UNIVERSITÀ
DI TRENTO

Dipartimento di Ingegneria e Scienza
dell'Informazione

Progetto:

Titolo del documento:

Documento di architettura

Document Info:

Doc. Name	*D3-XBooks_architettura*	Doc. Number	*D3*
Description	*Il documento include diagrammi delle classi e codice OCL*		

Scopo del documento **2**
1. Diagramma delle classi **2**
1.1 Utenti 3
1.2 Autenticazione 3
1.3 Librerie 5
1.4 Ricerca 6
1.5 Database 6
1.6 Diagramma delle classi complessivo 7
2. OCL **8**
2.1 Utente 8
2.2 Librerie 8
2.3 Ricerca 9
3. Diagramma delle classi con OCL **10**

Scopo del documento

Questo documento delinea l'architettura del progetto XBooks, focalizzandosi su due aspetti principali: la definizione delle classi che verranno implementate e la logica che governa il comportamento del software. La progettazione è rappresentata attraverso diagrammi delle classi in UML e specificata mediante codice in OCL. Nel documento precedente sono stati esposti i diagrammi relativi ai casi d'uso, al contesto e ai componenti del sistema. Partendo da quella base, ora viene descritta l'architettura del sistema, approfondendo sia la struttura delle classi che la logica operativa del software. I diagrammi delle classi illustrano la struttura in UML, mentre la logica, esprimibile unicamente attraverso OCL nel contesto UML, viene formalizzata per garantire una chiara comprensione delle regole di funzionamento del sistema.

1. Diagramma delle classi

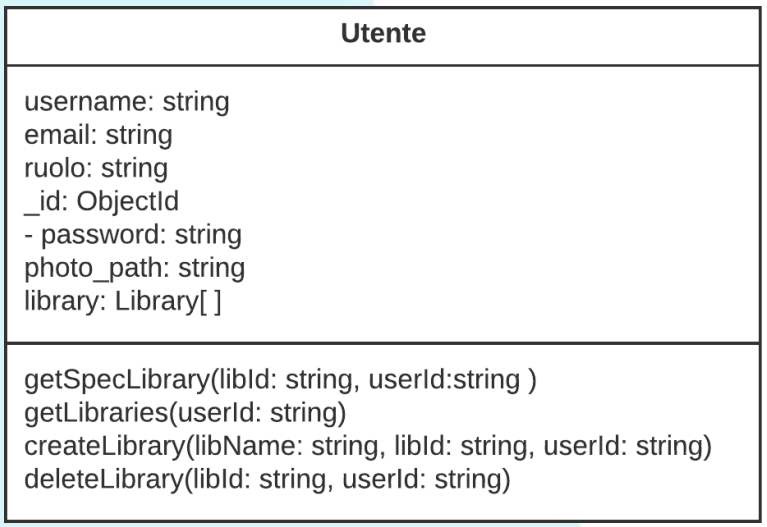
In questo capitolo vengono descritte le classi sviluppate per il progetto XBooks. Ogni componente del diagramma dei componenti è stato tradotto in una o più

classi, ciascuna dotata di un nome, un insieme di attributi che gestiscono i dati e una serie di metodi che definiscono le operazioni della classe. Le classi possono essere collegate tra loro attraverso associazioni, permettendo di delineare le relazioni e le interazioni tra le diverse entità.

Di seguito vengono elencate le classi derivate dai diagrammi di contesto e dei componenti. Durante la definizione delle classi, si è data particolare attenzione a massimizzare la coesione interna e a ridurre al minimo l'accoppiamento tra le classi, per garantire una struttura solida e modulare.

