Documento di Design

Sistema di Gestione di una Biblioteca Universitaria

Introduzione

Il seguente documento descrive l'architettura software, le classi, le interazioni e i flussi operativi del sistema di gestione di una biblioteca universitaria.
Deriva dai requisiti funzionali, e in generale dalle informazioni presenti nel documento SRS, ed è utilizzato come base per la fase di implementazione.

Ambito

Il sistema permette al bibliotecario di gestire:

- il catalogo della biblioteca (registrazione, modifica, ricerca, eliminazione dei libri);
- gli utenti (registrazione, modifica, ricerca, eliminazione);
- i prestiti (registrazione prestito, registrazione restituzione, visualizzazione prestiti attivi);

Attori

- Bibliotecario: unico utilizzatore del sistema.

Architettura del Sistema

Stile architetturale

Il sistema segue un'architettura a livelli.
Segue il paradigma Model-View-Controller (MVC). Permette una separazione della logica dell'applicazione in tre livelli principali:

- Model (package gruppo20.biblioteca.model)
    - Contiene la logica di business e le entità: Libro,Utente,Prestito e i gestori Libreria, Utenti e Prestiti;
- View (FXML + JavaFX)
    - Interfaccia grafica JavaFX sviluppata in FXML;
- Controller (package gruppo20.biblioteca.controller)
    - Controlli della View che gestiscono input, validazione e interazioni con il Model;

Vista logica

Il sistema è composto dai seguenti moduli:

1. Modulo Gestione Libri
    1.1. Responsabilità:

Il modulo Libri gestisce tutte le entità e operazioni relative ai libri presenti nella libreria.
Si occupa della registrazione, modifica, eliminazione dei dati dei libri della biblioteca.

1.2. Utilizza le entità: Libro, Autore, setLibreria.

2. Modulo Gestione Utenti
   2.1. Responsabilità:
   Il modulo Utenti gestisce tutte le entità e operazioni relative agli utenti che frequentano la biblioteca.
   Si occupa della registrazione, modifica, eliminazione dei dati nel catalogo utenti.
   2.2. Utilizza le entità: Utente, setUtenti.

3. Modulo Gestione Prestiti
   3.1. Responsabilità:
   Il modulo Prestiti gestisce tutte le entità e operazioni relative ai prestiti effettuati dagli utenti della biblioteca riguardanti i libri presenti nel catalogo della libreria.
   Si occupa della registrazione prestito, registrazione restituzione (con la specifica della data in cui avviene tale restituzione)
   3.2. Utilizza le entità: Prestito, setPrestiti.

Descrizione diagramma a livelli

1. **Model**
   1.1. Contiene la logica applicativa;
   1.2. Responsabile dei dati dell'applicazione, della risposta a richieste di informazioni provenienti da altri componenti, come la View e il Controller;
   1.3. Applica regole di business.
2. **View**
   2.1. Gestisce la presentazione, la visualizzazione delle entità gestite dal Model;
   2.2. Permette la ricerca;
   2.3. Controlla la validità dei dati inseriti;
   2.4. Riceve input dal bibliotecario;
   2.5. Mostra i dati provenienti dal Model all'utente e invia gli input dell'utente al Controller.
3. **Controller**
   3.1. Funge da intermediario tra il Model e la View;
   3.2. Riceve richieste dall'utente, le interpreta e aggiorna Model e View di conseguenza;
   3.3. Avvia il salvataggio dei dati dopo ogni operazione;
   3.4. Carica i dati all'avvio;

Vista di sviluppo

Il progetto utilizza Git come sistema per la gestione del codice sorgente. Permette di salvare le versioni del codice e tenerne traccia.

Il sistema di build utilizzato è Maven. Gestisce compilazione, dipendenze e struttura del progetto.

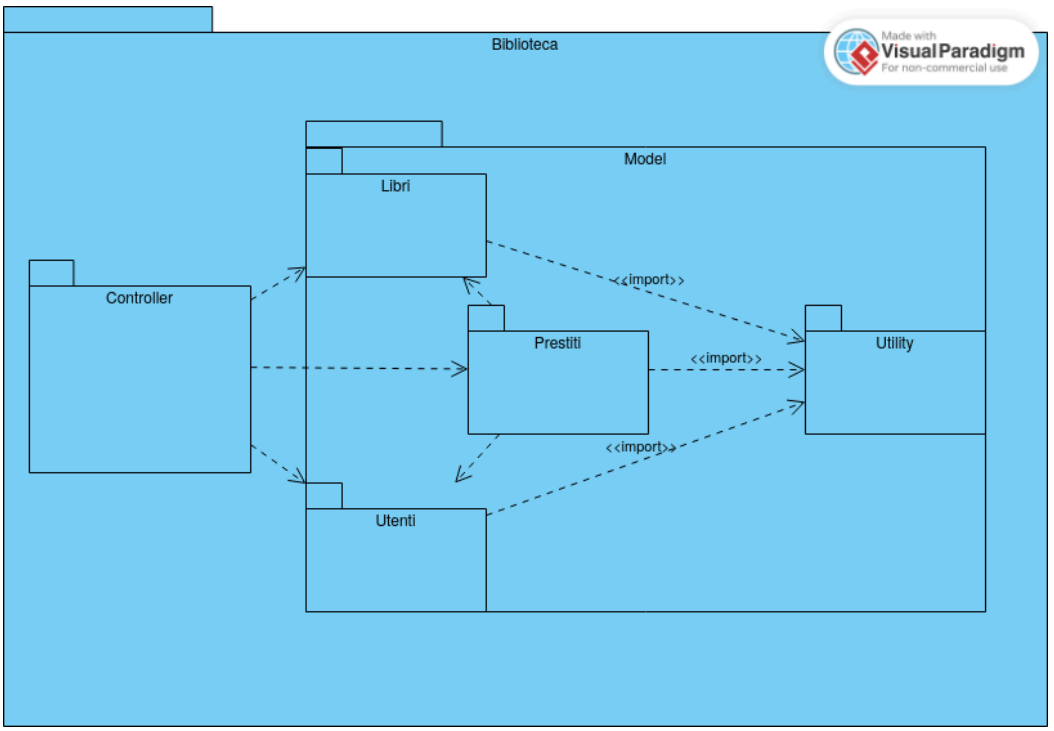
Per la gestione dei dati su disco si utilizza un database interno in sqlite.
Il salvataggio dei dati sul file è effettuato tramite le implementazioni dei metodi della classe astratta di GestioneDB, la cui descrizione esaustiva è possibile ritrovarla nella documentazione Doxygen.

Diagramma dei Package

Il sistema è strutturato in package. La struttura dei package è organizzata come segue:

- gruppo20.biblioteca.model
    - Main.java
    - App.java
- gruppo20.biblioteca.model.Utility
    - GestioneDB.java
- gruppo20.biblioteca.model.Libri
    - Libreria.java
    - Libro.java
    - Autore.java
- gruppo20.biblioteca.model.Prestiti
    - Prestiti.java
    - Prestito.java
    - EffettivaRestituzione.java
- gruppo 20.biblioteca.model.Utenti
    - Utente.java
    - Utenti.java
- gruppo 20.biblioteca.controller
    - Contesto.java
    - ControllerDashboard.java
    - ControllerLibreria.java
    - ControllerPrestiti.java
    - ControllerUtenti.java

Il diagramma relativo è riportato nella figura sottostante.



Modello statico

Il modello statico descrive la struttura del sistema, evidenziando le entità principali (classi), i loro attributi, metodi e le relazioni tra di esse. Questa parte del documento si concentra sul diagramma delle classi, strumento utilizzato per rappresentare la struttura del modello, permettendo di evidenziare la composizione e le interazioni tra le entità del sistema.

Per le dimensioni considerevoli del diagramma delle classi, l'immagine completa non è inserita direttamente nel documento. È possibile consultare il diagramma completo, presente nei file di documentazione allegati, al codice sorgente, su github.

Diagramma delle Classi - descrizione delle Classi

Il diagramma, realizzato con l'utilizzo dello strumento EasyUML, è la rappresentazione grafica delle interazioni tra i vari componenti del sistema.
I dettagli di tali componenti e delle loro interazioni sono di seguito riportati.

Classe Libro

- Responsabilità: rappresenta un'entità Libro presente in biblioteca.
- Attributi principali:
    - String titolo;
    - ArrayList<Autore> autori;
    - LocalDate annoPubblicazione;
    - int nCopie;
    - String isbn.
- Metodi pubblici:
    - Getters;

    - Setter di Autori ed nCopie;
    - equals;
    - hashCode;
    - toString;
- Relazioni rilevanti:
    - Usa la classe Autore;

Classe Autore

- Responsabilità: rappresenta un'entità semplice che modella un autore tramite le informazioni di base quali il nome e il cognome.
- Attributi principali:
    - String nome;
    - String cognome.
- Metodi pubblici:
    - Getters;
    - Autore convert(String s);
    - equals;
    - hashCode;
    - toString;
- Relazioni rilevanti:
    - Associata a Libro.

