

Diagramma delle classi

G06

Indice

Scopo del documento

Tipi introdotti

Classi derivate da componenti

OCL

Diagramma delle classi completo con codice OCL

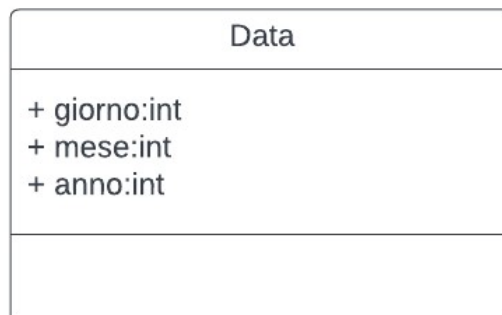
Scopo del Documento

Il presente documento ha lo scopo di descrivere l'architettura dell'applicazione EasyLib. Per riuscire a fare ciò, è stato usato un diagramma delle classi utilizzando UML (Unified Modeling Language) e del codice in OCL (Object Constraints Language) per riuscire a chiarire ogni modalità d'uso delle classi all'interno del diagramma. Molte classi sono derivate dai documenti precedenti, in particolare la maggior parte delle classi viene ricavata dal diagramma dei componenti e dal diagramma di contesto; qui ogni classe o gruppo di classe viene analizzato singolarmente o nel suo ambito e poi alla fine viene visualizzato il diagramma completo con anche il codice OCL.

Tipi introdotti

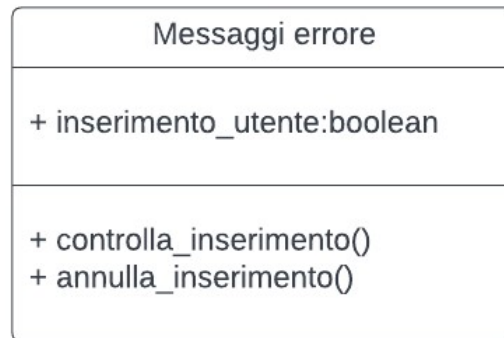
In questa sezione vengono descritti i tipi introdotti in questo documento che hanno lo scopo di semplificare e aiutare le interazioni tra le classi ricavate dai documenti precedenti.

1-Data: Viene introdotto il tipo Data, contenente solo gli attributi giorno, mese anno, di tipo intero. Questa classe è utile all'interno di tutto il diagramma quando viene richiesta una datazione (i.e. data di reso)

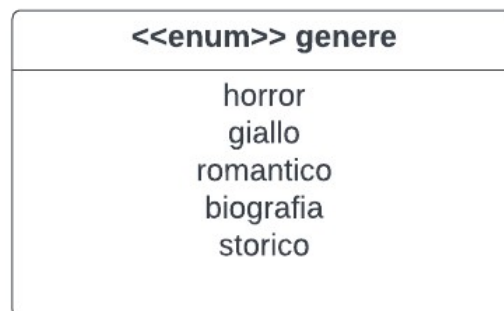


2- Messaggi di errore: Viene creato anche una classe messaggi di errore, che contiene un attributo inserimento utente con tipo boolean, dato che serve un inserimento perche ci sia un messaggio di errore. Questo tipo controlla inserimento dell'utente nei campi in cui gli è

richiesto di scrivere, se l'inserimento non rispetta i parametri del campo di testo, la classe annulla l'inserimento per permetterne quello corretto.

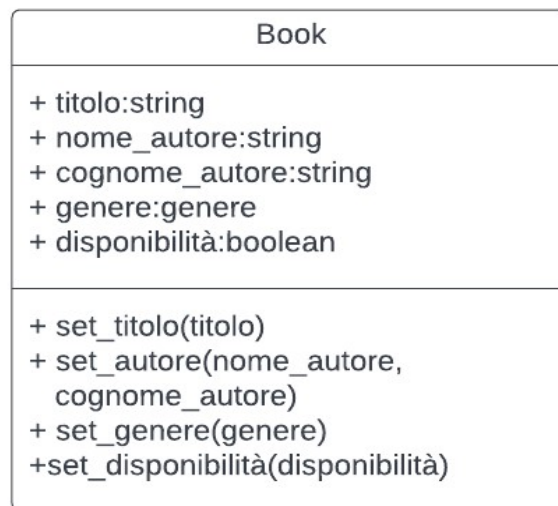


3- Genere: Viene creata una classe enum per descrivere i generi dei libri presenti in archivio, la classe viene usata sia per la parte di ricerca/ filtri che per l'archiviazione di nuovi libri all'interno del Database



4- Book: Per gestire al meglio le funzioni della biblioteca è stato necessario creare un tipo book con le caratteristiche dei libri presenti all'interno di EasyLib.

