



Università degli Studi di Ca' Foscari - Venezia  
Corso di Basi di Dati (Modulo 2)

---

Corso di Laurea Triennale in Tecnologie e scienze  
dell'Informazione

Relazione Progetto

## Progetto Basi di Dati - Traccia 1

10-08-2022

Candidati:

Marco Chinellato, 886217@stud.unive.it

Matricola 886217

Elisa Rizzo, 884784@stud.unive.it

Matricola 884784

Anno Accademico 2021-2022

# Indice

<b>1</b>	<b>Introduzione</b>	<b>3</b>
1.1	Descrizione dell'applicazione . . . . .	3
1.2	Ipotesi Considerate . . . . .	3
1.3	Struttura della relazione . . . . .	4
<b>2</b>	<b>Progettazione Concettuale</b>	<b>5</b>
2.1	Modelli . . . . .	6
2.2	Relazioni . . . . .	8
<b>3</b>	<b>Progettazione Logica</b>	<b>10</b>
3.1	Primary Keys . . . . .	11
3.2	Unique . . . . .	11
3.3	Foreigner Keys . . . . .	12
3.4	Checks . . . . .	12
3.5	Triggers . . . . .	13
3.6	Tabelle aggiuntive per la traduzione . . . . .	17
<b>4</b>	<b>Ruoli</b>	<b>18</b>
4.1	Ruoli DBSM . . . . .	18
4.2	Admin . . . . .	18
4.3	QrReader . . . . .	18
4.4	Professor . . . . .	19
4.5	Student . . . . .	20
4.6	Anonymus . . . . .	22
4.7	Ruoli nella WebApp . . . . .	24
<b>5</b>	<b>Funzionalità Principali</b>	<b>26</b>
5.1	Funzionalità a disposizione dei Professori . . . . .	26
5.2	Funzionalità a disposizione degli Studenti . . . . .	26
5.3	Anonymus . . . . .	26
5.4	Creazione dei Corsi . . . . .	27
5.5	QR Reader e QRCode . . . . .	28
5.6	Q&A . . . . .	30
5.7	Certificati . . . . .	33
5.8	Feedback . . . . .	33
5.9	Demografica . . . . .	36
5.10	Aggiunta di molteplici lezioni . . . . .	38
5.11	Registrazione Professori . . . . .	39
<b>6</b>	<b>Query e Funzioni Principali</b>	<b>41</b>
6.1	Query "get_lessons_bookable" . . . . .	41
6.2	Query "add_multiple_lessons" . . . . .	41
6.3	Query "update_lesson" . . . . .	42
6.4	Funzione "send_certificate_to_students" . . . . .	44
6.5	Query relative alla demografica . . . . .	45
6.5.1	Query "gender_subscribed" . . . . .	45
6.5.2	Query "age_subscribed" . . . . .	46
6.5.3	Query "city_subscribed" . . . . .	46
6.5.4	Query "type_school_subscribed" . . . . .	47

6.5.5	Query "hours_attended" . . . . .	48
<b>7</b>	<b>Navigazione Web App</b>	<b>49</b>
7.1	Registrazione e Login . . . . .	49
7.2	Visualizzazione lista dei corsi . . . . .	51
7.2.1	Visualizzazione i miei corsi – Studente Autenticato . . . .	52
7.2.2	Visualizzazione i miei corsi – Professore Autenticato . . .	53
7.3	Visualizzazione pagina del corso . . . . .	54
7.3.1	Studente Autenticato . . . . .	54
7.3.2	Professore Autenticato . . . . .	56
7.4	Visualizzazione pagina Lezioni . . . . .	57
7.4.1	Studente Autenticato . . . . .	57
7.4.2	Professore Autenticato . . . . .	58
7.5	Prenotazioni – Studente Autenticato . . . . .	59
7.5.1	Visualizzazione "Le mie prenotazioni" . . . . .	60
7.6	Feedback . . . . .	61
7.6.1	Studente Autenticato . . . . .	61
7.6.2	Professore Autenticato . . . . .	62
7.7	Pagina di demografica – Professore Autenticato . . . . .	63
7.8	Forum . . . . .	64
<b>8</b>	<b>Tecnologie Utilizzate</b>	<b>65</b>

# 1 Introduzione

Il nostro progetto sviluppa la prima traccia proposta, riguardante la Gestione delle Attività di Orientamento (PCTO) del dipartimento scientifico DAIS. Per facilitare l'installazione di tutte le librerie necessarie all'avvio della WebApp, è stato creato un file "setup.py" contenente tutti i requirements.

Il comando "pip install -e ." permetterà l'installazione automatica di tutte le librerie presenti all'interno del file.

## 1.1 Descrizione dell'applicazione

L'applicazione fornisce un'interfaccia di interazione con un database atto alla gestione delle attività di orientamento del DAIS, permettendo ai singoli utenti di registrarsi e loggarsi all'interno della stessa, e fornendo, in seguito al login, una serie di funzionalità.

Gli utenti, sia professori che studenti, hanno accesso ad un'area privata, dalla quale possono visualizzare i loro corsi, rispettivamente quelli da loro gestiti o frequentati, inoltre qualora l'utente autenticato sia uno studente saranno a sua disposizione anche le sue prenotazioni settimanali e una pagina riportante i certificati da lui conseguiti.

Nel caso in cui invece sia un professore oltre a poter modificare ed eliminare corsi e lezioni da lui aggiunte, potrà accedere, tramite un apposito bottone nella pagina di ogni suo corso, ad una pagina relativa ai feedback degli studenti e ad una che presenta uno studio demografico relativo agli iscritti a tale corso.

Inoltre è possibile, tramite la navigazione nella voce generica "Corsi" dell'interfaccia, prendere visione di tutti i corsi proposti e visualizzarne, tramite apposito bottone, le pagine associate, mentre la voce generica "Prenotazioni" permette di vedere tutte le lezioni settimanali per le quali sono aperte le prenotazioni.

Si noti che per lo sviluppo dell'applicazione è stato utilizzato il framework Flask mentre, per comunicare con il database, è stato utilizzato SqlAlchemy, che si interfaccia con **Postgresql**.

## 1.2 Ipotesi Considerate

Durante le fasi di progettazione dell'applicazione sono state prese in considerazione le seguenti asserzioni:

- ad ogni lezione viene associato un token utilizzato per confermare l'effettiva presenza degli studenti;
- gli studenti non hanno limiti rispetto al numero di corsi ai quali possono iscriversi;
- i corsi possono presentare o meno un limite massimo di iscritti;
- i corsi possono presentare o meno un numero di ore minime per ottenere un attestato di partecipazione;
- le aule in cui si svolgono le lezioni frontali o duali hanno una capienza massima;

- solo i professori e i collaboratori che hanno creato i corsi e le lezioni possono modificarli;
- i professori non sono abilitati a registrarsi manualmente, solo utenti speciale con il ruolo di admin sono abilitati a tale operazione;
- Il progetto è stato suddiviso in Blueprints per facilitare la suddivisione in molte route.  
Le blueprint utilizzate sono le seguenti:

1. **auth\_blueprint** : per gestire le pagine e le azioni legate all'autenticazione e alla registrazione;
2. **courses\_blueprint** : per gestire le pagine dei corsi e le relative azioni;
3. **lessons\_blueprint** : per gestire le pagine delle lezioni e le relative azioni;
4. **qna\_blueprint** : per gestire i forum;
5. **feedback\_blueprint** : per gestire le funzionalità relative ai feedback.

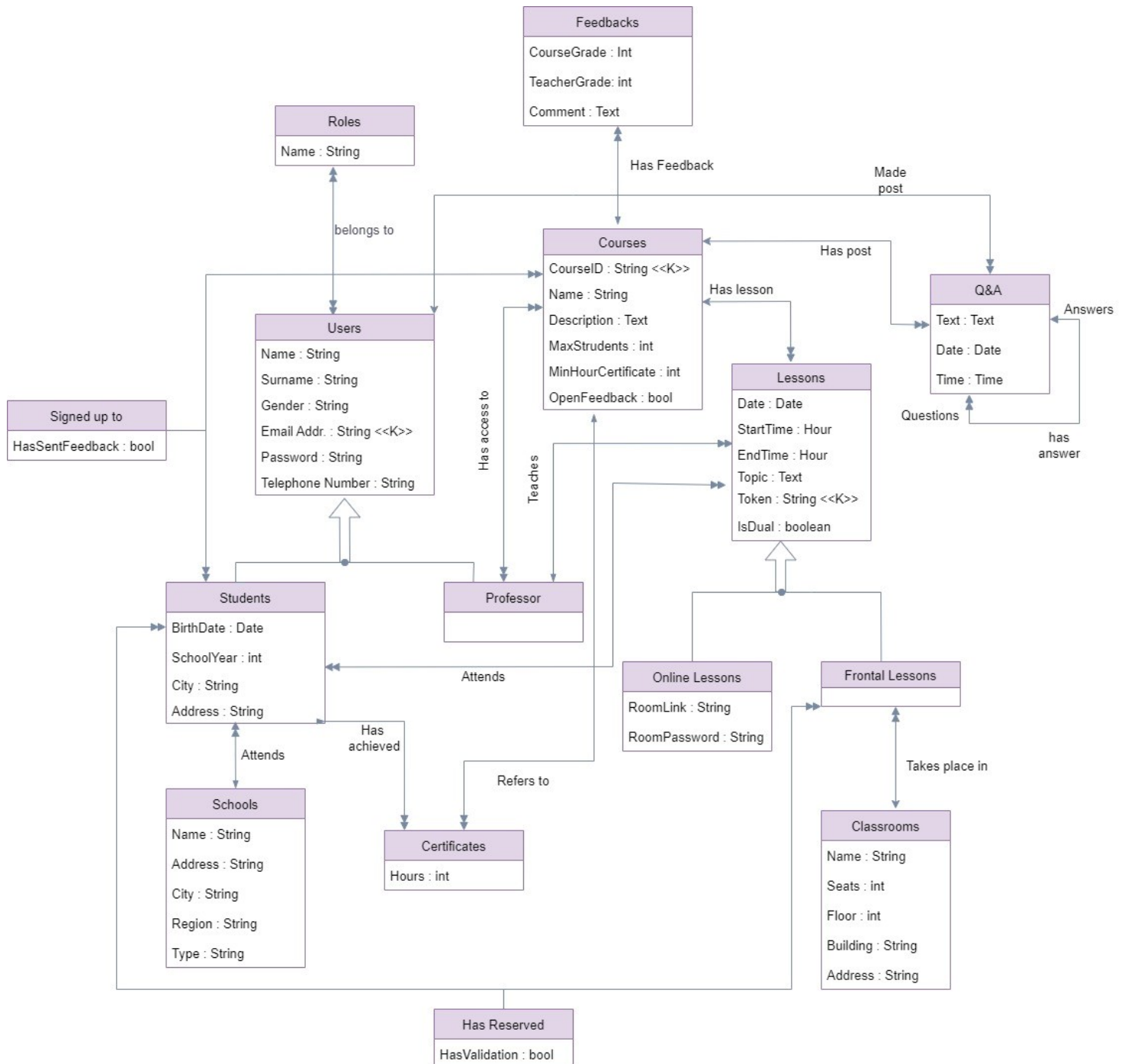
La decisione di non permettere ai professori di registrarsi autonomamente è stata dettata dalla volontà di garantire una maggiore sicurezza all'applicazione.

### 1.3 Struttura della relazione

La relazione verterà principalmente sull'esposizione e la spiegazione degli aspetti più interessanti relativi all'applicazione, partendo da una panoramica del processo di sviluppo della base di dati che la supporta, analizzato prima dal punto di vista concettuale e successivamente dal punto di vista logico, per poi trattare aspetti più specifici relativi alle scelte di sviluppo e all'effettiva progettazione .

## 2 Progettazione Concettuale

In questa sezione trattiamo la progettazione concettuale della base di dati



A partire dalla traccia del progetto :

Vi viene chiesto di curare il design e l'implementazione di una web application per la gestione delle attività di orientamento (PCTO) del DAIS. L'applicazione deve permettere la creazione di nuovi corsi, ciascuno composto da una o più lezioni tematiche, da svolgersi in presenza oppure online. Gli studenti devono potersi iscriversi ai corsi, mentre i docenti devono avere accesso ad un'interfaccia di analitica relativa ai corsi, che riporti almeno il numero di iscrizioni per ciascun corso e la demografica degli studenti partecipanti. I docenti devono avere inoltre la possibilità di inserire nuovi corsi.

