludi


4.3) Gestione Prestiti - Ricerca

< Indietro REGISTRO PRESTITI Cerca in:   

Utente	Libro	Scadenza
Studente Numero 10	Libro di Ingegneria Vol. 3	2026-02-02
Studente Numero 11	Libro di Ingegneria Vol. 6	2026-02-03
Studente Numero 1	Libro di Ingegneria Vol. 1	2026-02-03

4

+ Nuovo Prestito  Elimina / Concludi

1  2 3