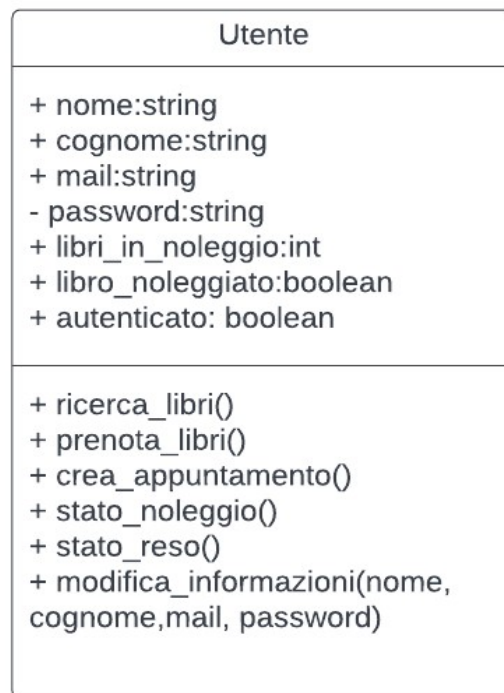
Gli attributi di questo tipo sono appunto: il suo titolo, genere (elencati nell'apposito enum) e le informazioni dell'autore. Questo tipo molto importante, viene usato in quasi tutte le classi essendo un sistema bibliotecario.



Classi derivate da componenti

In questa parte del documento vengono descritti, singolarmente o in gruppo, le varie classi ricavate dal diagramma dei componenti, dal diagramma di contesto o dai requisiti funzionali e non, dei due documenti precedenti

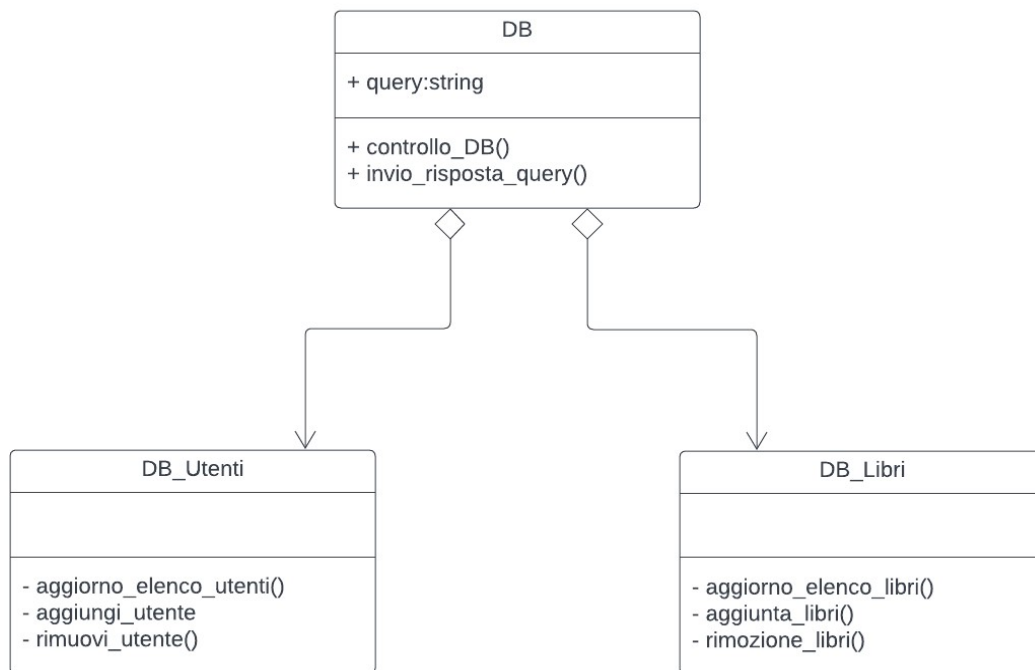
Utente: La parte più connessa di questo diagramma delle classi, contiene tutti gli attributi che ogni utente autenticato all'interno dell'applicazione, le sue funzioni sono tutte quelle descritte nei documenti precedenti e verranno prese in esame una a una in seguito in questo documento. Si vuole specificare che come utente è inteso anche l'utente admin ma che in questo diagramma è stato deciso di implementarlo come sottoclasse di un utente normale per semplificare la leggibilità del documento e farlo risultare più coeso



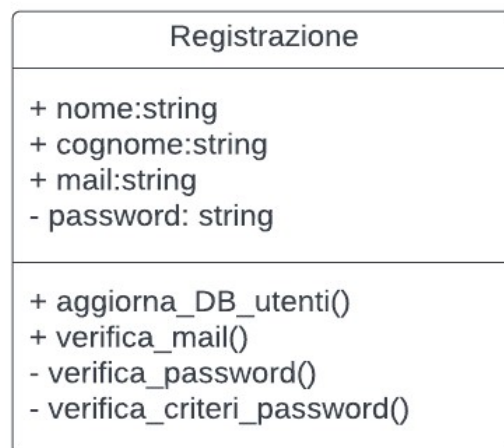
Database: La classe di accesso al database è stata divisa in 2 sottoclassi tramite un'aggregazione, per riuscire a dividere meglio le funzionalità e facilitare i collegamenti, avendo i nostri due database, libri e utenti, legami con classi diverse. La superclasse DB possiede come attributo principale la sua interrogazione tramite query. Successivamente con le funzioni di controllo nel database e invio risposta si vanno a differenziare le caratteristiche nei due rami.

