UNIVERSITÀ DEGLI STUDI DI TRENTO

Dipartimento di ingegneria e scienza dell'informazione

Corso di Laurea in Informatica

PROGETTO INGEGNERIA DEL SOFTWARE

AULE STUDIO UNITN

GRUPPO T43

Sarcletti Davide - Rizzi Mattia - Cattoni Alex

Anno accademico 2022/2023

INDICE

Obiettivi del documento

Il documento D3 mostra lo schema generale e specifico dell'architettura del sistema ASU (Aule Studio Unitn) attraverso l'uso del diagramma delle classi UML combinato al linguaggio OCL.
Grazie all'implementazione delle classi si potrà definire il codice da implementare per realizzare il portale, mentre attraverso il linguaggio OCL definiremo la logica dell'intero sistema.
Una descrizione in linguaggio naturale accompagna i diversi diagrammi aiutando a comprendere meglio i concetti rappresentati attraverso i diversi schemi.

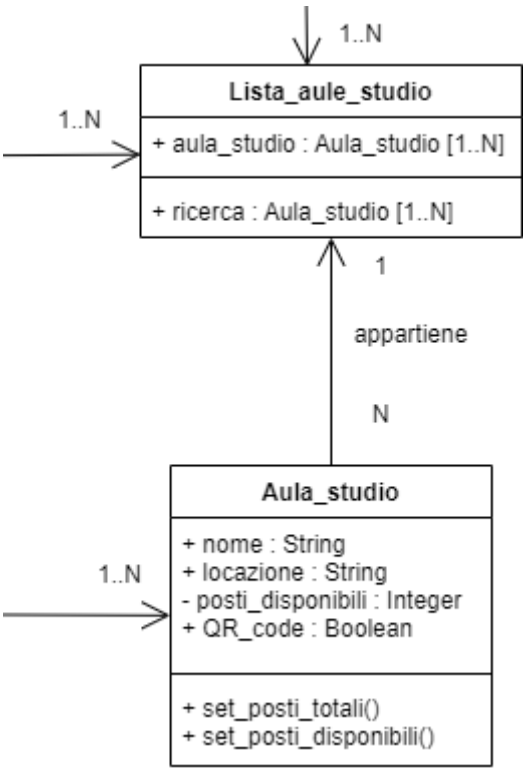
1. Diagramma delle classi

Attraverso il diagramma delle classi vengono rappresentate le diverse classi che porteranno all'implementazione del codice.

1. Lista Aule Studio

Il diagramma "Lista Aule Studio" illustra all'utente autenticato o non le varie aule studio presenti nella città, disponendo le varie informazioni come orari e posizione.
La lista delle aule studio viene estrapolata direttamente da MongoDB, essendo che tutte le istanze relative alle aule studio sono salvate al suo interno.

La classe Aula Studio gestisce le informazioni visibili unicamente agli utenti autenticati. Oltre a delle informazioni base saranno disposti anche i posti totali e quelli ancora disponibili in tale aula. Infine l'ultimo elemento importante della classe è il QR Code, indispensabile per registrare la presenza degli studenti nelle aule.
Le due funzioni `set_posti_totali()` e `set_posti_disponibili()` sono automatizzate e si occupano del "tracking" (tenere traccia) del numero di prenotazioni presso tale aula. Un utente amministratore potrà anche modificare il numero di posti totali e disponibili in caso di emergenza, necessità o casomai lo spazio a disposizione aumenti o diminuisca.