è stato sviluppato uno schema concettuale a partire dai modelli, dalle relazioni e dai vincoli che verranno trattati nelle prossime sezioni, al fine di permettere l'interazione con l'applicativo e un'adeguata trasposizione delle richieste del progetto.

## 2.1 Modelli

### 1. Roles

- Name : String

### 2. Users

- Name : String
- Surname : String
- Gender : String
- Email Address : String
- Password : String
- Telephone Number : String

### 3. Professors

### 4. Students

- BirthDate : Date
- SchoolYear : int
- City : String  
la città verrà poi utilizzata nello studio della demografica dei corsi
- Address : String

### 5. Schools

- Name : String
- Address : String
- City : String
- Region : String
- Type : String  
indica la tipologia dell'istituto di riferimento, come ad esempio Liceo, Istituto tecnico o Istituto Professionale

## 6. Certificates

- Hours : int  
indica il numero di ore del corso frequentate dallo studente

## 7. Courses

- CourseID : String
- Name : String
- Description : Text
- MaxStudents : int  
indica il numero massimo di studenti che possono essere iscritti al corso
- MinHourCertificates : int  
indica le ore minime da frequentare affinché lo studente possa ottenere l'attestato di partecipazione al corso
- OpenFeedback : boolean  
viene utilizzato per permettere agli studenti iscritti a un determinato corso di inviarne un feedback

## 8. Lessons

- Date : Date
- StartTime : Hour
- EndTime : Hour
- Topic : Text  
Nel topic non viene inserito solo l'argomento principale della lezione, ma anche materiali o link utili che il professore mette a disposizione degli studenti che seguono il corso!
- Token : String  
viene fornito dal professore e utilizzato dagli studenti per confermare la loro presenza a lezione
- IsDual : boolean

## 9. Frontal Lessons

## 10. Online Lessons

- RoomLink : String
- RoomPassword : String

## 11. Classrooms

- Name : String
- Seats : int
- Floor : int
- Building : String
- Address : String



## 12. Feedbacks

- CourseGrade : int
- TeacherGrade : int
- Comment : Text

## 13. Q&A

- Text : String
- Date : Date
- Time : Time

## 2.2 Relazioni

In questa sezione si analizzeranno le relazioni fra le tabelle che verranno così trascritte:

**Table1** «- R -» **Table2** Relazione M:M

**Table1** <- R -» **Table2** Relazione 1:M

**Table1** <- R -> **Table2** Relazione 1:1

- **Students** «— È iscritto —» **Courses**

Relazione atta a specificare l'iscrizione di uno studente a un determinato corso.

L'attributo "HasSentFeedback" della relazione indica se lo studente ha inviato o meno il feedback del corso.

Tale implementazione è stata scelta per far sì che ogni studente possa inviare uno ed un solo feedback per ciascun dei corsi a cui è iscritto;

- **Professors** «— Ha accesso —» **Courses**

Uno o più professori hanno piena libertà di modifica dei corsi che gestiscono, il che comprende la modifica dei campi, e l'aggiunta, la rimozione e la modifica delle lezioni;

- **Students** «— Ha prenotato —» **FrontalLessons**

Relazione atta a permettere agli studenti di prenotare un posto in aula per una specifica lezione frontale o duale.

Dovesse questa risultare piena, non sarà più possibile prenotare.

L'attributo della relazione "HasValidation" serve ad indicare se uno studente ha confermato con successo la sua presenza in aula;

- **Students** «— Ha seguito —» **Lessons**

A differenza della relazione descritta in precedenza, questa indica e certifica l'effettiva presenza di uno studente a una lezione, sia essa frontale od online.

La registrazione di tale presenza dovrà essere effettuata tramite l'utilizzo di un QRCode;

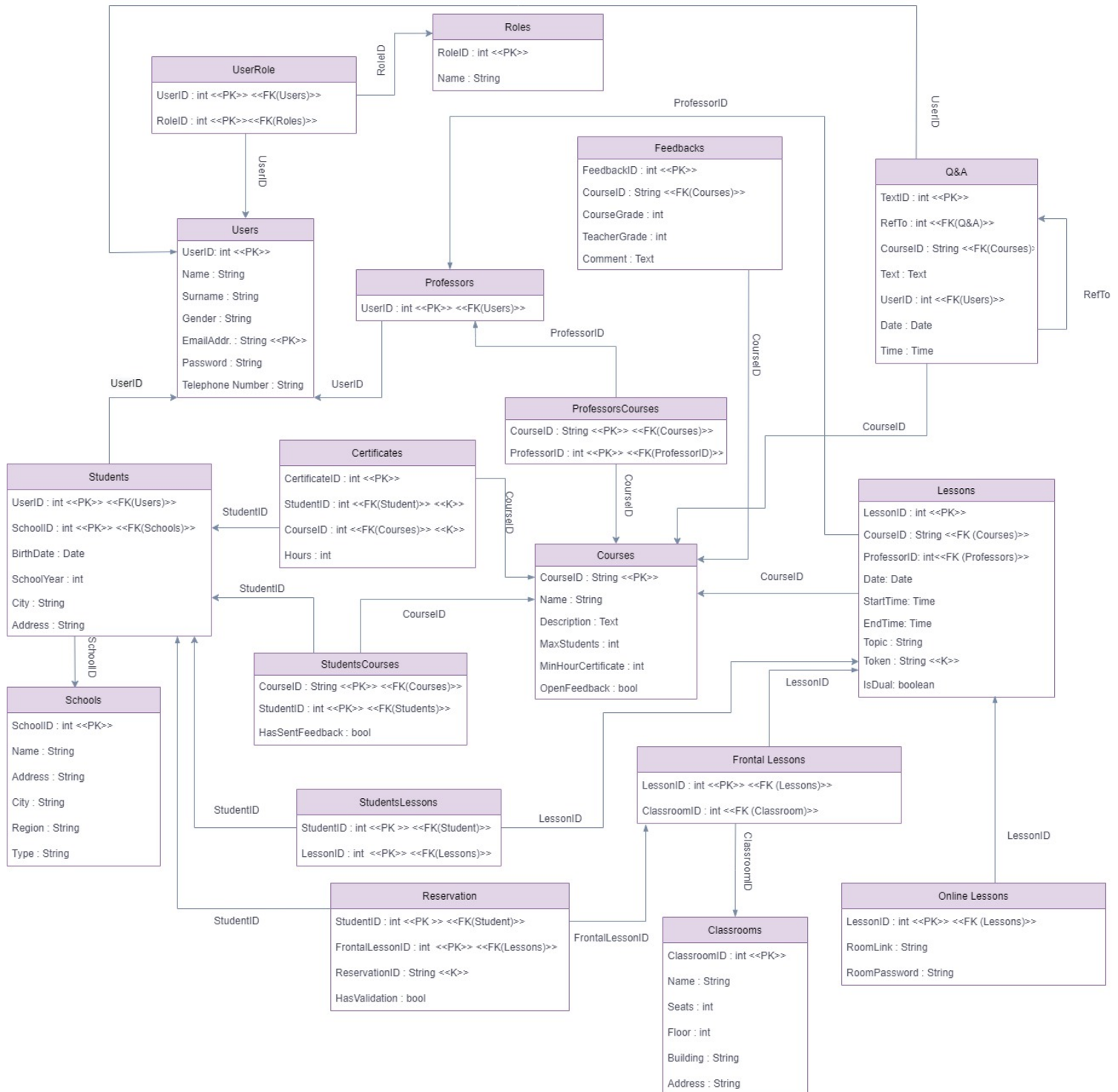
- **Professors** <— Insegna —» **Lessons**

I professori responsabili della creazione delle lezioni vengono indicati come insegnati delle stesse;

- **Students** «— Frequenta —> **Schools**  
Al momento della registrazione di uno studente ne viene richiesta la scuola superiore di provenienza;
- **FrontalLessons** «— Si svolge in —> **Classrooms**  
A ogni lezione frontale viene associata un'aula, nella quale essa si svolgerà;
- **Students** <— Ha Ottenuto —» **Certificates**  
A ogni studente vengono attribuiti degli attestati di partecipazione relativi ai corsi seguiti, se e solo se viene rispettata la frequenza obbligatoria minima;
- **Users** «— Appartiene a —» **Roles**  
A ogni utente registrato nella WebApp vengono assegnati uno o più ruoli, nel caso specifico i ruoli principali sono: Studente, Professore, Admin e QrReader, che verranno analizzati in maniera più dettagliata nei paragrafi successivi;
- **Feedback** «— Ha Feedback —> **Courses**  
Gli studenti hanno la possibilità di fornire un feedback anonimo dei corsi frequentati ai rispettivi docenti;
- **Courses** <— Ha post —» **QnA**  
Ad ogni corso è associata una sezione forum, che permette agli studenti e ai professori di comunicare fra di loro, al fine di risolvere eventuali dubbi;
- **Users** <— Ha creato il post —» **QnA**  
Ogni post creato viene associato allo studente che lo ha scritto;
- **Certificates** «— Fa riferimento a —> **Courses**  
A ogni corso portano essere associati gli attestati di partecipazione di tutti gli studenti frequentanti;
- **Courses** <— Ha lezione —» **Lessons**  
Ogni corso è composto da molteplici lezioni;
- **Q&A** <— Ha Risposta —> **Q&A**  
Un utente può rispondere ad un post già presente nel forum, creando così una vera e propria discussione.

### 3 Progettazione Logica

Analizziamo ora lo schema logico della base di dati.



A partire dallo schema concettuale illustrato nel precedente paragrafo e dalla sua traduzione, è stato successivamente strutturato il seguente schema logico.

### 3.1 Primary Keys

Elenchiamo ora le PRIMARY KEYS dei relativi modelli, le chiavi delle tabelle associative sono state escluse in quanto specificate nella sezione 3.6.

1. Roles - **RoleID** : int
2. Users - **UserID** : int
3. Professors - **UserID** : int
4. Students - **UserID** : int
5. Schools - **SchoolID** : int
6. Certificates - **CertificateID** : int
7. Courses - **CourseID** : String
8. Lessons - **LessonID** : int
9. Frontal Lessons - **LessonID** : int
10. Online Lessons - **LessonID** : int
11. Classrooms - **ClassroomID** : int
12. Feedbacks - **FeedbackID** : int
13. Q&A - **TextID** : int

### 3.2 Unique

I vincoli di UNIQUE definiti sulle tabelle sono:

1. **Users** - Email : String
2. **Lessons** - Token : String
3. **Certificates** - StudentID : int && CourseID : String
4. **Classroom** - Name : String && Building : String
5. **Reservations** - ReservationID : String
6. **Schools**  
SchoolName : String && Address : String && City : String

### 3.3 Foreigner Keys

Elenchiamo ora le FOREIGNER KEYS dei relativi modelli, quelle relative alle tabelle associative sono state escluse giacchè spiegate nella sezione 3.6.