Classe Utente

- Responsabilità: rappresenta un'entità Utente registrata al sistema.
- Attributi principali:
    - String nome;
    - String cognome;
    - String matricola;
    - String mail;
    - int nPrestiti;
- Metodi pubblici:
    - Getters;
    - equals;
    - hashCode;
    - toString;

Classe Prestito

- Responsabilità: traccia l'assegnazione di un libro a un utente, rappresenta un singolo prestito.
- Attributi principali:
    - LocalDate dataPrestito;
    - LocalDate dataPrevistaRestituzione;
    - EffettivaRestituzione effettivaRestituzione;
    - String isbn;
    - String titoloLibro;
    - String matricola;
    - int ritardo.
- Metodi pubblici:

    - Getters;
    - setEffettivaRestituzione;
    - equals;
    - isRitardo;
    - setRitardo;
    - hashCode.
- Relazioni rilevanti:
    - Usa la classe EffettivaRestituzione.
    - Associazione uno-a-uno con Utente (un prestito appartiene a un solo utente);
    - Associazione uno-a-uno con Libro (ogni prestito riguarda un solo libro).

Classe EffettivaRestituzione

- Responsabilità: rappresenta lo stato di restituzione effettiva di un libro e gestisce la data effettiva di restituzione.
- Attributi principali:
    - boolean restituito;
    - LocalDate dataEffettivaRestituzione;
- Metodi pubblici:
    - getter;
    - setEffettivaRestituzione;
    - isRestituito;
- Relazioni rilevanti:
    - Associata a Prestito.

Gestori

Quali Libreria, Utenti, Prestiti. Permettono l'aggiunta di dati, l'eliminazione e la modifica. Tutti i gestori lavorano con un ObservableSet al cui interno,per motivi di efficienza, c'è un HashSet.

Libreria

- Responsabilità: gestisce l'insieme dei libri attraverso un ObservableSet, gestisce operazioni sui libri, quali aggiungi, elimina e modifica e ne gestisce la coordinazione delle operazioni per il salvataggio e il caricamento su file.
- Metodi:
    - aggiungi;
    - elimina;
    - modifica;
    - getSetLibreria;
    - carica;
    - addRestituzione;
    - addPrestito.
- Relazioni rilevanti:
    - Con Libro: gestisce una collezioni di libri - associazione uno a molti;
    - Estende GestioneDB: implementa le operazioni di aggiunta, modifica, elimina, carica e utilizza la funzione tableExists della superclasse;

Utenti

- Responsabilità: gestisce l'insieme degli utenti attraverso un ObservableSet, gestisce operazioni sugli utenti, quali aggiungi, elimina e modifica e ne gestisce la coordinazione delle operazioni per il salvataggio e il caricamento su file.
- Metodi:
    - aggiungi;
    - elimina;
    - modifica;
    - getSetUtenti;
    - carica;
    - addRestituzione;
    - addPrestito;
    - toString.
- Relazioni rilevanti:
    - Con Utente: gestisce una collezioni di utenti- associazione uno a molti;
    - Estende GestioneDB: implementa le operazioni di aggiunta, modifica, elimina, carica e utilizza la funzione tableExists della superclasse;

Prestiti

- Responsabilità: gestisce l'insieme dei prestiti attraverso un ObservableSet, gestisce operazioni sui prestiti, quali aggiungi, elimina e modifica e ne gestisce la coordinazione delle operazioni per il salvataggio e il caricamento su file.
- Metodi:
    - aggiungi;
    - elimina;
    - modifica;
    - restituisci;
    - getSetPrestiti;
    - carica.
- Relazioni rilevanti:
    - Con Prestito: gestisce un insieme di prestiti - associazione uno a molti;
    - Estende GestioneDB: implementa le operazioni di aggiunta, modifica, elimina, carica e utilizza la funzione tableExists della superclasse;

Controller

Fungono da intermediari tra la View e i Model.

ControllerLibreria

- Responsabilità: gestisce gli input del bibliotecario.
- Metodi:
    - Initialize;
    - pageDashboard;
    - pageUtenti;
    - pageLibreria;
    - pagePrestiti.
- Relazioni rilevanti:
    - Con Contesto: può accedere al contesto del programma.

ControllerUtenti

- Responsabilità: gestisce gli input del bibliotecario.
- Metodi:
    - Initialize;
    - pageDashboard;
    - pageUtenti;
    - pageLibreria;
    - pagePrestiti.
- Relazioni rilevanti:
    - Con Contesto: può accedere al contesto del programma.

ControllerPrestiti

- Responsabilità: gestisce gli input del bibliotecario.
- Metodi:
    - Initialize;
    - pageDashboard;
    - pageUtenti;
    - pageLibreria;
    - pagePrestiti.
- Relazioni rilevanti:
    - Con Contesto: può accedere al contesto del programma.

Contesto

- Responsabilità: Contiene i dati necessari al programma, utilizzata per inizializzare il tutto una singola volta piuttosto che ricaricare i dati ad ogni cambio di interfaccia
- Metodi:
    - getGestLibreria;
    - getGestUtenti;
    - getGestPrestiti.
- Relazioni rilevanti:
    - Con Prestiti,Utenti e Libreria: vengono inizializzate da contesto all'avvio del programma.

GestioneDB

- Responsabilità: fornisce una funzione per verificare l'esistenza di una tabella in un database sqlite e un'interfaccia, tramite metodi astratti, per la gestione del database e delle rappresentazioni interne al programma.
  La classe fornisce una documentazione su quello che i metodi dovrebbero poter fare; tutte le classi che ereditano da questa si impegnano a rispettare quel contratto.
- Metodi:
    - aggiungi;
    - modifica;
    - elimina;
    - carica;
    - tableExists.
- Relazioni rilevanti:
    - Viene estesa da Libreria,Prestiti e Utenti che ne utilizzano anche il metodo tableExists.

Scelte progettuali

Coesione

Il sistema ha un'alta coesione tra le classi grazie alla progettazione effettuata assegnando ad ogni classe un'unica responsabilità. Ciò aumenta la leggibilità e la manutenibilità del sistema,
In particolare abbiamo:

- classe Libro: rappresenta i dati del libro, non si occupa di operazioni sul catalogo o di prestito;
- classe Utente: rappresenta i dati dello studente e il numero dei suoi prestiti attivi, non si occupa di operazioni sulla lista utenti o dei prestiti;
- classe Prestito: rappresenta l'associazione tra un utente e una copia di un libro;
- Moduli di gestione (Libreria, Utenti, Prestiti): gestiscono solo la logica operativa. I singoli moduli (Libreria, Utenti, Prestiti) estendono GestioneDB e implementano singolarmente i metodi astratti per adattarne il comportamento alle specificità del modulo. In tal modo si garantisce coesione procedurale, è garantita la presenza dei metodi di GestioneDB nelle classi che la estendono ed il loro comportamento generale descritto nella superclasse.
- Controller: gestiscono la comunicazione tra interfaccia e modelli utilizzando il contesto;
- Contesto: Inizializza i gestori all'avvio del programma.

Accoppiamento

Il sistema ha un accoppiamento basso tramite le seguenti scelte progettuali:

- Riduzione al minimo di passaggio di dati tra i moduli;
- Dipendenze dirette solo tra classi che devono necessariamente collaborare, ad esempio:
    - Prestito dipende necessariamente da Libri e Utenti;
- Gli oggetti non accedono ai dati interni di altre classi, ma utilizzano metodi pubblici;
- I moduli di gestione (Libreria, Utenti, Prestiti) fungono da intermediari, riducendo le relazioni dirette tra classi;
- L'interazione con l'interfaccia utente, gestita tramite JavaFX, controlla gli input dell'utente e comunica al controller le operazioni da eseguire;

Principi di buona progettazione

I principi di buona progettazioni seguiti sono:

- Principio della singola responsabilità (SRP): ogni classe ha un unico compito ben definito;
- Principio aperto/chiuso(OCP):
    - Incapsulamento, gli attributi delle classi sono privati (private) e sono accessibili tramite metodi pubblici (getter e setter);
    - Polimorfismo, l'aggiunta di nuove funzionalità avviene senza modificare le entità esistenti (si utilizza l'overriding in classe derivate per la modifica dei metodi);
- Privilegiare l'associazione rispetto all'ereditarietà:

