Manuale Tecnico

BookRecommender

Babini Ariele

10 luglio 2025

Indice

1 Introduzione

Questo manuale descrive dettagliatamente la struttura tecnica e progettuale del sistema **BookRecommender**. Il documento è rivolto a sviluppatori e manutentori del sistema, fornendo una panoramica approfondita delle scelte architetturali, algoritmiche e strutturali adottate.

2 Struttura dell'Applicazione

L'applicazione è organizzata secondo un'architettura modulare in pacchetti. I principali componenti sono:

- `controller` – gestione logica dell'applicazione
- `model` – rappresentazione dei dati (libri, utenti, raccomandazioni)
- `view` – interfaccia grafica (JavaFX/Swing)
- `utils` – classi di supporto

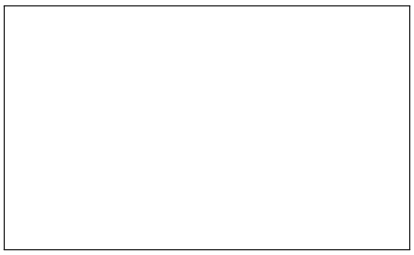

Figura 1: Diagramma delle classi principali (UML)

3 Scelte Progettuali

3.1 Diagrammi UML

Il sistema è basato su un modello ad oggetti. Di seguito è riportato il diagramma delle classi e delle interazioni principali.

- Class Diagram
- Sequence Diagram (per l'uso della raccomandazione)

3.2 Schema ER

Nel caso di uso di una base di dati:

Figura 2: Schema ER: utenti, libri, raccomandazioni

4 Scelte Architetturali

L'applicazione è costruita secondo il pattern MVC:

- Separazione tra logica, presentazione e gestione dati
- Facilità di estensione e testabilità

Altri aspetti architetturali:

- Modularità (uso dei package)
- Dipendenze gestite tramite Maven
- JavaFX come UI framework

5 Strutture Dati Utilizzate

Sono state utilizzate strutture dati standard della Java Collection Framework:

- `List<Book>` per collezioni di libri
- `Map<String, List<Book>>` per raccomandazioni per utente
- `Set<String>` per evitare duplicati

6 Scelte Algoritmiche

Il sistema di raccomandazione può includere (esempi):

- Similarità basata su contenuto (coseno tra vettori di parole chiave)
- Filtraggio collaborativo semplice (similitudine tra utenti)
- Ordinamento per punteggio

```
public List<Book> recommend(String userId) {
    List<Book> liked = getLikedBooks(userId);
    Map<Book, Double> scores = new HashMap<>();
    for (Book book : allBooks) {
        double score = computeSimilarity(book, liked);
        scores.put(book, score);
    }
    return scores.entrySet().stream()
        .sorted(Map.Entry.comparingByValue(Comparator.
           reverseOrder()))
        .limit(5)
        .map(Map.Entry::getKey)
        .collect(Collectors.toList());
}
```

Listing 1: Algoritmo semplificato di raccomandazione

7 Gestione dei File

Il sistema carica/salva file di dati nei seguenti formati:

- `CSV` per dati tabellari (libri, utenti)
- `JSON` opzionale per configurazioni

```
src/
|-- main/
|    |-- resources/
|         |-- data/
|              |-- books.csv
|              |-- users.csv
```

8 Pattern Utilizzati

8.1 Singleton

Utilizzato per gestire l'accesso centralizzato al sistema di raccomandazione.

```
public class RecommendationEngine {
    private static RecommendationEngine instance;
    private RecommendationEngine() {}

    public static synchronized RecommendationEngine
       getInstance() {
        if (instance == null)
            instance = new RecommendationEngine();
        return instance;
    }
}
```

Listing 2: Pattern Singleton per RecommendationEngine

8.2 MVC

- Model: classi `Book`, `User`
- View: classi GUI JavaFX
- Controller: logica di interazione

9 Documentazione JavaDoc

Tutte le classi sono documentate con JavaDoc. Esempio:

```
/**
* Classe che rappresenta un libro.
* Contiene titolo, autore, genere e valutazione.
```

```
*/
public class Book {
    private String title;
    private String author;
    private String genre;
    private double rating;
    // costruttori, metodi getter/setter, ecc.
}
```

Listing 3: JavaDoc per classe Book

10 Conclusione

Il sistema BookRecommender è progettato con modularità e chiarezza per facilitarne l'estensione e la manutenzione. La documentazione tecnica, insieme alla JavaDoc generata automaticamente, permette un agevole accesso alle componenti del codice.