UNIVERSITÀ DEGLI STUDI
DELL'INSUBRIA

Book Recommender

Technical Manual

Laurea Triennale in Informatica
Progetto Laboratorio B: Book Recommender
Sviluppato da: Matteo Corda, Jacopo Loni, Simone Cirillo, Gabriele Schioppa e Simone Diano

Sommario

1. **Introduzione**

    1.1 Tecnologie utilizzate

    1.2 Obiettivi del sistema

2. **Progettazione della soluzione**

    2.1 Architettura del sistema

    2.2 Componenti principali

3. **Documentazione UML del Sistema**

    3.1 Class Diagram

    3.2 Sequence Diagram

    3.3 State Diagram

4. **Progettazione del DATABASE**

    4.1 Processo di progettazione

    4.2 Schema relazionale implementato

    4.3 Tabelle principali

5. **Architettura e Implementazione**

    5.1 Scelte architetturali

    5.2 Strutture dati utilizzate

    5.3 Gestione della comunicazione client-server

    5.4 Gestione file e path

    5.5 metodi di ricerca

5.6 creazione e gestione delle librerie

5.7 Sistema di valutazioni e suggerimenti

6. **Limiti della soluzione sviluppata**

6.1 Limiti di sicurezza

6.2 Limiti funzionali

6.3 Limiti interfaccia grafica

6.4 Limiti del sistema operativo

7. **Sitografia**

7.1 tecnologie utilizzate

7.2 programmazione socket e multi-thread

7.3 DATABASE e UML

7.4 GIT e GitHub

7.5 sviluppo GUI

1. Introduzione

Book Recommender è un'applicazione sviluppata in Java che implementa un sistema di gestione e raccomandazione di libri. Il sistema è stato riprogettato per utilizzare un'architettura client-server con persistenza dei dati su database PostgreSQL, sostituendo il precedente approccio basato su file locali. Il progetto è sviluppato in Java 12, usa un'interfaccia grafica costruita con Java Swing ed è stato sviluppato e testato su sistema operativo Windows 10 e 11.

1.1 Tecnologie utilizzate:

Per lo sviluppo di questo progetto sono state utilizzate diverse tecnologie tra cui:

- **Java 12+:** Linguaggio di programmazione principale
- **Java Swing**: Framework per l'interfaccia grafica utente
- **PostgreSQL:** Sistema di gestione database relazionale
- **JDBC:** API per la connessione al database
- **Supabase**: Servizio cloud per hosting del database
- **Socket Programming:** Comunicazione client-server

1.2 Obiettivi del sistema:

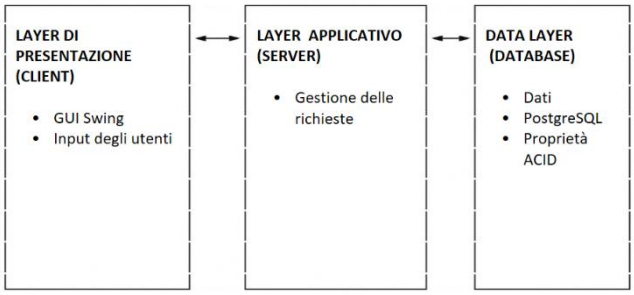
- Fornire un sistema scalabile di gestione e raccomandazione libri
- Implementare un'architettura robusta client-server
- Garantire persistenza e integrità dei dati
- Supportare accesso concorrente di più utenti

2. progettazione della soluzione

2.1 architettura del sistema:

Il sistema implementa un'architettura client-server a tre livelli:

- **Presentation Layer (Client):** Interfaccia grafica Swing
- **Application Layer (Server):** Logica e gestione richieste
- **Data Layer (Database):** Persistenza dati PostgreSQL



2.2 componenti principali:

Componenti Server-Side:

- **Server.java**: Server principale che gestisce le connessioni
- **ServerSlave.java**: Thread worker per gestire singole connessioni client

Componenti Client-Side:

- **Classi GUI**: Interfacce grafiche per interazione utente:
    - *HomeMainFrame*
    - *LibrerieMainFrame*
    - *LgMainFrame*
    - *RgMainFrame*
- **Proxy**: Gestione comunicazione con server

Classi Parametri: entità di sistema (Utente, Libro, Autore, Librerie, SuggerimentoLibro e ValutazioneLibro)