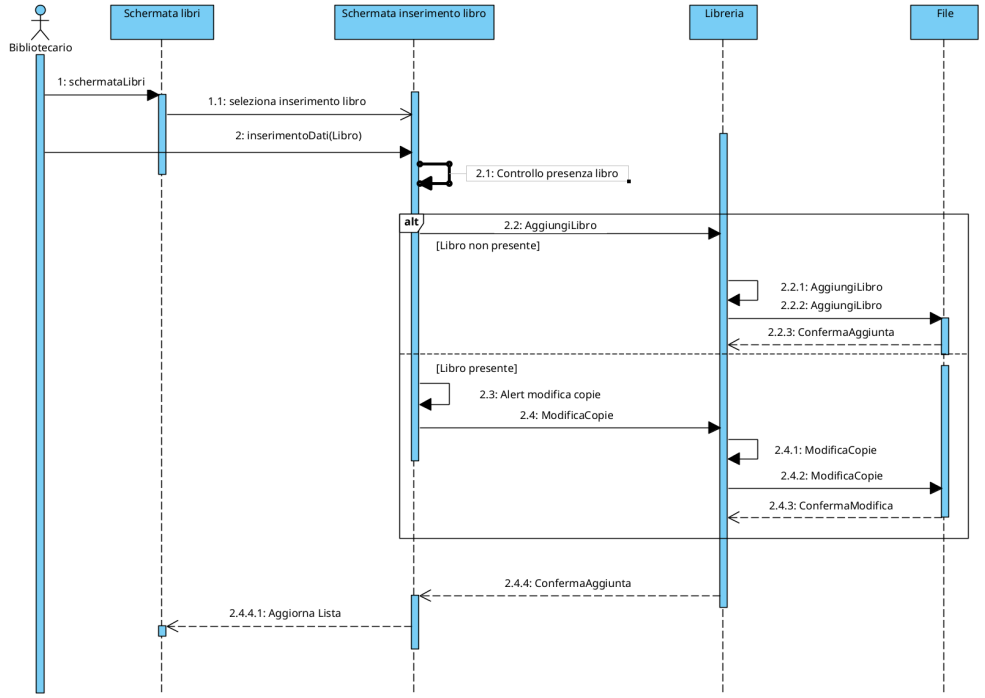
    - abbiamo invece un’associazione con la classe GestioneDB, estesa dalle classi di gestione (Libreria, Utenti, Prestiti), che ne implementano i metodi, per evitare la scrittura di un codice troppo complesso e poco manutenibile (in conformità del principio KISS);
- Robustezza: eventuali input non validi non compromettono il funzionamento dell’applicazione, ciò è possibile grazie a:
    - durante la registrazione (di libri, utenti o prestiti) avviene la verifica dei dati inseriti:
    - in caso di valori non inseriti (campo lasciato vuoto) o valori non validi, l’interfaccia impedisce di schiacciare il tasto per il salvataggio ed evidenzia il settore con informazioni errate/mancanti;
    - il prestito viene registrato solo se lo studente esiste e il numero di prestiti attivo è inferiore a 3, il libro esiste e ha copie disponibili;
    - le operazioni di prestito e restituzione aggiornano sempre sia lo studente che il libro,così da evitare inconsistenze;

Modello dinamico - Diagrammi di Sequenza

Di seguito sono riportate le descrizioni del comportamento del sistema durante l’esecuzione dei casi d’uso più significativi.
Diversamente dal modello statico, il modello dinamico si concentra sulle interazioni temporali tra gli oggetti, mostrando come collaborano per realizzare le funzionalità principali della biblioteca.
Per la rappresentazione di tali interazione vengono utilizzati diagrammi di sequenza UML, che illustrano il flusso dei messaggi tra le classi e interfacce coinvolte in ciascun caso d’uso.
Ogni diagramma è accompagnato da un commento descrittivo che ne spiega:

- lo scopo del caso d’uso;
- i partecipanti coinvolti;
- il flusso principale dei messaggi.

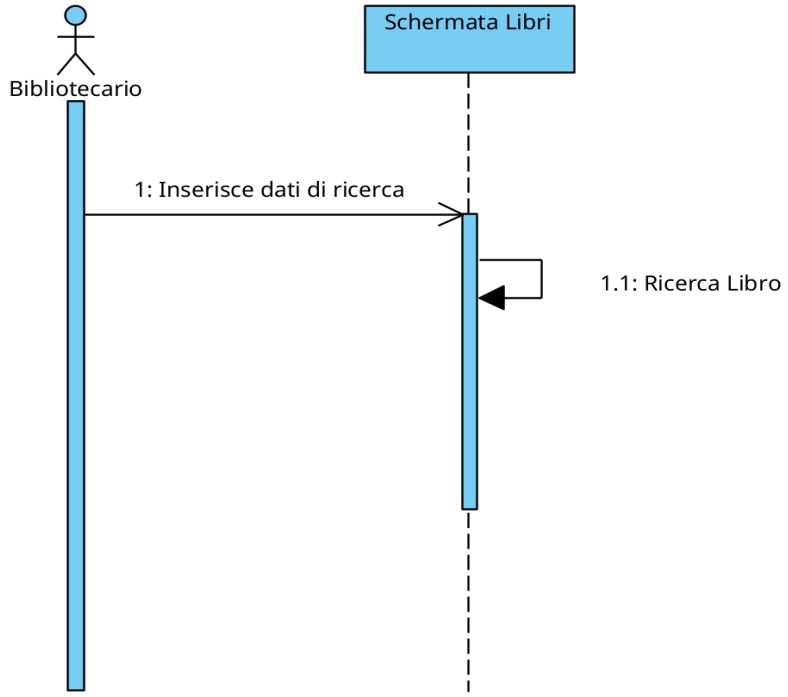
Inserimento di un nuovo libro



- Scopo: Questo diagramma rappresenta il flusso delle interazioni che permettono al bibliotecario di registrare un nuovo libro nel catalogo.
- Partecipanti:
    - Bibliotecario;
    - View schermata gestione libri;
    - View schermata di inserimento dati libro;
    - Gestore Libreria.
- Flusso principale:
    - Il bibliotecario accede alla sezione di gestione dei libri tramite le view;
    - Il bibliotecario richiede la registrazione di un nuovo libro tramite la view;
    - L'interfaccia passa alla SchermataInserimentoLibro, che permette di inserire i dati del volume;
    - Dopo l'inserimento dei dati corretti, la view invia i dati al ControllerLibreria;
    - Il controller controlla se il libro è già presente;
    - Se non è presente, viene chiamato il metodo aggiungi per aggiungere un nuovo libro;
        - Il controller Libreria richiede al gestore l'aggiunta del libro;
        - Il gestore aggiorna la rappresentazione interna ed il file con il nuovo stato del repertorio;
        - Libreria informa la schermata dell'avvenuto inserimento del libro;
        - La view aggiorna i dati mostrati e segnala al bibliotecario che l'operazione è completata.
    - Se il libro è già presente nel catalogo, viene chiesto al bibliotecario se vuole aggiornarne le copie;
        - Se annulla non vengono apportate modifiche;
        - Se conferma il controller chiede al gestore di aggiornare le copie;

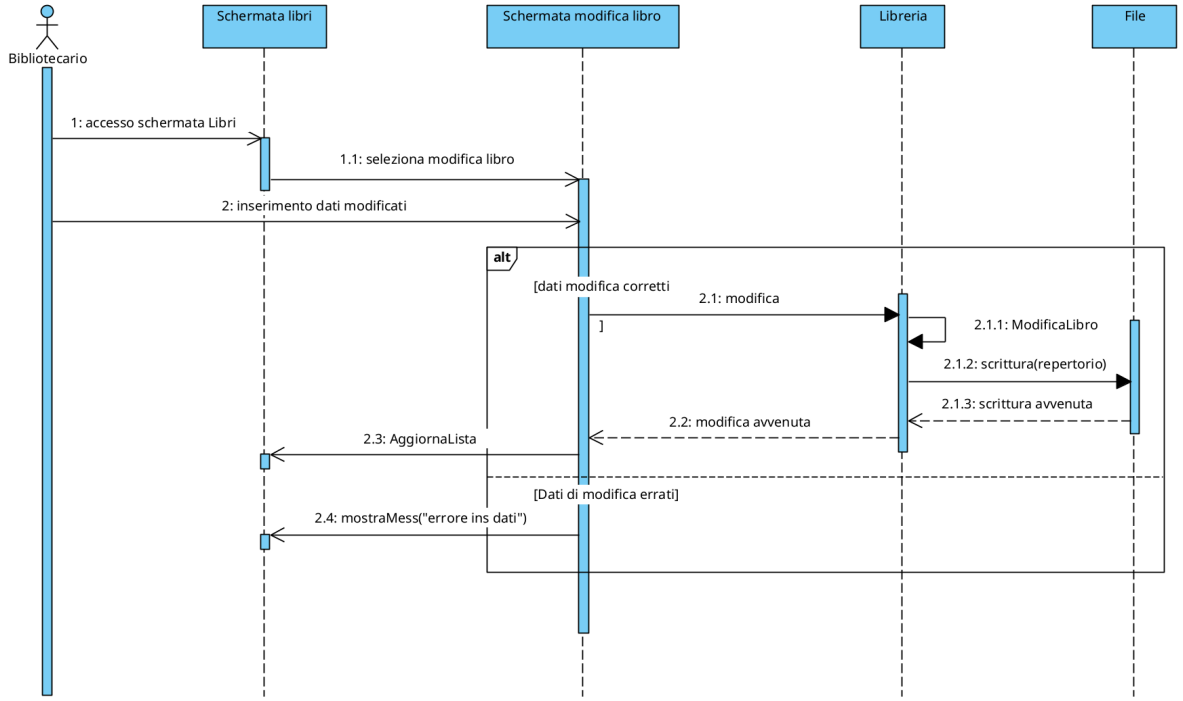
- Il gestore aggiorna la rappresentazione interna ed il file con il nuovo stato del repertorio;
- Libreria informa la schermata dell'avvenuto aggiornamento del libro;
- La view aggiorna i dati mostrati e segnala al bibliotecario che l'operazione è completata.

