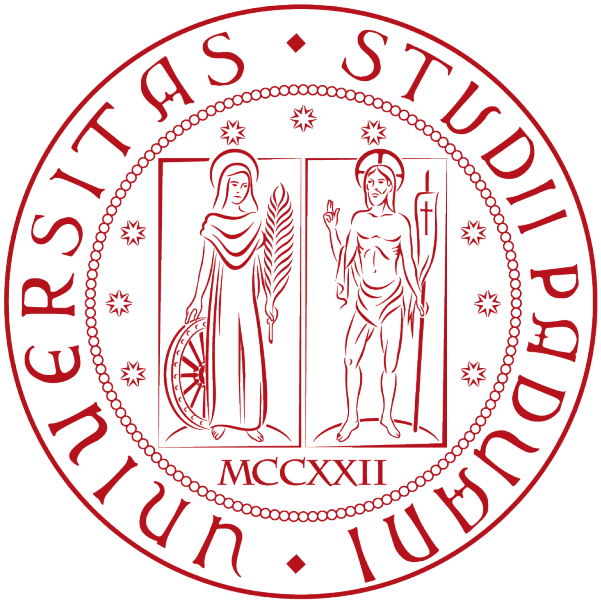

Relazione Prog. ad Oggetti

Cristian Pirlog, Matricola 1097011 - 10 Settembre 2017

MyBookshelves

Indice

1 Scopo del progetto

MyBookshelves si prefigge lo scopo di dare una mano ai lettori nella categorizzazione dei libri di proprio interesse.
Sono presenti diverse categorie di utenti che avranno un livello di accesso al contenuto basato sulla tipologia del loro account.
La funzionalità principale offerta da MyBookshelves è la possibilità di cercare all'interno del database i libri interessati, osservarne i dettagli (insieme ad una breve descrizione sulla trama), lasciare una recensione (se permesso) e/o vedere quelle degli altri ed infine aggiungere questi libri all'interno di categorie personalizzabili per tenerne traccia.
Gli utenti Moderatori inoltre potranno anche inserire libri all'interno del database e modificarli.
La stessa funzionalità è offerta anche all'admin (che è unico). L'admin può inoltre manipolare la maggior parte dei dati degli utenti, creare utenti ed eliminarli.

2 Struttura

2.1 Contenitore

Nella scelta del contenitore è stata posta attenzione in modo da avere un contenitore che fosse principalmente efficiente ed inoltre che permettesse un facile accesso ai dati. Per questo motivo è stato utilizzato un contenitore associativo, il **QMap**, il quale offre un dizionario basato sugli red-black-trees. Il costo per la consultazione dei dati al suo interno, grazie alla struttura ad albero, è di *O(logn)*.
Il QMap è stato utilizzato in diverse occasioni, ma principalmente per i due contenitori principali contenuti all'interno della classe **Database**, ovvero:

1. **QMap<QString,User*>** che rappresenta una QMap di puntatori a **User** identificati dalla chiave **Username** in formato **QString**.
2. **QMap<QString,Book*>** che rappresenta una QMap di puntatori a **Book** identificati dalla chiave **ISBN** sempre in formato **QString**.

2.2 Pattern Model-View-Controller

Per lo sviluppo è stato scelto, per quanto possibile, il pattern **MVC** in modo da separare la parte grafica dall'implementazione logica. Di fatti, nel Model vengono gestiti direttamente i dati, mentre nella View vengono visualizzati i dati contenuti nel model offrendo loro inoltre una rappresentazione grafica, che si occupa dell'interazione con gli utenti, come ad esempio l'inserimento dei dati; in fine, all'interno del Controller viene definita la logica di controllo.

3 Gerarchie di tipi e altre classi

Di seguito verranno illustrate le scelte progettuali necessarie alla realizzazione del modello dei dati di MyBookshelves:

3.1 Class Database

Si tratta di una classe d'utilità, utilizzata per le operazioni di manipolazione dei campi dati necessari per lo scopo del progetto.