1. **Students :**  
UserID FK(Users) : int  
SchoolID FK(Schools) : int
2. **Professors :**  
UserID FK(Users) : int
3. **Certificates :**  
StudentID FK(Students) : int            ONDELETE=CASCADE  
CourseID FK(Courses) : String
4. **Lessons :**  
ProfessorID FK(Users) : int            ONDELETE = CASCADE  
CourseID FK(Courses) : String
5. **FrontalLessons :**  
LessonID FK(Lessons) : int            ONDELETE=CASCADE  
ClassroomID FK(Classrooms) : int
6. **OnlineLessons :**  
LessonID FK(Lessons) : int            ONDELETE=CASCADE
7. **Feedbacks :**  
CourseID FK(Courses) : int
8. **Q&A :**  
CourseID FK(Courses) : String            ONDELETE = CASCADE  
RefTo FK(QnA) : int            ONDELETE = CASCADE  
UserID FK(Users) : int            ONDELETE = CASCADE

### 3.4 Checks

Presentiamo ora come sono stati sviluppati i relativi check constraint :

- un check sulla tabella Users per fare in modo che nel caso di modifiche o aggiunte, il campo Gender degli utenti sia "Male", "Female", "NonBinary" o "Other";

```
1 ALTER TABLE Users
2 ADD CONSTRAINT Check_Gender CHECK (Gender == 'Male' OR Gender ==
   'Female' OR Gender == 'NonBinary' OR Gender == 'Other')
```

- un check sulla tabella Feedbacks per garantire che il voto assegnato agli insegnanti e al corso sia maggiore di 0;

```
1 ALTER TABLE Feedbacks
2 ADD CONSTRAINT Check_grade_course CHECK (CourseGrade >= 0)
3 ADD CONSTRAINT Check_grade_prof CHECK (TeacherGrade >= 0)
```

- un check sulla tabella Lessons per garantire che l'orario di inizio delle lezioni preceda quello di conclusione;

```
1 ALTER TABLE Lessons
2 ADD CONSTRAINT Check_time CHECK (StartTime < EndTime)
```

- un check su Classrooms per garantire che il numero di posti sia maggiore di 0.

```
1 ALTER TABLE Classrooms
2 ADD CONSTRAINT Check_seats CHECK (Seats > 0)
3 ADD CONSTRAINT Check_floor CHECK (Floor >= 0)
```

### 3.5 Triggers

Per garantire un adeguato funzionamento dell'applicazione si è fatto ricorso inoltre serviti dei seguenti trigger:

1. il seguente trigger viene utilizzato per garantire che ogni corso abbia un numero massimo di iscritti, indicato nel campo "MaxStudents".  
Se quest'ultimo dovesse essere settato a zero, il corso non presenterà alcun limite di iscrizioni;

```
1 CREATE TRIGGER max_students_check
2 BEFORE INSERT ON StudentCourses
3 FOR EACH ROW EXECUTE max_student_check_func()
4
5 CREATE FUNCTION max_students_check_func() RETURNS trigger AS
6 $BODY$BEGIN
7     IF ( ( (SELECT COUNT(*)
8             FROM StudentsCourses AS sc JOIN Courses AS c
9             USING(CourseID)
10            WHERE NEW.CourseID = c.CourseID) >=
11            (SELECT c2.MaxStudents
12             FROM Courses AS c2
13             WHERE c2.CourseID = NEW.CourseID)) AND
14            (SELECT c3.MaxStudents
15             FROM Courses AS c3
16             WHERE CourseID = NEW.CourseID) > 0)
17     THEN
18         RETURN NULL;
19     END IF;
20     RETURN NEW;
21 END$BODY$
22 LANGUAGE plpgsql
```

2. il seguente trigger viene utilizzato per evitare che ci siano due lezioni dello stesso corso in orari incompatibili e nella stessa data;

```
1 CREATE TRIGGER check_self_course_overlapping_lesson
2 BEFORE INSERT OR UPDATE ON Lessons
3 FOR EACH ROW EXECUTE FUNCTION
4     check_self_course_overlapping_lesson_func()
5 CREATE FUNCTION check_self_overlapping_lesson_func() RETURNS
6     trigger AS
7 $BODY$BEGIN
8     IF(EXISTS(SELECT *
9         FROM Lessons AS l
10        WHERE l.LessonID <> NEW.LessonID
11            AND l.CourseID = NEW.CourseID
12            AND l.Date = NEW.Date
13            AND (
14                (NEW.StartTime BETWEEN l.StartTime AND
15                  l.EndTime)
16
17                OR
18
19                (NEW.EndTime BETWEEN l.StartTime AND
20                  l.EndTime)
21
22                OR
23
24                (NEW.StartTime <= l.StartTime
25                  AND
26                  NEW.EndTime >= l.EndTime)
27            ))
28 THEN
29     RAISE 'SameCourseOverlapping';
30 END IF;
31 RETURN NEW;
32 END$BODY$
33 LANGUAGE plpgsql
```

3. questo trigger viene utilizzato per impedire a due lezioni frontali di svolgersi nella medesima aula, nella stessa data e nello stesso range d'orario;

```
1 CREATE TRIGGER check_no_overlapping_lesson
2 BEFORE INSERT OR UPDATE ON FrontalLesson
3 FOR EACH ROW EXECUTE FUNCTION check_no_overlapping_lesson_func()
4
5 CREATE FUNCTION check_no_overlapping_lesson_func() RETURNS
  trigger AS
6 $BODY$BEGIN
7   IF ( EXISTS(SELECT *
8               FROM FrontalLessons AS fl1 JOIN Lessons AS l
9               USING (LessonID) JOIN Lessons AS lnow ON
10                  (NEW.LessonID = lnow.LessonID)
11                  WHERE fl1.ClassroomID = NEW.ClassroomID
12                      AND lnow.Date = l.Date
13                      AND (
14                        (l.StartTime BETWEEN lnow.StartTime AND
15                          lnow.EndTime)
16                        OR (l.EndTime BETWEEN lnow.StartTime AND
17                          lnow.EndTime)
18                        OR (l.StartTime <= lnow.StartTime AND
19                          l.EndTime >= lnow.EndTime)
20                      )
21                  ))
22   THEN
23     RAISE EXCEPTION 'LessonOverlapping';
24   END IF;
25   RETURN NEW;
26 END$BODY$
27 LANGUAGE plpgsql
```



4. questo trigger ha lo stesso scopo di quello precedente, con l'unica differenza che viene attivato ogniqualvolta venga effettuato un update su Lessons;

```
1 CREATE TRIGGER check_lesson_update_overlapping
2 BEFORE UPDATE ON Lessons
3 FOR EACH ROW EXECUTE FUNCTION
4     check_lesson_update_overlapping_func()
5 CREATE FUNCTION check_lesson_update_overlapping_func() RETURNS
6     trigger AS
7 $BODY$BEGIN
8     IF (EXISTS (SELECT *
9                 FROM Lessons AS l JOIN FrontalLessons AS fl
10                  USING(LessonID) JOIN FrontalLessons AS fl2 ON
11                     (fl2.LessonID = NEW.LessonID)
12                     WHERE l.Date = NEW.Date
13                        AND fl.ClassroomID = fl2.ClassroomID
14                        AND l.LessonID <> NEW.LessonID
15                        AND (
16                            (NEW.StartTime BETWEEN l.StartTime AND
17                             l.EndTime)
18                            OR (NEW.EndTime BETWEEN l.StartTime AND
19                             l.EndTime)
20                            OR (NEW.StartTime <= l.StartTime AND
21                             NEW.EndTime >= l.EndTime)
22                        )
23                ))
24 THEN
25     RAISE EXCEPTION 'LessonOverlapping';
26 END IF;
27 RETURN NEW;
28 END$BODY$
29 LANGUAGE plpgsql
```

5. questo trigger impedisce agli studenti di prenotare un posto per una lezione frontale o duale, qualora l'aula in cui si svolga risulti essere piena.

```
1 CREATE TRIGGER check_reservation_classroom_seats
2 BEFORE INSERT ON Reservations
3 FOR EACH ROW EXECUTE FUNCTION
4     check_reservation_classroom_seats_func()
5
6 CREATE FUNCTION check_reservation_classroom_seats_func() RETURNS
7     trigger AS
8 $BODY$BEGIN
9     IF ( (SELECT COUNT(*)
10         FROM "public"."Reservation" AS r
11         WHERE NEW."FrontalLessonID" = r."FrontalLessonID") >=
12         (SELECT c."Seats"
13         FROM "public"."Classrooms" AS c JOIN
14             "public"."FrontalLessons" AS fl2
15             USING("ClassroomID")
16         WHERE fl2."LessonID" = NEW."FrontalLessonID")
17     )
18     THEN
19         RAISE EXCEPTION 'SeatsNoMore';
20     END IF;
21     RETURN NEW;
22 END$BODY$
23 LANGUAGE plpgsql
```

### 3.6 Tabelle aggiuntive per la traduzione

Riportiamo le seguenti tabelle, integrate nello schema logico per tradurre le relazioni molti a molti presenti nello schema concettuale.

Tutte le tabelle riportate di seguito presentano due campi, allo stesso tempo sia Primary Keys sia Foreign Keys, che fanno riferimento ai modelli tra i quali, come indicato nello schema concettuale, era presente la relazione in questione.

- **UserRole** : la tabella "UserRole" si relaziona con "Utenti" e "Ruoli";
- **StudentsCourses** : la tabella oltre ad avere le due Foreign Keys che si riferiscono a "Students" e "Courses" e fungono anche da Primary Keys, presenta anche il campo "HasFeedback", precedentemente associato alla relazione molti a molti tra le due tabelle;
- **StudentsLessons** : la tabella è nata per ufficializzare l'effettiva presenza a lezione di uno studente;
- **Reservation** : la tabella oltre ad avere le due Foreign Keys, che si riferiscono a Students e Lessons e che fungono anche da Primary Keys, ha altri due campi: ReservationID che identifica ogni singola reservation e il campo HasValidation precedentemente associato alla relazione tra le due tabelle;
- **ProfessorsCourses** : in questa tabella ogni docente viene associato ai propri corsi di riferimento.

## 4 Ruoli

### 4.1 Ruoli DBSM

In questo capitolo descriveremo la suddivisione dei ruoli fornita dall'applicazione.

Ad ogni utente vengono assegnati permessi e ruoli differenti, i quali influenzano anche le funzionalità a loro disposizione.

In particolare, possiamo distinguere 5 ruoli:

1. Admin
2. QrReader
3. Professor
4. Student
5. Anonymus

I ruoli non vengono creati solo tramite DBMS, ma anche tramite Webapp. Grazie alla tabella UserRole, la WebApp sarà sempre a conoscenza dell'utente che la sta utilizzando e, in base a ciò, andrà a creare delle sessioni con DBMS e permessi differenti.

Riportiamo nei successivi paragrafi le funzionalità a disposizione dei singoli ruoli, soffermandoci maggiormente su quelle che riteniamo essere più interessanti.

### 4.2 Admin

L'admin ha accesso al database con tutti i permessi.

### 4.3 QrReader

Il ruolo del "QrReader" è così definito:

```
1 CREATE ROLE 'QrReader' WITH LOGIN ENCRYPTED PASSWORD '123456'
```

e presenta i seguenti permessi:

- leggere la tabella Courses;

```
1 GRANT SELECT ON Courses TO QrReader
```

- leggere la tabella Lessons;

```
1 GRANT SELECT ON Lessons TO QrReader
```

- leggere la tabella StudentCourses;

```
1 GRANT SELECT ON StudentsCourses TO QrReader
```

- leggere la tabella Classrooms;

```
1 GRANT SELECT ON Classrooms TO QrReader
```

- leggere la tabella StudentLessons ed effettuare eventuali inserimenti;

```
1 GRANT SELECT, INSERT ON StudentsLessons TO QrReader
```

- leggere la tabella FrontalLessons;

```
1 GRANT SELECT ON FrontalLessons TO QrReader
```

- leggere e modificare la tabella Reservation.

```
1 GRANT SELECT, UPDATE ON Reservation TO QrReader
```

## 4.4 Professor

Il ruolo del "Professor" è definito come segue :

```
1 CREATE ROLE 'Professor' WITH LOGIN ENCRYPTED PASSWORD '12345678'
```

I permessi a lui garantiti sono :

- leggere la tabella Schools;

```
1 GRANT SELECT ON Schools TO Professor
```

- leggere la tabella Professors;

```
1 GRANT SELECT ON Professors TO Professor
```

- leggere e modificare la tabella Users;

```
1 GRANT SELECT, UPDATE ON Users TO Professor
```

- leggere la tabella Students;

```
1 GRANT SELECT ON Students TO Professor
```

- leggere la tabella UserRole;

```
1 GRANT SELECT ON UserRole TO Professor
```

- leggere la tabella Feedback;

```
1 GRANT SELECT ON Feedback TO Professor
```

- leggere e modificare la tabella Courses ed effettuare eventuali cancellazioni e inserimenti;

```
1 GRANT SELECT, INSERT, DELETE, UPDATE ON Courses TO Professor
```

- leggere e modificare la tabella Lessons ed effettuare eventuali cancellazioni e inserimenti;

```
1 GRANT SELECT, INSERT, DELETE, UPDATE ON Lessons TO Professor
```

```
1 GRANT USAGE, SELECT ON SEQUENCE LessonID_seq TO Professor
```

- leggere e modificare la tabella FrontalLessons ed effettuare eventuali cancellazioni e inserimenti;

```
1 GRANT SELECT, INSERT, DELETE, UPDATE ON FrontalLessons TO Professor
```