Ricerca libro



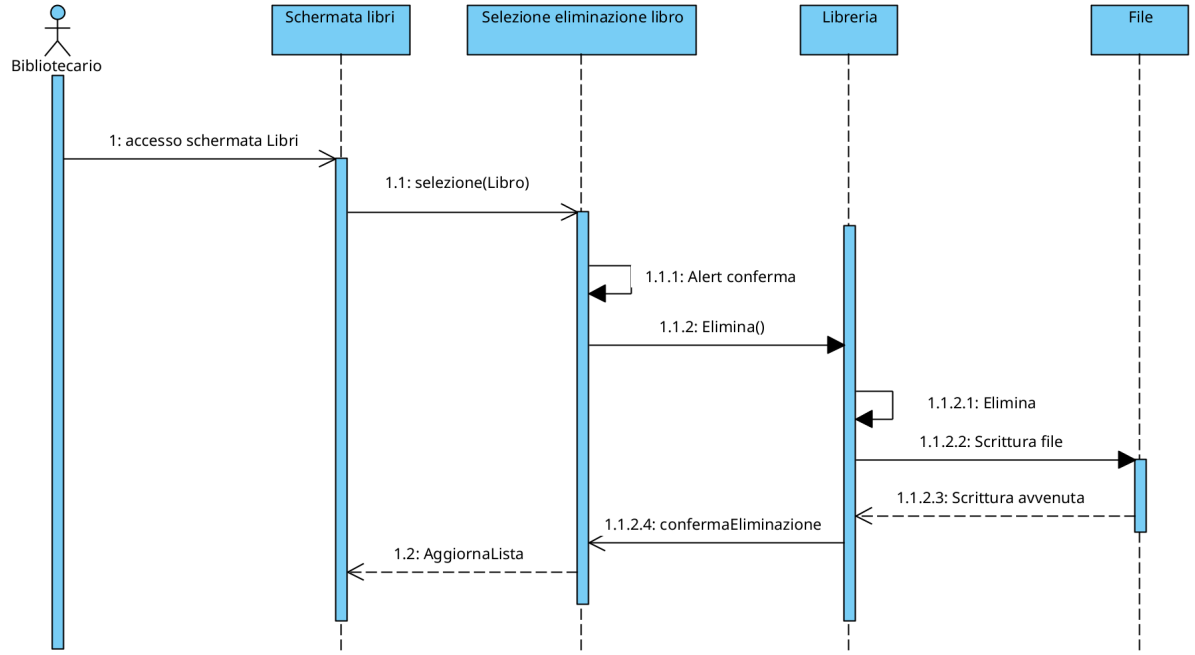
- Scopo: Questo diagramma rappresenta il flusso delle interazioni che permettono al bibliotecario di ricercare un libro nel catalogo.
- Partecipanti:
  - Bibliotecario;
  - View schermata gestione libri.
- Flusso principale:
  - Il bibliotecario accede alla sezione di gestione dei libri tramite le view;
  - Il bibliotecario richiede la ricerca di un libro tramite la barra di ricerca;
  - L'interfaccia permette di inserire il dato o dati del volume da ricercare;
  - Dopo l'inserimento dei dati la view, se il libro è presente, restituisce la relativa voce;
  - Se quei dati non corrispondono ad alcun libro nel catalogo, la schermata non restituisce risultati.

Modifica Libro



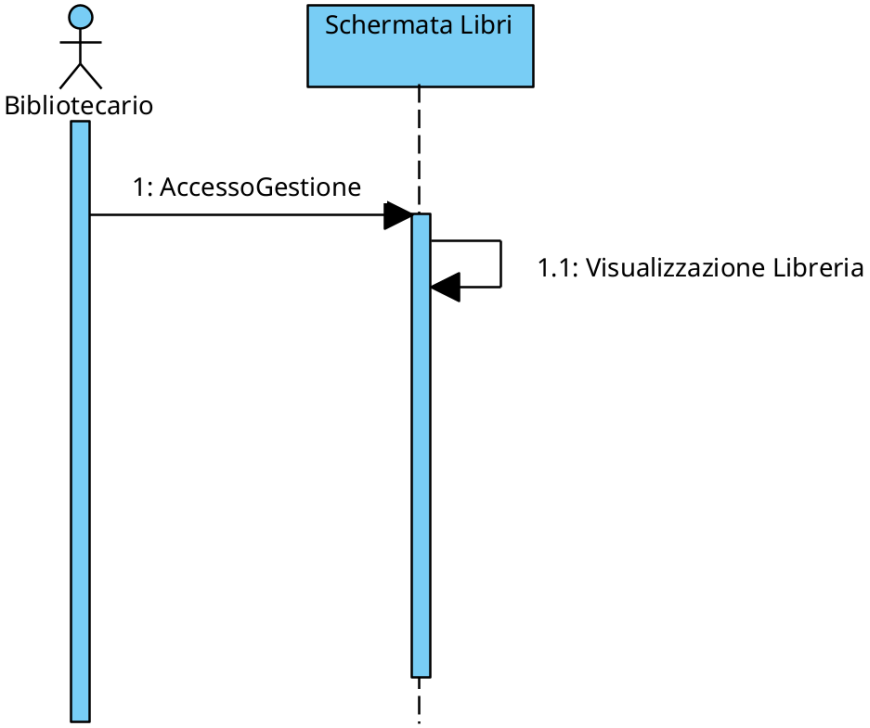
- Scopo: Questo diagramma rappresenta il flusso delle interazioni che permettono al bibliotecario di modificare uno o più dati di un libro presente nel catalogo.
- Partecipanti:
    - Bibliotecario;
    - View schermata gestione libri;
    - View schermata di inserimento dei dati del libro modificati;
    - Gestore Libreria.
- Flusso principale:
    - Il bibliotecario accede alla sezione di gestione dei libri tramite le view;
    - Il bibliotecario richiede la modifica di un libro tramite la view;
    - Dopo l'inserimento dei dati la view controlla che siano corretti;
    - Se risultano corretti, la view invia i dati al ControllerLibreria;
    - Il controller invia i dati al gestore Libreria;
        - Viene chiamato il metodo modifica;
        - Il gestore aggiorna la rappresentazione interna ed il file con il nuovo stato del repertorio;
        - La Libreria informa la schermata libri dell'avvenuta modifica;
        - La view aggiorna i dati visualizzati e segnala al bibliotecario che l'operazione è completata.
    - Se i dati inseriti sono errati la view comunica al bibliotecario l'erroneo inserimento dei dati.

Elimina libro



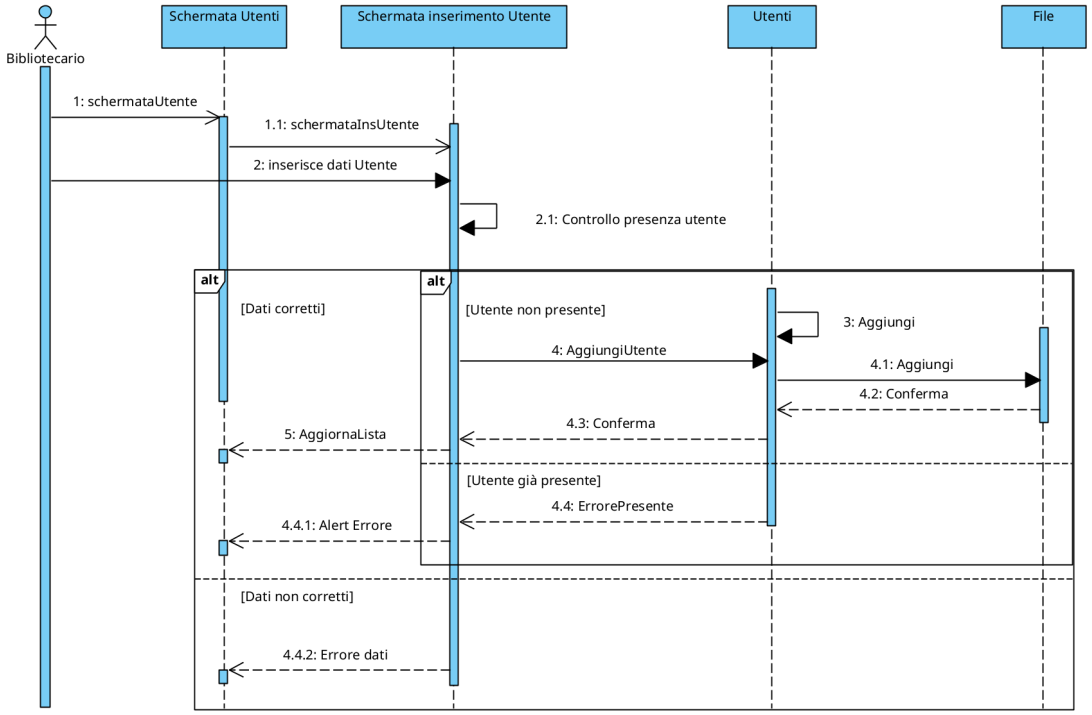
- Scopo: Questo diagramma rappresenta il flusso delle interazioni che permettono al bibliotecario di eliminare i dati di un libro presente nel catalogo.
- Partecipanti:
    - Bibliotecario;
    - View schermata gestione libri;
    - Selezione di eliminazione dei dati del libro;
    - Gestore Libreria.
- Flusso principale:
    - Il bibliotecario accede alla sezione di gestione dei libri tramite le view;
    - Il bibliotecario seleziona il libro da eliminare tramite la schermata di gestione libri;
    - La view manda un messaggio di conferma dell'operazione;
    - Dopo la conferma da parte dell'attore, la view invia i dati da eliminare al ControllerLibreria;
    - Il controller invia i dati al gestore Libreria;
    - Il gestore Libreria aggiorna la rappresentazione interna ed il file;
    - La Libreria informa la schermata libri dell'avvenuta eliminazione;
    - La view aggiorna i dati mostrati e segnala al bibliotecario che l'operazione è completata.