Database Utenti: è utilizzato ogni volta che avvengono delle variazioni tra gli utenti registrati, viene usato principalmente nelle sue funzioni di modifica dall'utente amministratore.

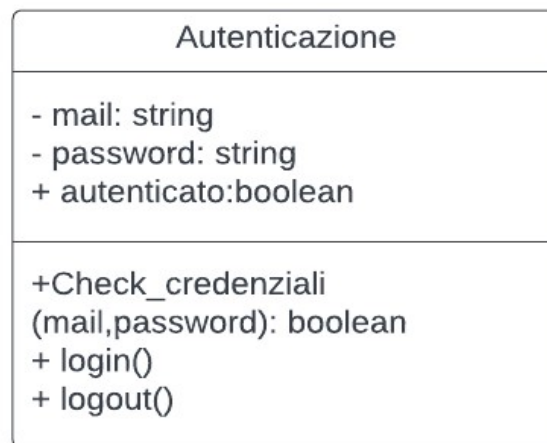
Database Libri: la classe è utilizzata ogni volta che si ha bisogno di informazioni riguardanti i libri in archivio, quindi anche per una ricerca normale dell'utente, inoltre la parte di aggiunta e rimozione libri viene anche qui gestita dall'utente amministratore.



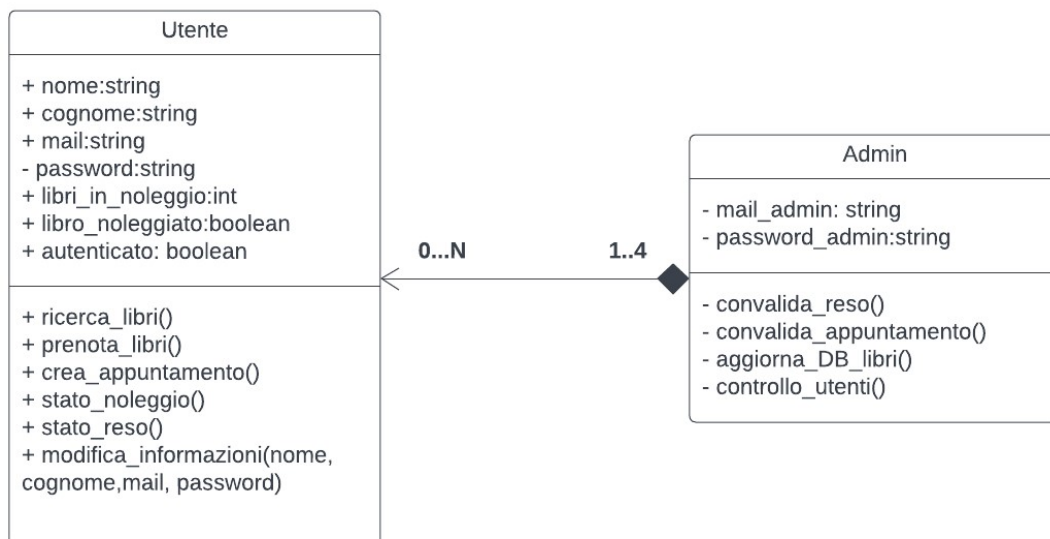
Registrazione: La classe registrazione gestisce la creazione di un nuovo profilo utente all'interno dell'archivio di EasyLib. gli attributi più importanti (mail, password) vengono verificati secondo i loro parametri, da le due funzioni verifica mail/password; in più la password deve rispettare i criteri di password forte ed è presente una funzione adibita a quello scopo. Si noti che la classe è direttamente collegata alla sottoclasse del database riguardante l'utente perche è lì che verranno inseriti i dati registrati dai nuovi utenti.



Autenticazione: La classe descrive le funzioni di autenticazione, sfrutta come attributi le credenziali create in fase di registrazione, gestisce le operazioni di login e logout se l'utente non effettuerà l'operazione di logout le sue credenziali non verranno richieste finché accederà all'applicazione frequentemente. La classe ha una funzione booleana che restituisce True se le credenziali inserite dall'utente sono corrette.

