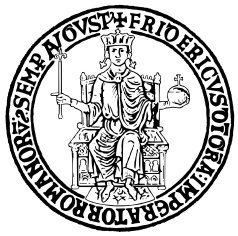
UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE
DELL'INFORMAZIONE



CORSO DI LAUREA IN INFORMATICA

INSEGNAMENTO DI BASI DI DATI - ANNO ACCADEMICO 2021/2022

Sistema di Gestione per Corsi di Formazione

Autori

Matteo Gennaro GIORDANO
N86003551
matte.giordano@studenti.unina.it

Gian Marco ADDATI
N86003795
gi.addati@studenti.unina.it

Docenti

Prof. Adriano PERON

22 gennaio 2022

Indice

Parte I
Introduzione

1 Informazioni generali sulla base di dati

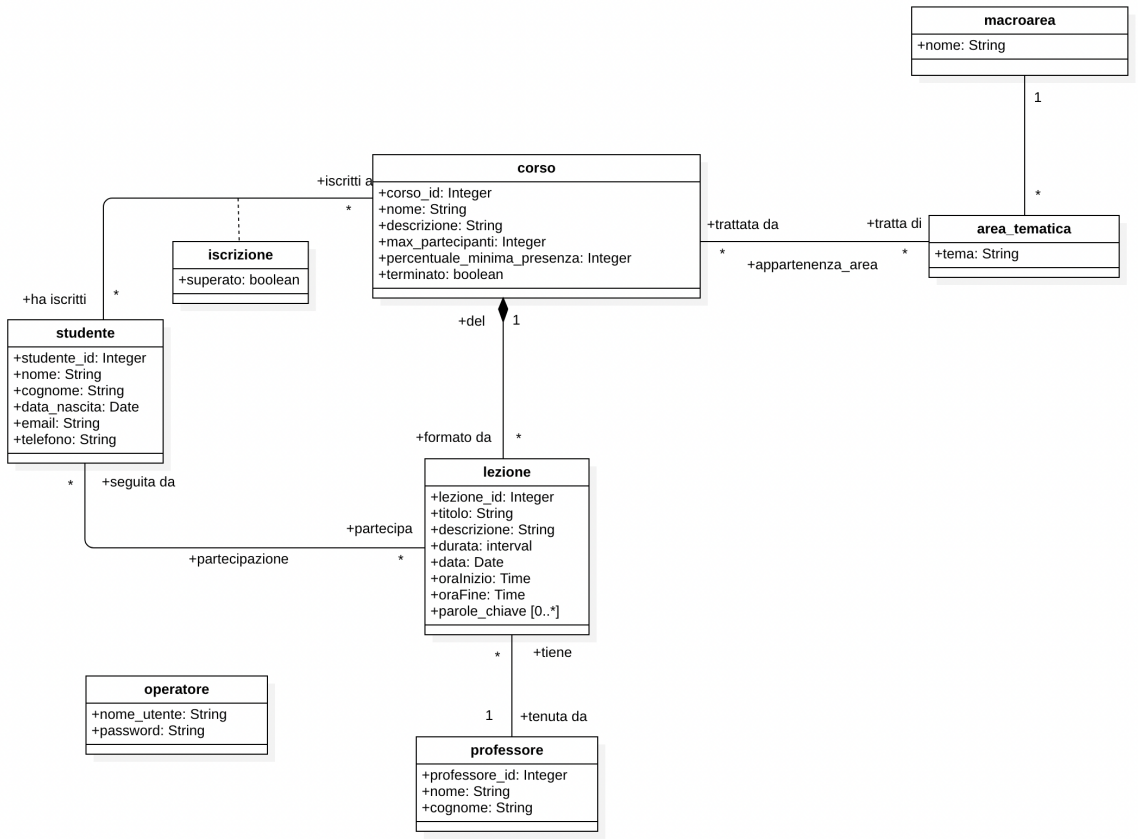
La base di dati relazionale progettata permette all'operatore la gestione dei dati relativi a dei corsi di formazione online.
Tali corsi prevedono un certo numero di lezioni (dalla durata massima di due ore) tenute da professori, a cui partecipano degli studenti (entro un numero massimo di iscrizioni stabilito dall'operatore).
Un corso è caratterizzato da una percentuale minima di presenze richieste per l'idoneità al superamento del corso, verificata al momento della sua terminazione.
Un corso non ha un numero prestabilito di lezioni, difatti può essere terminato in ogni momento dall'operatore.
E' possibile ottenere informazioni statistiche di un corso (ad esempio la percentuale media di presenze o il numero minimo di presente registrate durante il corso), oltre che le lezioni da cui è formato e gli studenti ad esso iscritti.
Ogni corso riguarda una o più aree tematiche, le quali sono contenute in macroaree. Un'area tematica può appartenere a una o più macroaree.
Le lezioni possono avere delle parole chiave associate.