1.1 Utenti

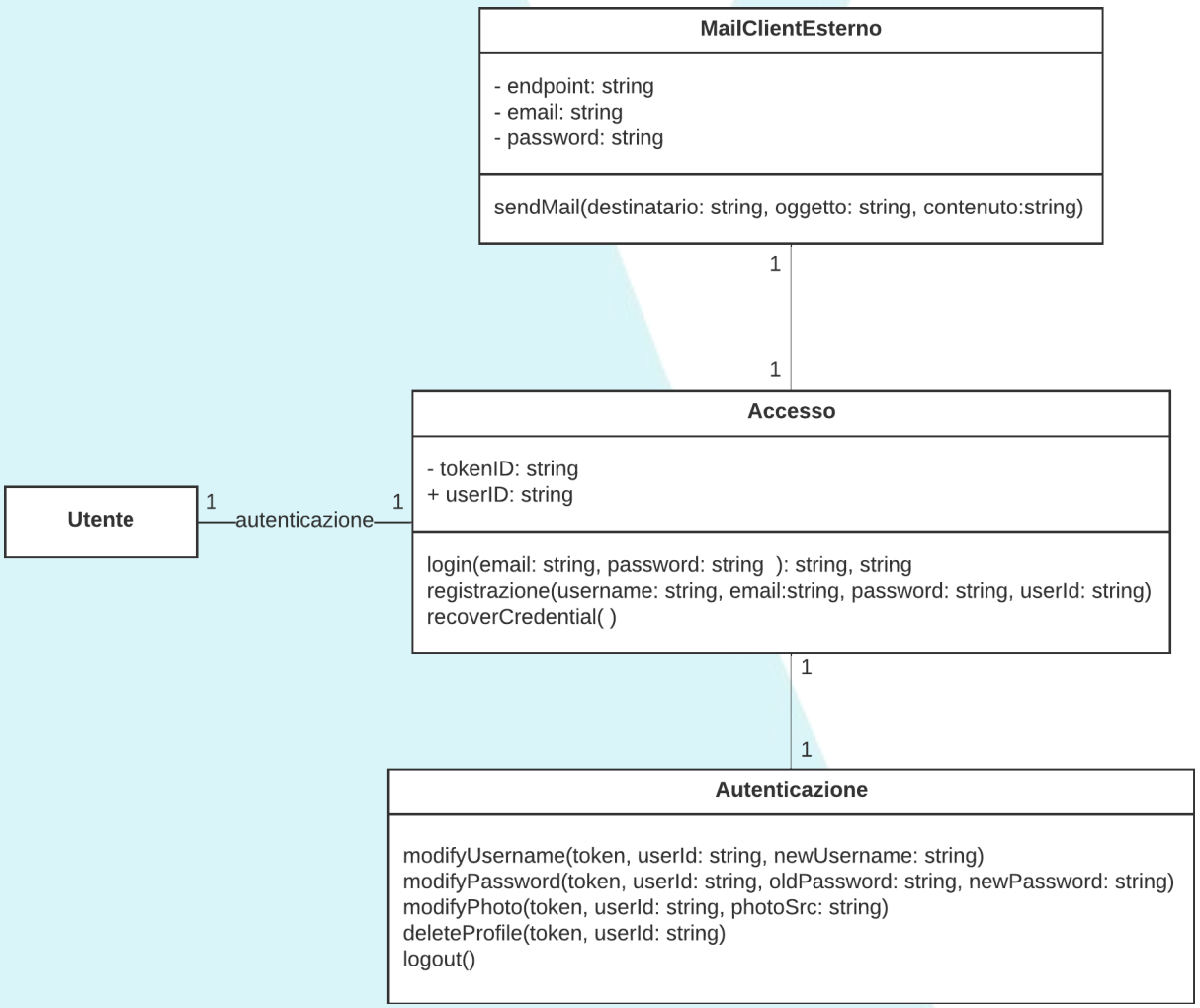
Analizzando i documenti precedenti è possibile notare la presenza di un solo attore, ovvero l'utente fruitore della webapp, di seguito è indicata la tabella in UML con gli attributi e le funzioni degli utenti.