Visualizza Libri



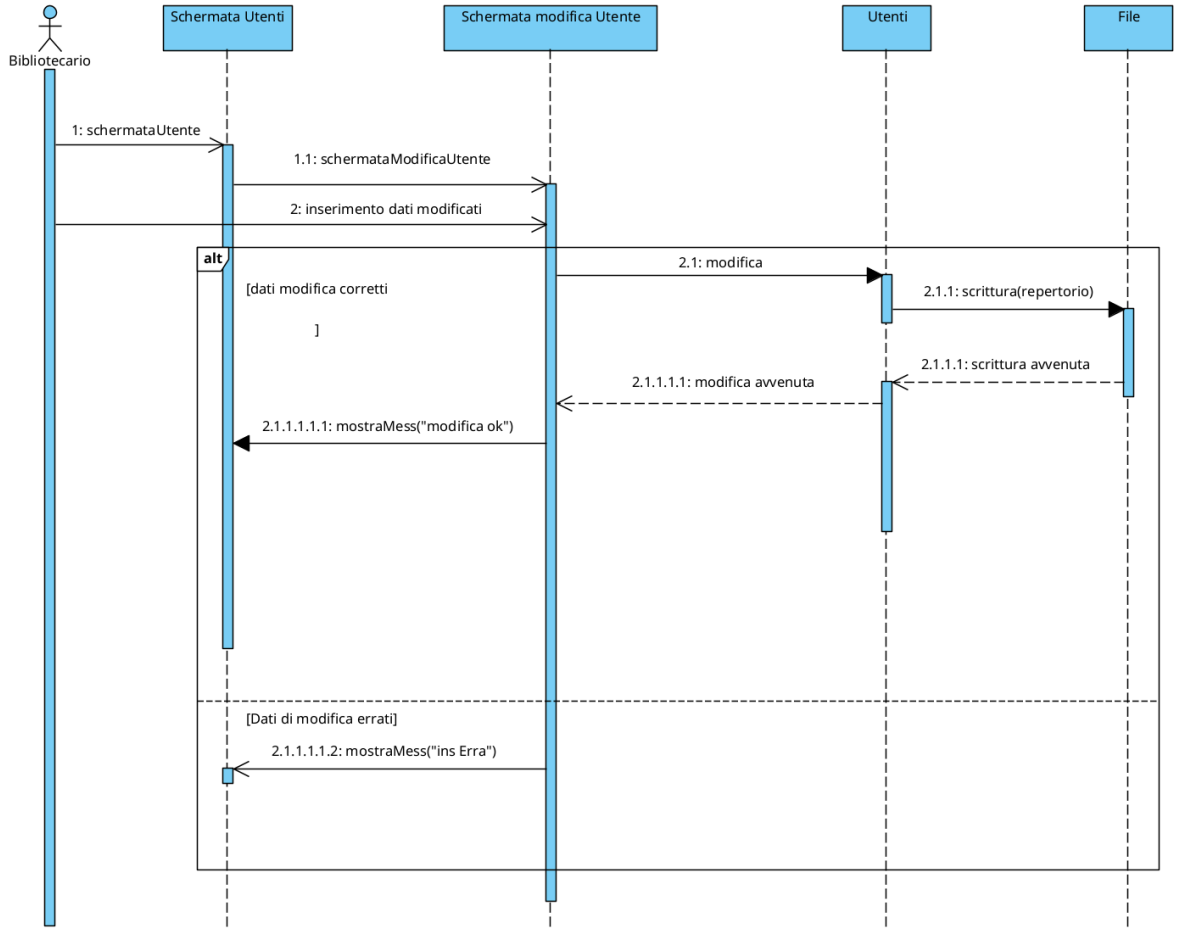
- Scopo: Questo diagramma rappresenta il flusso delle interazioni che permettono al bibliotecario di visualizzare la lista dei libri presenti nel catalogo.
  La visualizzazione avviene direttamente nella schermata iniziale della gestione libri. Questa schermata funge da view e ha la responsabilità di mostrare al bibliotecario l'elenco aggiornato dei libri disponibili nella struttura, all'interno di una tabella.
- Partecipanti:
  - Bibliotecario;
  - View schermata gestione libri.
- Flusso principale:
  - Il bibliotecario accede alla sezione di gestione dei libri tramite le view;
  - Quando viene caricata la schermata di gestione libri viene mostrata automaticamente la lista.

Registrazione utente



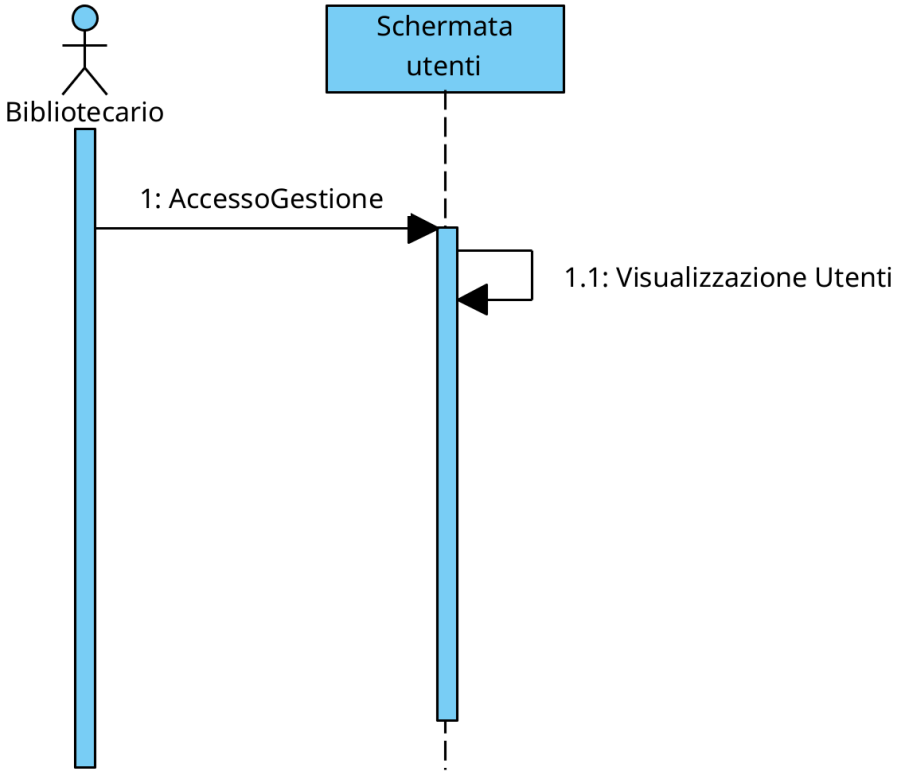
- Scopo: Questo diagramma rappresenta il flusso delle interazioni che permettono al bibliotecario di registrare un nuovo utente nel catalogo.
- Partecipanti:
  - Bibliotecario;
  - View schermata gestione utenti;
  - View schermata di inserimento dati utente;
  - Gestore Utenti.
- Flusso principale:
  - Il bibliotecario accede alla sezione di gestione degli utenti tramite le view;
  - Il bibliotecario richiede la registrazione di un nuovo utente tramite la view;
  - L'interfaccia passa alla SchermataInserimentoUtente, che permette di inserire i dati dello studente da registrare;
  - Dopo l'inserimento dei dati corretti, la view verifica se l'utente è già presente;
    - Se non è presente invia i dati al ControllerUtenti;
    - Nel caso in cui i dati sono errati o l'utente è presente, la view informerà il bibliotecario;
  - Il controller invia i dati al gestore Utenti;
  - Viene chiamato il metodo aggiungi per aggiungerlo;
    - Il gestore Utenti aggiorna la rappresentazione interna ed il file;
    - Utenti informa la schermata dell'avvenuta registrazione utente;
    - La view aggiorna i dati mostrati e segnala al bibliotecario che l'operazione è completata.