- leggere e modificare la tabella OnlineLessons ed effettuare eventuali cancellazioni e inserimenti;

```
1 GRANT SELECT, INSERT, DELETE, UPDATE ON OnlineLessons TO Professor
```

- leggere la tabella StudentCourses;

```
1 GRANT SELECT ON StudentCourses TO Professor
```

- leggere e modificare la tabella Certificates;

```
1 GRANT SELECT, UPDATE ON Certificates TO Professor
```

```
1 GRANT USAGE, SELECT ON SEQUENCE CertificateID_seq TO Professor
```

- leggere la tabella ProfessorsCourses ed effettuare eventuali cancellazioni e inserimenti;

```
1 GRANT SELECT, DELETE, INSERT ON ProfessorsCourses TO Professor
```

- leggere la tabella Classrooms;

```
1 GRANT SELECT ON Classrooms TO Professor
```

- leggere e modificare la tabella Q&A ed effettuare eventuali cancellazioni e inserimenti;

```
1 GRANT SELECT, DELETE, INSERT ON QnA TO Professor
```

```
1 GRANT USAGE, SELECT ON SEQUENCE TextID_seq TO Professor
```

- leggere la tabella StudentsLessons

```
1 GRANT SELECT ON StudentsLessons TO Professor
```

## 4.5 Student

Il ruolo dello "Student" è creato come segue:

```
1 CREATE ROLE 'Student' WITH LOGIN ENCRYPTED PASSWORD '1234'
```

I permessi a sua disposizione sono:

- leggere la tabella Schools;

```
1 GRANT SELECT ON Schools TO Student
```

- leggere la tabella Professors;

```
1 GRANT SELECT ON Professors TO Student
```

- leggere e modificare la tabella Users;

```
1 GRANT SELECT, UPDATE ON Users TO Student
```

- leggere la tabella UserRole;

```
1 GRANT SELECT ON UserRole TO Student
```

- leggere la tabella Students;

```
1 GRANT SELECT ON Students TO Student
```

- attuare inserimenti nella tabella Feedback e leggere il campo "FeedbackID";

```
1 GRANT INSERT ON Feedback TO Student
```

```
1 GRANT SELECT (FeedbackID) (FeedbackID) ON Feedback TO Students
```

Il precedente permesso di lettura del campo "FeedbackID" è stato assegnato per permettere il funzionamento della sequence di seguito riportata.

```
1 GRANT USAGE, SELECT ON SEQUENCE FeedbackID_Seq TO Student;
```

- leggere la tabella Courses;

```
1 GRANT SELECT ON Courses TO Student
```

- leggere la tabella Lessons;

```
1 GRANT SELECT ON Lessons TO Student
```

- leggere la tabella FrontalLessons;

```
1 GRANT SELECT ON FrontalLessons TO Student
```

- leggere la tabella OnlineLessons;

```
1 GRANT SELECT ON OnlineLessons TO Student
```

- lettura la tabella StudentsCourses ed effettuare eventuali cancellazioni e inserimenti;

```
1 GRANT DELETE, INSERT, SELECT, UPDATE ON StudentsCourses TO Student
```

- leggere la tabella Certificates  
1 GRANT SELECT ON Certificates TO Student
- leggere la tabella ProfessorsCourses;  
1 GRANT SELECT ON ProfessorCourses TO Student
- leggere la tabella Classrooms;  
1 GRANT SELECT ON Classrooms TO Student
- leggere e modificare la tabella Q&A ed effettuare eventuali cancellazioni e inserimenti;  
1 GRANT DELETE, INSERT, SELECT, UPDATE ON Q\&A TO Student  
1 GRANT USAGE, SELECT ON SEQUENCE TextID\_Seq TO Student;
- leggere la tabella StudentsLessons ed effettuare eventuali inserimenti;  
1 GRANT SELECT, INSERT ON StudentsLessons TO Student
- leggere la tabella Reservation ed effettuare eventuali cancellazioni e inserimenti.  
1 GRANT SELECT, DELETE, INSERT ON Reservation TO Student

## 4.6 Anonymus

Il ruolo "Anonymus" è creato come segue:

```
1 CREATE ROLE 'Anonymus' WITH LOGIN ENCRYPTED PASSWORD '123'
```

I permessi a disposizione degli utenti Anonimi sono:

- leggere la tabella Schools;  
1 GRANT SELECT ON Schools TO Anonymus
- leggere la tabella Professors;  
1 GRANT SELECT ON Professors TO Anonymus
- leggere la tabella Users ed effettuare eventuali inserimenti;  
1 GRANT SELECT, INSERT ON Users TO Anonymus  
1 GRANT USAGE, SELECT ON SEQUENCE User.UserID\_Seq TO Anonymus;
- leggere la tabella Students ed effettuare eventuali inserimenti;

```
1 GRANT SELECT, INSERT ON Students TO Anonymus
```

- effettuare inserimenti nella tabella UserRole;

```
1 GRANT INSERT ON UserRole TO Anonymus
```

- leggere la tabella Courses;

```
1 GRANT SELECT ON Courses TO Anonymus
```

- leggere la tabella Lessons;

```
1 GRANT SELECT ON Lessons TO Anonymus
```

- leggere la tabella FrontalLessons;

```
1 GRANT SELECT ON FrontalLessons TO Anonymus
```

- leggere la tabella OnlineLessons;

```
1 GRANT SELECT ON OnlineLessons TO Anonymus
```

- leggere la tabella StudentsCourses;

```
1 GRANT SELECT ON StudentsCourses TO Anonymus
```

- leggere la tabella ProfessorsCourses;

```
1 GRANT SELECT ON ProfessorsCourses TO Anonymus
```

- leggere la tabella Classrooms.

```
1 GRANT SELECT ON Classrooms TO Anonymus
```



## 4.7 Ruoli nella WebApp

I ruoli nella WebApp vengono utilizzati attraverso una tabella Role, la quale contiene tutti i ruoli disponibili, ed una tabella associativa UserRole che associa uno o più ruoli a uno o più utenti.

Ogni volta che viene effettuata una transazione, di qualsiasi tipo, viene controllato il ruolo dell'utente corrente sfruttando l'ID del "current\_user" e, in base ad esso, viene assegnata l'engine corretta per tale transazione.

In base al ruolo corrente, non solo viene assegnata l'engine corretta, ma cambia anche ciò che viene visualizzato nella Webapp al fine di facilitarne la navigazione.

Ad esempio, nel caso in cui l'utente corrente sia un professore, comparirà nell'home page un tasto nella navbar chiamato "Aggiungi Corso", che non risulterà visibile alle altre categorie di utenti.

Il ruolo dell'utente viene controllato anche in fase di caricamento della pagina;

infatti se un utente con un ruolo diverso da "Professore" proverà ad accedere alla pagina per creare un corso, verrà lanciato un errore 401.

Il controllo verrà effettuato tramite il metodo "hasRole(...)" della classe User, il quale a seconda delle situazioni richiamerà le funzioni "exists\_role\_user()" e "get\_users\_role()".

```
24 class User(Base, UserMixin):
25     __tablename__ = "Users"
26
27     UserID = Column(Integer, primary_key=True)
28     Name = Column(String)
29     Surname = Column(String)
30     Gender = Column(String)
31     email = Column(String, unique=True)
32     Password = Column(String)
33     PhoneNumber = Column(String)
34
35     Roles = relationship('Role', secondary = "user_role")
36     Posts = relationship("QnA", backref = "User")
37
38     __table_args__ = (
39         CheckConstraint(or_(Gender == 'Female', Gender=='Male', Gender=='Non Binary', Gender == 'Other')),
40     )
41
42     def get_id(self):
43         return self.UserID
44
45     def hasRole(self, role):
46
47         from DaisPCTO.db import exists_role_user #per evitare il circular import va messo dentro la funzione
48         return exists_role_user(self.get_id(), role)
49
```

```

03 class Role(Base):
04     __tablename__ = "roles"
05
06     RoleID = Column(Integer, primary_key=True)
07     Name = Column(String)
08
09     __table_args__ = ()
10
11 class UserRole(Base):
12     __tablename__ = 'user_role'
13
14     RoleID= Column(Integer, ForeignKey("roles.RoleID", ondelete="CASCADE"), primary_key=True)
15     UserID = Column(Integer, ForeignKey("Users.UserID", ondelete="CASCADE"), primary_key=True)
16
17     __table_args__ = ()

```

```

engine = {
    "Admin" : create_engine("postgresql://postgres:123456@localhost/testone8", echo=False, pool_size=50, max_overflow=0),
    "Student" : create_engine("postgresql://Student:studente01@localhost/testone8",echo=False, pool_size=20, max_overflow=0),
    "Professor" : create_engine("postgresql://Professor:123456@localhost/testone8",echo=False, pool_size=20, max_overflow=0),
    "Anonymous" : create_engine("postgresql://Anonymous:12345678@localhost/testone8", echo=False, pool_size=20, max_overflow=0),
    "QrReader" : create_engine("postgresql://QrReader:123456@localhost/testone8", echo=False, pool_size=20, max_overflow=0),
}

# studente = create_engine("postgresql://Student:ewfdwefd@localhost/testone8",echo=False, pool_size=20, max_overflow=0)
# engine2 = create_engine("postgresql://Student:studente01@localhost/testone8", echo=False, pool_size=20, max_overflow=0)

Session = sessionmaker()

```

## 5 Funzionalità Principali

In questo capitolo verranno descritte le principali funzionalità dell'applicazione, prestando particolare attenzione alle più interessanti.

### 5.1 Funzionalità a disposizione dei Professori

Le funzionalità a disposizione dei Professori sono le seguenti:

- creare ed modificare corsi;
- creare, modificare ed eliminare lezioni;
- visualizzare una pagina di demografica contenente grafici statistici relativi agli studenti iscritti per ognuno dei loro corsi;
- visualizzare una pagina relativa ai feedback degli studenti per ognuno dei loro corsi;
- interagire con studenti e altri professori attraverso i forum.

### 5.2 Funzionalità a disposizione degli Studenti

Le funzionalità a disposizione degli Studenti sono le seguenti:

- iscriversi e disiscriversi dai corsi;
- effettuare e disdire prenotazioni alle lezioni frontali o duali;
- scannerizzare i QRCODE, forniti a lezione dagli insegnanti, per confermare la loro presenza a lezione;
- interagire con altri studenti e professori attraverso i forum;
- fornire un feedback ai corsi da loro seguiti.

### 5.3 Anonymus

Gli utenti anonimi sono coloro che non hanno ancora eseguito il login.

Le funzionalità a loro disposizione sono:

- visitare e navigare i corsi disponibili nella webapp;
- iscriversi e accedere all'applicazione.

## 5.4 Creazione dei Corsi

L'aggiunta dei corsi viene gestita tramite una sezione apposita nella quale il professore può specificare il nome del corso, il suo codice identificativo (univoco per ogni corso) e una descrizione.

Nome *	ID Corso *
<input type="text" value="Nome"/>	<input type="text" value="ID Corso"/>
Numero massimo di studenti	Ore minime per ottenere il certificato
<input type="text" value="0"/>	<input type="text" value="0"/>
Descrizione	
<input type="text" value="Descrizione"/>	
<input type="button" value="Aggiungi Corso"/>	

Modificando il valore attribuito al campo "Numero massimo di studenti" l'insegnante può fissare un numero massimo di iscritti al corso. Se questo viene lasciato al valore di default 0, non viene impostato alcun limite. Allo stesso modo modificando il valore associato al campo "Ore richieste per ottenere il certificato", l'insegnante può fissare un numero di ore di frequenza obbligatoria necessarie affinché gli studenti ottengano il certificato di frequenza. Tale valore deve essere maggiore di 0.