1.2 Autenticazione

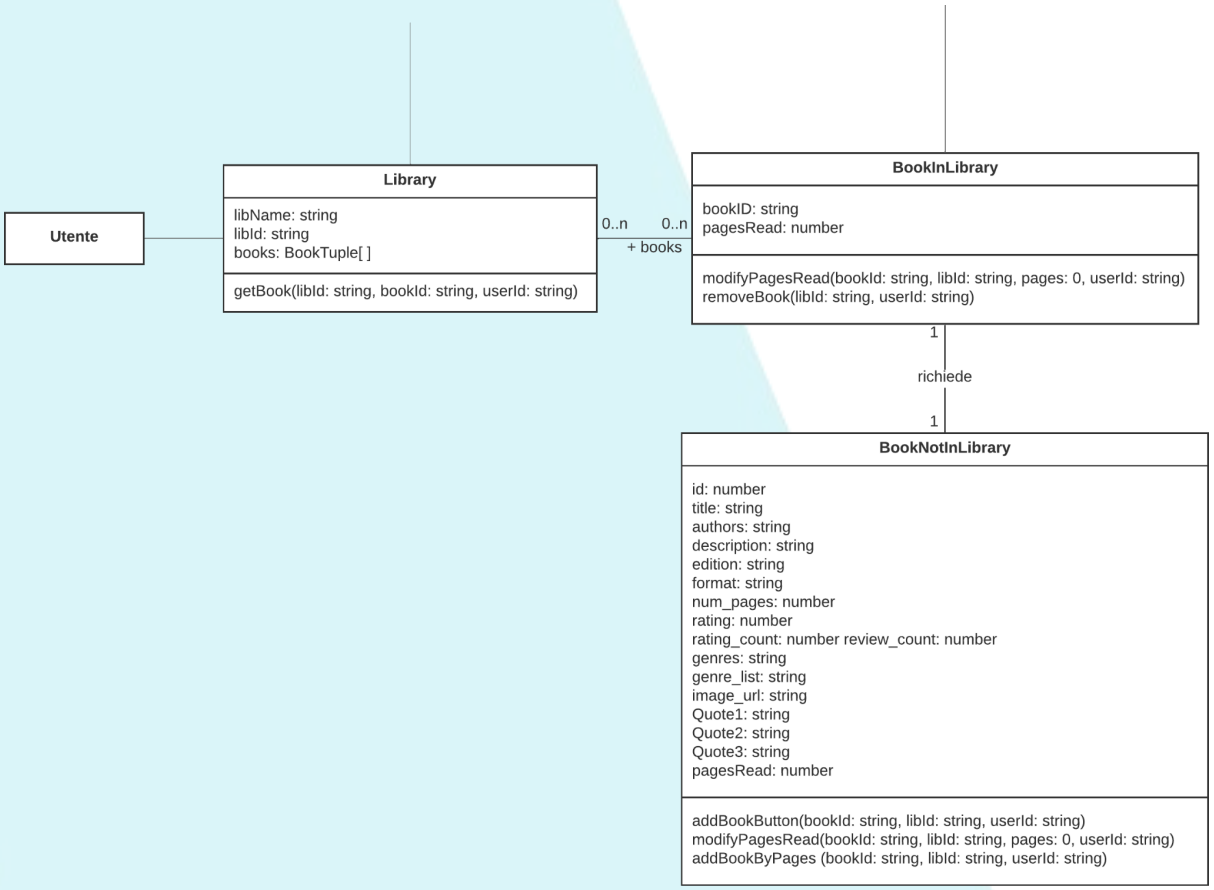
La fase di **autenticazione** comprende le fasi iniziali di approccio alla web app tramite registrazione e login, il quale è il componente più importante di questo blocco in quanto fornisce un **Token** di accesso poi necessario per utilizzare gli altri servizi di autenticazione. Oltre alle fasi iniziali l'autenticazione comprende anche i processi di modifiche relative al profilo quindi la possibilità di modificare

username, password, foto profilo e la possibilità di eliminare l'account. Il ruolo del Token è quello di verificare l'identità dell'utente alla richiesta delle funzioni elencate precedentemente, questo viene implementato per incrementare i livelli di sicurezza della web app. Di seguito viene presentato il diagramma delle classi in UML con i rispettivi attributi e metodi. Nel diagramma troveremo l`**attore utente** e le classi **Accesso** ed **Autenticazione**, facciamo questa distinzione in classi in quanto i metodi login e registrazione, come spiegato precedentemente avvengono prima di avere un Token di accesso mentre le altre funzioni per essere eseguite lo richiedono e quindi richiedono di aver utilizzato i 2 metodi di accesso. Oltre alle classi appena descritte troviamo la classe rappresentante il client esterno (**MailClientEsterno**) utilizzato per l'invio della mail nel recupero delle credenziali dell'utente non ancora autenticato.



