Progetto PCTO Ca'Foscari

Pietro Visconti, Diego Passarella, Tudor Croitor

July 1, 2022

Contents

1 Descrizione progetto

La web application permette la gestione dei corsi riguardanti le **attività PCTO** dell'Università.

1.1 Funzionalità principali

1.1.1 Registrazione ed accesso alla web application

All'interno dell'applicazione è possibile registrarsi come **studente** o come **professore**. Nella procedura di registrazione vengono richiesti Nome, Cognome, Data di nascita, Email, Password (e conferma password) ed infine la tipologia di utente.

Al termine della registrazione, tuttavia, sarà possibile accedere alla propria area riservata solo dopo aver aperto il **link di attivazione** (spiegato meglio in seguito) che verrà inviato alla email inserita durante la procedura. Questa è una forma di sicurezza che viene adottata con lo scopo di prevenire che l'utente possa registrarsi con una email a cui egli non ha accesso. Una volta attivato l'account, si potrà accedere alla web application tramite il form di login, inserendoci email e password.

Come anticipato, una volta premuto il tasto "Registrati" viene automaticamente inviata una email all'indirizzo specificato nel form, a cui viene passato come parametro un **token alfanumerico univoco**. Tale token viene poi associato all'utente nella tabella tokens. Nel frattempo, viene inserito l'utente nella tabella users, nella quale viene salvato un flag **"$\mathbf{is_{active}}$"** per verificare che l'utente abbia attivato il proprio account. Fintanto che il flag rimane a False, l'utente non può accedere alla web application. Subito dopo che l'utente ha cliccato il link di attivazione, viene verificato che il token presente nei parametri del link e quello già salvato nel database corrispondano: in caso positivo, il flag "is_{active}" viene impostato a True e l'utente potrà accedere.

Nella procedura di registrazione dei professori esiste un controllo addizionale: viene verificato che il **dominio dell'email** inserita nel form sia "unive.it" per accertarsi che quantomeno l'utente faccia parte dell'Università.

Tutti gli utenti possono eventualmente effettuare il **logout** dalla web application.

1.1.2 Accesso come Professore

Se l'utente effettua l'accesso in qualità di professore avrà a diposizione:

- Sezione **Profilo**, in cui verranno visualizzate tutte le informazioni personali.
- Sezione **Gestione Corsi**, una dashboard in cui sono esposti tutti i corsi creati dall'utente e visualizzarne le specifiche: Titolo, Categoria, Data di creazione, Numero di iscrizioni massimo e la sezione "Azioni", spiegata in seguito.
- Sezione **Crea nuovo corso**, nella quale è possibile appunto creare un nuovo corso, inserendo queste caratteristiche: Titolo, Descrizione, Massimo numero di partecipanti, Minimo numero di partecipanti, Minimo numero di lezioni, Durata delle lezioni, Categoria.

- Per ogni corso, sezione **Azioni**: sono presenti dei bottoni per accedere alle sezioni Lezioni, Informazioni Corso, Modifica Corso.
- Sezione **Lezioni**, in cui l'utente può aggiungere nuove lezioni relative al corso che sta modificando. Per ogni lezione sono richiesti Descrizione, Data, Orario, Modalità, Edificio, Aula.
- Sezione **Informazioni Corso**, in cui viene presentata un'overview del corso in questione: sono presenti la descrizione, le lezioni presenti (con possibilità di aggiungerne), la **mini-mappa** con il marker che indica il luogo di svolgimento della lezione più vicina temporalmente e la lista di studenti partecipanti.
- Sezione **Modifica Corso**, in cui è possibile modificare in tempo reale (senza dover ricaricare la pagina) le informazioni relative al corso in questione.