3.1.0.0.1 Campi dati I campi dati presenti sono:

- **QMap<QString,User*> container;**: come spiegato nel sottoparagrafo **2.1** rappresenta il contenitore degli utenti.
- **QMap<QString,Book*> book_container;**: come spiegato nel sottoparagrafo **2.1** rappresenta il contenitore dei books.
- **static QString db_file;**: viene utilizzato per settare staticamente il nome del file richiesta per la memorizzazione dei dati degli utenti.
- **static QString bc_file;**: viene utilizzato per settare staticamente il nome del file richiesta per la memorizzazione dei dati dei books.

3.1.0.0.2 Metodi Oltre ai soliti metodi get e set sono presenti diversi metodi per la manipolazione dei dati e la ricerca degli stessi all'interno dei contenitori.
Inoltre sono presenti i metodi **Save()** e **Load()** che si occupano rispettivamente, della memorizzazione su file dei dati alla fine della vita del **Database** e del caricamento degli stessi durante la costruzione dell'oggetto **Database**.

Per la lettura e scrittura su file la scelta è ricaduta sulle librerie offerte da qt, cioè **QXmlStreamReader** e **QXmlStreamWriter** per la velocità e semplicità d'utilizzo.

3.2 Classi Book & Bookshelf

La classe **Book** rappresenta il fulcro di questo progetto. I campi dati sono: **isbn,title,author, edition,publisher, format,description, rating,publishing_date**.

Inoltre abbiamo:

- **QVector<QString> genres**: che serve a raccogliere un vector di elementi che rappresentano i vari generi possibili del libro.
- **QVector<int> ratings**: raccoglie l'insieme di ratings dati dall'utenti.
- **QMap<QString,QString> quotes**: raccoglie le citazioni riguardanti quel libro con **Author, Quote**.
- **QMap<QString,QString> reviews**: raccoglie l'insieme di recensioni date dall'utente con **Username, Review**.

E' stato scelto di utilizzare il **vector** perchè non c'era bisogno di tenere traccia dell'ordine degli elementi al suo interno. Il **QMap** è stato invece scelto per la facilità d'utilizzo nella ricerca della coppia chiave/valore.
I metodi presenti sono i soliti **get e set**, richiesti per prendere e modificare i dati del **Book** in questione.
NB: Le **quotes** e i **genres NON** sono stati implementati a livello visivo data la loro aggiunta alla fine del progetto e la mancanza di tempo per portarla a termine.

Per quanto concerne la classe **Bookshelf**, che rappresenta la categoria di libri che alcuni tipi può aggiungere al proprio account, presenta i seguenti campi dati:

- **name:** rappresenta il nome della stessa, che è unico per l'utente.
- **description:** rappresenta una breve descrizione personale del **Bookshelf** in questione.
- **QVector<QString> bl**: rappresenta la lista di libri al suo interno identificati dall'attributo **ISBN** in formato QString.

I metodi invece sono i soliti **get/set** ed alcuni per la manipolazione più approfondita dei dati:

- *void addBook(const QString&)*: che serve ad aggiungere un libro all'interno del vettore.

- *void remBook_BS(const QString&)*: che serve ad eliminare un libro dall vettore.

3.3 Class Administrator

La classe **Administrator** rappresenta l'amministratore il quale è unico. Egli può aggiungere o rimuovere utenti, oltre che modificare parte dei loro dati. Stesse funzionalità sono applicate anche ai libri del contenitore C.
Come unico campo dato abbiamo: **Database* database;**. Questo serve a poter accedere al database per la manipolazione dei dati.
Per quanto riguarda i metodi utilizzati questi sono in gran parte metodi che servono al settaggio, alla rimozione e ai controlli sui dati. In particolare abbiamo invece il metodo:
void PromoteUser(const QString&,const QString&): serve per la promozione o retrocessione dell'utente.