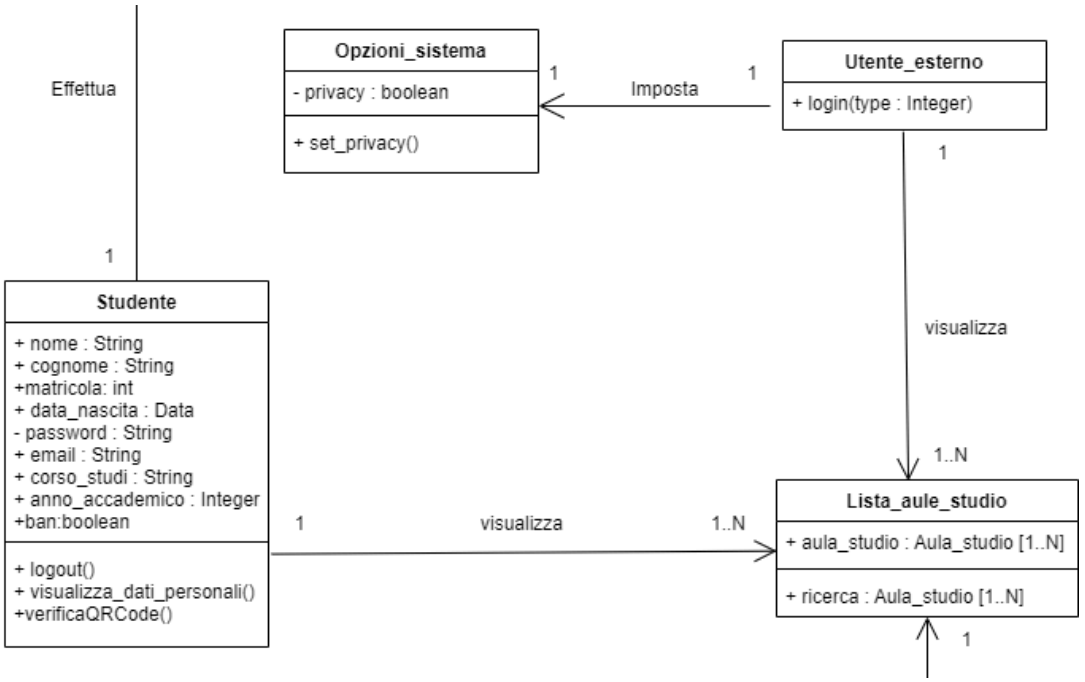


2. Studente - Utente esterno

Per gestire gli utenti autenticati e salvare le loro informazioni sfruttiamo la classe Studente. Tramite essa recuperiamo dal database Unitn le varie informazioni personali dello studente necessarie al completamento del profilo, nel nostro sito infatti non sarà possibile registrarsi, ma solo effettuare il login. Tali informazioni sono le seguenti: nome, cognome, data di nascita, matricola, email, corso di studio e anno accademico. Lo studente sarà in grado unicamente di visualizzare tramite visualizza_informazioni_personali() i propri dati e non sarà in grado in alcun modo di modificarli per ovvi motivi di sicurezza e di accessibilità. L'attributo booleano +ban gestisce la sospensione temporanea dello studente dall'effettuare nuove prenotazioni.
La funzione *logout()* permetterà allo studente di disconnettersi dal proprio account.
Quando uno studente effettua una prenotazione, dovrà confermare la sua presenza tramite un QR code presente all'interno dell'aula che ha prenotato, ovvero tramite il metodo verificaQRCode().

Quando un utente esterno (non loggato) accede al sito, le uniche azioni che potrà compiere saranno visualizzare le aule studio presenti (senza effettuare prenotazioni) ed effettuare il login() con le credenziali UniTN, in modo tale da poter effettuare e gestire le sue prenotazioni.