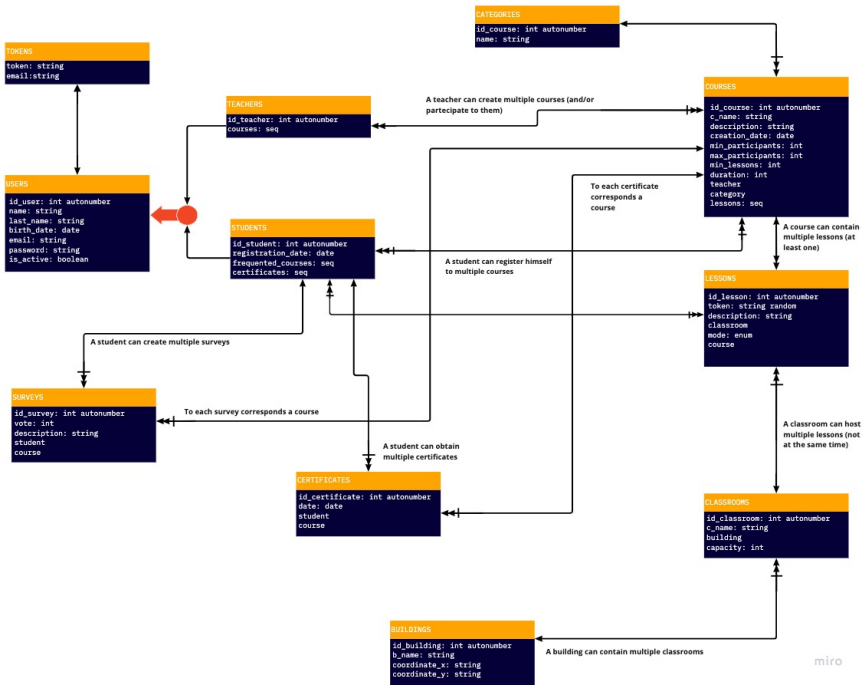
1.1.3 Accesso come Studente

Se l'utente effettua l'accesso in qualità di studente, avrà a disposizione:

- Sezione **Tutti i corsi**, in cui viene esposta una lista di tutti i corsi creati dai professori suddivisi in categorie. Per ogni corso, lo studente può visualizzarne i dettagli, tra cui Titolo, Descrizione, Creatore, Data di creazione, Numero di posti e il Numero di posti disponibili.
- Sezione **I miei corsi**, nella quale vengono mostrati i corsi ai quali lo studente è iscritto. Viene data la possibilità di cancellare la propria iscrizione e di visualizzare i dettagli relativi al corso (vedi Informazioni Corso).

1.2 Progettazione

1.2.1 Schema concettuale



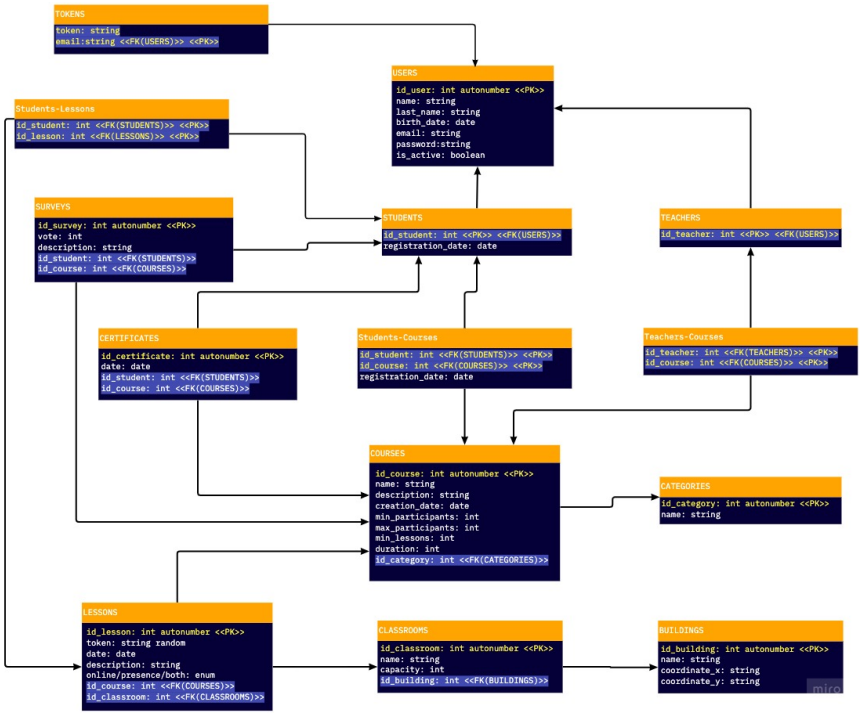
NOTA: Attualmente le funzionalità riguardanti le tabelle **Surveys** e **Certificates** non sono ancora state implementate.