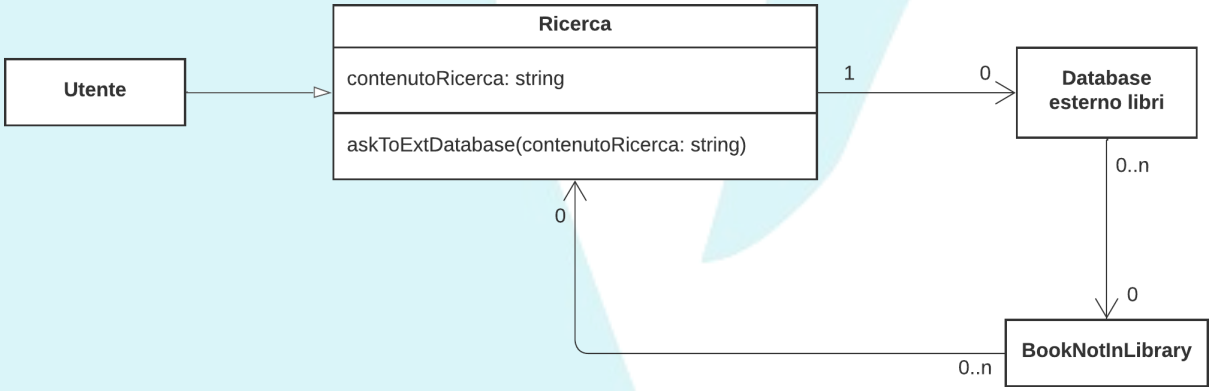
1.3 Librerie

Il gruppo di funzionalità al centro della web app è quello responsabile della manipolazione delle librerie di un utente, in particolare nello schema UML sottostante vediamo le 3 classi principali più l'utente. Risalendo lo schema a ritroso troviamo i libri nella versione estesa ottenuti da una API esterna (**BookNotInLibrary**), per memorizzare i libri all'interno delle librerie viene preso il campo id ed affiancato a **pagesRead** per creare un nuovo modello più compatto e leggero (**BookInLibrary**) per il database interno, il sistema poi risalirà agli altri dati tramite l'id dei libri. Una volta inserito un libro in una libreria sarà possibile modificare il numero di pagine lette come vediamo dal metodo modifyPagesRead.
La classe library contiene attributi e metodi delle singole librerie mentre i metodi per creare od eliminare librerie sono contenute nell'attore utente come abbiamo visto nel capitolo 1.1.



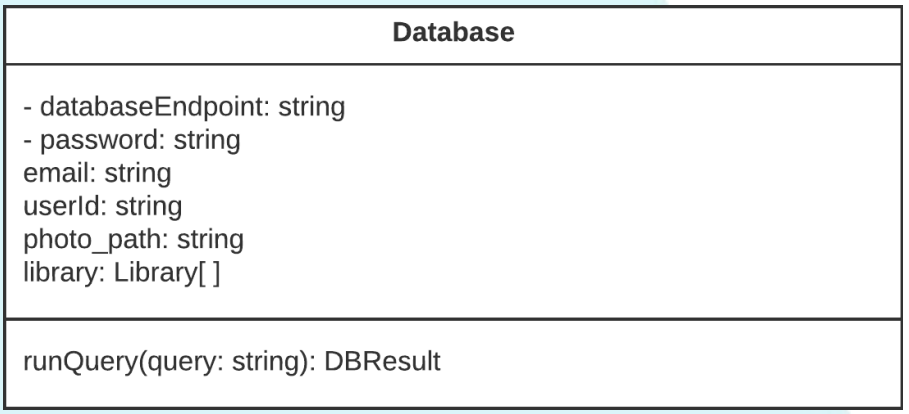
1.4 Ricerca

L'utente nella web app, come specificato negli scorsi documenti, potrà effettuare ricerche per cercare i libri desiderati nel database esterno dei libri. Per analizzare ciò sfruttiamo la classe di ricerca che, preso come parametro il contenuto della barra di ricerca, richiede al database esterno i risultati della ricerca. Il **database esterno dei libri** restituirà un array di **BookNotInLibrary**, componente precedentemente analizzato, alla classe ricerca che poi li restituirà all'**utente**. L'utente poi, come abbiamo visto al punto precedente potrà agire sui risultati attraverso il blocco delle Librerie.