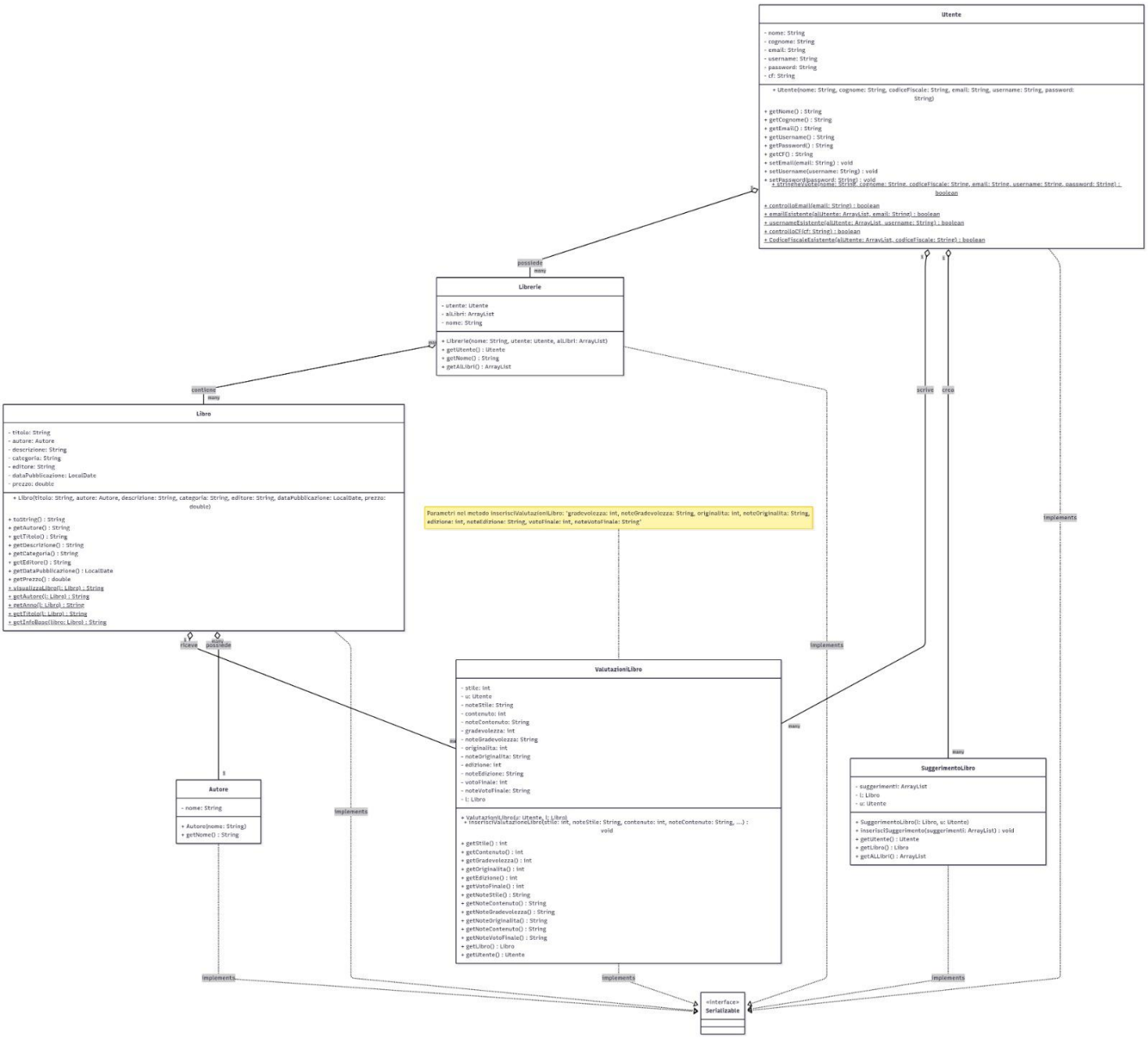
3. Documentazione UML del Sistema

La documentazione UML fornisce una rappresentazione formale dell'architettura software a diversi livelli di astrazione. I diagrammi UML sono stati sviluppati per catturare sia gli aspetti statici che dinamici del sistema, fornendo una visione completa dell'architettura client-server implementata.

3.1 Class Diagram

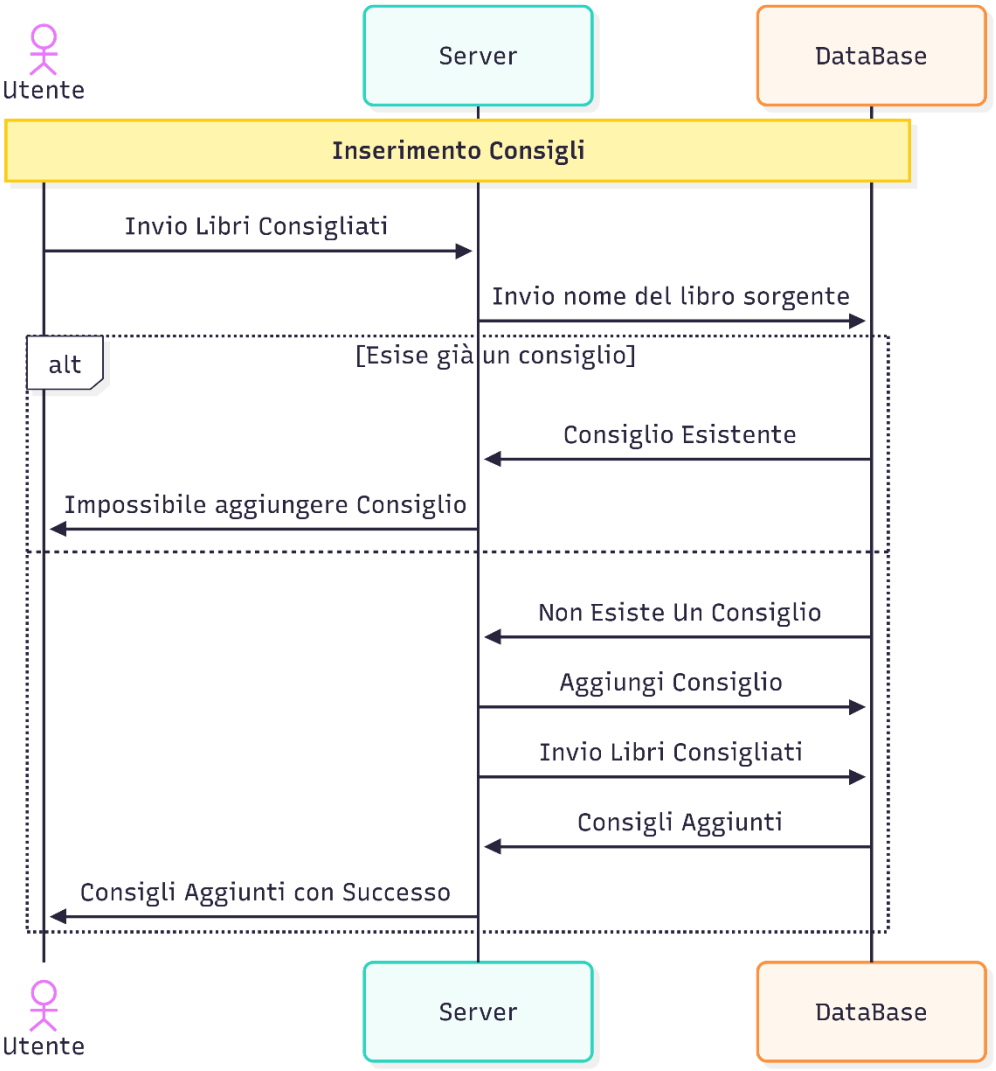
Il Class Diagram mostra l'architettura delle classi del dominio della Book Recommender App, evidenziando le relazioni tra le entità principali (Utente, Libro, Autore, Libreria, Valutazione, Consigli) e definendo gli attributi e metodi necessari per la gestione delle funzionalità di raccomandazione e valutazione dei libri:

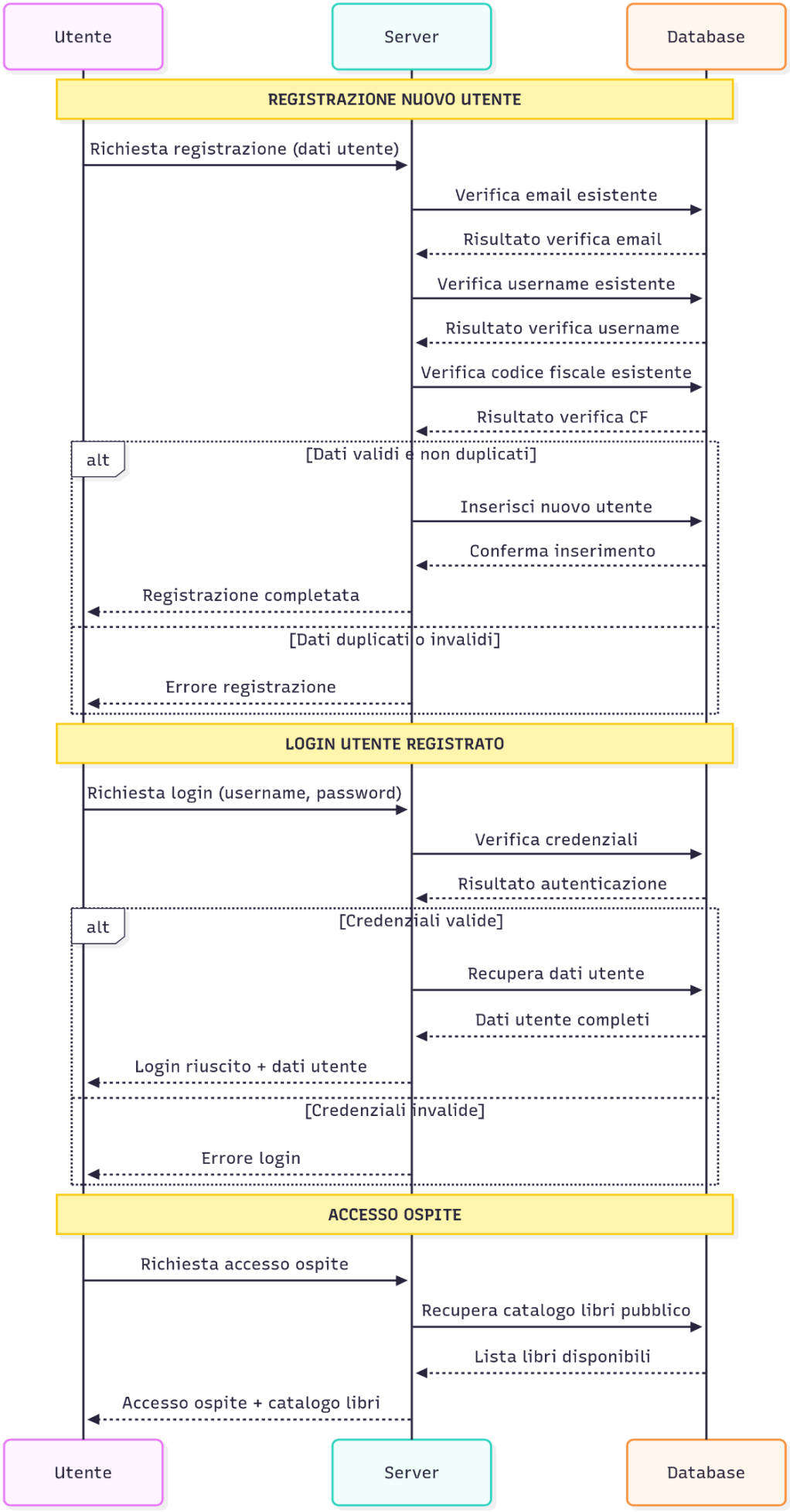
Lo si può visualizzare nel dettaglio in allegato alla documentazione.