Nello schema concettuale qui mostrato, si evidenziano le seguenti relazioni:

1. Gli utenti si suddividono in Studenti e Professori.
2. Ad ogni utente viene associato un token (nella tabella tokens).
3. Ogni Professore può creare molteplici corsi.
4. Ogni Studente può partecipare a molteplici corsi.
5. Ad ogni corso corrisponde una categoria.
6. Ad ogni corso sono associate molteplici lezioni.
7. Uno studente può partecipare a più lezioni.

8. Uno studente può partecipare a molteplici sondaggi.
9. Uno studente può possedere molteplici certificati di partecipazione.
10. A ogni lezione viene associata un'aula, e all'aula un edificio

1.2.2 Schema logico



1.3 Scelte progettuali

1.3.1 Politiche di integrità dei dati

Per prevenire eventuali incongruenze durante l'inserimento/aggiornamento di nuovi dati all'interno delle tabelle, sono stati effettuati dei controlli tramite i **check constraints** su attributo o tupla.

Alcuni esempi:

```sql
-- tabella corsi
CREATE TABLE courses (
id_course INT AUTO_INCREMENT,
c_name VARCHAR(30) NOT NULL,
description VARCHAR(100),
creation_date DATE NOT NULL,
max_partecipants INT CHECK(max_partecipants > 0),
min_partecipants INT CHECK(min_partecipants > 0),
min_lessons INT NOT NULL CHECK(min_lessons >= 0),
duration INT NOT NULL CHECK(duration > 0),
id_category INT NOT NULL,
PRIMARY KEY(id_course),
FOREIGN KEY(id_category) REFERENCES categories(id_category)
);


-- tabella aule
CREATE TABLE classrooms (
id_classroom INT AUTO_INCREMENT,
c_name VARCHAR(15) NOT NULL,
capacity INT NOT NULL CHECK(capacity > 0),
id_building INT NOT NULL,
PRIMARY KEY(id_classroom),
FOREIGN KEY(id_building) REFERENCES buildings(id_building)
);

-- tabella sondaggi
CREATE TABLE surveys (
id_survey INT AUTO_INCREMENT,
vote INT NOT NULL CHECK(vote >= 0 AND vote <= 5),
description VARCHAR(100),
id_student INT NOT NULL,
id_course INT NOT NULL,
PRIMARY KEY(id_survey),
FOREIGN KEY(id_student) REFERENCES students(id_student),
FOREIGN KEY(id_course) REFERENCES courses(id_course)
);
```

Altri controlli più complessi, invece, sono stati effettuati tramite codice, per renderli più capibili e flessibili.

Esempio in cui viene verificato che l'aula inserita non sia già occupata da un'altra lezione nello stesso orario:

```
87    # inserimento di una lezione
88    if single_lesson.validate_on_submit():
89
90        #Verifica che non esista un'altra lezione nella stessa aula e nello stesso orario
91        if not check_lesson_availability(lesson_base, single_lesson):
92
93            flash("L' aula inserita è già prenotata", "danger")
94            return render_template('teachers/new_lesson.html', id_course=id_course, form_base=lesson_base
95
96        #In caso di slot libero, la lezione viene inserita
97        insert_lesson(lesson_base, single_lesson, id_course)
98
99        return redirect(url_for('main.lessons', id_course=id_course, path='teacher'))
```

1.3.2 SQLAlchemy ORM

E' stato scelto di adottare la tecnologia **Object-Relational-Mapping** in quanto permette di incapsulare le tabelle e le rispettive relazioni in classi, rendendo il codice e le query facili da scrivere e da capire. Un altro aspetto importante da sottolineare è che SQLAlchemy ORM garantisce la **protezione da SQL-Injection**.

Anche le relazioni tra le classi ORM sono state affidate completamente a SqlAlchemy: in questo modo ogni modifica fatta allo schema del database viene riflessa immediatamente nelle classi python. Questo garantisce protezione da eventuali bug dovuti ad una modifica di attributi/relazioni, **rendendo sicuro ogni cambiamento** allo schema relazionale.

Esempio:

```
19
20 class Student(db.Model):
21     __table__ = db.Model.metadata.tables['pcto_db.students']
22
```

1.4 Scelte tecnologiche

1.4.1 JQuery?

1.4.2 Folium?