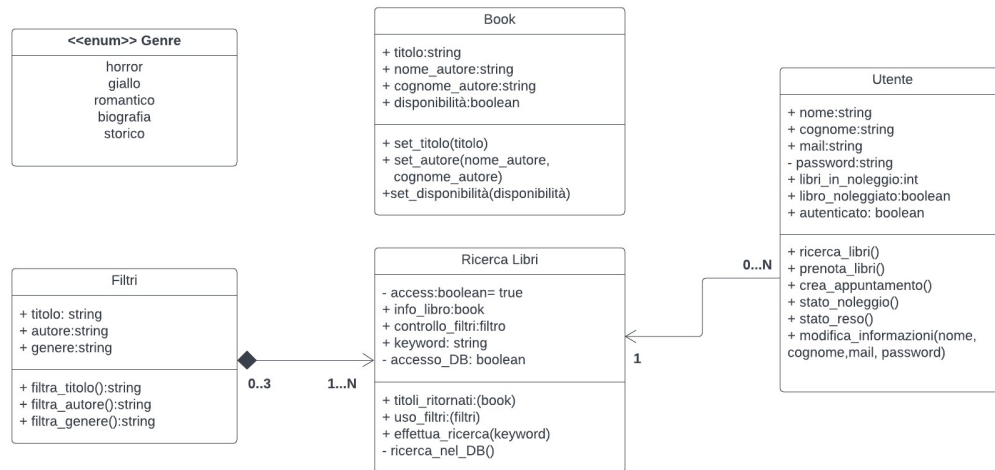


Utente-Admin: E' stato deciso di rappresentare la classe admin come una sottoclasse di utente così da distinguere bene le funzionalità dell'admin e fare in modo che le due condividano le stesse funzioni e attributi della classe utente; va specificato che l'admin viene considerato un utente normale quando accede con le sue credenziali utente, mentre assume ruolo di amministratore e tutte le sue capacità nel momento in cui accede con le credenziali speciali admin in una schermata apposita. Il numero massimo di admin previsti per la gestione dell'applicazione è di 4 così da non avere troppe persone in grado di modificare l'applicazione. La classe admin contiene tutte le funzioni amministrative e di gestione, quali l'accesso al database libri e il controllo utenti, come rappresentato nel diagramma dei componenti.

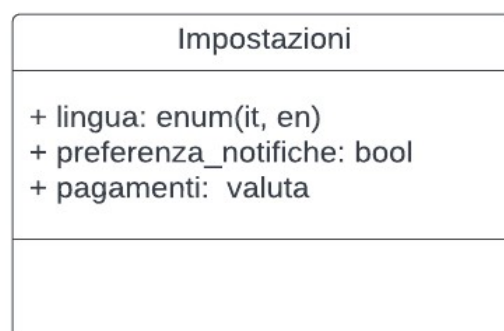


Utente-ricerca, filtri: funzionalità più importante del nostro sistema bibliotecario è la ricerca libri, abbinata alla classe utente (ogni tipologia di utente può effettuare una ricerca libri). In questa parte viene descritta insieme alla sua classe filtri apposta creata per soddisfare le funzionalità descritte nei documenti precedenti, e che permette all'utente di cercare uno o più libri secondo al massimo tre criteri (autore, titolo, genere) descritti come attributi della

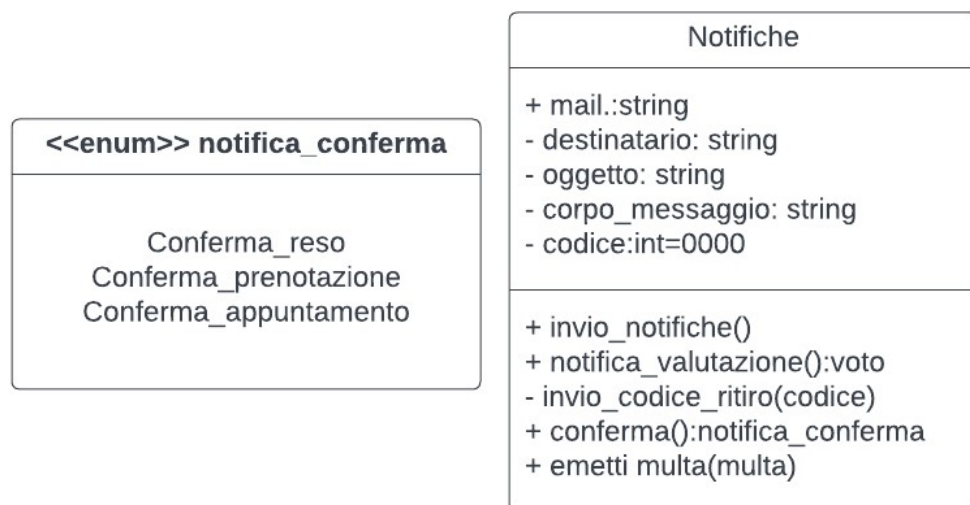
classe. Il tipo book aiuta il sistema a descrivere le informazioni riguardanti i volumi e la ricerca viene affinata grazie all'enum genere, specificato all'inizio, e all'utilizzo dei 3 filtri disponibili, rappresentati come una composizione di ricerca libri, senza la ricerca infatti non avrebbe senso avere una classe filtri.