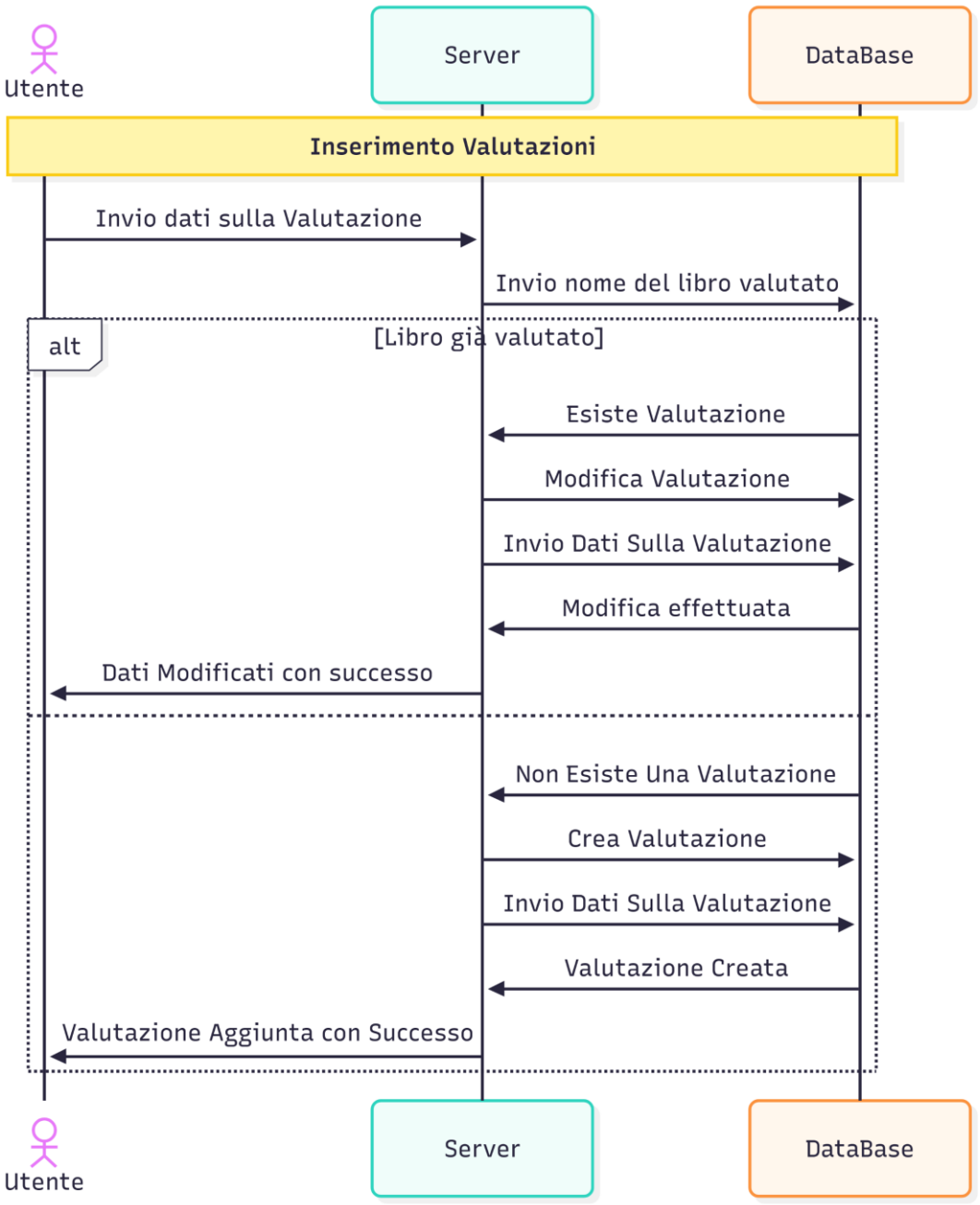


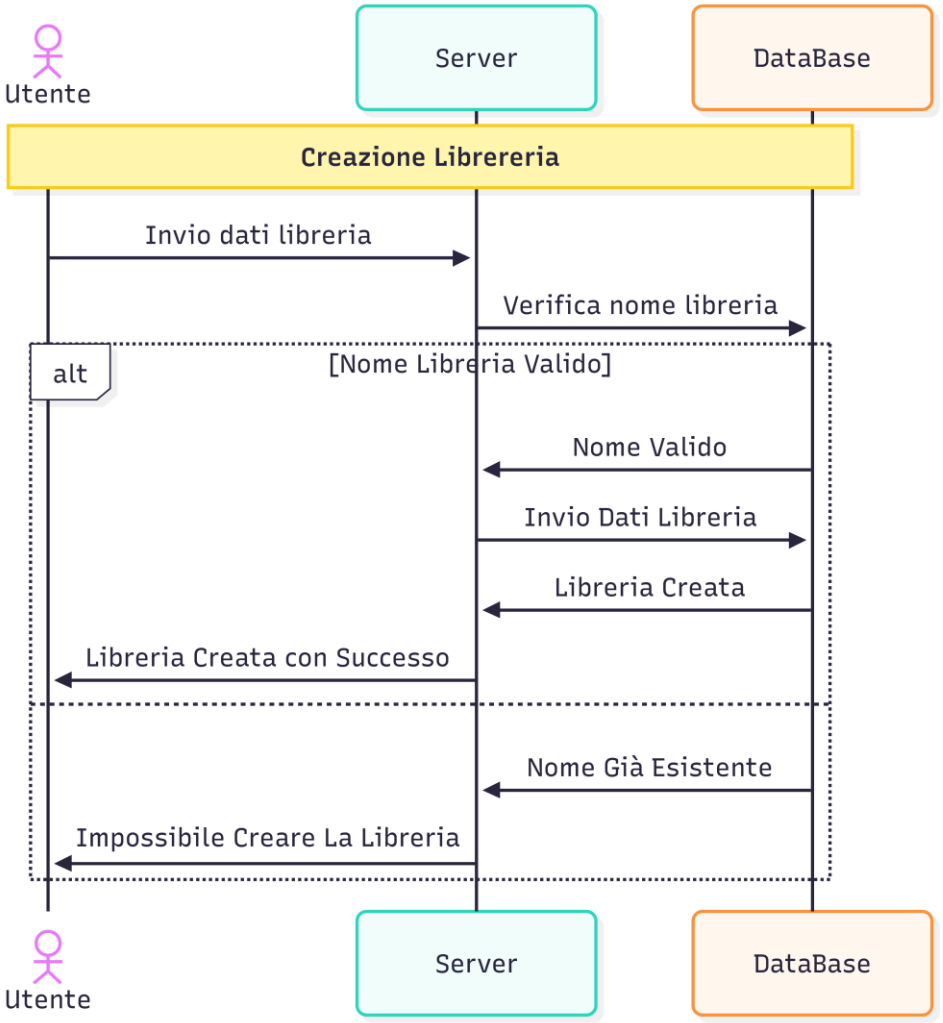
3.2 Sequence Diagram

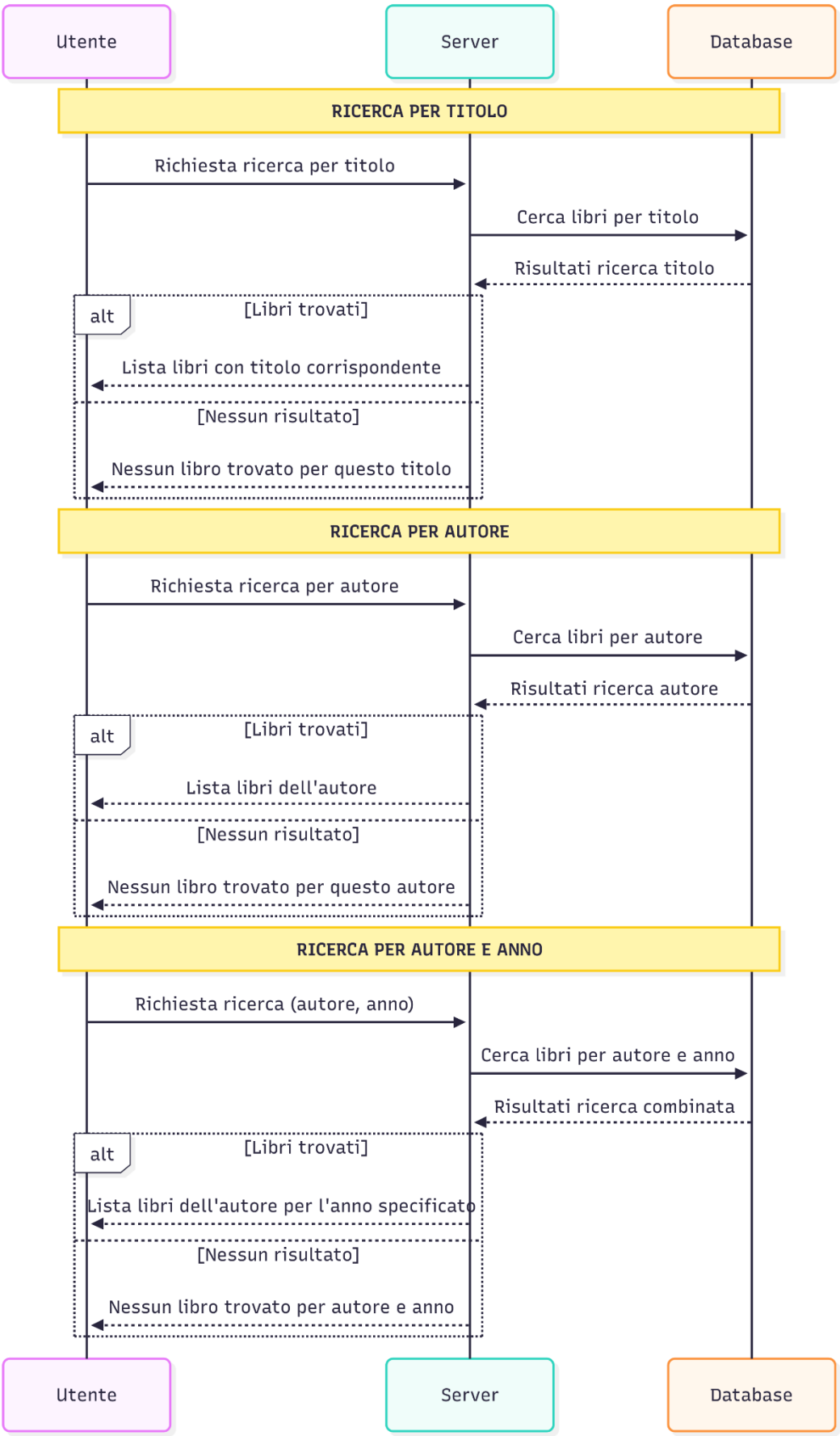
Il Sequence Diagram rappresenta il flusso di comunicazione temporale tra il client e il server nelle operazioni fondamentali dell'applicazione, come l'autenticazione utente, la ricerca di libri nel database PostgreSQL e la sincronizzazione delle librerie personali attraverso l'architettura client-server:




Utente
Server
Database
REGISTRAZIONE NUOVO UTENTE
Richiesta registrazione (dati utente)
Verifica email esistente
Risultato verifica email
Verifica username esistente
Risultato verifica username
Verifica codice fiscale esistente
Risultato verifica CF
alt
[Dati validi e non duplicati]
Inserisci nuovo utente
Conferma inserimento
Registrazione completata
[Dati duplicati o invalidi]
Errore registrazione
LOGIN UTENTE REGISTRATO
Richiesta login (username, password)
Verifica credenziali
Risultato autenticazione
alt
[Credenziali valide]
Recupera dati utente
Dati utente completi
Login riuscito + dati utente
[Credenziali invalide]
Errore login
ACCESSO OSPITE
Richiesta accesso ospite
Recupera catalogo libri pubblico
Lista libri disponibili
Accesso ospite + catalogo libri
Utente
Server
Database