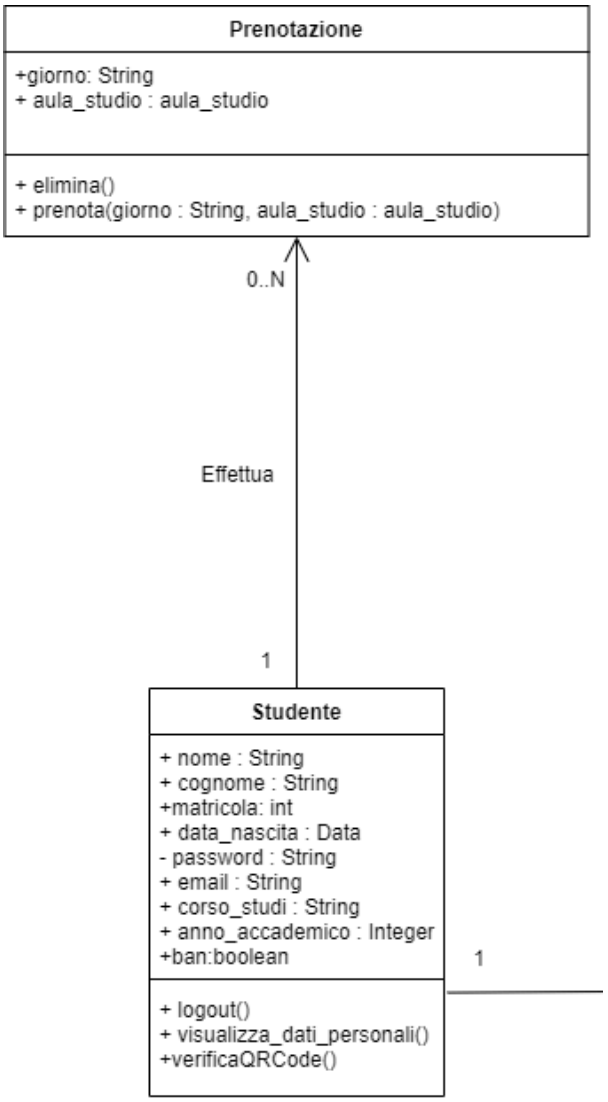
Infine Opzioni_Sistema rappresenta semplicemente le impostazioni scelte dagli utenti, i quali potranno confermare il consenso della privacy come richiesto nel GDPR (set_privacy()).



3. Prenotazione

La classe Prenotazione è una delle più importanti all'interno del diagramma. Essa è quella che permette agli studenti di prenotare (tramite prenota()) una certa aula studio in una data a scelta offerta dal sito web.. Una volta confermata la prenotazione l'utente avrà la possibilità tramite elimina()) di disdire la prenotazione, ci sarà infatti una sezione per gestire le prenotazioni e per poterle eliminare.

All’avvenuta prenotazione o disdetta il counter che tiene conto del numero di prenotazioni presso una determinata aula verrà aggiornato automaticamente, in modo tale da far sapere quanti posti disponibili rimangono agli studenti in ogni aula.

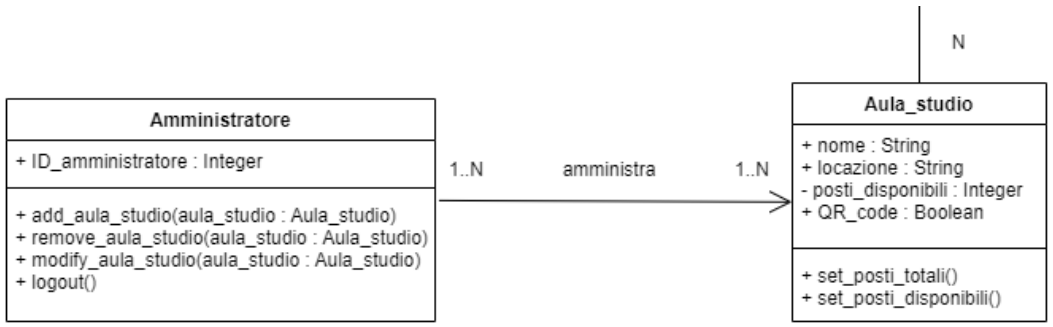


4. Amministratore

L'amministratore è l'utente che si occuperà di gestire le varie informazioni riguardanti le aule studio e le prenotazioni. Esso è identificato da un +ID_amministratore in modo da tener traccia di chi effettua le modifiche. L'amministratore e lo studente sono due utenze diverse, le credenziali per accedere come amministratore saranno presenti nel database MongoDB e non in quello Unitn. In questo modo se uno studente divenisse amministratore egli non avrebbe dei permessi differenti sul proprio account unitn ma delle credenziali separate per la gestione delle aule. Lo stesso vale per un eventuale gestore che non fa parte del sistema unitn ma che necessita un modo per accedere ed effettuare le modifiche.