3.4 Gerarchia degli Utenti

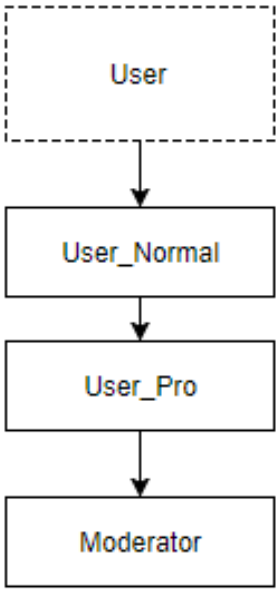

Figura 1: Gerarchia degli User

3.5 Abstract Class User

Questa gerarchia rappresenta l'insieme degli utenti che andranno ad utilizzare il contenitore C rappresentante la classe **Book** che vedremo più avanti.
Si parte da un generico utente, elemento contenuto all'interno della classe base astratta **User**, presente alla base della gerarchia.
I campi dati presenti sono:

- **username:** Username dell'utente.
- **name:** Name dell'utente.
- **surname:** Surname dell'utente.
- **password:** Password dell'utente.
- **email:** Email dell'utente.

- **type:** Tipologia di account dell'utente.
- **bsl:** Vettore contenente i puntatori agli oggetti della classe **Bookshelf**.

Ogni **User** viene identificato univocamente attraverso l'attributo *username*. Questa classe diventa astratta grazie ai metodi virtuali puri:

```
virtual std::list<const Book*> searchBooks(Database*,const
QString&)=0;
```

e

```
virtual QString accountType() const =0;
```

Grazie alla dichiarazione virtuale di questi metodi saranno facilitati in futuro gli upgrade/downgrade riguardanti le possibilità offerte agli utenti. Attraverso la prima si potrà aggiornare le funzioni di ricerca di ogni singolo tipo di **User**, mentre attraverso la seconda si potrà mostrare agli utenti queste e altre possibili funzioni loro offerte.

Oltre ai precedenti metodi sono presenti altri utili ad impostare i campi dati e estrarre le informazioni quando necessario (i soliti metodi get/set). Ci sono inoltre alcuni metodi che permettono di controllare se un **Bookshelf** esiste:

```
bool checkIfBSExists(const QString&) const;
```

e se un **Book** è presente all'interno di un determinato **Bookshelf.**

NB: all'interno del corpo del costruttore vengono aggiunti 3 **Bookshelves** che rappresentano le categorie dove saranno inseriti i libri che sono già stati letti**(Read)**, i libri che si vogliono leggere **(To read)** ed i libri che si stanno leggendo. **(Reading)**

3.6 Classe User Normal

Questa tipologia di utente, oltre alle possibilità offerte dalla classe base astratta, implementa entrambi i metodi puri sopra descritti.
Il primo metodo, che permette la ricerca dei **Books** all'interno del database, permette all'**User_Normal** di cercare il libro solamente in base al **ISBN** del titolo.
Il metodo *accountType* mostra all'utente le sue capacità.

3.7 Classe User Pro

Questa tipologia di utente ha la possibilità di aggiungere nuovi Bookshelves e modificarli attraverso il metodo:

```
void addBookshelf(const QString&, const QString&);
```

Inoltre, il metodo *searchBooks* sopra descritto, offre la capacità di cercare i libri anche in base ai campi dati **Title, Author, Publisher, Format e Description**.
Il metodo *accountType* mostra all'utente le sue capacità.

3.8 Classe Moderator

Questa tipologia di utente, oltre alle possibilità offerte dalle superclassi sopra descritte, può aggiungere nuovi libri attraverso l'utilizzo del metodo:

```
    virtual void AddBook(const QString &i, const QString &bt,
  const QString &ba, const QString &e, const QString &publ, const
   QString &bf, const QString &des, const QDate &publ_date, const
   double &br,Database* db);
```

Inoltre, anche questa classe implementa il metodo *accountType* aggiornando le funzionalità offertagli.

4 Uso del polimorfismo

Il polimorfismo è presente all'interno della gerarchia degli **User**.