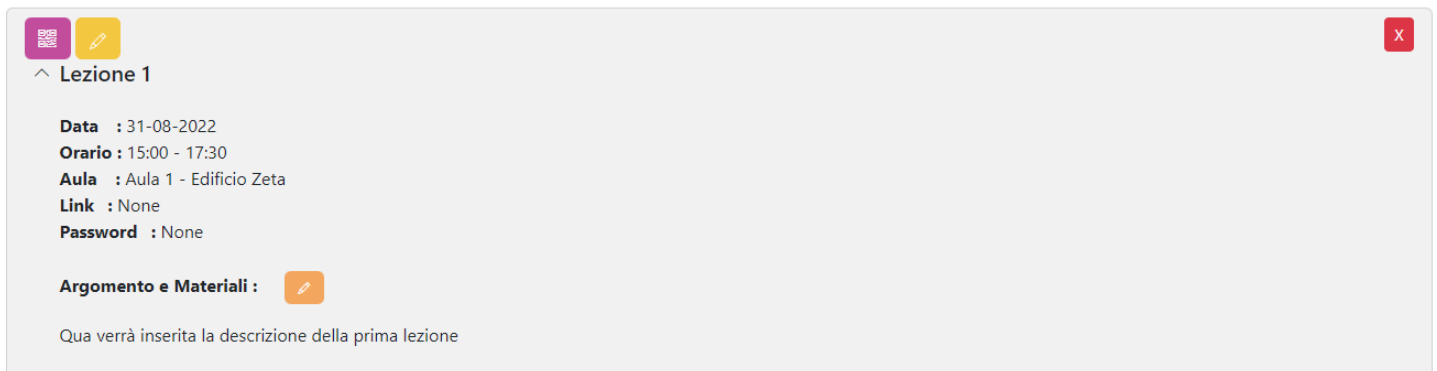
## 5.5 QR Reader e QRCode

Una funzionalità particolarmente degna di nota è il QRCode.

Gli utenti dotati del ruolo di studente, utilizzando uno scanner incorporato all'applicazione e accessibile tramite la voce "Registra presenza" della navbar, possono confermare la loro presenza alle lezioni scannerizzando il qr-code fornito dall'insegnante.

Questo codice avrà validità solo dall'inizio alla fine della lezione.

A ogni lezione è associato un diverso qr-code che sarà visualizzabile solo dal professore di riferimento: dovrà accedere alla voce "Lezioni" del corso e, per ogni lezione, premere l'apposito bottone viola.



: Visualizzazione di una lezione se l'utente corrente è un professore che ha accesso al corso

È stata introdotta un'altra modalità per permettere agli studenti di confermare la propria presenza in aula : le prenotazioni effettuate dagli studenti saranno provviste di qr-code, gli studenti dovranno scannerizzarlo all'ingresso dell'edificio come accade negli edifici del campus scientifico.

Tuttavia, anche questa modalità ha dei limiti, la scannerizzazione del codice sarà permessa a partire da due ore dall'inizio della lezione e non una volta cominciata.

Per accedere alle prenotazioni lo studente dovrà cliccare sul bottone "Prenotazioni" situato nella navbar e poi su "Le mie prenotazioni". Si noti che le prenotazioni passate non verranno visualizzate, seppur vengano mantenute nel database. Di seguito un esempio di come uno studente visualizza una prenotazione.

Basi di Dati	
<b>Aula :</b> Aula 1 - Edificio Zeta	
<b>Data :</b> 01-09-2022	
<b>Orario :</b> 16:00 - 17:30	
<b>Annulla Prenotazione</b>	<b>Posti disponibili :</b> 149 / 150

: Visualizzazione delle prenotazioni effettuate se l'utente corrente è uno studente

Interagendo con il pulsante relativo al qr-code gli studenti lo visualizzeranno attraverso un popup e scaricarlo premendo sul pulsante "Salva QRcode".



: Un esempio di come viene visualizzato un qrcode di una lezione

## 5.6 Q&A

Un'ulteriore funzionalità che riteniamo essere interessante è la presenza di un Forum in ogni corso, per permettere uno scambio tra gli utenti dell'applicativo.

La sezione Q&A è strutturata in una serie di thread, ognuno composto da un post iniziale che funge da domanda e dalle relative risposte.

La pagina del forum è strutturata nel seguente modo:



[Home](#) [Corsi](#) [Prenotazioni](#) [Registra Presenza](#) [Elisa](#) ▾

### Basi di Dati - CT0006-2022

[Docenti e informazioni](#)

[Lezioni](#)

[Forum](#)

#### Forum

Scrivi un nuovo post...

Invia post

---



Elisa Rizzo (You) -- 01-09-2022 - 10:54

[Modifica](#)

Ciao oooo

[Rispondi](#)


La parte superiore della pagina è dedicata ad una sezione atta alla creazione di nuovi post.

### Forum

Scrivi un nuovo post...

Invia post

---



Elisa Rizzo -- 01-09-2022 - 10:54

Ciaooooo

Rispondi

Ciao anche a te!


Invia risposta

### Forum

Scrivi un nuovo post...

Invia post


---



Elisa Rizzo -- 01-09-2022 - 10:54

Ciaooooo

Rispondi



Marco Chinellato (You) -- 01-09-2022 - 10:59 [Modifica](#)

Ciao anche a te!

Salva modifiche

In corrispondenza di ogni domanda postata nel forum l'utente potrà interagire con un pulsante "Rispondi", che gli permetterà di creare un post in risposta. Gli utenti potranno inoltre modificare tutti i loro post tramite l'apposito pulsante di "Modifica".



### Forum

Scrivi un nuovo post...

Invia post

---



Elisa Rizzo -- 01-09-2022 - 10:54

Ciaooooo

Rispondi



Marco Chinellato **(You)** -- 01-09-2022 - 10:59 [Modifica](#)

Ciao anche a te, ma modificato!

Per eliminare un post basta modificarlo rimuovendone il contenuto e creando quindi un port vuoto.

Nel caso in cui il post eliminato fosse una domanda principale saranno eliminati anche tutti i post di risposta.

Le domande sono ordinate dalla più recente alla meno recente. Le risposte per ogni domanda invece sono ordinate dalla meno recente alla più recente, per dare un senso di ordine nella sotto-discussione.

## 5.7 Certificati

Ogni studente può ricevere un attestato di partecipazione relativo ai corsi frequentati, inviato dal professore una volta terminato il corso e aperti i Feedback. Il certificato viene inviato a tutti coloro che hanno frequentato almeno l'ammontare di ore minime richieste dal corso.

Ogni studente potrà visualizzare i suoi certificati nell'apposita pagina "I miei Certificati".

Ecco un esempio di come uno studente visualizza i suoi certificati.

### Basi di Dati

**Congratulazioni Elisa!!** É ora disponibile l'attestato di partecipazione al corso Basi di Dati - CT0006

---

Hai seguito questo corso per **17.0** Ore su **15.0** ore minime richieste

Scaricami!

## 5.8 Feedback

Allo scopo di garantire una migliore interazione con gli studenti e fornire agli insegnanti un resoconto delle impressioni degli stessi, per quanto riguarda i corsi da loro tenuti, è stato inoltre implementato un sistema di feedback.

Ogni studente iscritto a un corso, alla fine di questo potrà :

- assegnare un voto al corso;
- assegnare un voto all'insegnante;
- lasciare dei commenti facoltativi.

Ogni feedback e ogni commento lasciato dagli studenti sarà anonimo. Non essendo stata creata alcuna relazione fra la tabella "Students" e la tabella "Feedbacks" non si potrà risalire in alcun modo all'autore del commento.

Di seguito come viene presentata la pagina agli studenti :

**Assegna un voto da 0 a 10 al corso :**

- ☐ 0 - Per niente soddisfatto
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7
- ☐ 8
- ☐ 9
- ☐ 10 - Completamente soddisfatto

**Assegna un voto da 0 a 10 al professore :**


- ☐ 0 - Per niente soddisfatto
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7
- ☐ 8
- ☐ 9
- ☐ 10 - Completamente soddisfatto

**Commenti aggiuntivi (facoltativo) :**

Commenta

Invia feedback

I professori possono visualizzare i feedback in una pagina apposita, alla quale possono accedere tramite il pulsante "Leggi feedback studenti!" presente nelle pagine dei corsi da loro tenuti.



Home Corsi Alessandra ▾ Aggiungi Corso

## Basi di Dati - CT0006-2022

Leggi feedback studenti!

Docenti e informazioni

Lezioni

Forum

### Docenti

Stefano Calzavara

Alessandra Raffaetà

### Informazioni Corso

Iscritti attuali: 5/55

**Descrizione Corso:**  
Corso di basi di dati!

Per ottenere l'attestato di partecipazione al corso, lo studente dovrà seguire almeno **15** ore di lezione.

Vuoi chiudere i feedback?

In questa pagina possono visualizzare il voto medio che gli studenti hanno assegnato loro dagli studenti, il voto medio assegnato al corso e una tabella contenente tutti i commenti.

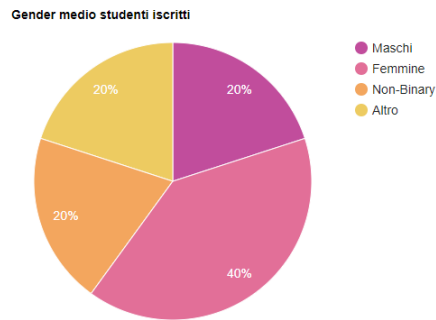
#	Valutazione media al corso	Valutazione media all'insegnante
1	8.12	8.67

#	Commenti
1	Prof disponibile e attento
2	Questo è un feedback ma non so cosa scrivere scusate
3	Un altro feedback, ma anche stavolta non so che dire

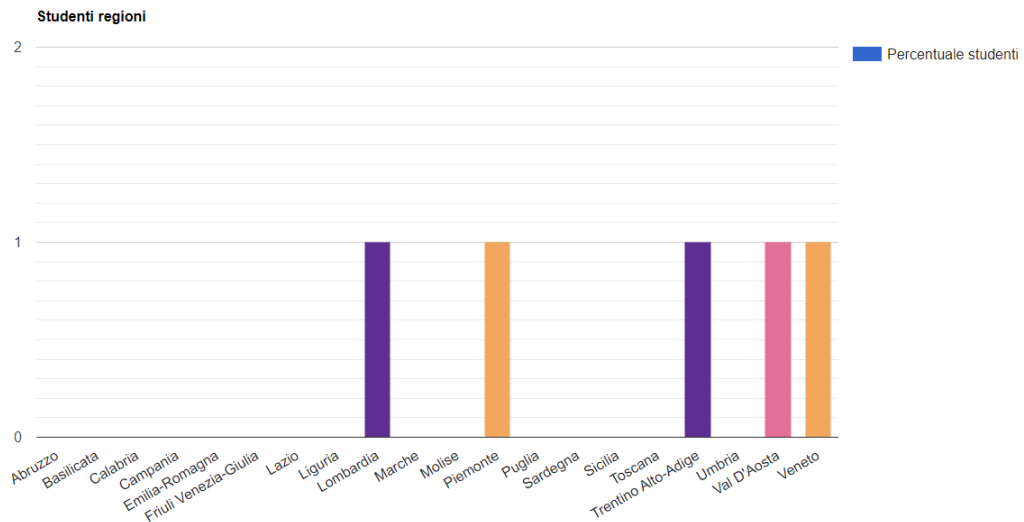
## 5.9 Demografia

Il professore può inoltre visualizzare una sezione dove vengono inseriti grafici statistici riguardanti gli studenti che stanno seguendo il corso:

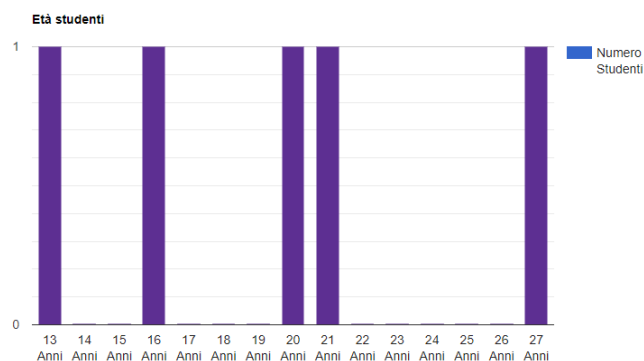
- un grafico "Gender";



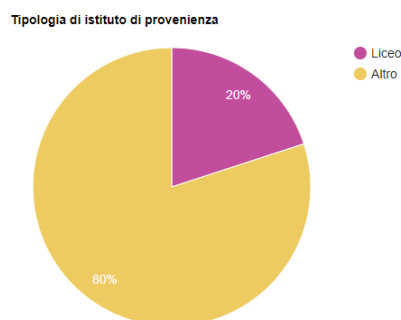
- un grafico di provenienza regionale "Regioni studenti", che riporta il numero di studenti provenienti da ogni regione italiana;



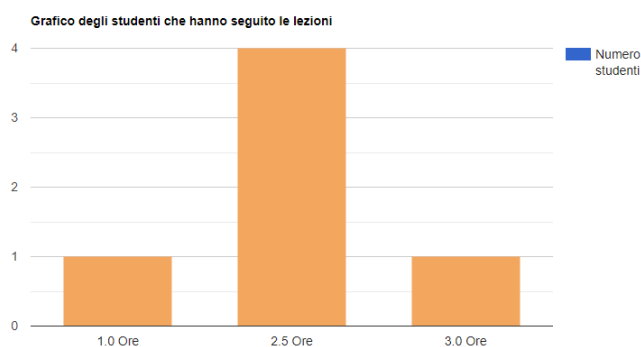
- un grafico "Età studenti", che riporta il numero di studenti per ogni fascia di età. È stato deciso di non inserire fasce di età dato che in un contesto di alternanza scuola lavoro, la differenza anagrafica non sarà rilevante;



- un grafico per distinguere la tipologia degli istituti di provenienza (Liceo, ITIS, IPSIA o altro);



- un grafico riportante il numero di ore frequentate dagli studenti.