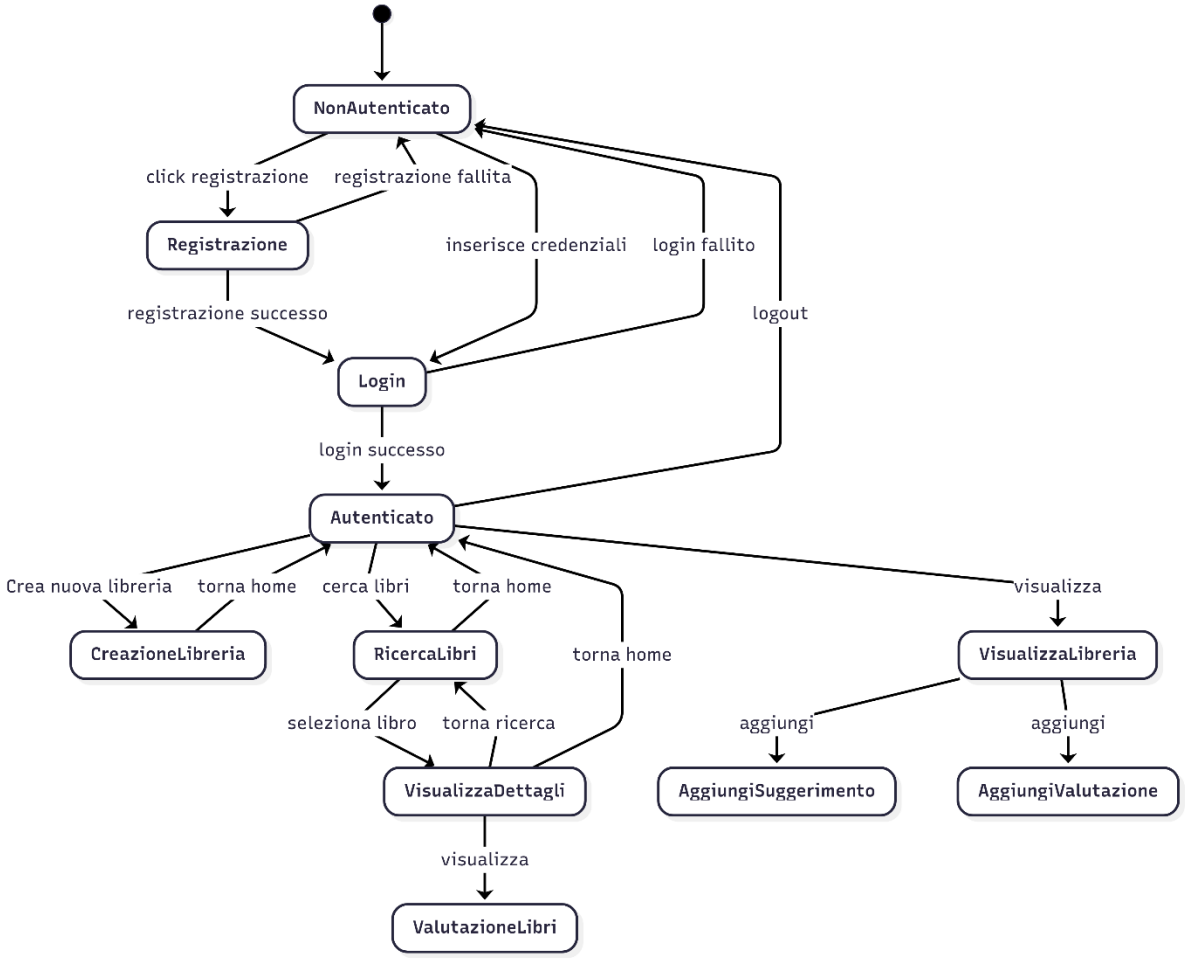

Utente
Server
DataBase
Inserimento Valutazioni
Invio dati sulla Valutazione
Invio nome del libro valutato
alt
[Libro già valutato]
Esiste Valutazione
Modifica Valutazione
Invio Dati Sulla Valutazione
Modifica effettuata
Dati Modificati con successo
Non Esiste Una Valutazione
Crea Valutazione
Invio Dati Sulla Valutazione
Valutazione Creata
Valutazione Aggiunta con Successo
Utente
Server
DataBase


Utente
Server
DataBase
Creazione Librereria
Invio dati libreria
Verifica nome libreria
alt
[Nome Libreria Valido]
Nome Valido
Invio Dati Libreria
Libreria Creata
Libreria Creata con Successo
Nome Già Esistente
Impossibile Creare La Libreria
Utente
Server
DataBase


Utente
Server
Database
RICERCA PER TITOLO
Richiesta ricerca per titolo
Cerca libri per titolo
Risultati ricerca titolo
alt
[Libri trovati]
Lista libri con titolo corrispondente
[Nessun risultato]
Nessun libro trovato per questo titolo
RICERCA PER AUTORE
Richiesta ricerca per autore
Cerca libri per autore
Risultati ricerca autore
alt
[Libri trovati]
Lista libri dell'autore
[Nessun risultato]
Nessun libro trovato per questo autore
RICERCA PER AUTORE E ANNO
Richiesta ricerca (autore, anno)
Cerca libri per autore e anno
Risultati ricerca combinata
alt
[Libri trovati]
Lista libri dell'autore per l'anno specificato
[Nessun risultato]
Nessun libro trovato per autore e anno
Utente
Server
Database

3.3 State Diagram

Lo State Diagram modella i diversi stati di una sessione utente nel Book Recommender, dalla connessione iniziale al server fino alla gestione delle operazioni di ricerca e valutazione, evidenziando le transizioni di stato in base alle azioni dell'utente:



4. progettazione del DATABASE

4.1 processo di progettazione

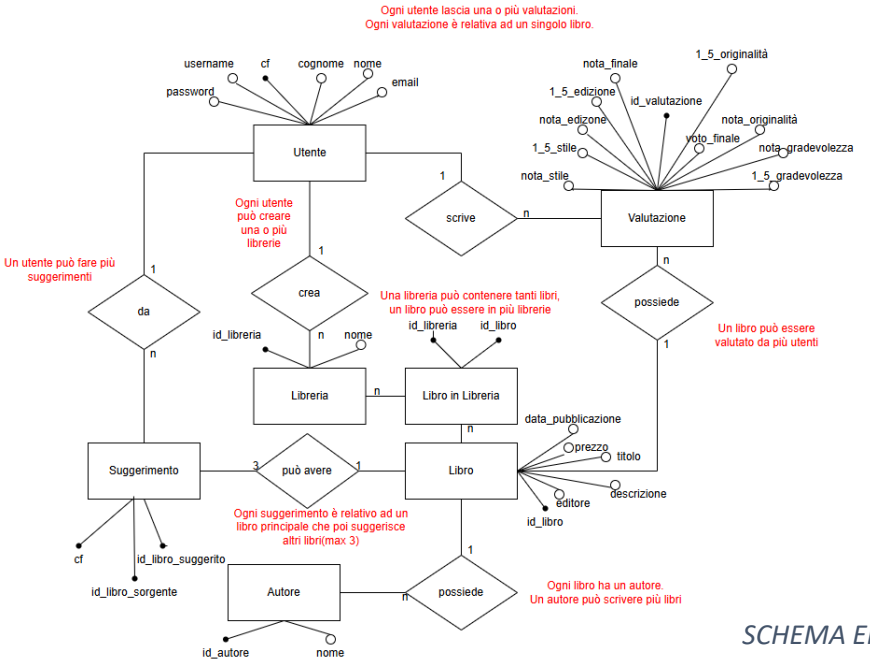
La progettazione del database ha seguito un approccio metodico strutturato in diverse fasi:

1. **Analisi dei Requisiti:** Il processo è iniziato con l'identificazione delle entità principali del dominio applicativo e delle loro relazioni. Sono stati analizzati i requisiti funzionali per determinare le operazioni che il sistema deve supportare.