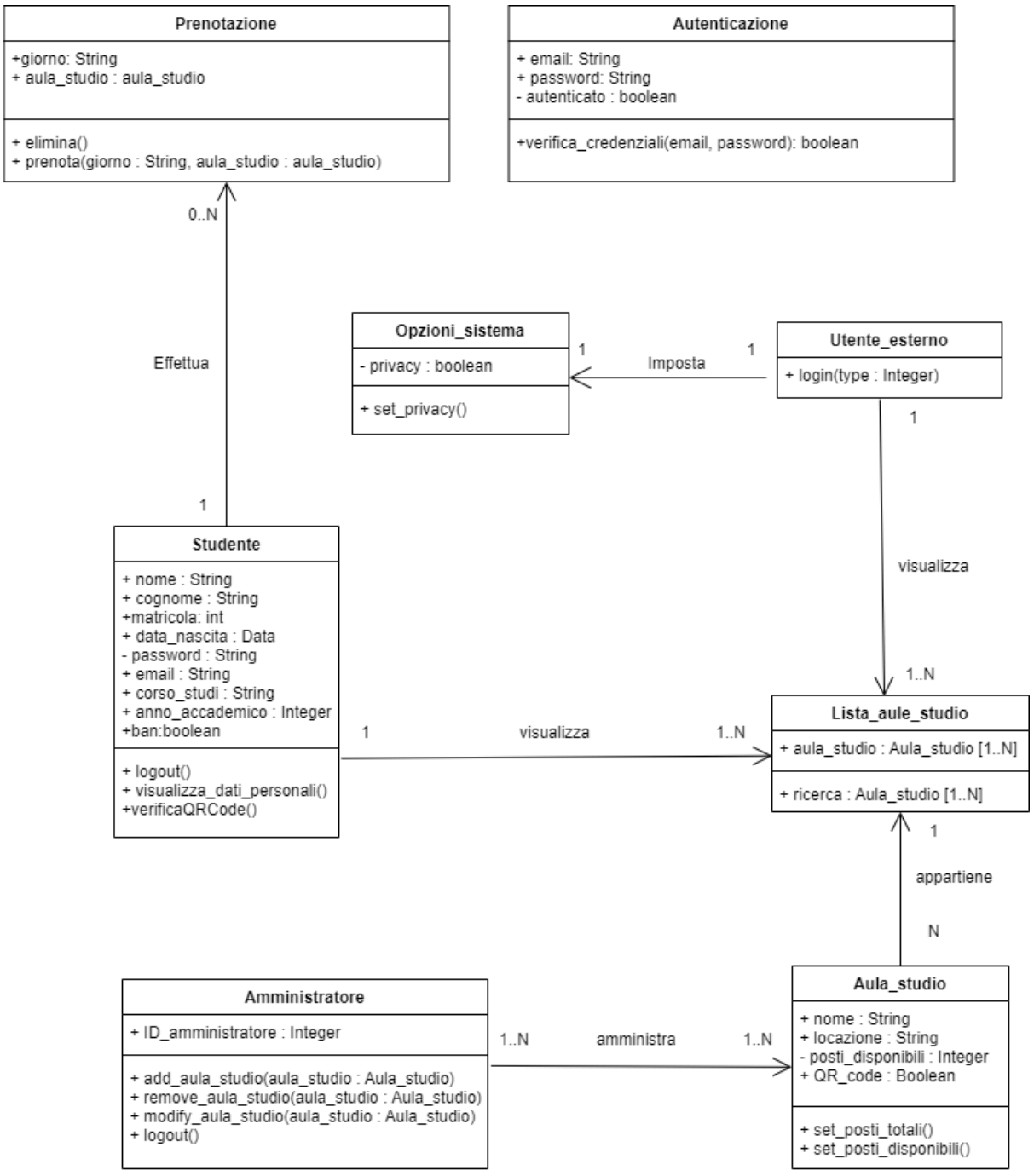
Di preciso l'amministratore tramite le seguenti funzioni definite effettuerà:

- add_aula_studio() con la quale potrà creare all'interno del database una nuova aula studio con le relative informazioni necessarie.
- remove_aula_studio() che permette di eliminare una aula studio esistente casomai non fosse più utilizzabile per qualsivoglia motivo.
- modify_aula_studio() permette di modificare i dati delle sale. Tramite esso ci si ricollega a due funzioni presenti in Aula_Studio accessibili unicamente a un utente amministratore: set_posti_totali e set_posti_disponibili. Tramite esse si è in grado di modificare i posti totali e disponibili in una locazione.



5. Diagramma delle classi complessivo

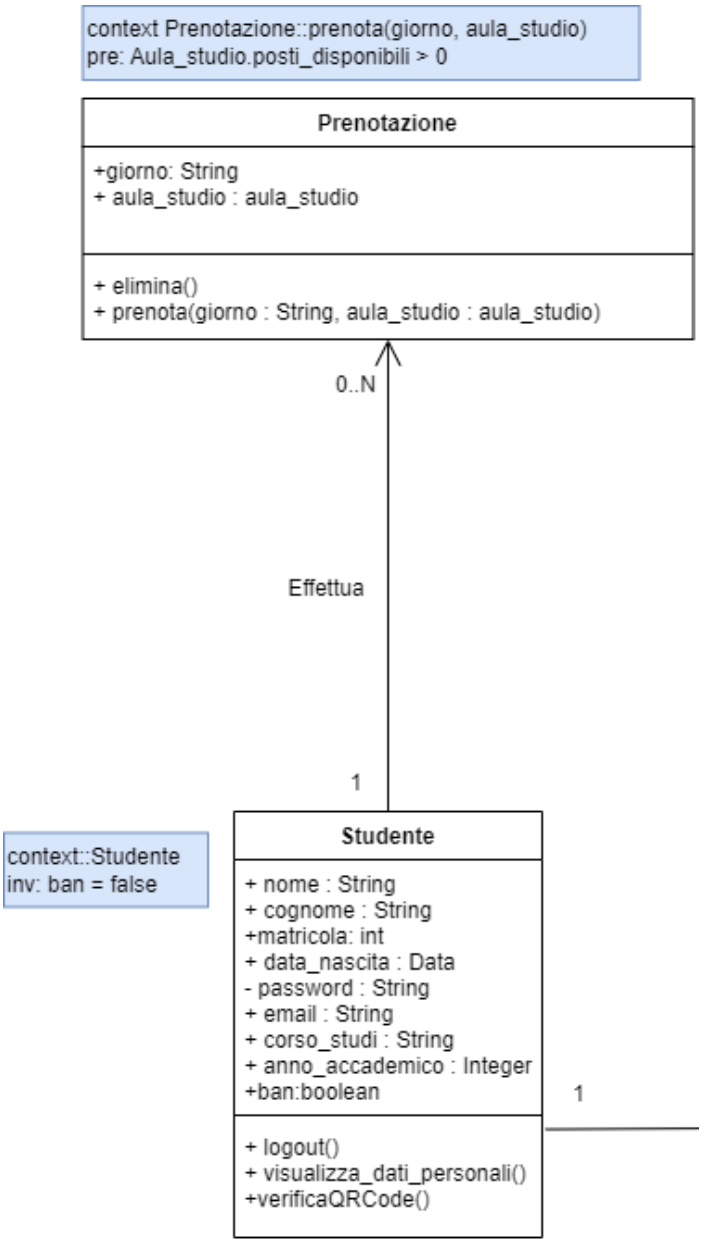
Il diagramma delle classi nella sua interezza unisce le definizioni delle classi elencate precedentemente, mostrando anche le classi satellite utilizzate come appoggio per distribuire metodi e attributi.