Si è ritenuto che basarsi sul numero di ore, piuttosto che sul numero di lezioni, fosse più utile al professore al fine di avere maggiori informazioni su quanti studenti hanno ottenuto o stanno per ottenere l'attestato di partecipazione.

Gli insiemi degli studenti sono disgiunti, quindi uno studente che ha seguito 2.5 ore di lezione, verrà inserito solo nella colonna relativa al valore 2.5, e non in quelle con valori inferiori.

## 5.10 Aggiunta di molteplici lezioni

I professori oltre ad avere la possibilità di aggiungere singole lezioni, possono anche decidere di aggiungere più lezioni, tramite l'apposita sezione all'interno della pagina dei loro corsi, sotto la voce "Lezioni".

^ Aggiungi Lezioni

Data

gg/mm/aaaa

Seleziona Aula

--Seleziona un tipo --

Orario di Inizio

--:--

Orario di Fine

--:--

Argomento e Materiali

...

Modalità erogazione

--Seleziona un tipo--

Link lezione

Password Lezione

Conferma

Aggiungi più lezioni insieme

Data Inizio

gg/mm/aaaa

Data Fine

gg/mm/aaaa

Orario di Inizio

--:--

Orario di Fine

--:--

Seleziona Aula

--Seleziona un tipo--

Modalità erogazione

--Seleziona un tipo--

Conferma

Come mostrato nell'immagine proposta qui sopra, tramite la sezione "Aggiungi più lezioni insieme", l'insegnante potrà impostare un orario e un intervallo di tempo all'interno del quale programmare le proprie lezioni nel giorno della settimana corrispondente al giorno di inizio intervallo (per esempio prendendo un intervallo con data di inizio lunedì, le lezioni verranno programmate tutti i lunedì interni all'intervallo indicato).

Le lezioni aggiunte tramite questa opzione dovranno però essere tutte erogate nella stessa modalità e nel caso in cui queste siano frontali o duali dovranno svolgersi nella stessa aula.

Modifiche della modalità di erogazione, degli orari, delle aule e l'aggiunta dell'argomento, dei materiali e del link e della password delle lezioni dovranno essere

fatte manualmente ad ogni lezione.

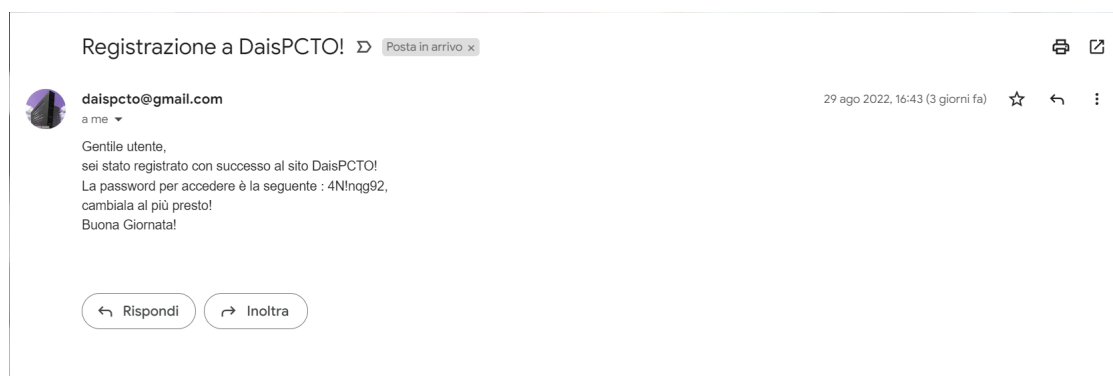
### 5.11 Registrazione Professori

Come accennato nella sezione 1.2, i professori non hanno la possibilità di registrarsi autonomamente all'applicazione per motivi di sicurezza.

La loro registrazione viene realizzata da un utente con il ruolo di Admin, il quale accedendo alla seguente pagina registra l'insegnante indicandone nome, cognome ed e-mail.

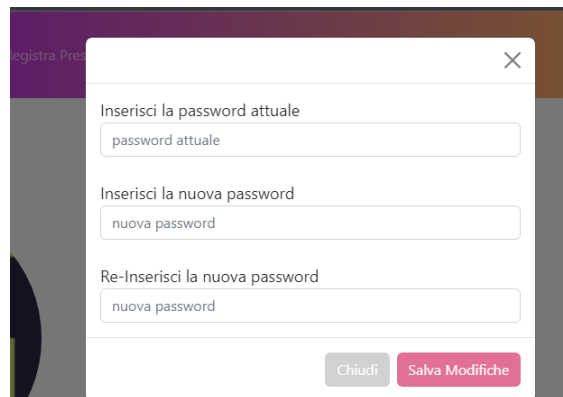
Nome	Cognome
<input type="text" value="Nome"/>	<input type="text" value="Cognome"/>
Email	
<input type="text" value="@ Email"/>	
<input type="button" value="Registra professore"/>	

A seguito della registrazione verrà generata una password casuale, successivamente inviata alla mail utilizzata dal professore durante il processo di registrazione, per permettere l'accesso all'app.





La password potrà poi essere modificata tramite l'apposita voce nella home page dell'applicazione.



A modal form for changing a password. It has a close button (X) in the top right corner. The form contains three input fields: 'Inserisci la password attuale' with placeholder 'password attuale', 'Inserisci la nuova password' with placeholder 'nuova password', and 'Re-Inserisci la nuova password' with placeholder 'nuova password'. At the bottom right, there are two buttons: 'Chiudi' (grey) and 'Salva Modifiche' (pink).

Inserisci la password attuale  
password attuale

Inserisci la nuova password  
nuova password

Re-Inserisci la nuova password  
nuova password

Chiudi Salva Modifiche

## 6 Query e Funzioni Principali

In questa sezione della relazione verranno presentate le query e le funzioni più interessanti e più importanti utilizzate nel progetto.

In particolare verranno prese in considerazione quelle relative a :

- la prenotazione delle lezioni da parte degli studenti;
- l'aggiunta di molteplici lezioni in contemporanea;
- la modifica delle lezioni da parte degli insegnanti;
- l'invio dei certificati agli studenti.;
- la costruzione dei grafi relativi alla demografica.

### 6.1 Query "get\_lessons\_bookable"

Questa query permette di selezionare, a partire da un utente passato in input attraverso il parametro "user\_id", tutte le lezioni prenotabili da tale utente, che si svolgeranno entro i successivi 7 giorni.

```
def get_lessons_bookable(user_id):
    try:
        session = Session(bind=get_engine())

        return session.query(Course.Name.label("CourseName"), Lesson.Date, Lesson.StartTime, Lesson.EndTime, Classroom.Name, Classroom.Building, Classroom.Seats,
                              Lesson.LessonID, func.coalesce(Reservation.StudentID, -1).label("StudentID"), Reservation.ReservationID)\
            .join(Course, Course.CourseID == Lesson.CourseID)\
            .join(StudentCourse, and_(StudentCourse.StudentID == user_id, StudentCourse.CourseID == Course.CourseID))\
            .join(FrontalLesson, FrontalLesson.LessonID == Lesson.LessonID)\
            .join(Classroom, Classroom.ClassroomID == FrontalLesson.ClassroomID)\
            .join(Reservation, and_(Reservation.FrontalLessonID == FrontalLesson.LessonID, Reservation.StudentID == user_id, isouter=True))\
            .filter(or_(
                and_(Lesson.Date - func.current_date() <='7', Lesson.Date - func.current_date() > '0'),
                and_(Lesson.StartTime > func.current_time(), Lesson.Date - func.current_date() == '0')
            ))\
            .order_by(Lesson.Date, Lesson.StartTime, Course.Name)\
            .all()

    except exc.SQLAlchemyError as e:
        return None
```

### 6.2 Query "add\_multiple\_lessons"

La query permette di aggiungere molteplici lezioni in contemporanea.

Nel caso in cui alcune delle lezioni da inserire non possano essere aggiunte, non verrà fatta una rollback dell'intera operazione, ma verranno invece aggiunte le restanti lezioni.

Nel caso di mancata aggiunta di alcune lezioni, alla fine dell'operazione, verrà fatto un flash a schermo che indicherà quali non è stato possibile aggiungere.

```
def add_multiple_lesson(form, course_id, professor):
    """
    VOGLIAMO FARE IN MODO CHE AGGIUNGA TUTTE LE LEZIONI CHE SI POSSONO AGGIUNGERE, IGNORANDO QUELLE CHE NON SI POSSONO AGGIUNGERE
    """
    start_date = form.start_date_2.data
    end_date = form.end_date_2.data
    diff_date = start_date

    is_dual = True if form.lesson_type_2.data == "Duale" else False

    while diff_date < end_date:
        token = generate_password_hash(f'{current_user.get_id()}{course_id}{diff_date}{form.start_time_2.data}{form.end_time_2.data}{form.classroom_2.data}{random.randint(0, 1000)}')\
            .decode('utf-8')
        lesson_new = Lesson(Date = diff_date, StartTime = form.start_time_2.data, EndTime = form.end_time_2.data, CourseID = course_id, IsDual = is_dual, ProfessorID=professor,
            Token=token)

        result_query = _add_lesson(lesson_new, form.lesson_type_2.data, form.classroom_2.data)

        if result_query == 'ClashError' or result_query == "SameCourseClashError":
            flash(f'Lezione del {diff_date} non aggiunta per sovrapposizione')

        diff_date = diff_date + datetime.timedelta(days=7)
```

Per effettuare l'effettiva aggiunta delle singole lezioni viene richiamata la query "`_add_lesson`" di seguito riportata.

```
def _add_lesson(lesson_new, type_lesson, classroom, link=None, password=None):
    try:
        session = Session(bind=get_engine())
        session.add(lesson_new)
        session.flush()

        if (type_lesson == "Frontale"):
            session.add(FrontalLesson(LessonID = lesson_new.LessonID, ClassroomID = classroom))

        elif (type_lesson == "Online"):
            session.add(OnlineLesson(LessonID=lesson_new.LessonID, RoomLink = link, RoomPassword = password))

        elif (type_lesson == "Duale"):
            session.add(FrontalLesson(LessonID = lesson_new.LessonID, ClassroomID = classroom))
            session.add(OnlineLesson(LessonID=lesson_new.LessonID, RoomLink = link, RoomPassword = password))

        session.commit()
    except exc.SQLAlchemyError as e:
        session.rollback()

        if e.orig.diag.message_primary == "LessonOverlapping":
            # delete Lesson(lesson_new.LessonID)
            return "ClashError"

        if e.orig.diag.message_primary == "SameCourseOverlapping":
            return "SameCourseClashError"

        if int(e.orig.pgcode) == 23514: #il code 23514 indica un check error che ritorna il dbms
            return "DataError"

        return "UnknownError"
    finally:
        session.close()
    return "Success"
```

### 6.3 Query "`update_lesson`"

Questa query permette la modifica delle lezioni.

