

## 1. Introduzione:

Il tema scelto è il numero 1 proposto per l' anno 2022, ovvero la **gestione delle attività di orientamento**.

Il progetto implementa la gestione delle attività all' interno di un ambiente universitario, permette a professori, studenti di avere un proprio account, un' area personale dove prenotare, aggiungere corsi, lezioni, visualizzare statistiche.

Alla registrazione ogni utente è considerato uno studente, i ruoli possono essere modificati solo dall' admin.

Le informazioni vengono registrate su database PostgreSQL, la password viene criptata per migliorare la sicurezza.

L' applicazione è dotata di un template base (base.html) e ogni pagina ha il suo contenuto.

## 2. Funzionalità principali:

### Struttura delle pagine:

Il sito si presenta strutturato: header, contenente il nome del sito, della menu navbar per la navigazione all'interno del sito, una parte centrale con il contenuto di ogni pagina, ed un footer con dei link ad altri siti unive.

### Funzionalità:

L'applicazione fornisce una serie di funzionalità in base al ruolo dell'utente che visita il sito.

Alcune funzionalità sono condivise come ad esempio ogni utente può visionare la homepage del sito.

### Registrazione:

Un utente si può registrare accedendo alla pagina "Registrati" in cui vengono chieste una serie di informazioni:

1. nome -> con `input type="text"`
2. cognome -> con `input type="text"`
3. codice fiscale -> con `input type="text"`
4. età -> con `input type="number"`
5. numero di cellulare -> con `input type="text"`
6. sesso -> con `input type="radio"`
7. password e conferma password -> `<input type = "text"`

Ogni informazione è settata come required sull'HTML per essere sicuri che l'utente non si dimentichi alcuni dati, la password possiede controlli per assicurarsi che sia almeno di 8 caratteri e contenga un numero, almeno un carattere in minuscolo e maiuscolo.

La password viene salvata dopo essere stata criptata, inoltre l'utente deve inserire la password 2 volte per conferma.

Vi è inoltre un controllo sul formato della mail.

Vengono creati allo start dell'applicazione alcuni utenti di default:

- Mario, studente, email: emailMario
- Maria, studente, email: emailMaria
- Stefano, professore, email: emailCALZAVARA
- Admin, admin, email: admin

Ogni utente possiede come password "admin" per semplificare il debugging.

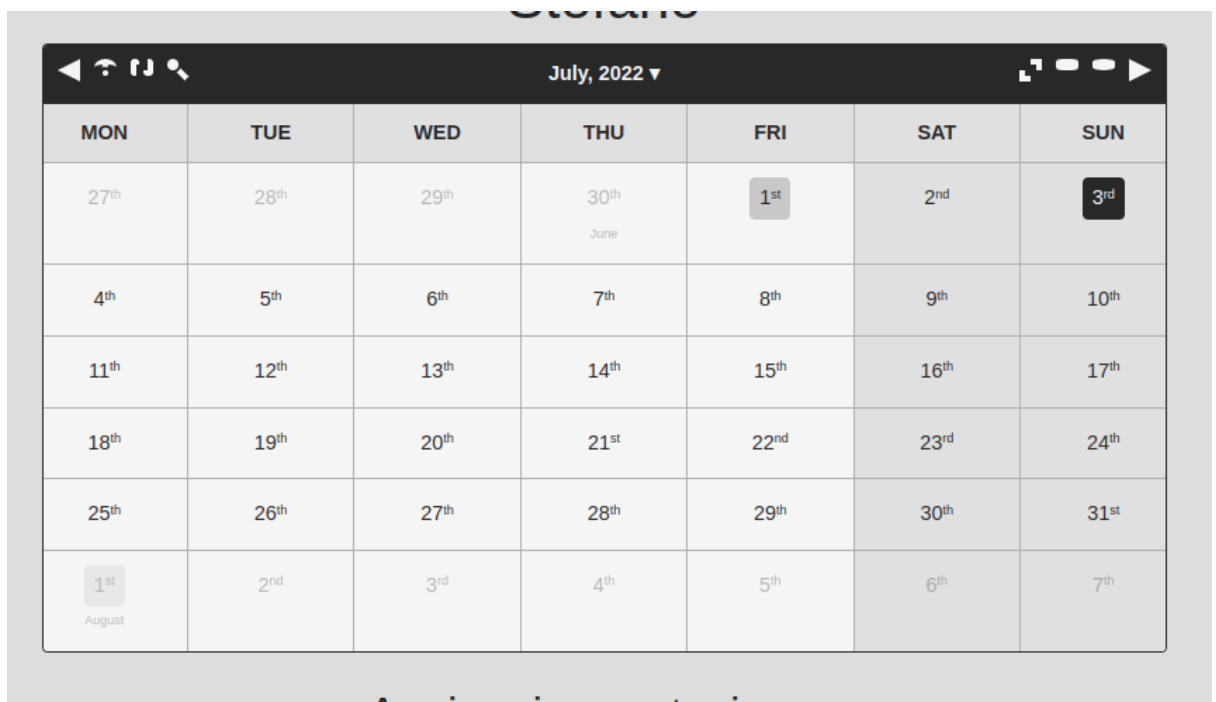
### Login:

Un utente registrato può effettuare il login per entrare nella sua area riservata inserendo: email e password.

### Area Riservata:

L'area riservata si presenta differente in base al ruolo dell'utente, le diverse pagine per accedere alle funzionalità sono accessibili dal menu navbar.

Ogni utente possiede nella pagina principale dell'area riservata il calendario con le lezioni nelle varie aule:



MON	TUE	WED	THU	FRI	SAT	SUN
27 <sup>th</sup>	28 <sup>th</sup>	29 <sup>th</sup>	30 <sup>th</sup> June	1 <sup>st</sup>	2 <sup>nd</sup>	3 <sup>rd</sup>
4 <sup>th</sup>	5 <sup>th</sup>	6 <sup>th</sup>	7 <sup>th</sup>	8 <sup>th</sup>	9 <sup>th</sup>	10 <sup>th</sup>
11 <sup>th</sup>	12 <sup>th</sup>	13 <sup>th</sup>	14 <sup>th</sup>	15 <sup>th</sup>	16 <sup>th</sup>	17 <sup>th</sup>
18 <sup>th</sup>	19 <sup>th</sup>	20 <sup>th</sup>	21 <sup>st</sup>	22 <sup>nd</sup>	23 <sup>rd</sup>	24 <sup>th</sup>
25 <sup>th</sup>	26 <sup>th</sup>	27 <sup>th</sup>	28 <sup>th</sup>	29 <sup>th</sup>	30 <sup>th</sup>	31 <sup>st</sup>
1 <sup>st</sup> August	2 <sup>nd</sup>	3 <sup>rd</sup>	4 <sup>th</sup>	5 <sup>th</sup>	6 <sup>th</sup>	7 <sup>th</sup>

Inoltre ogni utente può:

- effettuare logout
- tornare alla home

### **Prof:**

Home crea corso prenota lezione calendario lezioni Statistiche Indietro LOGOUT

Un professore può:

- Creare un corso: tramite un menù a tendina inserendo le informazioni del corso da creare.
- Prenotare un aula: il prof può attraverso il calendario verificare quale aula sia occupata e prenotare un'aula non occupata per la lezione, l'applicazione controlla la disponibilità.
- Visionare Statistiche: la pagina fornisce una serie di informazioni sui: corsi, lezioni e demografia degli iscritti ai corsi, come numero totale iscritti ed età media

### **Studente:**

Home Lezioni prenotate iscrizione corsi Indietro LOGOUT

Uno studente può:

- Prenotare un posto ad una lezione tramite un menù a tendina sulla sua pagina di home area riservata.
- Iscrivere ad un corso inserendo il codice del corso.
- Visionare corsi in cui è iscritto sulla pagina iscrizione corsi.
- Vedere in quali lezioni ha prenotato un posto.