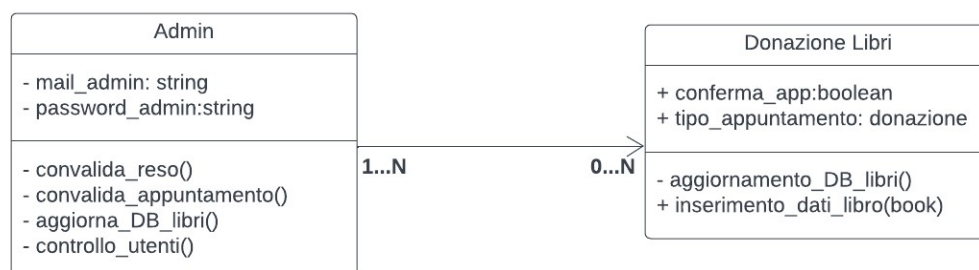
Impostazioni: Viene introdotta una classe impostazioni che servirà all'utente autenticato per cambiare alcune specifiche della sua web app. Tra cui la lingua se italiana o inglese, descritta da un semplice attributo enumerativo, le preferenze di notifiche (se attive o meno) e la valuta in uso all'interno dell'applicazione (che verrà trattata nelle parti successive).



Notifiche: La classe notifiche descrive le modalità di scambio di informazioni dall'applicazione verso l'utente. Per ogni funzionalità l'applicazione invia molteplici notifiche: di conferma, descritte dal tipo enum notifica conferma creato appositamente, notifiche che facilitino l'interazione con l'utente, come la mail che invia il codice di quattro cifre da mostrare al momento del ritiro in fisico, oppure la mail che notifica la possibilità di valutare il libro letto, tramite una classe descritta in seguito. Ogni notifica viene inviata per mail (all'indirizzo specificato dall'utente) e possono essere disattivate le notifiche nella sezione impostazioni. La classe nell'immagine sottostante è sgombra di collegamenti per aiutare la sua leggibilità ma va sottolineato che nel diagramma complessivo essa è fortemente connessa.

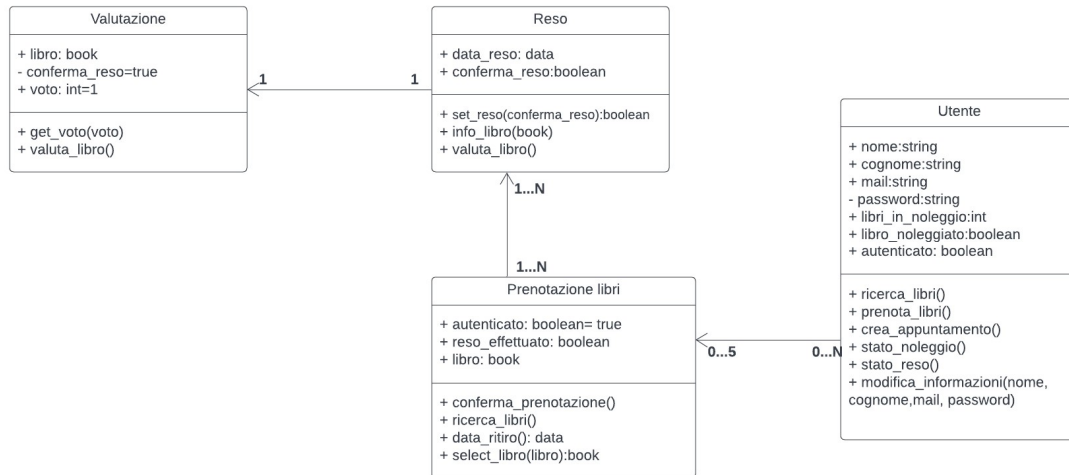


Donazione Libri: Una caratteristica della nostra applicazione è la sezione di donazione libri in cui un utente può portare alla nostra sede i propri volumi vecchi. La classe in questo è descritta accoppiata alla classe admin, il cui compito è regolare le donazioni tramite gli appuntamenti di questo tipo. Dopo aver confermato l'appuntamento, che verrà descritto meglio successivamente, l'admin potrà ritirare il libro in fisico e aggiungere le sue informazioni all'interno del database, tramite le informazioni che l'utente deve fornire riguardo il libro, di tipo book, in un modulo a parte.



Prenotazione, reso, valutazione: Funzionalità descritta in tutti i documenti fino ad ora è quella di prenotazione/reso. In questo caso le classi partono dall'utente che deve essere per forza autenticato, e procedono a cascata, infatti: dopo che il nostro user si è autenticato può prenotare ogni libro il cui stato è disponibile, il numero massimo di libri prenotabili è cinque. Ad avvenuta prenotazione il noleggio passerà nella classe di reso, creando la data di reso

per terminare il noleggio. A reso avvenuto l'applicazione sbloccherà la funzionalità di valutazione libro (cosicché le valutazioni provengano solo da utenti che hanno noleggiato e riportato il libro) in cui l'utente potrà valutare, da una a cinque stelle, la sua lettura. Il dato verrà conservato nella sua area personale. Il codice sarà più chiaro dopo l'inserimento del codice OCL (vedi sotto):