Non si occupa solo della modifica di informazioni quali data e orario, ma anche della modifica della modalità di erogazione e della conseguente aggiunta o rimozione della lezione dalle tabelle "`FontalLessons`" e "`OnlineLessons`".

```

def update_lesson(lesson_id, form):
    try:
        session = Session(bind=get_engine())

        is_current_lesson_frontal = session.query(FrontalLesson).filter(FrontalLesson.LessonID == lesson_id).first() is not None
        is_current_lesson_online = session.query(OnlineLesson).filter(OnlineLesson.LessonID == lesson_id).first() is not None

        is_current_lesson_dual = is_current_lesson_frontal and is_current_lesson_online

        is_new_lesson_dual = True if form.lesson_type_update.data == 'Duale' else False

        if form.lesson_type_update.data == "Frontale" and (is_current_lesson_dual or is_current_lesson_online):
            session.query(OnlineLesson).filter(OnlineLesson.LessonID == lesson_id).delete()
            session.query(FrontalLesson).filter(FrontalLesson.LessonID == lesson_id).update({FrontalLesson.ClassroomID : form.classroom_update.data})
            if not is_current_lesson_frontal:
                session.add(FrontalLesson(LessonID = lesson_id, ClassroomID = form.classroom_update.data))

        elif form.lesson_type_update.data == "Online" and (is_current_lesson_frontal or is_current_lesson_dual):
            session.query(FrontalLesson).filter(FrontalLesson.LessonID == lesson_id).delete()
            session.query(OnlineLesson).filter(OnlineLesson.LessonID == lesson_id).update({OnlineLesson.RoomLink : form.link_update.data, OnlineLesson.
            RoomPassword : form.password_update.data})
            if not is_current_lesson_online:
                session.add(OnlineLesson(LessonID = lesson_id, RoomLink = form.link_update.data, RoomPassword = form.password_update.data))

        elif form.lesson_type_update.data == "Duale" and not is_current_lesson_online:
            session.add(OnlineLesson(LessonID= lesson_id, RoomLink=form.link_update.data, RoomPassword = form.password_update.data))
            session.query(FrontalLesson).filter(FrontalLesson.LessonID == lesson_id).update({FrontalLesson.ClassroomID : form.classroom_update.data})

        elif form.lesson_type_update.data == "Duale" and not is_current_lesson_frontal:
            session.add(FrontalLesson(LessonID=lesson_id, ClassroomID = form.classroom_update.data))
            session.query(OnlineLesson).filter(OnlineLesson.LessonID == lesson_id).update({OnlineLesson.RoomLink : form.link_update.data, OnlineLesson.
            RoomPassword : form.password_update.data})

        session.query(Lesson).filter(Lesson.LessonID == lesson_id).update({Lesson.Date : form.date_update.data, Lesson.StartTime : form.start_time_update.data,
        Lesson.EndTime : form.end_time_update.data, Lesson.IsDual : is_new_lesson_dual})
        session.commit()

    except exc.SQLAlchemyError as e:
        session.rollback()

        if e.orig.diag.message_primary == "LessonOverlapping":
            flash("Lesson overlapping error")
            # delete_lesson(Lesson_new.LessonID)
            return "ClashError"

        if e.orig.diag.message_primary == "SameCourseOverlapping":
            flash("Lesson overlapping error")
            return "SameCourseClashError"

        if e.orig.pgcode == '23514': #check constraint error
            flash("Date error")
            return "DataError"

        flash("Unknown error")
        return "UnknownError"

    finally:
        session.close()

    return "Success"

```

In particolare:

- se la lezione era Duale e viene modificata per essere Frontale :
  - verrà eliminata dalla tabella "OnlineLessons";
  - verrà modificato l'elemento corrispondente nella tabella "FrontalLessons";
- se la lezione era Duale e viene modificata per essere Online :
  - verrà eliminata dalla tabella "FrontalLessons";
  - verrà modificato l'elemento corrispondente nella tabella "OnlineLessons";

- se la lezione era Frontale e viene modificata per essere Online :
  - verrà eliminata dalla tabella "FrontalLessons";
  - verrà aggiunta nella tabella "OnlineLessons";
- se la lezione era Frontale e viene modificata per essere Duale :
  - verrà aggiunta nella tabella "OnlineLessons";
  - verrà modificato l'elemento corrispondente nella tabella "FrontalLessons";
- se la lezione era Online e viene modificata per essere Frontale :
  - verrà eliminata dalla tabella "OnlineLessons";
  - verrà aggiunta nella tabella "FrontalLessons";
- se la lezione era Online e viene modificata per essere Duale :
  - verrà aggiunta nella tabella "FrontalLessons";
  - verrà modificato l'elemento corrispondente nella tabella "OnlineLessons".

Se non avverranno modifiche alla modalità di erogazione, verrà semplicemente modificato l'elemento corrispondente nella tabella "OnlineLessons" nel caso in cui la lezione fosse Online, nella tabella "FrontalLessons" nel caso in cui la lezione fosse Frontale, oppure in entrambe nel caso in cui la lezione fosse Duale. Nel caso in cui si verificano delle sovrapposizioni di orario con altre lezioni dello stesso corso o relative all'utilizzo di un'aula già occupata da un'altra lezione, verranno lanciati appositi errori e le modifiche verranno annullate.

## 6.4 Funzione "send\_certificate\_to\_students"

La seguente funzione viene utilizzata per inviare i certificati agli studenti.

```
def send_certificate_to_students(course_id):
    # hours = session.query(Course).filter(Course.CourseID == course_id.upper()).first().MinHourCertificate
    hours = get_course_by_id(course_id).MinHourCertificate
    # student_courses = session.query(StudentCourse).filter(course_id == StudentCourse.CourseID).all()
    student_courses = get_students_by_course(course_id)

    for student in student_courses:
        students_courses_hours = get_student_courses(student.UserID) #dentro i corsi dello studente ci salviamo ANCHE quante ore ha seguito di suddetto corso
        for course in students_courses_hours:
            if course.CourseID == course_id.upper() and course.Hours.total_seconds()/3600 >= hours:
                try:
                    session = Session(bind=get_engine())
                    session.add(Certificate(StudentID = student.UserID, CourseID = course_id.upper(), Hours = course.Hours.total_seconds()/3600))
                    session.commit()
                except exc.SQLAlchemyError as e:
                    session.rollback()
            finally:
                session.close()
```

A partire da uno specifico corso, identificato dal parametro "course\_id" passato in input alle successive funzioni, richiamando la query "get\_course\_by\_id" salva il numero di ore minime da frequentare per ottenere l'attestato di partecipazione.

```
def get_course_by_id(course_id):
    try:
        session = Session(bind=get_engine())
        return session.query(Course).filter(Course.CourseID == course_id).first()
    except:
        return None
```

Successivamente, richiama la query "get\_students\_by\_course" per salvare la lista degli studenti frequentanti il corso.

```
def get_students_by_course(course_id):
    try:
        session = Session(bind=get_engine())

        return session.query(User.Name, User.Surname, Student.birthDate, Student.City, User.UserID).join(Student, Student.UserID == User.UserID)\
            .filter(and_(StudentCourse.CourseID == course_id, StudentCourse.StudentID == Student.UserID))\
            .order_by(User.Surname, User.Name)\
            .all()
    except Exception as e:
        print(e)
        return []
```

Infine richiamando la query "get\_student\_courses" salva per ogni studente, appartenente alla lista precedentemente creata, il numero di ore da lui frequentate.

```
def get_student_courses(user_id):
    try:
        session = Session(bind=get_engine())

        query = session.query(Course.CourseID, Course.Name, Course.Description, Course.MinHourCertificate, func.sum(case((and_(Lesson.StartTime.isnot(None),
            Lesson.EndTime.isnot(None)), Lesson.EndTime-Lesson.StartTime), else_="00:00:00")).label("Hours")) \
            .join(StudentCourse, StudentCourse.CourseID == Course.CourseID)\
            .join(StudentLesson, StudentLesson.StudentID == StudentCourse.StudentID, isouter=True)\
            .join(Lesson, and_(Lesson.LessonID == StudentLesson.LessonID, Lesson.CourseID == Course.CourseID), isouter=True)\
            .filter(StudentCourse.StudentID == user_id)\
            .order_by(Course.Name)\
            .group_by(Course.CourseID, Course.Name, Course.Description, Course.MinHourCertificate).all()

        return query
    except Exception as e:
        print(e)
        return None
```

Solo dopo aver controllato che le ore seguite dagli studenti siano maggiori rispetto al numero di ore minime, invia i certificati.

## 6.5 Query relative alla demografica

La sezione di analisi demografica dei corsi richiama 5 query, utilizzando i dati da queste prodotti per costruire i diagrammi colonnali e i diagrammi a torta riportati nella sezione 5.9, identificando il corso di riferimento con il parametro "course\_id" passato in input alle funzioni.

### 6.5.1 Query "gender\_subscribed"

La query riportata di seguito viene utilizzata per creare il grafico a torta relativo al genere degli studenti.

```
def gender_subscribed(course_id):
    try:
        session = Session(bind=get_engine())
        return session.query(func.sum(case((and_(User.Gender.isnot(None), User.Gender == "Male"), 1), else_=0)).label("Male"),\
            func.sum(case((and_(User.Gender.isnot(None), User.Gender == "Female"), 1), else_=0)).label("Female"),\
            func.sum(case((and_(User.Gender.isnot(None), User.Gender == "Non Binary"), 1), else_=0)).label("NonBinary"), \
            func.sum(case((and_(User.Gender.isnot(None), User.Gender == "Other"), 1), else_=0)).label("Other"))\
            .join(StudentCourse, User.UserID == StudentCourse.StudentID).filter(StudentCourse.CourseID == course_id).first()
    except Exception as e:
        print(e)
        return None
```

### 6.5.2 Query "age\_subscribed"

Questa query restituisce una lista contenente l'età di tutti gli studenti iscritti al corso.

```
def age_subscribed(course_id):
    try:
        session = Session(bind=get_engine())
        return session.query(Student.birthDate).join(StudentCourse, StudentCourse.StudentID == Student.UserID).filter(StudentCourse.CourseID == course_id).all()
    except:
        return []
```

### 6.5.3 Query "city\_subscribed"

Questa query serve per ottenere le città di provenienza di tutti gli studenti che seguono il corso.

```
def city_subscribed(course_id):
    try:
        session = Session(bind=get_engine())
        return session.query(Student.City).join(StudentCourse, StudentCourse.StudentID == Student.UserID).filter(StudentCourse.CourseID == course_id).all()
    except:
        return
```

Successivamente vengono convertite in regioni attraverso la seguente funzione, la quale esegue inoltre un conteggio del numero di studenti provenienti da ognuna di esse :

```

@courses.route('/action/get/region')
@role_required("Professor")
def get_student_region():

    course_id = request.args.get("course_id").upper()
    c = city_subscribed(course_id)

    res = {}
    with open("province.json") as f:
        data = json.load(f)
        for city in c:
            for i in data:
                if i["nome"].upper() == city.City:
                    if i["regione"] not in res:
                        res[i["regione"]] = 1
                    else:
                        res[i["regione"]] += 1

    return jsonify({"success" : True, "Regioni" : res, "Total" : len(c)})

```

#### 6.5.4 Query "type\_school\_subscribed"

La seguente funzione serve per conteggiare il numero di studenti provenienti da diverse tipologie di istituti superiori di secondo grado che seguono il corso.

```

def type_school_subscribed(course_id):
    try:
        session = Session(bind=get_engine())
        return session.query(func.sum(case((and_(School.Type.contains("LICEO"), School.Type.isnot(None)), 1), else_=0)).label("Liceo"),\
            func.sum(case((and_(School.Type.contains("ISTITUTO TECNICO"), School.Type.isnot(None)), 1), else_=0)).label("Tecnico"),\
            func.sum(case((and_(School.Type.contains("PROFESSIONALE"), School.Type.isnot(None)), 1), else_=0)).label("Professionale"), \
            func.sum(case((and_(not_(School.Type.contains("LICEO")), not_(School.Type.contains("ISTITUTO TECNICO")), not_(School.Type.contains(
                "PROFESSIONALE"))), School.Type.isnot(None)), 1), else_=0)).label("Altro"))\
            .join(Student, Student.SchoolID == School.SchoolID)\
            .join(StudentCourse, and_(StudentCourse.StudentID == Student.UserID, StudentCourse.CourseID == course_id))\
            .first()
    except Exception as e:
        return None

```

Le tipologie di istituti che sono state considerate per semplificare l'operazione sono :

- Licei
- Istituti Tecnici
- Istituti Professionali

Altre possibili tipologie di istituti sono state raggruppate nella categoria "**Altro**".