2 Informazioni generali sulla documentazione

Nella presente documentazione è possibile trovare progettazione concettuale e logica della base di dati, oltre che la trascrizione della progettazione fisica.
Seguono poi il popolamento della base di dati e una serie di esempi d'uso dell'applicativo che la adotta.

Parte II
Progettazione Concettuale

3 Class Diagram non Ristrutturato



(Dovesse risultare difficile leggere il class diagram, all'interno della cartella ne è presente una copia in formato più leggibile.)

3.1 Commenti al Class Diagram non Ristrutturato

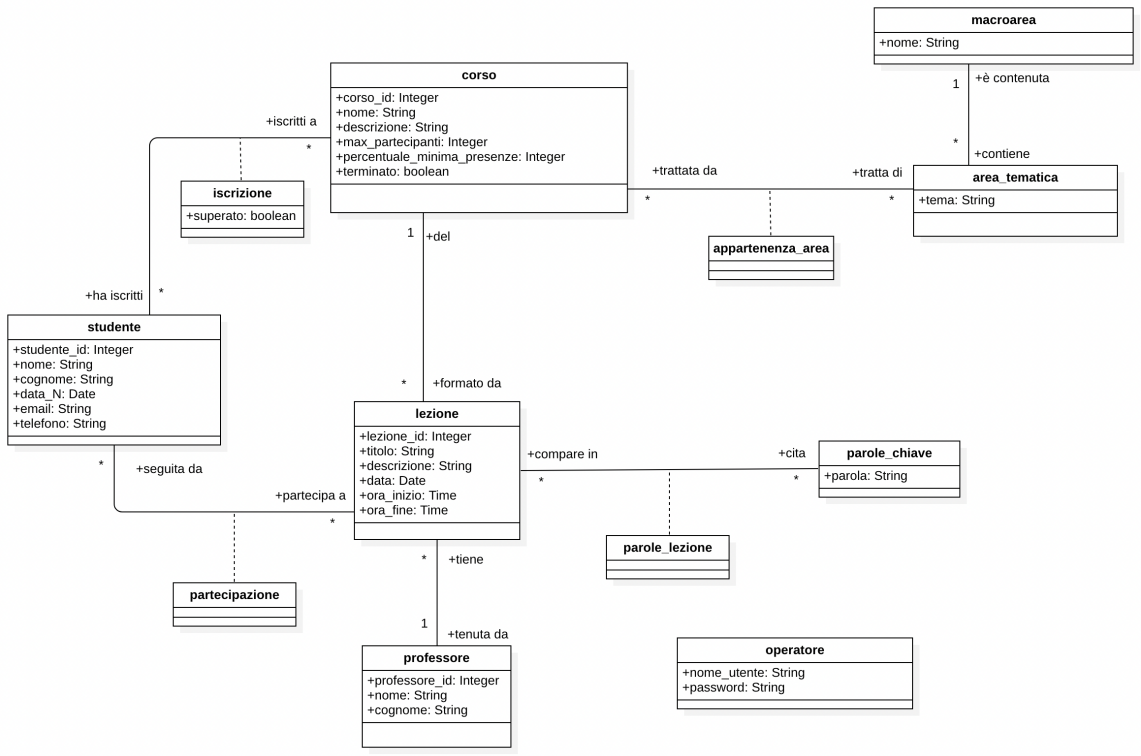
1. Tra corso e lezione vige una relazione di composizione, in quanto la base di dati non consente la presenza di lezioni indipendenti da un corso, ed ogni lezione appartiene ad un solo corso. Nella base di dati ciò è garantito dall'associazione (1,*), che prevede quindi un riferimento obbligatorio al corso di appartenenza in ogni istanza di una lezione. Nell'applicativo sarà inoltre obbligatorio specificare il corso di appartenenza di una lezione al momento della sua creazione.