Modifica Utente



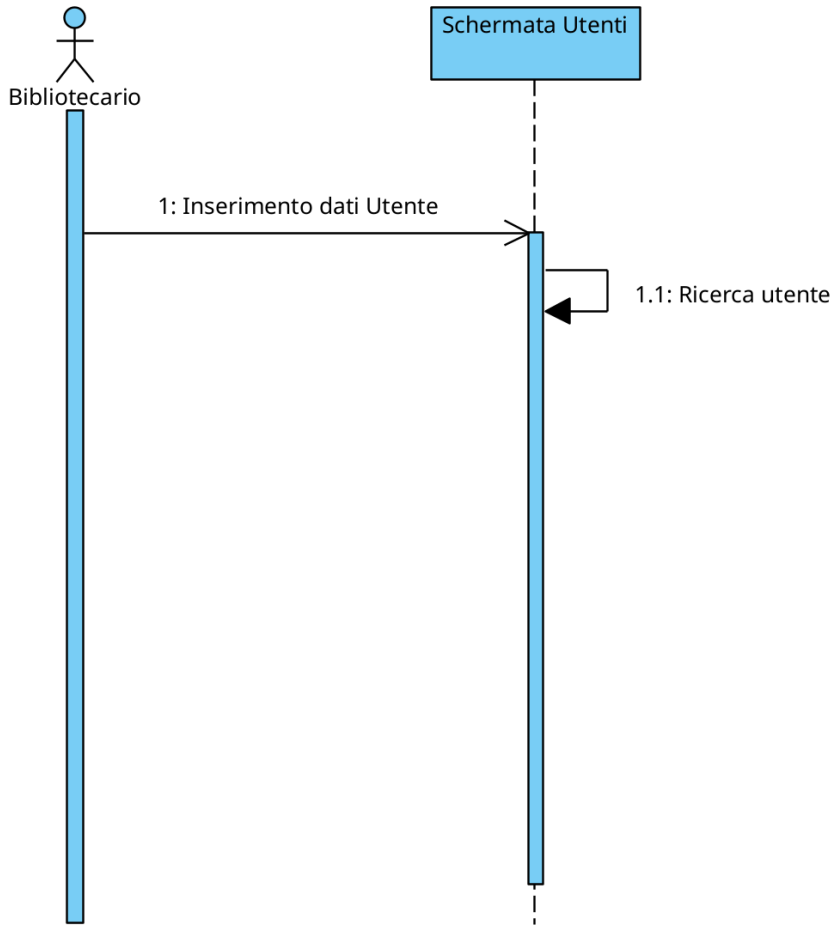
- Scopo: Questo diagramma rappresenta il flusso delle interazioni che permettono al bibliotecario di modificare uno o più dati di un utente presente nell'anagrafica.
- Partecipanti:
    - Bibliotecario;
    - View schermata gestione libri;
    - View schermata di inserimento dei dati dell'utente modificati;
    - Gestore Utenti;
- Flusso principale:
    - Il bibliotecario accede alla sezione di gestione degli utenti tramite le view;
    - Il bibliotecario richiede la modifica;
    - La view lo rimanda ad un'altra schermata, nella quale ci saranno i campi dei dati relativi all'utente, che il bibliotecario potrà modificare;
    - Dopo l'inserimento dei dati la view controlla che siano corretti;
        - Se risultano corretti, la view invia i dati al ControllerUtenti;
        - Se i dati inseriti sono errati la view comunica, con un messaggio, al bibliotecario l'erroneo inserimento dei dati;
    - Il controller invia i dati al gestore Utenti;
    - Utenti aggiorna la rappresentazione interna ed il file;
    - Il gestore Utenti informa la schermata dell'avvenuta modifica;
    - La view aggiorna i dati visualizzati e segnala al bibliotecario che l'operazione è completata.

Visualizzazione Utenti



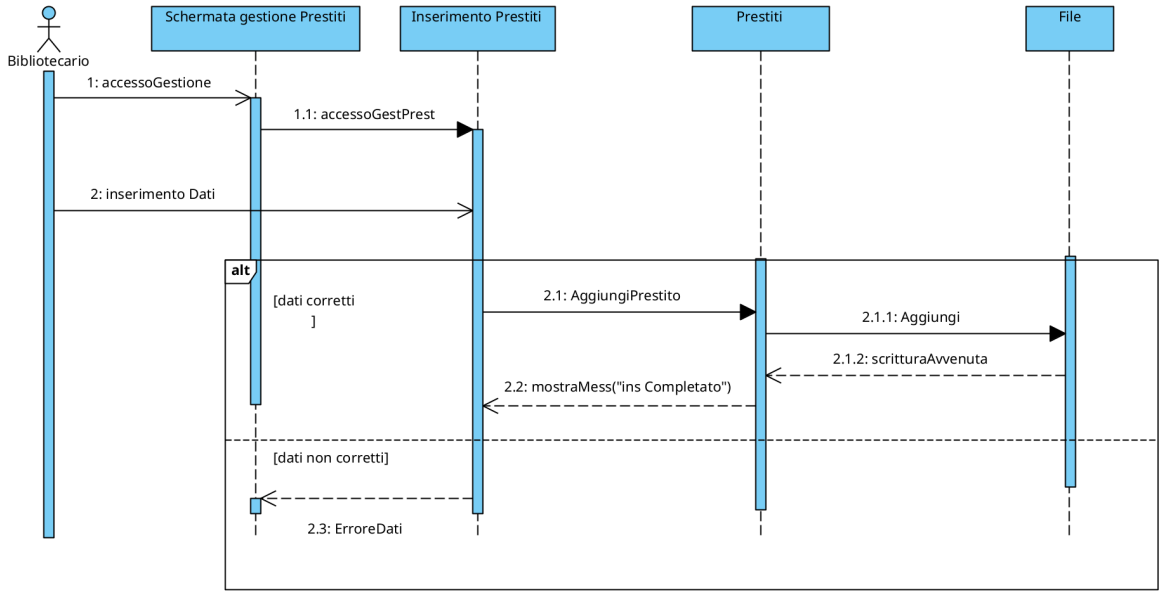
- Scopo: Questo diagramma rappresenta il flusso delle interazioni che permettono al bibliotecario di visualizzare la lista degli utenti registrati.
  La visualizzazione avviene direttamente nella schermata iniziale della gestione utenti. Questa schermata funge da view e ha la responsabilità di mostrare al bibliotecario l'elenco aggiornato, all'interno di una tabella.
- Partecipanti:
  - Bibliotecario;
  - View schermata gestione utenti.
- Flusso principale:
  - Il bibliotecario accede alla sezione di gestione degli utenti tramite le view;
  - Quando viene caricata la schermata di gestione viene mostrata automaticamente la lista.

Ricerca utente



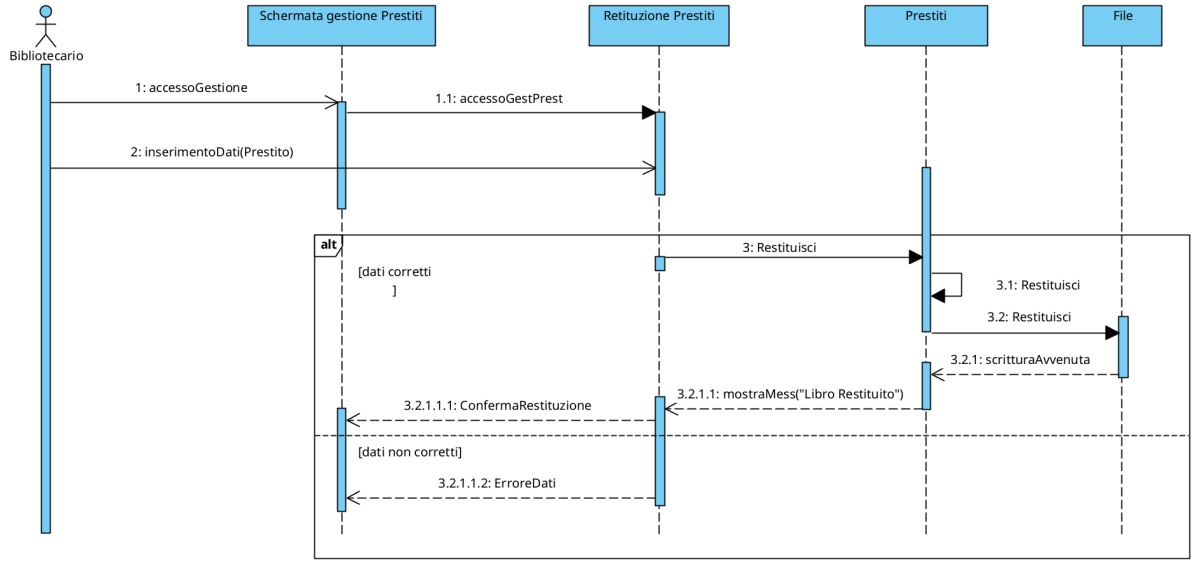
- Scopo: Questo diagramma rappresenta il flusso delle interazioni che permettono al bibliotecario di ricercare un utente registrato.
- Partecipanti:
    - Bibliotecario;
    - View schermata gestione utenti;
- Flusso principale:
    - Il bibliotecario accede alla sezione di gestione degli Utenti tramite le view;
    - Il bibliotecario richiede la ricerca di un utente tramite la barra di ricerca;
    - L'interfaccia permette di inserire il dato o dati dell'utente da ricercare;
    - Dopo l'inserimento dei dati la view, se l'utente è presente, restituisce la relativa voce;
    - Se quei dati non corrispondono ad alcun utente nel catalogo, la schermata non restituisce risultati.

Inserimento prestito



- Scopo: Questo diagramma rappresenta il flusso delle interazioni che permettono al bibliotecario di registrare un nuovo prestito di un libro da parte di un utente registrato.
- Partecipanti:
  - Bibliotecario;
  - View schermata gestione prestiti;
  - View schermata di inserimento dati del prestito;
  - Prestiti.
- Flusso principale:
  - Il bibliotecario accede alla sezione di gestione dei prestiti tramite le view;
  - Il bibliotecario richiede la registrazione di un nuovo prestito tramite la view;
  - L'interfaccia passa alla SchermataInserimentoPrestiti, che permette di inserire i dati del prestito da registrare;
  - Dopo l'inserimento dei dati se risultano corretti, la view invia i dati al ControllerPrestiti;
    - Se i dati inseriti sono errati la view comunica, con un messaggio, al bibliotecario l'erroneo inserimento dei dati;
  - ControllerPrestiti invia tali dati al gestore Prestiti;
  - Il gestore aggiorna la rappresentazione interna ed il file.