2. **Schema Entità-Relazioni (ER)** È stato sviluppato uno schema ER che modella:
    a. Le entità principali del dominio
    b. Le relazioni tra entità con cardinalità appropriate
    c. Gli attributi di ogni entità con domini e vincoli
    d. Le chiavi primarie

3. **Modellazione Comportamentale**: In parallelo alla modellazione ER, sono stati sviluppati diversi diagrammi UML.

4. **Processo di Trasformazione ER → Relazionale:** La trasformazione dallo schema ER al modello relazionale ha comportato diverse decisioni progettuali:
    a. **Mappatura Entità**: Ogni entità del modello ER è stata trasformata in una tabella relazionale
    b. **Gestione Relazioni**: Le relazioni many-to-many sono state risolte attraverso tabelle di congiunzione
    c. **Chiavi Primarie**: Scelta tra chiavi naturali (es. codice fiscale) e chiavi surrogate (es. ID sequenziali)
    d. **Vincoli di Integrità**: Implementazione di tutti i vincoli identificati nel modello ER



SCHEMA ER ristrutturato

4.2 schema relazionale implementato

Il database è stato implementato su **PostgreSQL** utilizzando la piattaforma cloud **Supabase**, che offre:

- *Hosting gestito del database PostgreSQL*
- *Interfaccia web per gestione e monitoraggio*
- *Backup automatici e alta disponibilità*

4.3 Tabelle principali

Tabella utente:

```
CREATE TABLE utente (
    cf VARCHAR(16) PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    cognome VARCHAR(100) NOT NULL,
    email VARCHAR(255) NOT NULL UNIQUE,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL
);
```

***Descrizione**: Memorizza tutti gli utenti registrati nel sistema*

***Funzione**: Gestisce l'anagrafica degli utenti che possono possedere librerie, valutare libri e fornire suggerimenti di lettura*

***Chiave primaria**: cf (Codice Fiscale)*

***Osservazioni**: username ed email sono indicate come unique in quanto devono essere uniche nel sistema*

Tabella autore:

```
CREATE TABLE autore (
    idautore SERIAL PRIMARY KEY,
    nome VARCHAR(255) NOT NULL
);
```

***Descrizione**: Catalogo di tutti gli autori presenti nel sistema*

***Funzione**: Mantiene l'anagrafica degli autori dei libri*

***Chiave primaria**: idautore (auto-incrementale)*

Tabella libri:

```
CREATE TABLE libro (
    idlibro SERIAL PRIMARY KEY,
    titolo TEXT NOT NULL,
    descrizione TEXT,
    categoria TEXT,
    editore TEXT,
    datapubblicazione DATE,
    prezzo DECIMAL(10,2),
    idautore INTEGER NOT NULL,
    FOREIGN KEY (idautore) REFERENCES autore(idautore)
);
```

***Descrizione**: Catalogo completo di tutti i libri del sistema*

***Funzione**: Memorizza le informazioni bibliografiche di ogni libro*

***Chiave primaria**: idlibro (auto-incrementale)*

***Dipendenze**: Collegata ad AUTORE tramite idautore*

Tabella libreria:

```
CREATE TABLE libreria (
    idlibreria SERIAL PRIMARY KEY,
    nome VARCHAR(255) NOT NULL,
    cf VARCHAR(16) NOT NULL,
    FOREIGN KEY (cf) REFERENCES utente(cf)
);
```

***Descrizione**: Rappresenta le librerie possedute dagli utenti*

***Funzione**: Ogni utente può possedere una o più librerie personali*

***Chiave primaria**: idlibreria (auto-incrementale)*

***Dipendenze**: Collegata ad UTENTE tramite cf (codice fiscale del proprietario)*

Tabella Libroinlibreria:

```
CREATE TABLE libroinlibreria (
    idlibreria INTEGER NOT NULL,
    idlibro INTEGER NOT NULL,
    PRIMARY KEY (idlibreria, idlibro),
    FOREIGN KEY (idlibreria) REFERENCES libreria(idlibreria),
    FOREIGN KEY (idlibro) REFERENCES libro(idlibro)
);
```

***Descrizione:** tabella di associazione molti-a-molti tra libri e librerie*

***Funzione**: Gestisce quali libri sono presenti in quali librerie*

***Chiave primaria**: Composita (idlibreria, idlibro)*

***Dipendenze**: Collegata a LIBRERIA e LIBRO*

Tabella valutazione:

```
CREATE TABLE valutazione (
    idvalutazione SERIAL PRIMARY KEY,
    idlibro INTEGER NOT NULL,
    cf VARCHAR(16) NOT NULL,
    -- Voti da 1 a 5 per ogni criterio
    stile INTEGER CHECK (stile BETWEEN 1 AND 5),
    notestile VARCHAR(500),
    contenuto INTEGER CHECK (contenuto BETWEEN 1 AND 5),
    notecontenuto VARCHAR(500),
    gradevolezza INTEGER CHECK (gradevolezza BETWEEN 1 AND 5),
    notegradevolezza VARCHAR(500),
    originalita INTEGER CHECK (originalita BETWEEN 1 AND 5),
    noteoriginalita VARCHAR(500),
    edizione INTEGER CHECK (edizione BETWEEN 1 AND 5),
    noteedizione VARCHAR(500),
    votofinale INTEGER CHECK (votofinale BETWEEN 1 AND 5),
    notevotofinale VARCHAR(500),
    FOREIGN KEY (idlibro) REFERENCES libro(idlibro),
    FOREIGN KEY (cf) REFERENCES utente(cf)
);
```

***Descrizione:** Memorizza le valutazioni dettagliate dei libri da parte degli utenti*

***Funzione**: Sistema di recensioni con voti da 1 a 5 su diversi criteri*

***Chiave primaria**: idvalutazione (auto-incrementale)*

***Dipendenze**: Collegata a LIBRO e UTENTE*

***criteri di valutazione**: stile, contenuto, gradevolezza, originalità, edizione*

Tabella suggerimento:

```
CREATE TABLE suggerimento (
    idlibrosorgente INTEGER NOT NULL,
    idlibrosuggerito INTEGER NOT NULL,
    cf VARCHAR(16) NOT NULL,
    PRIMARY KEY (idlibrosorgente, idlibrosuggerito, cf),
    FOREIGN KEY (idlibrosorgente) REFERENCES libro(idlibro),
    FOREIGN KEY (cf) REFERENCES utente(cf)
);
```

***Descrizione:** Sistema di raccomandazioni tra libri da parte degli utenti*

***Funzione**: Permette agli utenti di suggerire libri basandosi su altri libri*

***Chiave primaria**: Composta da: (idlibrosorgente, idlibrosuggerito, cf)*

***Dipendenze**: Collegata a LIBRO e UTENTE*

5. Architettura e implementazione

5.1 scelte architetturali

Il sistema adotta diverse scelte architetturali:

Architettura Client-Server Multi-thread:

```
public class Server{
    private ServerSocket serverSocket;
    public final static int PORT = 1090;

    public void exec(){
        while(true){
            Socket socket = serverSocket.accept();
            new ServerSlave(socket);
        }
    }
}
```