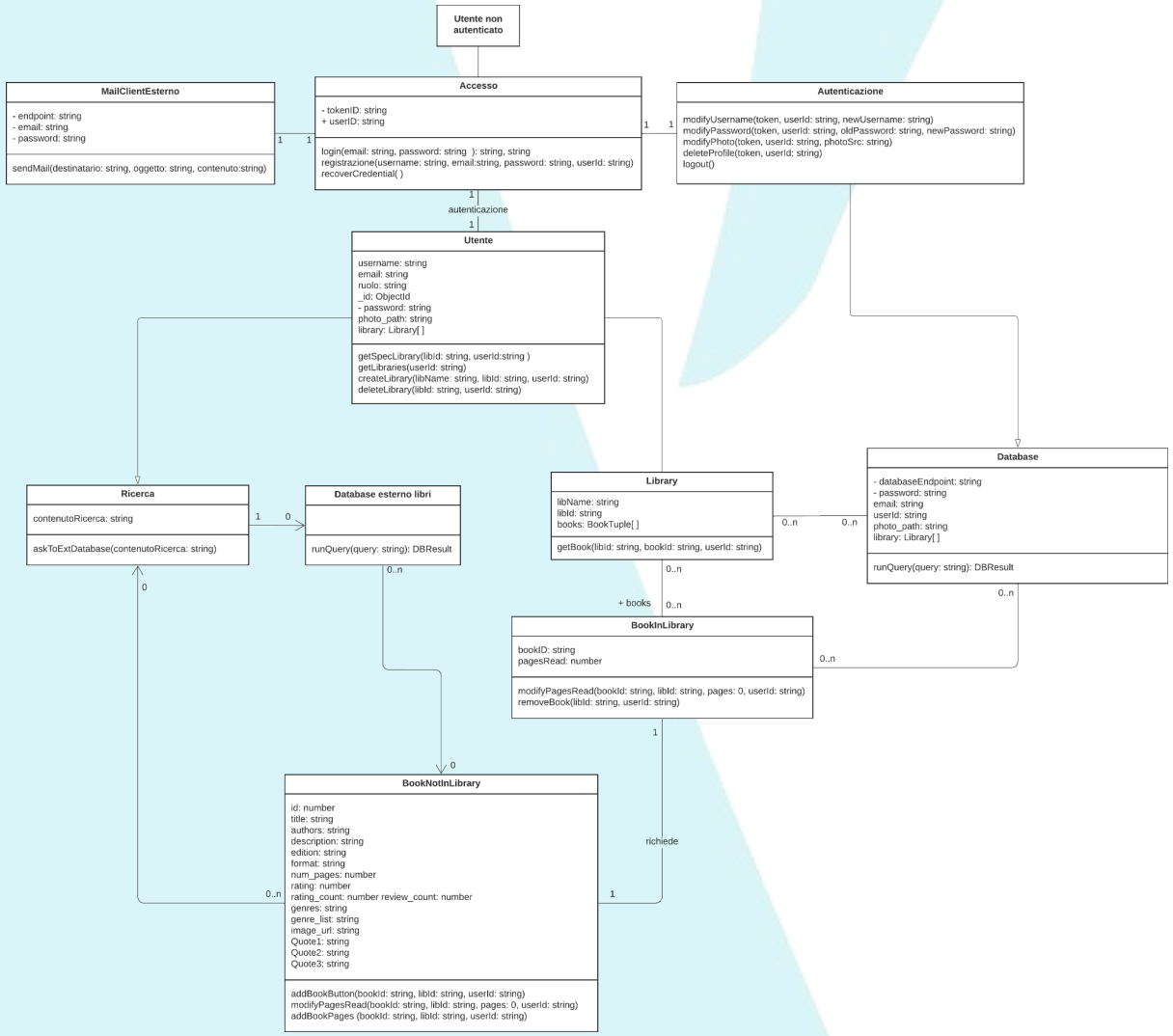
1.5 Database

Questa classe ha il compito di comunicare con il database, questa comunicazione avviene tramite l'invio di Query e la ricezione della risposta.