### **Admin:**

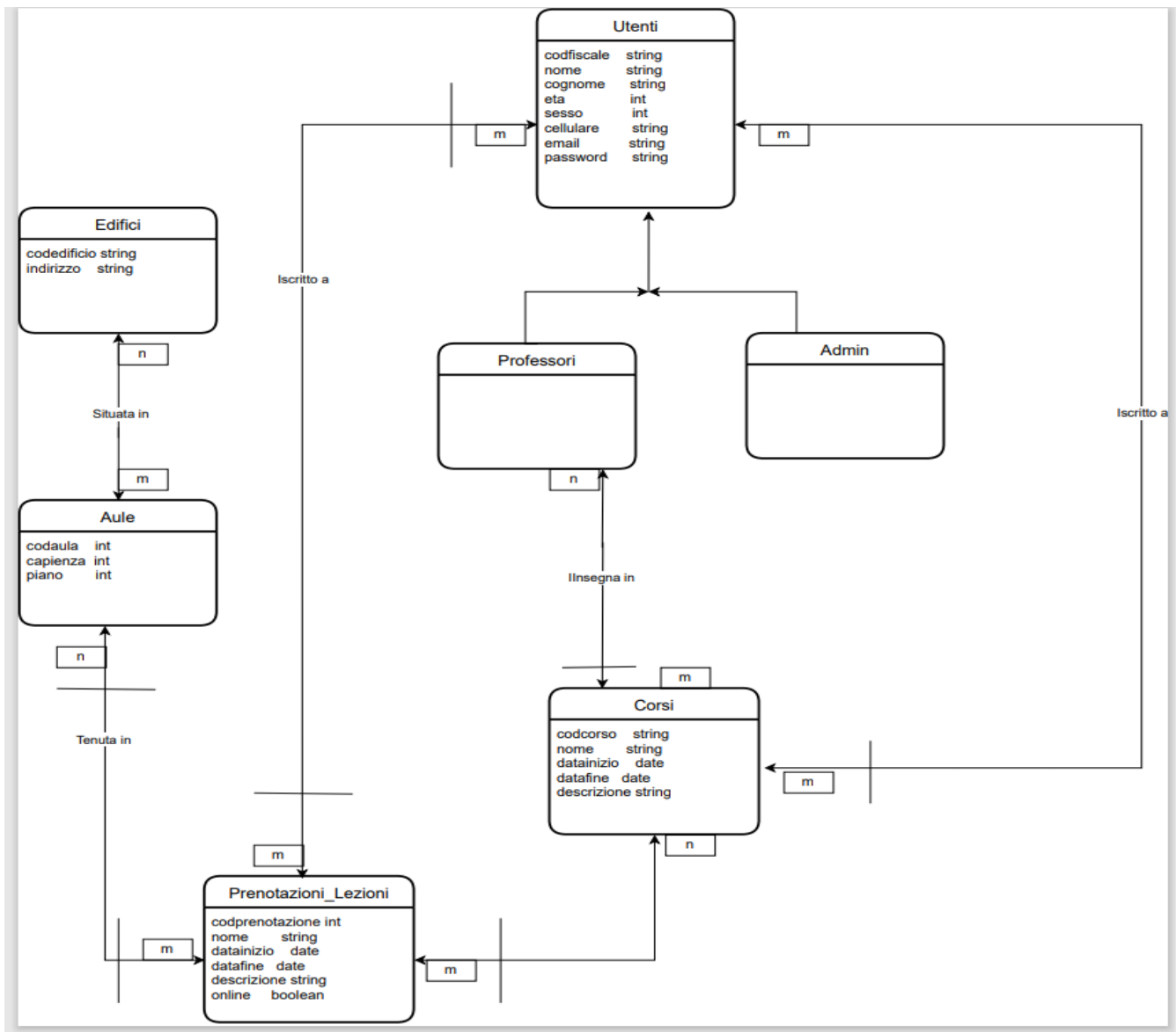
tipi account Prenota aula per lezioni Edifici Aule Corsi Indietro LOGOUT

Un Admin può:

- Visionare tutti gli utenti e modificare il ruolo di ognuno.
- Aggiungere: un edificio, un'aula, creare corsi inserendo info del corso e il professore responsabile.

### 3. Progettazione concettuale e logica:

#### a. Modello a oggetti:



#### Descrizione:

Gli utenti presenti nel database si dividono in 3 categorie: Utenti, Professori, admin.

