BiBooks: prima implementazione

Componenti del gruppo e link al repository:

Andrea Pascuzzi: a.pascuzzi2@campus.unimib.it

Simone Pierro: s.pierro@campus.unimib.it

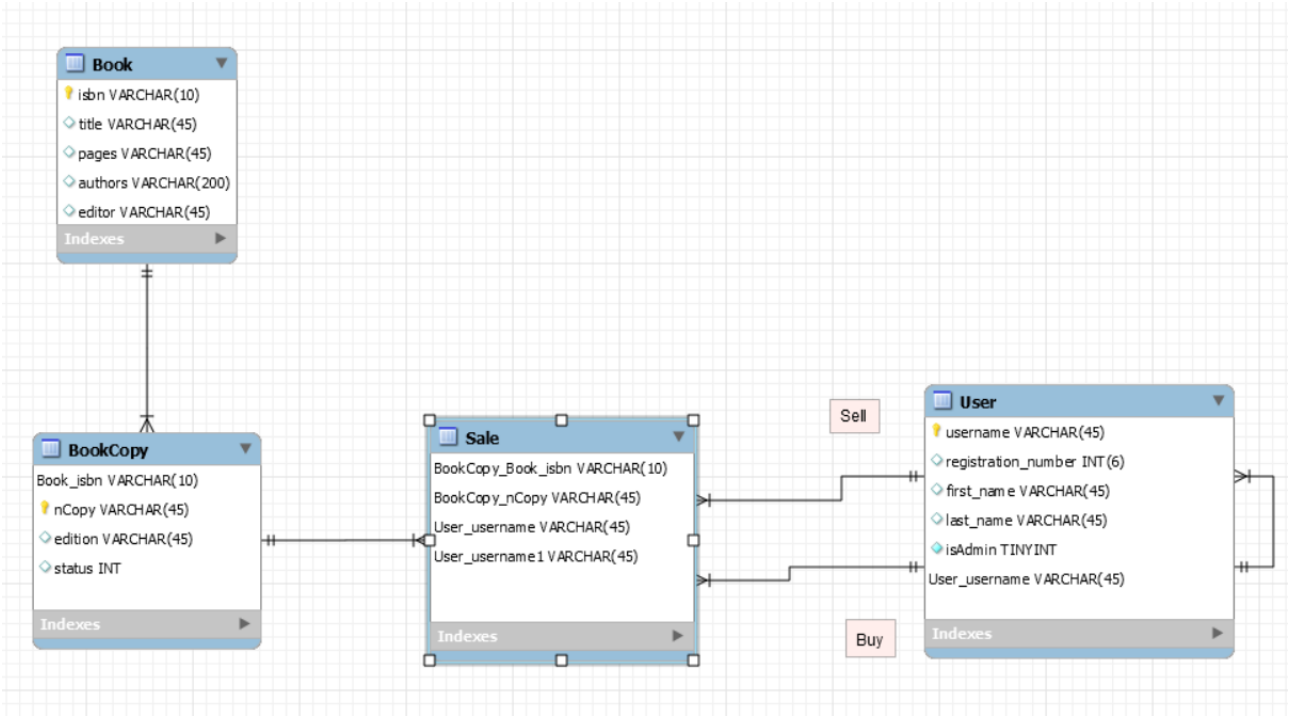
David Gargaro: d.gargaro@campus.unimib.it

Repository GitLab: https://gitlab.com/spierro/2022_assignment3_bibooks

A seguito dell'utilizzo delle tecniche di elicitazione si vuole iniziare a sviluppare un primo prototipo di BiBooks.