1.6 Diagramma delle classi complessivo

Di seguito è riportato il diagramma delle classi complessivo di tutte le classi analizzate precedentemente.

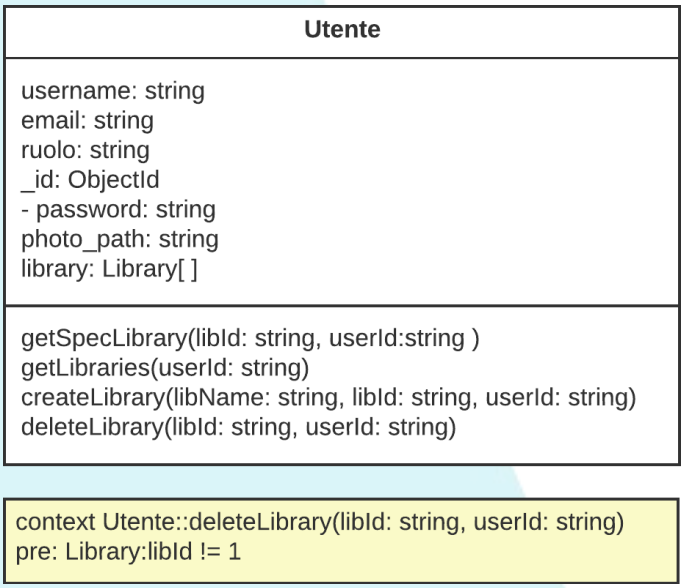


2.OCL

Questo capitolo presenta una descrizione formale della logica adottata per alcune operazioni di specifiche classi. La rappresentazione di questa logica avviene tramite Object Constraint Language (OCL), poiché i concetti trattati non possono essere formalizzati attraverso altri strumenti all'interno del contesto UML.

2.1 Utente

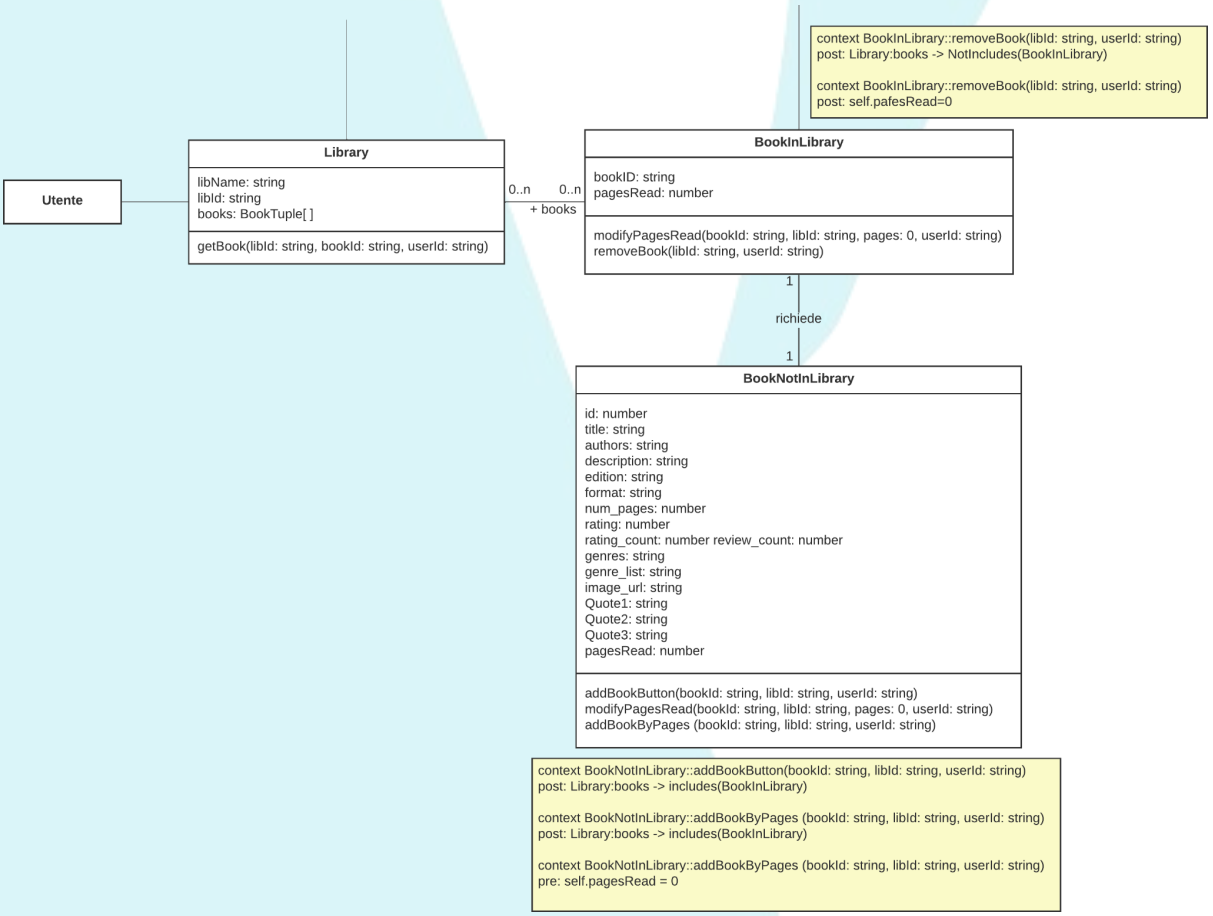
Nella classe Utente analizziamo il metodo deleteLibrary. Nella nostra webapp non dovrà essere possibile eliminare le librerie di base ovvero: to read, currently reading e finished. In OCL specifichiamo che l'id della libreria da eliminare deve essere diverso da 1 in quanto esso sarà l'id utilizzato per riconoscere le librerie fondamentali