Un professore può insegnare in uno o più corsi. I corsi hanno una data di inizio e fine. Le prenotazioni delle lezioni vengono effettuate da professore e la lezione quindi è tenuta da quest' ultimo, la prenotazione è legata ad un corso, infatti un corso può avere più lezioni , inoltre una lezione viene svolta in un'aula (che si trova in un determinato edificio) oppure si svolge in via telematica. La prenotazione della lezione può essere effettuata da più studenti(utenti) e gli studenti possono prenotarsi a più lezioni.

```

classDiagram
    class Edifici {
        codedificio string <pk>
        indirizzo string
    }
    class Aule {
        codaula int <pk>
        capienza int
        edificio string <fk>
        piano int
    }
    class Utenti {
        codfiscale string <pk>
        nome string
        cognome string
        eta int
        sesso int
        cellulare string
        email string
        password string
        ruolo enum(admin, professore, utente)
    }
    class Prenotazioni_utenti {
        lezione int <pk> <fk>
        utente string <pk> <fk>
    }
    class Prenotazioni_Lezioni {
        codprenotazione int <pk>
        nome string
        datainizio date
        datafine date
        descrizione string
        online boolean
        corso string <fk>
        aula int <fk>
    }
    class Corsi {
        codcorso string <pk>
        nome string
        datainizio date
        datafine date
        descrizione string
        professore string <fk>
    }
    class Iscrizione_corsi {
        corso string <pk> <fk>
        studente string <pk> <fk>
    }

    Edifici "1" -- "N" Aule : edificio
    Aule "1" -- "N" Prenotazioni_Lezioni : aula
    Utenti "1" -- "N" Prenotazioni_utenti : utente
    Utenti "1" -- "N" Corsi : professore
    Prenotazioni_utenti "1" -- "N" Prenotazioni_Lezioni : lezione
    Prenotazioni_Lezioni "1" -- "N" Corsi : corso
    Iscrizione_corsi "1" -- "N" Utenti : Studente
    Iscrizione_corsi "1" -- "N" Corsi : corso

```

La tabella Utenti comprende un discriminatore di tipo enum(utente,professore,admin) per la gestione degli account e relativi privilegi.

Prenotazioni utenti che è la tabella che permette di registrare l'iscrizione di un utente a più lezioni e le lezioni hanno più utenti (le chiavi primarie sono anche le chiavi esterne)

Iscrizione corsi permette allo stesso modo di registrare l'iscrizione degli utenti ai vari corsi e tenere traccia di tutti gli utenti iscritti a quel determinato corso.

#### 4. Alcune query:

Le query si trovano nel file python "functions.py" nelle varie funzioni. Di seguito un elenco di alcune query del progetto:

```
SELECT pl.corso,pl.aula,pl.datainizio,pl.durata, pl.maxpresenti, pl.online, COUNT(pp.utente) AS prenotati
FROM Prenotazioni_posti pp RIGHT JOIN Prenotazioni_lezioni pl
ON pp.lezione = pl.codprenotazione
GROUP BY pp.lezione, pl.corso, pl.aula, pl.datainizio, pl.durata, pl.maxpresenti, pl.online
```

##### 1. Query funzione get\_prenotati():

###### **Descrizione:**

la query restituisce sulla pagina statistiche professore ogni prenotazione lezione con i vari attributi , in più nella tabella si visualizza il conto delle persone che si sono prenotate per quella determinata lezione e la capienza massima della lezione.

N.B se non ci sono iscritti la RIGHT JOIN restituisce lo stesso la lezione con il conto prenotati a 0.

##### 2. Query funzione jcorsiprenotazioni():

```
SELECT c.codcorso,c.nome,u.nome AS professore,c.datainizio,c.datafine,
COUNT(ic.studente) AS prenotati, AVG(u2.eta) AS media
FROM Iscrizioni_corsi ic RIGHT JOIN Corsi c ON ic.corso = c.codcorso
JOIN Utenti u ON c.professore = u.codfiscale
JOIN Utenti u2 ON ic.studente = u2.codfiscale
GROUP BY c.codcorso,c.nome,u.nome,c.datainizio,c.datafine
```

###### **Descrizione:**

la query restituisce tutti i corsi presenti nel database, con il rispettivo numero iscritti e la media dell' età degli studenti. E' stato necessario fare una doppia JOIN con Utenti per recuperare il nome del professore, la seconda per fare il conto degli iscritti. Se non ci sono iscritti, grazie alla RIGHT JOIN, la query restituisce lo stesso il record del corso.