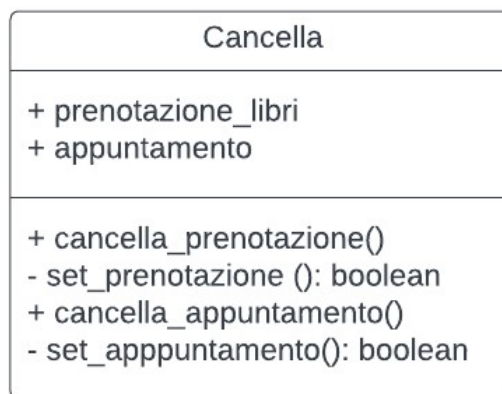
Multa, valuta: Nell'analisi dei componenti e del contesto viene descritta una parte di pagamento presente solo in caso di ritardo di riconsegna dei libri che invia multa come forma di tutela del sistema di biblioteca per riuscire a rientrare in possesso del volume non resituito. Queste classi descrivono le specifiche di questa funzionalità, tramite la classe multa, che scatta nel momento in cui scadono le 24 ore della data di reso del libro noleggiato, il sistema emette una contravvenzione inviata per mail (quella fornita dall'utente, vedi parte notifiche sopra). Un enum valuta permette all'utente la scelta di con quale valuta l'utente visualizza il quantitativo della multa.



Appuntamento: La classe appuntamento è dotata di un attributo che ne specifica la data, e un attributo che ne specifica la tipologia; le opzioni di appuntamento sono 2 e vengono descritte nell'enum tipo_app, che è stato creato per rendere più leggibile il diagramma complessivo, esso comprende l'appuntamento per la donazione e l'appuntamento che avviene al momento del reso. Sia specificato che ogni utente può prenotare anche le 2 tipologie di appuntamento separatamente da questo la cardinalità massima di 2 appuntamenti per ogni utente.



Cancella la classe cancella è utile all'utente per annullare tutte le tipologie di prenotazione che può fare nel caso in cui siano state fatte per errore, o non siano più necessarie. Nel diagramma infatti è posizionata nel mezzo tra gli appuntamenti e le prenotazioni reso.

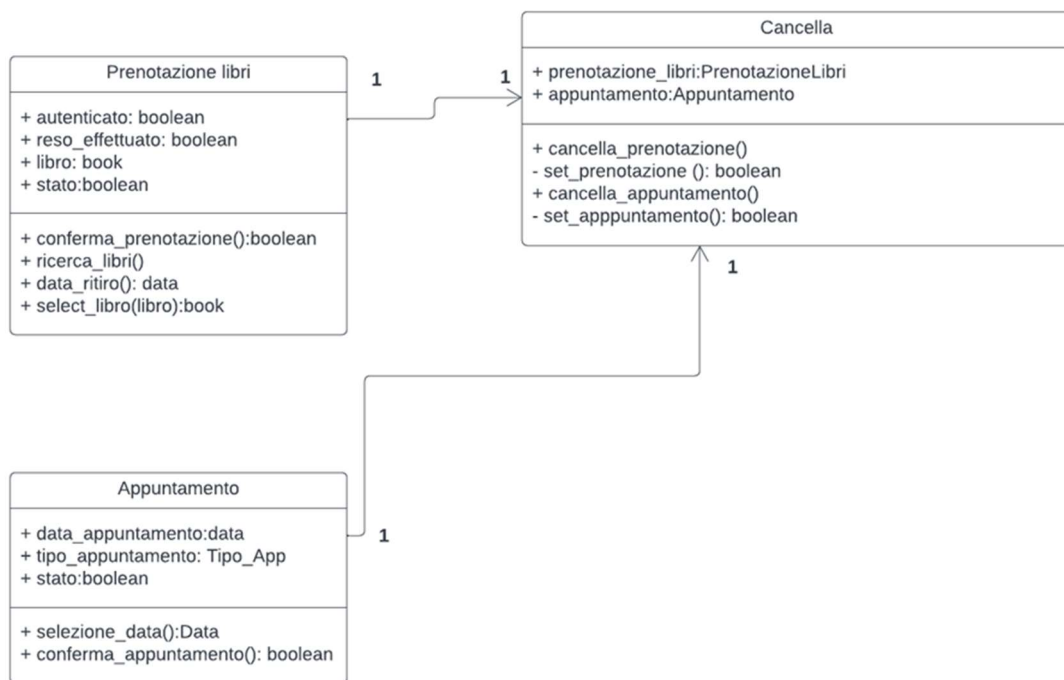


Codice in Object Constraint Language

In questo paragrafo del documento è descritta la logica prevista su alcune operazioni di classi. Questa logica viene descritta utilizzando l'Object Constraint Language (OCL) in modo tale da esprimere queste operazioni nel contesto UML.