- ***Server centralizzato**: Un singolo server sulla porta 1090 gestisce tutte le richieste*
- ***Thread dedicati**: Ogni connessione client viene gestita da un ServerSlave thread separato*
- ***Comunicazione Socket TCP**: Utilizzo di socket per comunicazione affidabile*
- ***Separazione delle responsabilità**: Netta divisione tra logica di presentazione (client) e logica di business (server)*

Proxy per la comunicazione

```
public class Proxy{
    // Gestisce la comunicazione tra client e server
    // Utilizza flussi di oggetti per lo scambio di dati
}
```

5.2 strutture dati utilizzate

ArrayList come struttura principale

Il sistema utilizza intensivamente ArrayList<> per la gestione delle collezioni**:**

```
// Dalla classe Librerie
private final ArrayList<Libro> alLibri;
public ArrayList<Libro> getAlLibri(){
    return this.alLibri;
}

// Esempi di utilizzo nelle query
ArrayList<Libro> listaLibri = new ArrayList();
ArrayList<Utente> alUtenti = new ArrayList<>();
```

Vantaggi dell'ArrayList nel contesto del progetto:

- ***Accesso indicizzato O(1)****: Essenziale per operazioni di visualizzazione rapida nelle GUI*
- ***Dimensione dinamica****: Permette di gestire collezioni di dimensioni variabili*
- ***Compatibilità Swing****: Integrazione diretta con componenti GUI Java (JTable, JList, DefaultListModel)*

Serializable

Tutte le classi parametro implementano l'interfaccia Serializable per permettere il trasferimento efficiente via socket. Questa scelta tecnica permette la trasmissione di oggetti complessi mantenendo l'integrità dei dati e le relazioni tra entità:

```
public class Librerie implements Serializable{
    private final Utente utente;
    private final ArrayList<Libro> alLibri;
    private final String nome;
}

public class ValutazioniLibro implements Serializable{
    // Contiene le valutazioni multi-criterio dei libri
}
```

5.3 gestione della comunicazione Client-Server

Protocollo di comunicazione basato su stringhe

Il ServerSlave gestisce le richieste tramite un sistema di comandi string:

```
String operazione = (String)in.readObject();

if(operazione.equals("getLibriSuggeriti")){
    Utente u = (Utente)in.readObject();
    int idLibro = (int)in.readObject();
    // Logica per recuperare suggerimenti
}
else if(operazione.equals("GetLibrerieDaCf")){
    // Logica per recuperare librerie utente
}
```

- ***Estensibilità**: Facile aggiunta di nuove operazioni senza modificare l'architettura*
- ***Debug facilitato**: I comandi sono leggibili e tracciabili nei log*
- ***Flessibilità**: Permette operazioni con parametri variabili*

Query SQL integrate nel ServerSlave

Tutte le query SQL sono integrate direttamente nel ServerSlave, permettendo ottimizzazioni specifiche per ogni operazione:

```
// Esempio query per ottenere libri suggeriti
String operazioneQuery = "SELECT l.titolo as titolo, l.descrizione as descrizione, " +
    "l.categoria as categoria, l.editore as editore, l.datapubblicazione as data, " +
    "l.prezzo as prezzo, a.nome as nome FROM suggerimento s " +
    "join libro l on (s.idlibrosuggerito = l.idlibro) " +
    "join utente u on(u.cf = s.cf) " +
    "join autore a on (l.idautore = a.idautore) " +
    "WHERE u.cf = ? AND idlibrosorgente = ?";
```

5.4 Gestione file e path

Struttura del progetto:

```
   BookRecommenderUni/
├── src/
│   ├── frames/           # Package GUI e Proxy
│   │   ├── HomeMainFrame.java
│   │   ├── LibrerieMainFrame.java
│   │   ├── LgMainFrame.java
│   │   ├── RgMainFrame.java
│   │   └── Proxy.java
│   ├── parametri/        # Package entità del dominio
│   │   ├── Utente.java
│   │   ├── Libro.java
│   │   ├── Autore.java
│   │   ├── Librerie.java
│   │   └── ValutazioniLibro.java
│   ├── server/           # Package server
│   │   ├── Server.java
│   │   └── ServerSlave.java
│   └── Main.java
├── docs/                 # Documentazione JavaDoc
└── doc/                  # User Manual e Technical Manual
```

5.5 metodi di ricerca

Il sistema implementa tre modalità di ricerca per consentire agli utenti di trovare libri nel database in modo efficiente e flessibile.

Esempio Metodo ricercaPerTitolo:

```
public ArrayList<Libro> ricercaPerTitolo(String titolo) {
    ArrayList<Libro> alLibro = new ArrayList<>();
    try {
        out.writeObject("CercaPerTitolo");
        out.flush();
        out.writeObject(titolo);
        out.flush();
        alLibro = (ArrayList<Libro>)in.readObject();
    } catch (IOException | ClassNotFoundException e) {
        // Gestione errori di comunicazione
    }
    return alLibro;
}
```

Utilizzo nell'interfaccia grafica:

```
// Implementazione nel LibrerieMainFrame
searchByTitleButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        risultatoRicercaLibrerie.removeAll();
        String testo = searchField.getText();
        ArrayList<Libro> risultati = proxy.ricercaPerTitolo(testo);
        mostraRisultati(risultati, 0, risultatoRicercaLibrerie);
    }
});
```

5.6 Creazione e gestione delle librerie

Creazione di una Libreria

Il processo di creazione avviene tramite l'interfaccia grafica LibrerieMainFrame:

```
// Processo di validazione e creazione
if (libriDaAggiungere.isEmpty()) {
    JOptionPane.showMessageDialog(frame, "Inserire almeno un libro");
    return;
}

// Creazione dell'oggetto Libreria
Librerie libreria = new Librerie(nomeLibreria, u, libriDaAggiungere);

// Invio al server tramite proxy
proxy.aggiungiLibreria(libreria);

// Conferma all'utente
JOptionPane.showMessageDialog(frame, "Libreria Creata con successo");
```

Gestione Aggiunta Libri alla Libreria

Il sistema previene duplicati attraverso un controllo di validazione:

```
// Controllo duplicati
for(int i = 0; i < libriDaAggiungere.size() && libroAggiunto; i++) {
    if(libriDaAggiungere.get(i).getTitolo().equals(libro.getTitolo())) {
        libroAggiunto = false;
    }
}

if(libroAggiunto) {
    libriDaAggiungere.add(libro);
    dialog.dispose();
} else if(!libroAggiunto && firstTime) {
    JOptionPane.showMessageDialog(frame, "Libro già inserito");
}
```

5.7 Sistema di valutazione e suggerimento

Valutazioni multi-criterio:

```
public class ValutazioniLibro implements Serializable{
    // Campi per valutazione (1-5) su:
    // - Stile
    // - Contenuto
    // - Gradevolezza
    // - Originalità
    // - Edizione
    // - Voto finale
}
```

metodo di Inserimento Valutazione:

```
public void inserisciValutazioneLibro(int stile, String noteStile,
                                      int contenuto, String noteContenuto,
                                      int gradevolezza, String noteGradevolezza,
                                      int originalita, String noteOriginalita,
                                      int edizione, String noteEdizione,
                                      int votoFinale, String noteVotoFinale) {
    this.stile = stile;
    this.contenuto = contenuto;
    this.gradevolezza = gradevolezza;
    this.originalita = originalita;
    this.edizione = edizione;
    this.votoFinale = votoFinale;
    // Assegnazione delle note
    this.noteStile = noteStile;
    this.noteContenuto = noteContenuto;
    this.noteGradevolezza = noteGradevolezza;
    this.noteOriginalita = noteOriginalita;
    this.noteEdizione = noteEdizione;
    this.noteVotoFinale = noteVotoFinale;
}
```