Backend:

Per far ciò ci si serve del framework SpringBoot e di alcuni pacchetti per la persistenza dei dati, quali JPA e H2.
In particolare H2 è un database di tipo in-memory, ciò significa che se l'applicazione è stoppata o riavviata i dati legati all'istanza runtime vengono distrutti.
Questa scelta deriva dal fatto che al momento si vuole effettuare uno showcase dell'applicazione BiBooks, in modo da iniziare a ragionare sui componenti essenziali del progetto.
Lo schema Entity-Relationship che modella il dominio del progetto è il seguente:

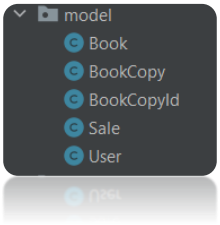


In particolare le entità coinvolte sono le seguenti:

- **Book**: ogni libro viene identificato dall'ISBN come chiave primaria.
- **BookCopy:** ogni libro può avere una o più copie associate (identificata/e da una chiave composta dalla chiave esterna di Book e un identificativo autogenerated)

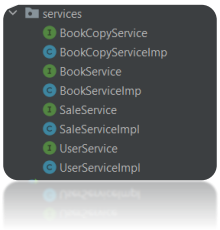
- **Sale:** ogni BookCopy può essere coinvolta nel processo di vendita tra utenti. Ogni Sale viene identificato dalla chiave di BookCopy e dalla chiave primaria degli utenti Buyer e Seller.
- **User:** rappresenta l'entità associata all'utente registrato su BiBooks. Un utente registrato può visualizzare le copie di Book disponibili e vendere/acquistare libri. Viene utilizzato il numero di matricola come chiave primaria.

Vi è un self-loop tra le entità **User** (1-n), questa relazione rappresenta la possibilità per un utente di essere amministratore di altri utenti.
Partendo dallo strato architetturale inferiore, ogni entità ha associato un modello.



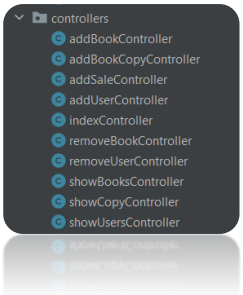
Ogni classe Model ha il suo Repository, un'interfaccia che estende la classe CrudRepository del package `org.springframework.data.repository` .
In questo modo alcuni metodi CRUD sono già implementati e possono essere richiamati dagli strati architetturali superiori.

Ad ogni entità/modello è poi associata l'interfaccia Service che dichiara i metodi astratti utilizzati poi dal REST API Controller, ossia l'ultima classe dell'architettura la quale espone i metodi HTTP per compiere le operazioni CRUD sui dati.



Ogni service ha la sua implementazione, ossia una classe che implementa i metodi dichiarati dalle interfacce Service.
Ogni xxServiceImpl richiama i metodi CRUD esposti dai rispettivi Repository.

Vengono poi dichiarate le classi di tipo Controller.



I controller si occupano di fornire la pagine web arricchite dei dati recuperati attraverso

l'invocazione dei metodi delle classi Service.
Il progetto utilizza il design pattern Page Controller che vede la generazione di un controller per ogni pagina web html.

All'interno del progetto figura anche un BootStrapper (`BookLoader.java`) che si occupa di caricare in prima istanza alcuni dati fittizi per il modello **Book**.

Una volta avviato il progetto è possibile creare, leggere, modificare e cancellare istanze dei modelli per la quale vi è una concreta implementazione.

Fontend e filtri:

In prima istanza il Frontend di BiBooks è molto semplice, composto da Form per poter raccogliere i dati e creare le istanze associate alle entità di dominio.

La pagina web statica `index.html` contiene alcuni pulsanti per navigare all'interno del sito BiBooks.
Vi sono poi altre pagine web statiche per gestire la visualizzazione o aggiunta di oggetti nel relativo campo di interesse (User, Book, BookCopy).
Questa parte del progetto è molto standard, per questo motivo ci siamo limitati ad implementare le pagine web necessarie a soddisfare le richieste dell'assignment.
In particolar modo la pagina web statica `showCopy.html` contiene al suo interno una input di tipo testuale per poter effettuare una ricerca sul set di istanze del modello BookCopy a disposizione del sistema.
Infatti questa ricerca avviene utilizzando il parametro ISBN fornito come QueryParam durante la richiesta HTTP del client (browser).
La ricerca quindi avviene sull'entità Book associata alle entità BookCopy, raggruppando quindi tutte le BookCopy per ISBN.

Questo tipo di matching avviene nella classe `showCopyController.java` che oltre ad effettuare la ricerca nel caso in cui il parametro isbn venga passato, si occupa anche dell'esposizione delle copie disponibili.