Cancellazione prenotazione libri e appuntamenti

La cancellazione degli appuntamenti e delle prenotazioni vedono coinvolte tre classi differenti.



Le operazioni di `cancella_prenotazione` e `cancella_appuntamento` richiedono come prerequisiti che gli attributi stato della classe **Appuntamento** e della classe **PrenotazioneLibri** siano settati a `true`, in OCL vengono espressi come di seguito.

```

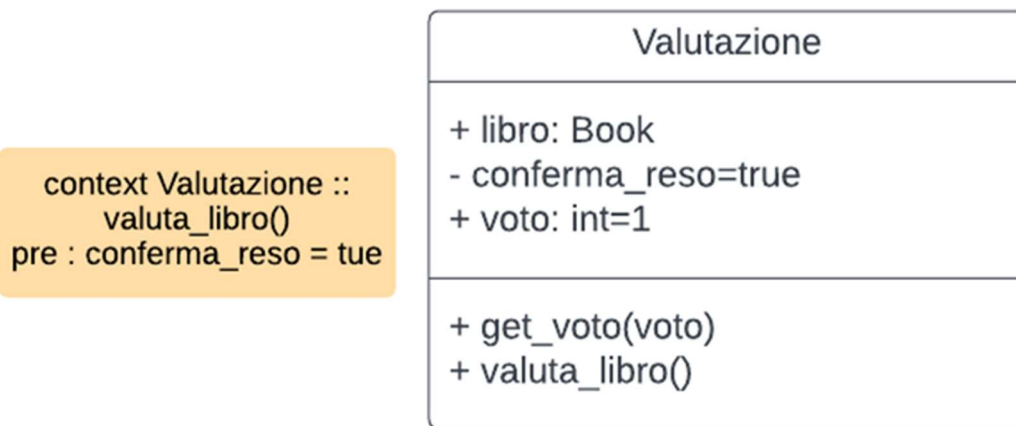
context Cancella ::
  cancella_appuntamento()
  pre : Appuntamento.stato = true
  
```

```

context Cancella ::
  cancella_prenotazione()
  pre : PrenotazioneLibri.stato = true
  
```

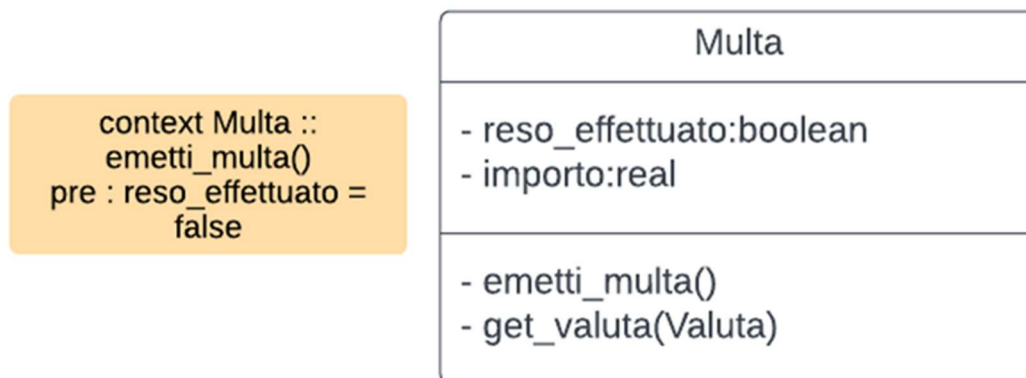
Valuta libri

Per effettuare il metodo `valuta_libro` si richiede che l'attributo `conferma_reso` sia uguale a `true`. L'operazione coinvolge la classe **Valutazione** e viene rappresentata in OCL come segue



Emetti multa

L'operazione `emetti_multa` richiede come prerequisito il fatto che l'attributo `reso_effettuato` sia **settato** `false`, coinvolge la classe **Multa**. La sua rappresentazione in OCL è la seguente



Messaggi di errore e annullamento input utente

Le operazioni di `annulla_inserimento` e `set_messaggio` coinvolgono la classe **MessaggiErrore**

Messaggi errore
+ inserimento_utente:boolean + messaggio:string
+ controlla_inserimento() + annulla_inserimento() + set_messaggio()

Entrambe richiedono come prerequisito che l'attributo `inserimento_utente` venga posto a `false`. Vengono descritte in OCL nella seguente forma

```

context MessaggiErrore ::
    set_messaggio()
pre : inserimento_utente = false

context MessaggiErrore ::
    annulla_inserimento()
pre : inserimento_utente = false

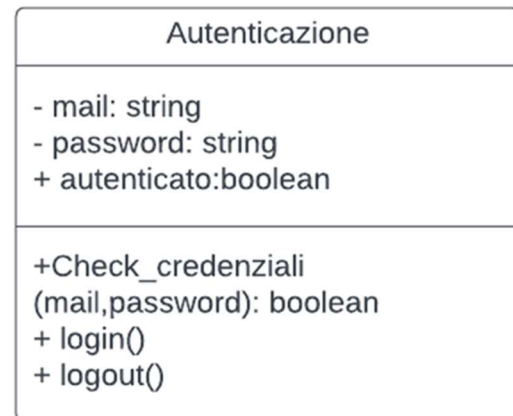
```