2. Codice in OCL

Nel capitolo seguente viene mostrata la logica delle operazioni che coinvolgono le diverse classi, tramite l'utilizzo del linguaggio OCL.

1. Prenotazione studente



In questa sezione andiamo a vedere i principali vincoli imposti allo studente quando va ad effettuare una prenotazione, oltre ai vincoli delle prenotazioni stesse.

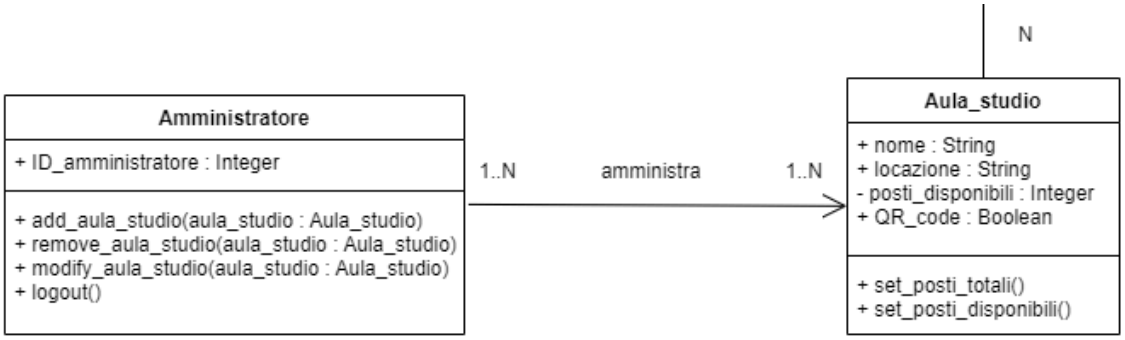
```
context Prenotazione::prenota(giorno, aula_studio)
pre: Aula_studio.posti_disponibili > 0
```

Il primo vincolo che andiamo ad analizzare è quello presente sul metodo prenota(giorno, aula studio), sul quale andiamo ad imporre al metodo un controllo su l'aula studio e la data prenotata.
Ci dovrà essere almeno un posto libero se si vorrà effettuare una prenotazione.

```
context::Studente
inv: ban = false
```

Infine con quest'ultimo vincolo verifichiamo dal database se lo studente ha la capacità di effettuare una prenotazione, ovvero che non sia attiva una sospensione temporanea. Questo controllo viene effettuato tramite il valore booleano ban, il quale se settato a false indica che non vi sono sospensioni attive. Casomai il valore fosse a true lo studente non sarebbe in grado di completare la prenotazione fino allo scadere della sospensione.

2. Amministratore



In questa sezione abbiamo i vincoli relativi al ruolo amministrativo, incentrati principalmente sui metodi di aggiunta e di modifica delle singole aule studio, che non dovranno andare a creare incongruenze nel database.

```
context Amministratore::add_aula_Studio(aula_studio)
pre: (aula_Studio -> includes(aula_Studio) = false)
```

Con il primo vincolo si effettua un controllo preventivo all'aggiunta di una nuova aula studio con lo scopo di ricercare se l'aula in fase di inserimento non sia già esistente. In questo modo non avremo problematiche legate a doppioni nel database.

```
context Amministratore::remove_aula_Studio(aula_studio)
pre: (aula_Studio -> includes(aula_Studio) = true)
```

Con questo vincolo, quando un amministratore va ad effettuare delle modifiche su una specifica aula studio tramite l'apposito form, ci sarà una verifica preventiva che andrà a controllare che l'aula studio che sta modificando sia esistente all'interno del database, in modo tale da non effettuare modifiche a vuoto.

3. Diagramma delle classi completo con OCL

Qui sotto è infine riportato il diagramma completo delle classi con l'aggiunta del codice in OCL.