Restituzione Prestito



- Scopo: Questo diagramma rappresenta il flusso delle interazioni che permettono al bibliotecario di registrare l'avvenuta restituzione di un prestito.
- Partecipanti:
  - Bibliotecario;
  - View schermata gestione prestiti;
  - View schermata di inserimento dati della restituzione;
  - Gestore Prestiti;
- Flusso principale:
  - Il bibliotecario accede alla sezione di gestione dei prestiti tramite le view;
  - Il bibliotecario seleziona il prestito che bisogna contrassegnare come restituito tramite la schermata di gestione;
  - La view lo rimanda ad un’altra schermata, nella quale si richiede di inserire la data di restituzione;
    - Se la data inserita è errata la view non permette il salvataggio
  - Dopo l’inserimento della data la view invia i dati al ControllerPrestiti;
    - ControllerPrestiti invia tali dati al gestore Prestiti;
    - Prestiti aggiorna la rappresentazione interna ed il file;
    - Il ControllerPrestiti informa la schermata dell’avvenuta modifica;
    - La view aggiorna i dati visualizzati e segnala al bibliotecario che l’operazione è completata.

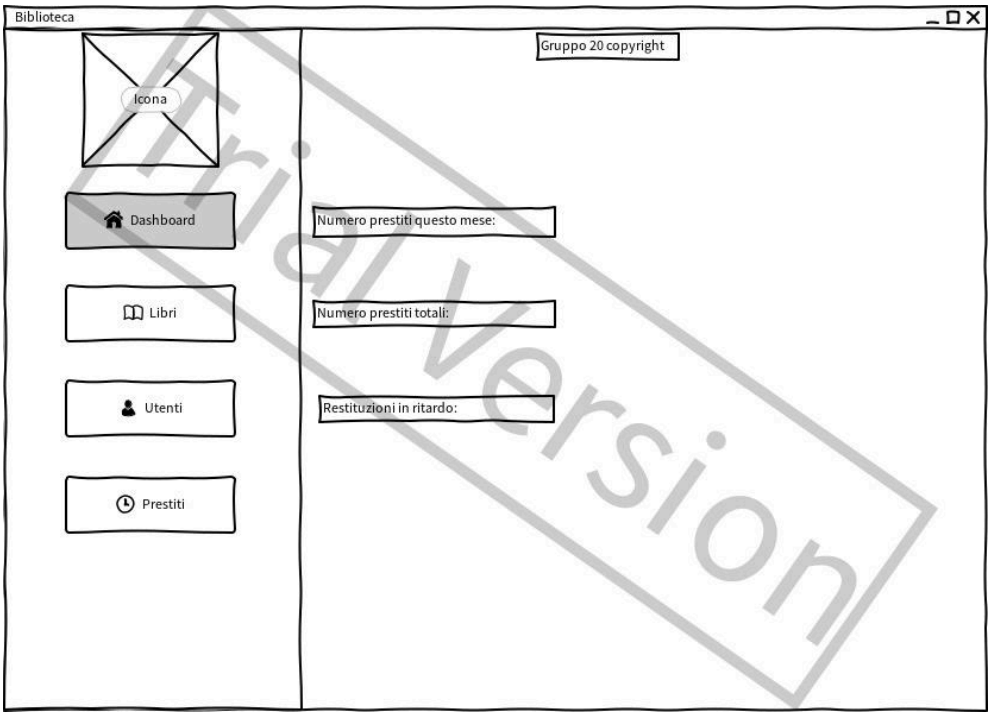
Design dell’interfaccia utente

L’interfaccia utente del sistema sarà sviluppata utilizzando JavaFx, adottando un’architettura MVC (Mdel-View-Controller).
Le view sono responsabili esclusivamente della presentazione grafica.
L’interfaccia è progettata per essere semplice, chiara e conforme al flusso naturale delle operazioni di gestione della libreria.

Mock-up delle schermate e descrizione

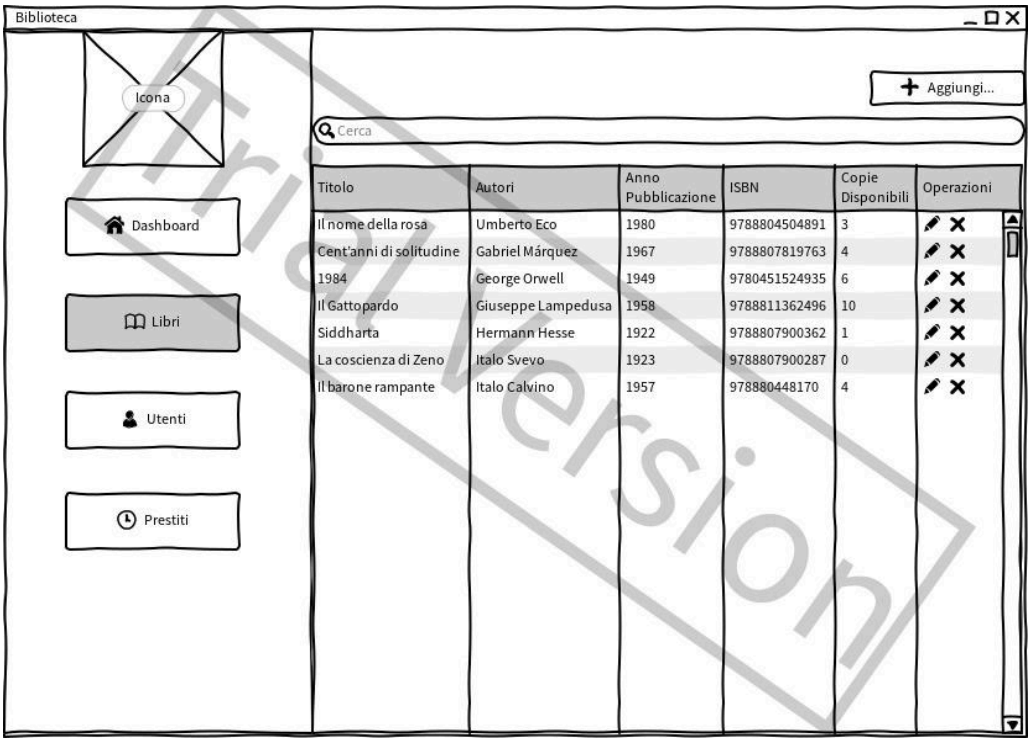
Di seguito sono riportate le schermate con cui l'utente interagirà. Si tratta di semplici schizzi realizzati con wireframesketcher.com/, finalizzati a mostrare in modo chiaro e sintetico l’interfaccia grafica del sistema.
Ogni schermata sarà accompagnata da una breve descrizione delle sue funzionalità e dei principali elementi visivi, evidenziando come l’utente possa interagire con il sistema.

Schermata principale



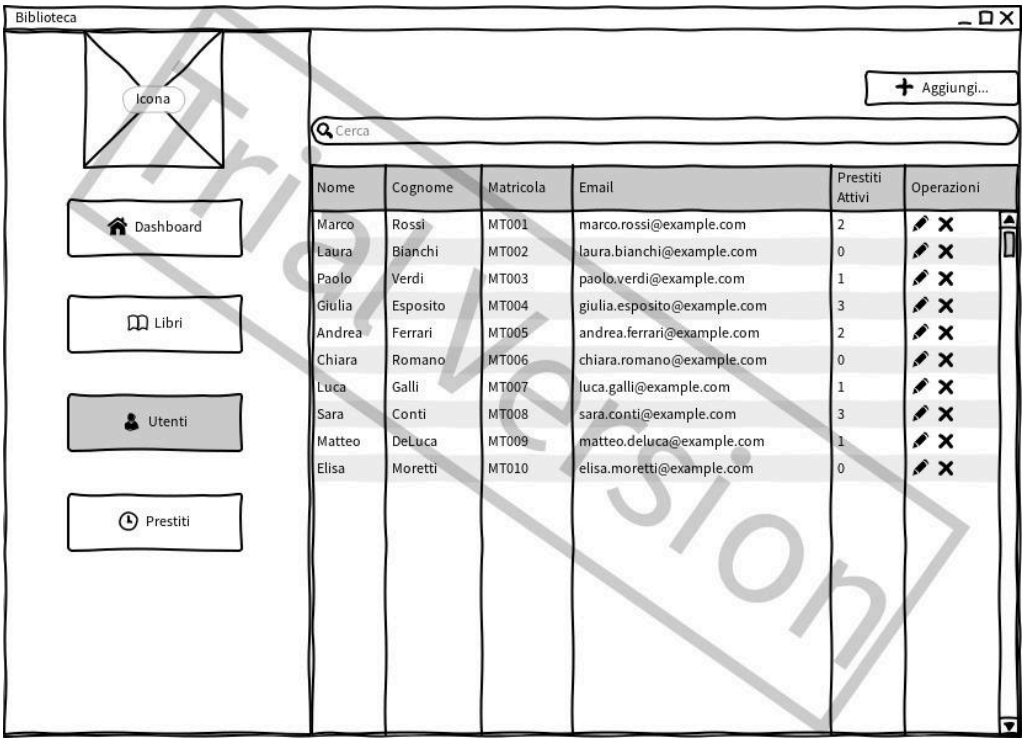
- Descrizione:
  - Funge da punto di ingresso dell’applicazione.
  - Consente al bibliotecario di scegliere quale modulo utilizzare
  - Permette di visualizzare i prestiti in scadenza alla data odierna e quelli scaduti.
- Elementi principali:
  - Pulsanti per accedere alle tre sezioni:
    - Gestione Libri
    - Gestione Utenti
    - Gestione Prestiti
  - Tabelle per visualizzare i dati dei prestiti.