2.2 Librerie

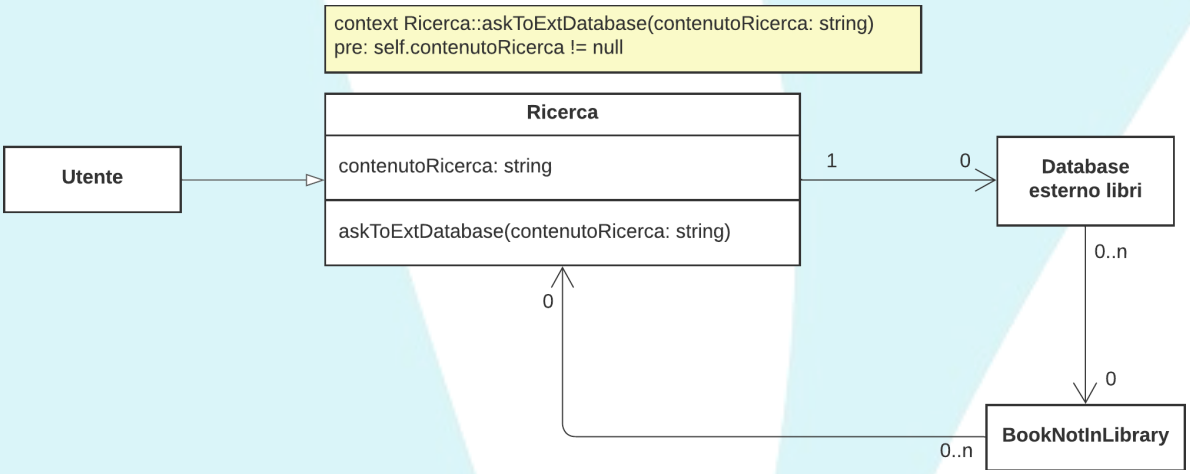
Nella categoria delle librerie analizziamo le due classi che rappresentano i libri.

Nella classe BookNotInLibrary tramite OCL viene specificata la logica dei metodi di aggiunta di un libro ad una libreria, in particolare, usare entrambi i metodi significa che il libro aggiunto diventerà un BookInLibrary e sarà inserito nella libreria selezionata. Nel metodo AddBookByPages viene indicato che il valore pagesRead deve essere originariamente =0 in quanto sarà all'incremento di tale numero che il metodo verrà utilizzato.



2.3 Ricerca

La webapp, come indicato dall'OCL seguente, permetterà di eseguire ricerche nel database esterno solo se il contenuto della barra di ricerca non è nullo.



3. Diagramma delle classi con OCL