4.1 Polimorfismo gerarchia User

I metodi che sono stati utilizzati sono:

```
virtual std::list<const Book*> searchBooks(Database*,const QString
   &)=0;
```

Questo metodo *virtuale puro* è stato implementato all'interno delle classi **User_Normal**, **User_Pro** e **Moderator**, ricordando in breve che all'interno della classe User_Normal viene permesso all'utente di cercare un libro solo in base all' ISBN, mentre per le altre sottoclassi le possibilità si estendono anche ai campi dati **Title, Author, Publisher, Format e Description**.
Questo metodo viene utilizzato all'interno dell'**User Controller** alla riga 28, dove gli elementi risultanti dalla ricerca vengono inseriti all'interno di una *std::list<const Book*> temp*. Questa lista verrà ritornata alla funzione chiamante presente all'interno della **User_View**. In questo modo, la tabella sottostante alla barra di ricerca verrà riempita successivamente con la lista di elementi risultanti.

```
virtual QString accountType() const =0;
```

Questo metodo *virtuale puro* è stato implementato all'interno delle classi **User_Normal**, **User_Pro** e **Moderator** dove vengono descritti, attraverso il ritorno di una **QString** i dettagli riguardanti le funzionalità offerte ad ogni tipologia di utente.
Questo metodo viene utilizzato all'interno della **User_view** alla riga 199, dove attraverso ogni utente, qualsiasi sia la sua tipologia, potrà aprire attraverso l'attivazione dell'apposito **QAction** all'interno della barra di stato in alto, una finestra dove potrà visualizzare le informazioni richieste.

4.2 Altri metodi

Nonostante il polimorfismo non sia stato implementato per i seguenti metodi, essi sono predisposti per un tale utilizzo.

```
virtual void addBookshelf(const QString&,const QString&);
```

Questo metodo presente all'interno della classe **User_Pro** permette all'utente di questo tipo di aggiungere una nuova **Bookshelf** personale. Viene utilizzato all'interno dell' **User_Controller** alla riga 148.

```
virtual void AddBook(const QString &i, const QString &bt, const
   QString &ba, const QString &e, const QString &publ, const
   QString &bf, const QString &des, const QDate &publ_date, const
   double &br,Database* db);
```

Questo metodo presente all'interno della classe **Moderator** permette all'utente di questo tipo di aggiungere un nuovo **Book**. Viene utilizzato

5 Manuale utente

Per la parte grafica è stato scelto l'utilizzo del Qt Designer.
Una volta aperta l'applicazione l'utente potrà decidere se premere Login in alto a destra oppure registrarsi premendo il tasto Register. La schermata che si vedrà, nel caso si premessero entrambi i tasti sarà questa:

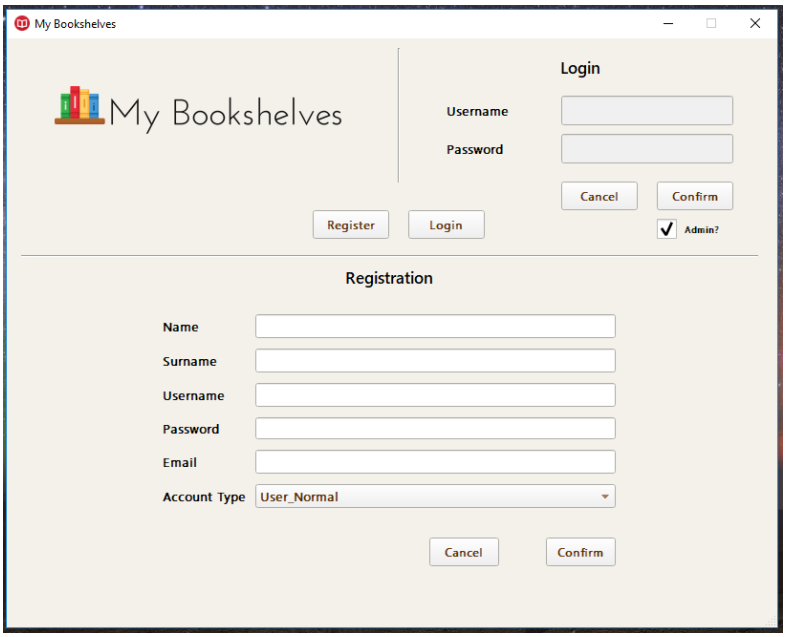

Figura 2: Finestra precedente al Login

5.1 Modulo Administrator

Attivando la checkbox **Admin?** e successivamente premendo Confirm si avrà accesso al modulo **Administrator**. In questo modo si avrà accesso alla **AdministratorMainWindow** seguente:

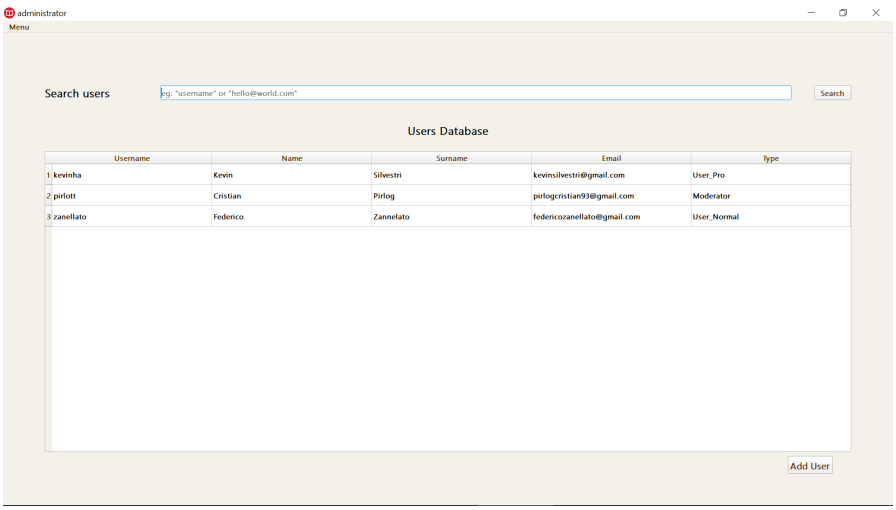

Figura 3: Finestra principale del modulo Administrator

In questa finestra è presente una barra in cui è presente il *Menu* a tendina, contenente gli actions **Open Books**, **Add User** e **Quit**.
In alto possiamo osservare la **Search bar** dove si potranno inserire i termini desiderati. Premendo **Search** la tabella sottostante verrà aggiornata con i soli utenti risultanti dalla ricerca. In basso

a destra abbiamo invece il tasto **Add User** presente anche all'interno del Menu, alla quale pressione mostrerà un modulo di registrazione di un nuovo utente.
Faccendo il doppio click all'interno di una riga della tabella verrà aperta la seguente finestra:

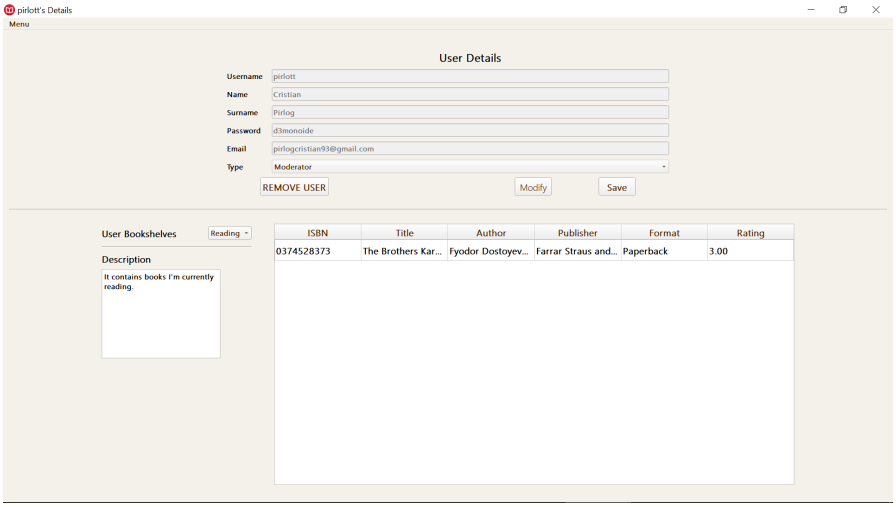

Figura 4: Vista delle informazioni dell'utente.

All'interno di questa finestra possiamo osservare i dettagli relativi all'utente precedentemente selezionato. Nella parte superiore avrà la possibilità di eliminare l'utente o modificarne i dati. Mentre in basso a destra potrà osservare la lista di libri presente in ogni **Bookshelf**. Dall'menu in alto si potrà invece chiudere la finestra.