### 6.5.5 Query "hours\_attended"

Questa query si occupa del calcolo del numero di ore frequentate da ogni studente iscritto al corso.

```
def hours_attended(course_id):  
    try:  
        session = Session(bind=get_engine())  
        q1 = session.query(func.sum(case((and_(Lesson.StartTime.isnot(None), Lesson.EndTime.isnot(None)), Lesson.EndTime-Lesson.StartTime), else_="00:00:00")).  
            label("Hours"))\  
            .join(StudentLesson, StudentLesson.LessonID == Lesson.LessonID)\  
            .filter(Lesson.CourseID == course_id)\  
            .group_by(StudentLesson.StudentID).all()
```

## 7 Navigazione Web App

### 7.1 Registrazione e Login



[Home](#) [Corsi](#) [Account](#) ▾

<b>Nome</b> <input type="text" value="Nome"/>		<b>Cognome</b> <input type="text" value="Cognome"/>	<b>Genere</b> <input type="text" value="--Seleziona un genere--"/>
<b>Email</b> <input type="text" value="@ Email"/>		<b>Reinserisci Email</b> <input type="text" value="@ Ripeti Email"/>	
<b>Indirizzo</b> <input type="text" value="Indirizzo"/>		<b>Provincia di provenienza</b> <input type="text" value="Provincia"/>	
<b>Password</b> <input type="text" value="Password"/>		<b>Reinserisci Password</b> <input type="text" value="Ripeti Password"/>	
<b>Numero di Telefono</b> <input type="text" value="Numero di Telefono"/>		<b>Data di Nascita</b> <input type="text" value="gg/mm/aaaa"/>	
<b>Nome scuola di provenienza</b> <input type="text" value="Nome scuola"/>		<b>Anno scolastico</b> <input type="text" value="Anno scolastico"/>	
<input type="button" value="Registrati"/>			

Come già detto in precedenza la pagina di registrazione interessa solo gli studenti.



email

email

password

password

☐ Remember me

Accedi!

## 7.2 Visualizzazione lista dei corsi



[Home](#) [Corsi](#) [Account](#) ▾

🕒 Tutti i corsi

🕒 I miei corsi

<b>ASD</b>  <b>Descrizione Corso:</b> ASD  <a href="#">Visualizza pagina del corso</a>	<b>Basi di Dati</b>  <b>Descrizione Corso:</b> Corso di basi di dati!  <a href="#">Visualizza pagina del corso</a>	<b>Basi di Dati 2</b>  <b>Descrizione Corso:</b> basi di dati mod 2  <a href="#">Visualizza pagina del corso</a>
<b>Programmazione</b>  <b>Descrizione Corso:</b> Programmazione 1, C  <a href="#">Visualizza pagina del corso</a>	<b>Programmazione ad Oggetti</b>  <b>Descrizione Corso:</b> Corso di programmazione ad oggetti, Java  <a href="#">Visualizza pagina del corso</a>	<b>Sicurezza</b>  <b>Descrizione Corso:</b> Corso di sicurezza  <a href="#">Visualizza pagina del corso</a>

### 7.2.1 Visualizzazione i miei corsi – Studente Autenticato



Home Corsi Prenotazioni Registra Presenza Elisa ▾

🔍 Tutti i corsi

🔍 I miei corsi

**Basi di Dati**

Disiscriviti

**Descrizione Corso:**  
Corso di basi di dati!

Visualizza pagina del corso

10%

**Basi di Dati 2**

Disiscriviti

**Descrizione Corso:**  
basi di dati mod 2

Visualizza pagina del corso

30%

## 7.2.2 Visualizzazione i miei corsi – Professore Autenticato



Home Corsi Alessandra ▾ Aggiungi Corso

Tutti i corsi

I miei corsi

### ASD

#### Descrizione Corso:

ASD

Visualizza pagina del corso

### Basi di Dati

#### Descrizione Corso:

Corso di basi di dati!

Visualizza pagina del corso

### Basi di Dati 2

#### Descrizione Corso:

basi di dati mod 2

Visualizza pagina del corso

### Programmazione

#### Descrizione Corso:

Programmazione 1, C

Visualizza pagina del corso

### Programmazione ad Oggetti

#### Descrizione Corso:

Corso di programmazione ad oggetti, Java

Visualizza pagina del corso

## 7.3 Visualizzazione pagina del corso

### 7.3.1 Studente Autenticato



[Home](#) [Corsi](#) [Prenotazioni](#) [Registra Presenza](#) [Elisa](#) ▼

## ASD - ASD01

[Docenti e informazioni](#) [Lezioni](#) [Forum](#)

### Docenti

Alessandra Raffaetà

### Informazioni Corso

**Iscritti attuali:** 1/69

[Iscriviti!](#)

#### Descrizione Corso:

FIORI

Per ottenere l'attestato di partecipazione al corso, lo studente dovrà seguire almeno **96** ore di lezione.

Nel caso in cui i feedback siano aperti la pagina visualizzata è la seguente.



## Basi - CT0006-2022

[Invia feedback!](#)

[Docenti e informazioni](#)

[Lezioni](#)

[Forum](#)

### Docenti

Stefano Calzavara

Alessandra Raffaetà

### Informazioni Corso

**Iscritti attuali:** 5/55

[Disiscriviti!](#)

#### Descrizione Corso:

Corso di basi di dati!

Per ottenere l'attestato di partecipazione al corso, lo studente dovrà seguire almeno **15** ore di lezione.



### 7.3.2 Professore Autenticato



[Home](#) [Corsi](#) [Alessandra](#) ▾ [Aggiungi Corso](#)

## Basi - CT0006-2023

[Leggi feedback studenti!](#)

[Docenti e informazioni](#)

[Lezioni](#)

[Forum](#)

### Docenti

[Aggiungi collaboratori](#)

Andrea Marin

Alessandra Raffaetà

### Informazioni Corso



Iscritti attuali: 1/15

#### Descrizione Corso:

Corso bielo

Per ottenere l'attestato di partecipazione al corso, lo studente dovrà seguire almeno **48** ore di lezione.

[Vuoi aprire i feedback?](#)

## 7.4 Visualizzazione pagina Lezioni

### 7.4.1 Studente Autenticato



[Home](#) [Corsi](#) [Prenotazioni](#) [Registra Presenza](#) [Elisa](#) ▾

## Basi - CT0006-2023

[Docenti e informazioni](#)

[Lezioni](#)

[Forum](#)

#### ^ Lezione 1 - (Online)

**Data** : 31-08-2022

**Orario** : 16:00 - 18:00

**Link** : None

**Password** : None

**Argomento e Materiali :**

None

#### ^ Lezione 2 - (Online)

**Data** : 07-09-2022

**Orario** : 16:00 - 18:00

**Link** : None

**Password** : None

**Argomento e Materiali :**

None

#### ^ Lezione 3 - (Online)

**Data** : 14-09-2022

## 7.4.2 Professore Autenticato



Home Corsi Alessandra ▾ Aggiungi Corso

### Basi - CT0006-2023

Docenti e informazioni Lezioni Forum



#### Lezione 1 - (Online)

**Data** : 31-08-2022

**Orario** : 16:00 - 18:00

**Link** : None

**Password** : None

**Argomento e Materiali** :



None



#### Lezione 2 - (Online)

**Data** : 07-09-2022

**Orario** : 16:00 - 18:00

**Link** : None

**Password** : None

**Argomento e Materiali** :



None

6-2023/lessons

## 7.5 Prenotazioni – Studente Autenticato



[Home](#) [Corsi](#) [Prenotazioni](#) [Registra Presenza](#) [Elisa](#) ▼

🕒 Prenotazione Future

🕒 Le mie Prenotazioni

### Basi

**Aula :** Aula 1 - Edificio Zeta

**Data :** 07-09-2022

**Orario :** 16:00 - 17:30

Prenota Posto in Aula

**Posti disponibili :** 150 / 150

### Basi

**Aula :** Aula 1 - Edificio Zeta

**Data :** 08-09-2022

**Orario :** 16:00 - 17:30

Prenota Posto in Aula

**Posti disponibili :** 150 / 150

### 7.5.1 Visualizzazione "Le mie prenotazioni"



[Home](#) [Corsi](#) [Prenotazioni](#) [Registra Presenza](#) [Elisa](#) ▼

🕒 Prenotazione Future

🕒 Le mie Prenotazioni

Basi	
<b>Aula :</b> Aula 1 - Edificio Zeta	
<b>Data :</b> 07-09-2022	
<b>Orario :</b> 16:00 - 17:30	
<a href="#">Annulla Prenotazione</a>	<b>Posti disponibili :</b> 149 / 150

## 7.6 Feedback

### 7.6.1 Studente Autenticato



[Home](#) [Corsi](#) [Prenotazioni](#) [Registra Presenza](#) [Elisa](#) ▼

**Assegna un voto da 0 a 10 al corso :**

- ☐ 0 - Per niente soddisfatto
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7
- ☐ 8
- ☐ 9
- ☐ 10 - Completamente soddisfatto

**Assegna un voto da 0 a 10 al professore :**

- ☐ 0 - Per niente soddisfatto
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- ☐ 7
- ☐ 8
- ☐ 9
- ☐ 10 - Completamente soddisfatto

**Commenti aggiuntivi (facoltativo) :**

Commenta

## 7.6.2 Professore Autenticato



[Home](#) [Corsi](#) [Alessandra ▾](#) [Aggiungi Corso](#)

#	Valutazione media al corso	Valutazione media all'insegnante
1	9.00	8.67

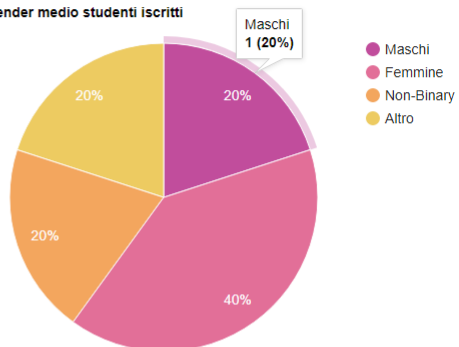
#	Commenti
1	Commento 1
2	q3wreqr3qr3
3	CIAOO COME VAAAA

## 7.7 Pagina di demografica – Professore Autenticato



Home Corsi Alessandra ▾ Aggiungi Corso

Gender medio studenti iscritti



Studenti regioni

2		Percentuale studenti



## 7.8 Forum



[Home](#) [Corsi](#) [Prenotazioni](#) [Registra Presenza](#) [Elisa](#) ▾

### Basi - CT0006-2023

[Docenti e informazioni](#)


[Lezioni](#)


[Forum](#)


#### Forum

Scrivi un nuovo post...

Invia post

 Elisa Rizzo **(You)** -- 01-09-2022 - 22:38 [Modifica](#)  
Ole ole ole [Rispondi](#)

 Alessandra Raffaetà -- 01-09-2022 - 16:04  
hbop [Rispondi](#)

 Elisa Rizzo **(You)** -- 01-09-2022 - 22:39 [Modifica](#)  
Ciaooo

64

## 8 Tecnologie Utilizzate

Per quanto riguarda **Python** sono stati utilizzati i seguenti framework e le seguenti librerie:

- **Flask**;
- **Flask-Login** : per gestire l'autenticazione degli utenti;
- **Flask-WTF** : per gestire al meglio i form, sfruttando anche il CSRF-Token per garantire una maggiore sicurezza;
- **Flask-Bcrypt** : per inserire un hash delle password all'interno del database, al fine di avere una maggiore sicurezza ed evitare di metterle in chiaro;
- **Flask-Gravatar** : non va a modificare la parte back-end ed è stato utilizzato per migliorare l'aspetto grafico della Webapp. È responsabile della creazione di immagini profilo attraverso un hash dell'indirizzo email;
- **Psycopg2** : libreria necessaria al funzionamento di SQLAlchemy con il DBMS PostgreSQL;
- **SqlAlchemy** : il cui ORM è stato utilizzato per compiere le query e avere maggiore sicurezza limitando la SQL-Injection.

In riferimento a HTML CSS è stato utilizzato **Bootstrap** per migliorare l'aspetto grafico della web app e facilitare la navigazione degli utenti.

In materia di Javascript sono stati utilizzati:

- **jQuery** : per semplificare l'utilizzo dello stesso;
- **Ajax** : per fare richieste al server in modo più facile e dinamico;
- **Html5-QrCode** : per leggere codici QR ed ottenerne una stringa;
- **Qrious** : per generare codici QR dato un token;
- **Google Charts** : per la creazione dei grafici statistici relativi alla demografica.