2. L'associazione iscrizione di tipo (*,*) tra corso e studente vede la presenza di una classe di associazione in quanto, oltre ai riferimenti a tali entità, è stato aggiunto un valore booleano "superato" che specifica se lo studente è idoneo o meno al superamento del corso. Questo valore verrà aggiornato al momento della terminazione del corso se lo studente rispetta il numero minimo di presenze.

3. Per quanto riguarda l'associazione professore-lezione, la base di dati prevede che un professore tenga una lezione; di conseguenza un corso può avere lezioni tenute da più professori diversi. Inoltre è possibile consultare una vista "insegnamenti" apposita che mostra in quali corsi un dato professore ha tenuto almeno una lezione.

4. L'entità operatore risulta non in associazione con alcuna entità in quanto contiene solamente istanze rappresentanti gli utenti che hanno accesso al database. Nell'applicativo saranno controllate tali credenziali in fase di login.

4 Class Diagram Ristrutturato



(Dovesse risultare difficile leggere il class diagram, all'interno della cartella ne è presente una copia in formato più leggibile.)

4.1 Commenti alla Ristrutturazione del Class Diagram

1. Nel class diagram non ristrutturato, l'entità lezione presentava, tra gli attributi, "durata", che qui non compare essendo un attributo calcolato a partire da "ora_inizio" e "ora_fine".

2. Inoltre, l'entità lezione presentava un attributo multiplo: "parole_chiave". Nella ristrutturazione si è scelto di implementarlo tramite un'associazione (*,*) con un'entità parole_chiave.

3. La composizione presente nel class diagram non ristrutturato è stata sostituita da una semplice associazione (1,*), più simile all'implementazione fisica.