Login e logout

Le operazioni di login e logout coinvolgono la classe **Autenticazione** hanno come prerequisito che l'attributo `autenticato` sia settato a `true` in OCL viene descritto come segue

```
context Autenticazione :: login()
pre : autenticato = true

context Autenticazione :: logout()
pre : autenticato = true
```



Conferma reso e convalida appuntamento

I metodi `convalida_reso` e `convalida_appuntamento` della classe **Admin** richiedono che l'attributo `autenticato`, derivato dalla super classe **Utente**, sia settato a `true` ad indicare che l'utente amministratore ha fatto accesso alla sua area riservata. In OCL viene rappresentato come segue

```
context Admin ::
convalida_reso() pre :
autenticato = true

context Admin ::
convalida_appuntamento()
pre : autenticato = true
```



Operazioni della classe Utente

I metodi della classe **Utente** prenota_libri, crea_appuntamento, stato_reso, stato_noleggio e modifica_informazioni richiedono tutte come prerequisito che l'attributo autenticato sia impostato a true, inoltre l'attributo libri_in_noleggio ha un vincolo per cui deve essere minore di 5, questo si riflette sul metodo prenota_libri che avrà come prerequisito anche libri_in_noleggio < 5. Queste operazioni sono descritte in OCL come segue

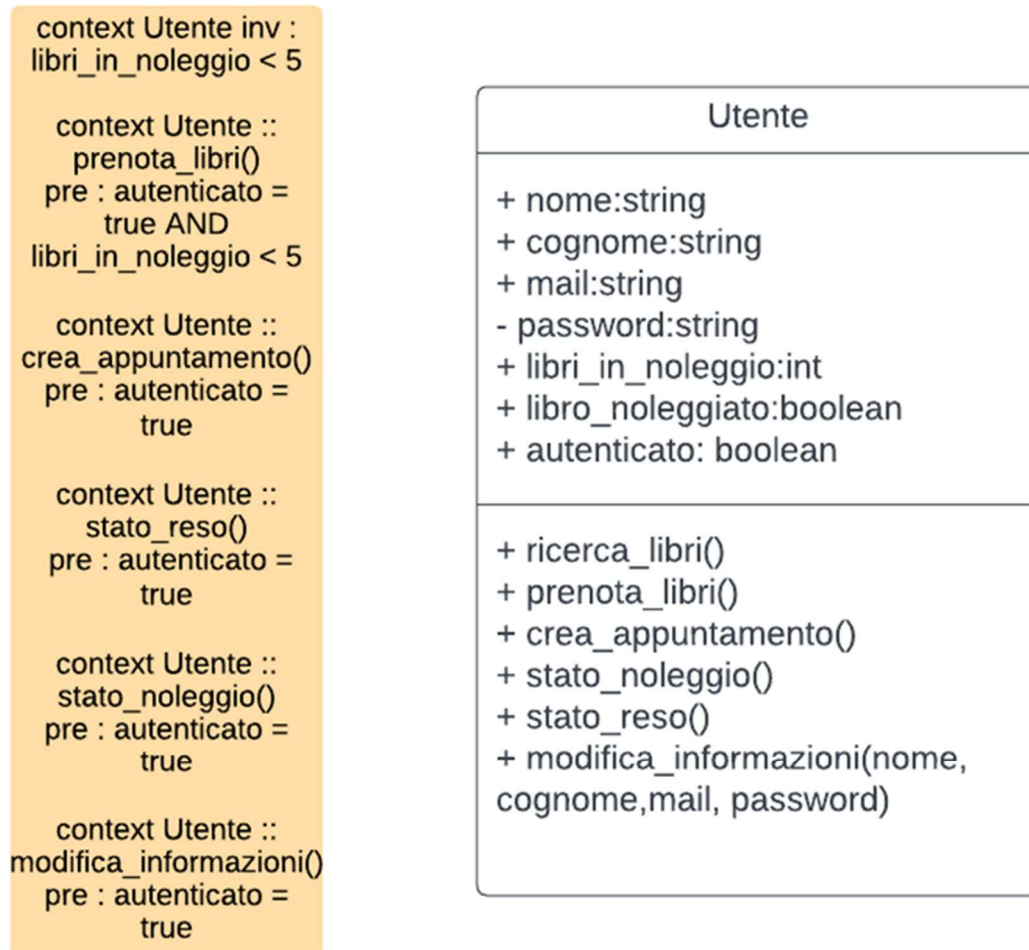


Diagramma delle classi con codice OCL

Infine, si riporta il diagramma delle classi completo con il codice OCL integrato