Dalla **AdministratorMainWindow** si potrà, attraverso il Menu in alto, premere **Open Books** per aprire una finestra molto simile all'attuale. In si potrà cercare un libro utilizzando una barra di ricerca e visualizzarne i risultati nella tabella sottostante. Doppio cliccando su una riga, si aprirà una finestra che offrirà la possibilità di vedere le informazioni **importanti** riguardanti il libro e modificarne alcuni dati.
Dall'menu in alto si potrà aggiungere un nuovo libro o chiudere l'attuale finestra.

5.2 Modulo User

Una volta loggati, si presenterà la seguente finestra:

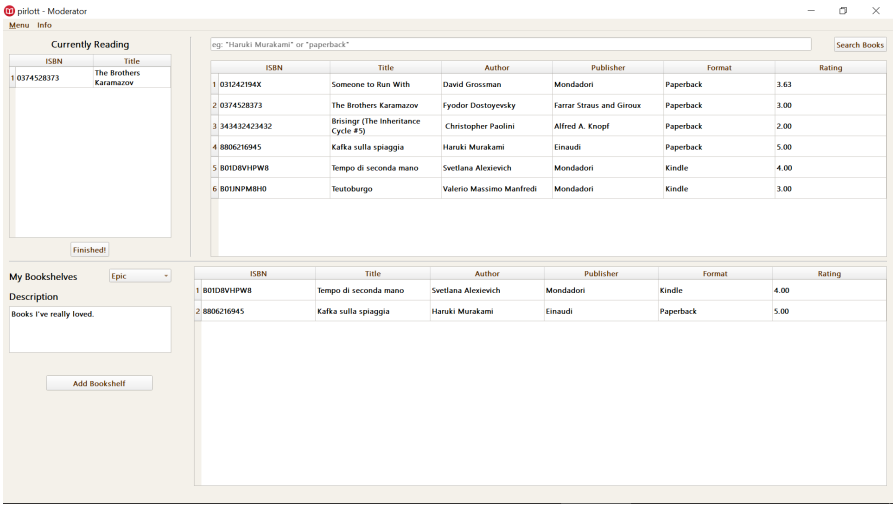

Figura 5: Finestra principale di un utente

Da questa finestra possiamo osservare come un utente di tipo **Moderator** vedrebbe questa finestra. In alto possiamo ricercare i libri che ci interessano all'interno del database. A sinistra possiamo osservare invece la **Bookshelf** standard **Reading** in cui saranno presenti i libri che si stanno leggendo. Nella parte inferiore della finestra possiamo osservare un menu a tendina dove si potrà scegliere la **Bookshelf** da visualizzare, esclusa la **Reading**, già presente in alto a sinistra. Si ha anche la possibilità per gli utenti **User_Pro** e **Moderator** di aggiungere un nuovo Bookshelf premendo il pulsante Add Bookshelf a sinistra. Questo pulsante non sarà visualizzabile da parte dell'**User_Normal**.
In alto a sinistra abbiamo il bottone **Finished**, il quale una volta premuto avendo selezionato un elemento della tabella sovrastante, se si è **User_Normal** il libro in questione verrà spostato nella **Bookshelf Reading**. Attraverso l'utilizzo dei **SIGNAL/SLOT** i campi dati all'interno delle tabelle verranno aggiornati automaticamente. Se si è invece un'utente di tipologia diversa, si avrà la possibilità di lasciare una **Review** ed un **Rating**.

Dal menu in alto si potrà aprire una finestra che conterrà le informazioni riguardanti il livello di accesso dell'attuale tipo di account premendo **Info**. Di fianco abbiamo **Menu**, attraverso il quale possiamo accedere alle informazioni dell'utente premendo **Account**, aggiungere un nuovo libro attraverso **Add new book** (non visualizzabile se non si è **Moderator**). Inoltre si potrà uscire dal programma premendo Logout.

Premendo **Account** nel menu in alto si aprirà una finestra contenente tutte le informazioni riguardanti il proprio account. Si avrà la possibilità di modificare i dati ed eliminare l'account. Inoltre nella sezione a destra si potranno modificare i dati delle **Bookshelves** (se la **Bookshelf** non fa parte delle 3 standard).