5 Dizionari

5.1 Dizionario delle Classi

CLASSE	DESCRIZIONE	ATTRIBUTI
Corso	Contiene informazioni relative ad un corso.	**corso_id** (Integer): PK che identifica univocamente un corso. **nome** (String): Nome del corso. **descrizione** (String): Breve descrizione del contenuto del corso. **max_partecipanti** (Integer): Numero massimo di partecipanti al corso. **percentuale_minima_presenze** (Integer): Percentuale minima di partecipazioni per uno studente alle lezioni di un corso per poterlo superare. **terminato** (Boolean): Stato di un corso: *true* se è terminato, *false* altrimenti.
Studente	Contiene le informazioni personali per ogni studente.	**studente_id** (Integer): PK che identifica univocamente uno studente. **nome** (String): Nome dello studente. **cognome** (String): Cognome dello studente. **data_nascita** (Date): Data di nascita dello studente. **email** (String): Email di riferimento dello studente. **telefono** (String): Numero di telefono dello studente.
Professore	Contiene le informazioni personali per ogni professore.	**professore_id** (Integer): PK che identifica univocamente un professore. **nome** (String): Nome del professore. **cognome** (String): Cognome del professore.
Lezione	Contiene le caratteristiche di una lezione.	**lezione_id** (Integer): PK che identifica univocamente una lezione. **titolo** (String): Titolo della lezione. **descrizione** (String): Breve descrizione del contenuto della lezione. **data** (Date): Data in cui si tiene la lezione. **ora_inizio** (Time): Ora a cui inizia la lezione. **ora_fine** (Time): Ora a cui finisce la lezione. **professore** (Integer): Riferimento all' id del professore che tiene la lezione. **del_corso** (Integer): Riferimento all' id del corso in cui si tiene la lezione.
Parole_Chiave	Contiene le parole chiave associabili ad una lezione.	**parola** (String): Nome della parola.
Macroarea	Contiene i nomi delle macroaree in cui possono essere organizzate le aree tematiche.	**nome** (String): Nome della macroarea.
Area_Tematica	Contiene i nomi delle aree tematiche in cui si organizzano i corsi.	**tema** (String): Nome dell' area tematica. **nome_macroarea** (String): Nome della macroarea a cui appartiene l'area tematica.
Iscrizione	Tiene traccia delle iscrizioni effettuate dagli studenti ai corsi.	**superato** (Boolean): Stato dell' iscrizione: *true* che lo studente è idoneo al superamento del corso, *false* indica il contrario. **studente_iscritto** (Integer): Riferimento all' id dello studente iscritto al corso. **corso_scelto** (Integer): Riferimento all' id del corso a cui lo studente si è iscritto.
Partecipazione	Tiene traccia delle iscrizioni effettuate dagli studenti alle lezioni di un dato corso.	**lezione_frequentata** (Integer): Riferimento all' id della lezione a cui partecipa lo studente. **studente_partecipante** (Integer): Riferimento all' id dello studente che partecipa alla lezione.
Parole_Lezione	Tiene traccia delle parole chiave che compaiono in una data lezione.	**parola** (String): Riferimento al nome della parola chiave che compare nella lezione. **lezione** (Integer): Riferimento all' id della lezione in cui compare la parola chiave.
Appartenenza_Area	Tiene traccia dell'appartenenza di un corso alle aree tematiche.	**corso** (Integer): Riferimento all' id del corso che appartiene all'area tematica. **tema_trattato** (String): Riferimento al nome dell' area tematica trattata dal corso.
Operatore	Contiene le credenziali di tutti gli operatori adibiti all'utilizzo della base di dati.	**nome_utente** (String): PK e username che identifica univocamente un operatore. **password** (String): Password per permettere l'accesso dell'operatore.

5.2 Dizionario delle Associazioni

ASSOCIAZIONE	DESCRIZIONE	CLASSI COINVOLTE
Iscrizione	Indica l'avvenuta iscrizione di uno studente ad uno o più corsi.	**Corso** [*] ruolo **ha iscritti**: indica gli studenti iscritti a quel corso. **Studente** [*] ruolo **iscritti a**: indica a quali corsi uno studente è iscritto.
Appartenenza_Area	Indica l'appartenenza di un corso ad una o più aree tematiche.	**Corso** [*] ruolo **tratta di**: indica le aree tematiche che il corso affronta. **Area_Tematica** [*] ruolo **trattata da**: indica da quali corsi un'area è trattata.
Partecipazione	Indica l'avvenuta partecipazione di uno studente ad una o più lezioni.	**Studente** [*] ruolo **partecipa a**: indica a quali lezioni lo studente partecipa. **Lezione** [*] ruolo **seguita da**: indica gli studenti che seguono la lezione.
Parole_Lezione	Indica l'appartenenza delle parole chiave alle lezioni.	**Lezione** [*] ruolo **cita**: indica le parole chiave che sono citate durante la lezione. **Parole_Chiave** [*] ruolo **compare in**: indica in quali lezioni compare la parola.
Composizione Corso	Indica la composizione di un corso. Ogni corso ha 0..* lezioni, e non ci sono lezioni non di un corso.	**Corso** [*] ruolo **formato da**: indica le lezioni che compongno il corso. **Lezione** [1] ruolo **del**: indica a quale corso appartiene la lezione.
Raggruppamento Aree	Indica per ogni macroarea quali aree tematiche comprende.	**Macroarea** [*] ruolo **contiene**: indica quali aree tematiche ricadono sotto la macroarea. **Area_Tematica** [1] ruolo **è contenuta**: indica sotto quale macroarea l'area tematica si trova.
Insegnamento	Indica l'assegnazione delle lezioni ai professori.	**Professore** [*] ruolo **tiene**: indica quali lezioni sono tenute dal professore. **Lezione** [1] ruolo **è tenuta**: indica quale professore tiene la lezione.