## 5. TRIGGER e CHECK :

Sono stati implementati dei vincoli check e dei trigger all' interno del progetto, di seguito verranno riportati i principali:

1. Trigger per il controllo se l' aula è libera per la lezione:

```
CREATE OR REPLACE FUNCTION disponibilita_lez()
RETURNS TRIGGER AS $$
BEGIN
    if NEW.aula IN (SELECT pl.aula
                    FROM Prenotazioni_lezioni pl
                    WHERE (NEW.durata >= pl.datainizio AND NEW.datainizio <= pl.durata)) THEN
        RETURN NULL;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER disponibilita_aula
BEFORE INSERT OR UPDATE ON Prenotazioni_lezioni
FOR EACH ROW EXECUTE PROCEDURE disponibilita_lez();
```

### Descrizione:

Nello specifico viene controllato in inserimento o aggiornamento della tabella Prenotazioni\_lezioni se in quella determinata data è già prevista una lezione in quell' aula.

2. Trigger che controlla se i posti disponibili per la lezione sono <= alla capienza massima dell' aula:

```
CREATE OR REPLACE FUNCTION capienza_max()
RETURNS TRIGGER AS $$
BEGIN
    if NEW.maxpresenti > (SELECT a.capienza
                         FROM Aule a
                         WHERE a.codaula = NEW.aula ) THEN
        RETURN NULL;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER posti_aule_prenotazioni
BEFORE INSERT OR UPDATE ON Prenotazioni_lezioni
FOR EACH ROW EXECUTE PROCEDURE capienza_max();
```

4. Trigger che controlla che il codice fiscale del professore che tiene un corso(chi viene inserito o aggiornato) corrisponda effettivamente ad un professore e non ad un altro tipo di utente.

```
CREATE OR REPLACE FUNCTION checkprof()
RETURNS TRIGGER AS $$
BEGIN
    if NEW.professore IN (SELECT u.codfiscale
                          FROM Utenti u
                          WHERE u.ruolo = 'professore') THEN
        RETURN NEW;

    END IF;

    RETURN NULL;

END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER corsoprof
BEFORE INSERT OR UPDATE ON Corsi
FOR EACH ROW EXECUTE PROCEDURE checkprof();
```

5. Vincoli CHECK :

```
CREATE TABLE Prenotazioni_lezioni(
    codprenotazione SERIAL NOT NULL,
    corso varchar(50) NOT NULL,
    aula int ,
    datainizio timestamp NOT NULL,
    durata timestamp NOT NULL,
    maxpresenti int,
    online boolean,
    CHECK (durata > datainizio),
    CHECK (online AND maxpresenti = 0 OR online != True AND maxpresenti > 0 ),
    PRIMARY KEY (codprenotazione),
    FOREIGN KEY (corso) REFERENCES Corsi(codcorso),
    FOREIGN KEY (aula) REFERENCES Aule(codaula)
);
```

#### Descrizione:

Vincoli check nella tabella Prenotazioni lezioni, il primo check ha il compito di controllare che la data inizio della lezione sia prima della data di fine della lezione, lo stesso check viene inserito nella tabella corsi.

Il secondo check invece controlla che se la lezione è online non c'è un massimo di presenti, se la lezione non è online ma in presenza invece è necessario inserire il numero massimo di presenti alla lezione.



## 6. Definizione ruoli:

```
CREATE TYPE role as ENUM('utente', 'professore', 'admin');
```

### Descrizione:

Tipo specifico per la definizione dei ruoli degli utenti, l'admin all'interno della web application ha la possibilità di cambiare il tipo di ruolo a un determinato account, infatti inizialmente dopo la registrazione tutti gli account sono settati a 'utente'.

## 6. Ulteriori informazioni

Abbiamo usato varie librerie per questo progetto, in particolare:

- **bcrypt** che serve a criptare la password in modo che non venga salvata in chiaro sul database.
- **Login Manager**
- Vengono utilizzate funzioni javascript per migliorare l'esperienza utente, soprattutto nel calendario.
- **flash** utilizzati sia per errori che per notifiche tramite popup all'utente
- **controllo email** usato libreria standard(re) per il match dei caratteri
- **controllo password** creato funzione che controlla lunghezza(8), carattere maiuscolo, carattere minuscolo, almeno un numero.

Gruppo cda ploid-curago  
De Battista Daniele 875632  
Panico Lorenzo 872986  
De Col Mattia 875635