Faccendo doppio su una delle righe all'interno di qualsiasi tabella si avrà accesso alla **book_view**:

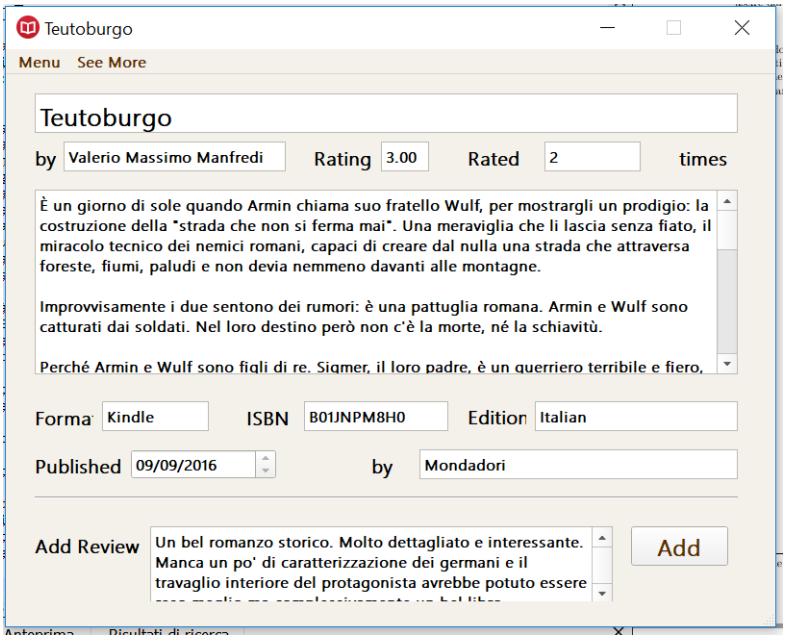

Figura 6: Vista del libro

All'interno di questa finestra possiamo vedere tutte le informazioni riguardanti il libro. Inoltre, se **non** si è **User_Normal** si potrà lasciare una review oppure editarne la vecchia.
Dal Menu in alto si ha accesso a diverse opzioni. Premendo **See More** si potranno visualizzare le recensioni degli utenti e facendo il doppio click sulla tabella verrà mostrata la recensione in dettaglio.

Premendo **Menu** invece, si potrà aggiungere il libro ad una delle proprie **Bookshelf** (le tabelle nella mainwindow verranno aggiornate in automatico) premendo **Add to Bookshelf.** Se invece si vuole eliminare il libro da tutte le **Bookshelves** c'è una **action** apposita chiamata **Remove from Bookshelves**.
Se si è Moderator, si avranno delle opzioni ulteriori che permetteranno di modificare i dati del libro.

A Indicazioni conclusive

A.1 Impegno temporale

Progettazione modello e GUI: 6h
Codifica modello e GUI: 48h
Debugging: 10 h

A.2 Informazioni tecniche

OS: Windows 10 Home Edition
QT Version: Qt Creator 4.3.1(Community) based on Qt 5.9.1
Compiler: MSVC 2015, 32 bit

A.3 Compilazione ed esecuzione

Per la corretta compilazione del progetto è richiesta l'esecuzione del comando **qmake MyBookshelves.pro**, contenuto all'interno della cartella MyBookshelves. Attraverso l'utilizzo di questo file sarà possibile la generazione automatica tramite qmake del Makefile. Il comando **make**, è il prossimo richiesto.
Per l'esecuzione dello stesso bisogna invece utilizzare il comando **./MyBookshelves**
Oltre ai file necessari, vengono inoltre consegnati i files **Book_Database.xml e User_Database.xml**, richiesti per l'accesso ad alcuni dati preinseriti. Entrambi i files si possono trovare nella cartella **./resources/Databases**, interna a quella del progetto.

A.4 Accesso utenti

Moderator
username: pirlott
password: d3monoide

User_Pro
username: kevinha
password: silvestri01

User_Normal
username: zanellato
password: fedebrichi