Interfaccia Grafica Valutazione:

L'interfaccia utilizza JComboBox per i punteggi (1-5) e JTextArea per le note:

```
String[] arrayPunteggio = {"1", "2", "3", "4", "5"};
JComboBox<String> boxStile = new JComboBox<>(arrayPunteggio);
JComboBox<String> boxContenuto = new JComboBox<>(arrayPunteggio);
JComboBox<String> boxGradevolezza = new JComboBox<>(arrayPunteggio);
JComboBox<String> boxOriginalita = new JComboBox<>(arrayPunteggio);
```

Sistema di raccomandazioni

Gli utenti possono suggerire libri basandosi su altri libri letti:

```
// Query per inserire suggerimenti
"INSERT INTO suggerimento(idlibrosorgente,idlibrosuggerito,cf) VALUES(?,?,?)"

// Query per recuperare utenti che hanno fatto suggerimenti
"SELECT DISTINCT u.* FROM SUGGERIMENTO s JOIN utente u ON (s.cf = u.cf) WHERE s.idlibrosorgente = ?"
```

Metodo aggiungiSuggerimento:

```
public class SuggerimentoLibro implements Serializable {
    private ArrayList<Libro> suggerimenti;
    private final Libro l;           // Libro sorgente
    private final Utente u;          // Utente che suggerisce

    public SuggerimentoLibro(Libro l, Utente u) {
        this.l = l;
        this.u = u;
        suggerimenti = new ArrayList<>();
    }
}
```

Processo di Inserimento Suggerimenti:

```
public void inserisciSuggerimento(ArrayList<Libro> suggerimenti) {
    this.suggerimenti = suggerimenti;
}
```

Validazione dei Suggerimenti:

```
// Controllo se esiste già un suggerimento per questo libro
boolean isSuggerito = proxy.getIfSuggerito(u, libro);

if(!(alSuggerimenti.isEmpty())) {
    SuggerimentoLibro sl = new SuggerimentoLibro(libro, u);
    sl.inserisciSuggerimento(alSuggerimenti);

    if(!isSuggerito) {
        proxy.aggiungiSuggerimentiLibroUtente(u, libro, alSuggerimenti);
        risultatoRicercaSuggerimenti.removeAll();
        dialog.dispose();
    } else {
        JOptionPane.showMessageDialog(frame, "Esiste già un suggerimento per questo libro");
    }
} else {
    JOptionPane.showMessageDialog(frame, "Inserire almeno un libro");
}
```

Controlli di Integrità

- **Limite massimo:** Massimo 3 libri suggeriti per libro sorgente
- **Prevenzione duplicati:** Controllo che il libro non sia già stato suggerito
- **Auto-referenza:** Impedisce di suggerire lo stesso libro di cui si stanno inserendo suggerimenti

```
// Controllo numero massimo suggerimenti
if(alSuggerimenti.size() >= 3) {
    aggiunto = false;
    JOptionPane.showMessageDialog(frame, "Hai inserito il numero massimo di libri");
}

// Controllo auto-referenza
if(libro.getTitolo().equals(libroCorrente.getTitolo())) {
    sameLib = false;
    JOptionPane.showMessageDialog(frame,
        "non puoi suggerire lo stesso libro di cui stai inserendo suggerimenti");
}
```

Calcolo medie valutazioni

Il sistema calcola automaticamente le medie delle valutazioni per ogni libro:

```
if(operazione.equals("getMediaValutazioniLibro")){
                int id = (int)in.readObject();
                String operazioneQuery = "SELECT AVG(stile), AVG(contenuto), AVG(gradevolezza),
                 AVG(originalita),  AVG(edizione), AVG(votofinale) FROM valutazione WHERE idLibro = ?";
                ArrayList<Double> alMedia = new ArrayList<>();
```

```
// Raccolta dei voti dai componenti GUI
int votoStile = Integer.parseInt(boxStile.getSelectedItem().toString());
int votoContenuto = Integer.parseInt(boxContenuto.getSelectedItem().toString());
int votoGradevolezza = Integer.parseInt(boxGradevolezza.getSelectedItem().toString());
int votoOriginalita = Integer.parseInt(boxOriginalita.getSelectedItem().toString());
int votoEdizione = Integer.parseInt(boxEdizione.getSelectedItem().toString());

// Calcolo della media
float mediaVoti = (votoStile + votoContenuto + votoGradevolezza +
                   votoOriginalita + votoEdizione) / 5;
votoFinale = Math.round(mediaVoti);
```

6. Limiti della soluzione sviluppata:

6.1 limiti di sicurezza

1. **Password Storage**: Password memorizzate in plain text nel database
2. **Comunicazione**: Nessuna crittografia per la comunicazione client-server

6.2 limiti funzionali

1. **Ricerca**: Funzionalità di ricerca limitate (solo per titolo/autore/autore e anno)
2. **Backup**: Nessun sistema automatico di backup database

6.3 limiti interfaccia grafica

1. **GUI**: Interfaccia non adattiva per diverse risoluzioni
2. **Lingue**: Nessun supporto per multiple lingue
3. **Mobile**: Nessun supporto per dispositivi mobi

6.4 limiti sistema operativo

1. **Dipendenza Windows**: Sviluppato e testato esclusivamente su Windows 10/11
2. **Path Windows**: Possibili problemi con separatori di path su sistemi Unix/Linux
3. **Font rendering**: Java Swing può avere comportamenti diversi su macOS e Linux

7. Sitografia:

7.1 tecnologie utilizzate

- **Java SE:** https://docs.oracle.com/en/java/
- **Java Swing:** https://docs.oracle.com/javase/tutorial/uiswing/
- **PostgreSQL:** https://www.postgresql.org/docs/
- **Supabase:** https://supabase.com/docs
- **JDBC:** https://docs.oracle.com/javase/tutorial/jdbc/

7.2 programmazione socket e multi-thread

- **Java Socket:** https://docs.oracle.com/javase/tutorial/networking/sockets/
- **Multi-threading:** https://docs.oracle.com/javase/tutorial/essential/concurrency/

7.3 DATABASE e UML

- **PostgreSQL:** https://www.postgresqltutorial.com/
- **SQL Reference:** https://www.w3schools.com/sql/
- **Lucidchart ER:** https://www.lucidchart.com/pages/er-diagrams
- **UML diagrams tool:** https://mermaid.live/

7.4 GIT e GitHub

- **Documentazione Git:** https://git-scm.com/doc
- **Documentazione GitHub:** https://docs.github.com/

7.5 Sviluppo GUI

- **Swing Layout:** https://docs.oracle.com/javase/tutorial/uiswing/layout/
- **Java AWT:** https://docs.oracle.com/javase/8/docs/api/java/awt/package-summary.html
- **Event Handling:** https://docs.oracle.com/javase/tutorial/uiswing/events/
- **Swing Components:** https://docs.oracle.com/javase/tutorial/uiswing/components/