Schermata Gestione libri



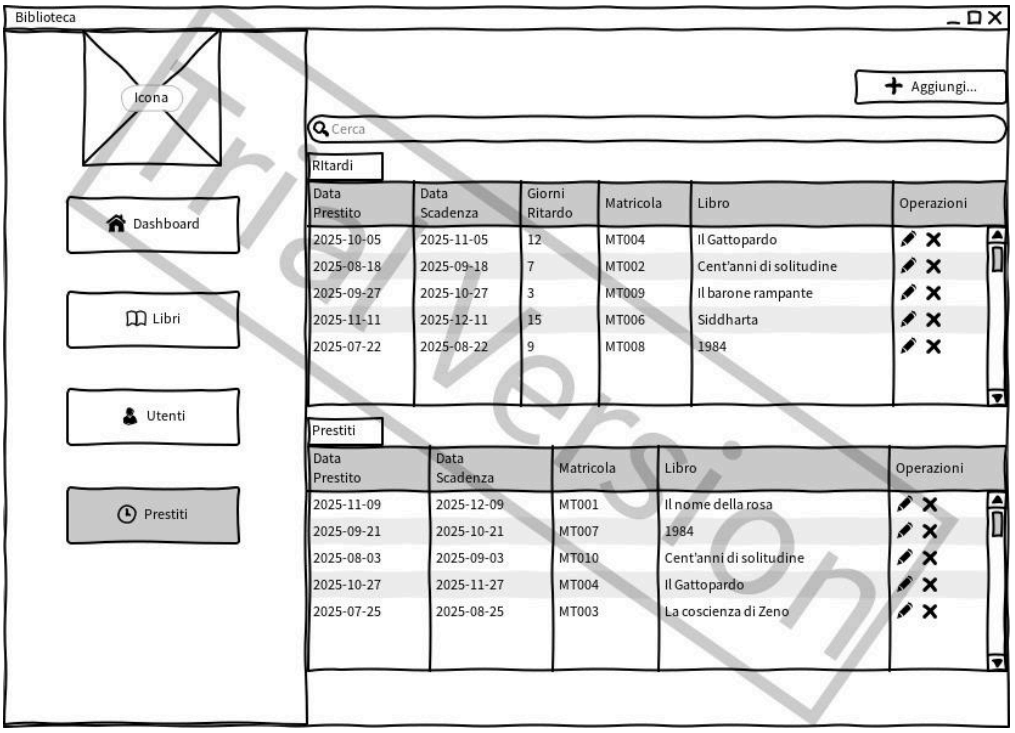
- Descrizione:
    - Consente al bibliotecario di operare sui libri;
    - Permette di visualizzare la lista dei libri presenti in biblioteca, ordinata per titolo;
    - Consente la navigazione verso la pagina precedente (la Dashboard) o verso le altre schermate (Utenti, Prestiti);
- Elementi principali:
    - Pulsanti per effettuare le operazioni di:
        - Registrazione nuovo libro (+ Aggiungi)
            - rimanda ad un'ulteriore schermata che permette l'inserimento dei dati;
        - Modifica dati libro (Icona penna)
            - rimanda ad un'ulteriore schermata che permette la modifica dei dati;
        - Elimina libro (Icona X);
        - Ritorno alla Dashboard;
        - Spostamento verso la Gestione Utenti;
        - Spostamento verso la Gestione Prestiti;
    - Sezione apposita per inserimento di un dato per la ricerca;
    - Tabella con l'elenco dei libri presenti nel catalogo, strutturata in colonne che rappresentano i diversi attributi del libro(titolo, lista degli autori, anno di pubblicazione,ISBN, numero copie disponibili);

Schermata Gestione utenti



- Descrizione:
    - Consente al bibliotecario di operare sugli utenti;
    - Permette di visualizzare la lista degli utenti registrati, ordinata per cognome e nome;
    - Consente la navigazione verso la pagina precedente (la Dashboard) o verso le altre schermate (Libreria, Prestiti);
- Elementi principali:
    - Pulsanti per effettuare le operazioni di:
        - Registrazione nuovo utente(+ Aggiungi)
            - rimanda ad un'ulteriore schermata che permette l'inserimento dei dati;
        - Modifica dati utente (Icona penna)
            - rimanda ad un'ulteriore schermata che permette la modifica dei dati;
        - Elimina utente (Icona X);
        - Ritorno alla Dashboard;
        - Spostamento verso la Gestione Utenti;
        - Spostamento verso la Gestione Prestiti;
    - Sezione apposita per inserimento di un dato per la ricerca;
    - Tabella con l'elenco degli utenti registrati, strutturata in colonne che rappresentano i diversi attributi dell'utente(nome, cognome, matricola, e-mail, numero prestiti attivi);

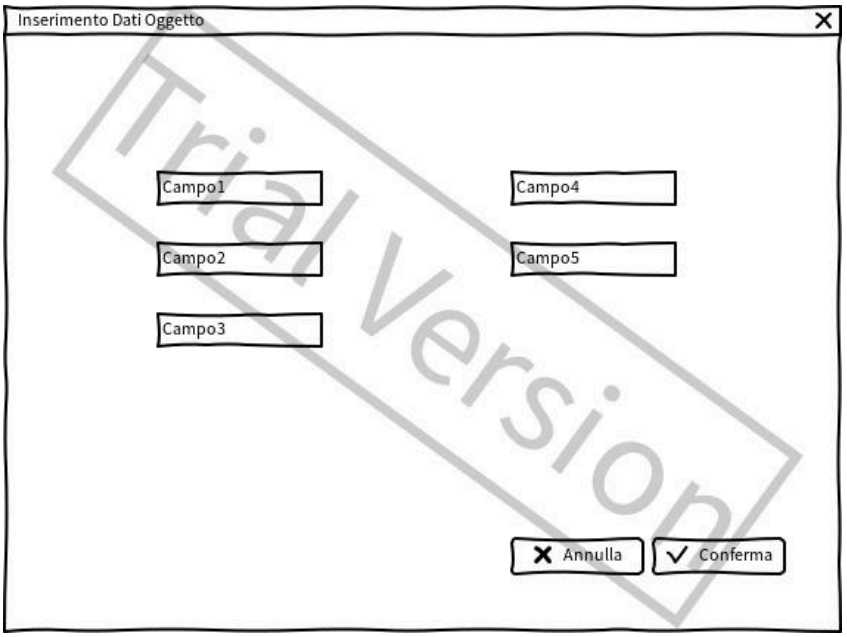
Schermata Gestione prestiti



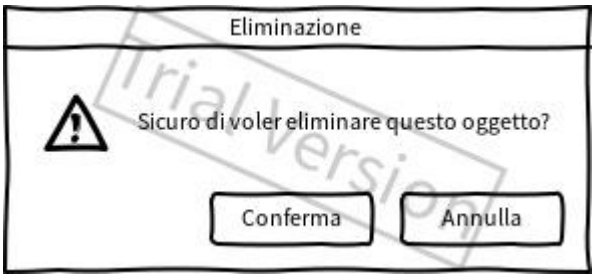
- Descrizione:
    - Consente al bibliotecario di operare sui prestiti;
    - Permette di visualizzare l'elenco dei prestiti attivi, ordinato per data prevista di restituzione ed evidenzia i prestiti in ritardo;
    - Consente la navigazione verso la pagina precedente (la Dashboard) o verso le altre schermate (Libreria, Utenti);
- Elementi principali:
    - Pulsanti per effettuare le operazioni di:
        - Registrazione nuovo prestito (+ Aggiungi)
            - rimanda ad un'ulteriore schermata che permette l'inserimento dei dati;
        - Modifica/Registrazione di una restituzione (Icona penna)
            - rimanda ad un'ulteriore schermata che permette l'inserimento dei dati della restituzione;
        - Ritorno alla Dashboard;
        - Spostamento verso la Gestione Utenti;
        - Spostamento verso la Gestione Prestiti;
    - Sezione apposita per inserimento di un dato per la ricerca;
    - Tabella con l'elenco dei prestiti attivi, strutturata in colonne che rappresentano i diversi attributi dell'utente(matricola utente, libro prestato, data avvenuto prestito, data prevista della restituzione);

Ulteriori Schermate

- Schermata generale di inserimento dati:



- Prototipo di un avviso per la conferma di eliminazione. Dopo che l'attore bibliotecario schiaccia il pulsante relativo all'eliminazione, presente in ognuna delle schermate relative alla gestione, apparirà prima tale messaggio, per evitare eliminazioni di dati indesiderati.



Logica degli eventi

Gli eventi generati dall'interfaccia (click sui pulsanti, inserimento dati per ricercare o per registrazione) verranno gestiti dai controller associati alle varie view.
La UI non accede direttamente ai dati, ma passa sempre tramite il controller, che funge da intermediario.

View (UI)	**Controller**
Dashboard	ControllerDashboard.java
Libreria	ControllerLibreria.java
Utenti	ControllerUtenti.java
Prestiti	ControllerPrestiti.java