5.3 Dizionario dei Vincoli

NOME	TIPO	DESCRIZIONE
numero_partecipanti_maggiore_di_0	Intrarelazionale	Un corso deve avere almeno un partecipante
iscrizione_unique	Intrarelazionale	Uno studente non può iscriversi due volte allo stesso corso
parole_lezione_unique	Intrarelazionale	Una parola chiave non può essere associata due volte alla stessa lezione
area_tematica_corso_unique	Intrarelazionale	Un'area tematica non può essere associata due volte allo stesso corso
lezione_studente_unique	Intrarelazionale	Uno studente non può partecipare due volte alla stessa lezione
no_orario_illegale_lezioni	Intrarelazionale	Una lezione non può avere un orario di fine precedente a quello di inizio
solo_orari_consentiti_lezioni	Intrarelazionale	Una lezione non può iniziare prima delle 8:00 e non può terminare dopo le 20:00
no_lezioni_più_di_2ore	Intrarelazionale	Una lezione non può durare più di 2 ore
no_lezioni_corso_terminato	Interrelazionale	Ad un corso terminato non possono essere associate delle lezioni
no_iscrizioni_corso_terminato	Interrelazionale	Non è possibile iscrivere uno studente ad un corso terminato
no_partecipazioni_lezione_corso_terminato	Interrelazionale	Non è possibile registrare partecipazioni ad una lezione di un corso terminato
no_partecipazioni_non_iscritti_corso	Interrelazionale	Non è possibile registrare partecipazioni ad una lezione di un corso a cui non si è iscritti
no_iscrizioni_corso_pieno	Interrelazionale	Non è possibile iscrivere uno studente ad un corso che ha raggiunto il numero massimo di iscritti
no_lezioni_contemporanee_stesso_corso	Interrelazionale	Un corso non può avere lezioni che si svolgono nello stesso arco di tempo
no_partecipazioni_contemporanee	Interrelazionale	Uno studente non può partecipare a lezioni di corsi diversi che si svolgono nello stesso arco di tempo
no_decremento_num_max_partecipanti	Interrelazionale	Un corso, quando viene modificato, non può avere un nuovo max_partecipanti minore del numero di iscrizioni già effettuate al corso stesso.

Parte III
Progettazione Logica

6 Schema Logico

6.1 Tabelle

- CORSO (**corso_id**, nome, descrizione, max_partecipanti, percentuale_minima_presenze, terminato)
- STUDENTE (**studente_id**, nome, cognome, data_nascita, email, telefono)
- PROFESSORE (**professore_id**, nome, cognome)
- LEZIONE (**lezione_id**, titolo, descrizione, data, ora_inizio, ora_fine, professore, del_corso)
- PAROLE_CHIAVE (**parola**)
- MACROAREA (**nome**)
- AREA_TEMATICA (**tema**, nome_macroarea)
- ISCRIZIONE (superato, studente_iscritto , corso_scelto)
- PARTECIPAZIONE (lezione_frequentata , studente_partecipante)
- PAROLE_LEZIONE (parola , lezione)
- APPARTENENZA_AREA (corso , tema_trattato)
- OPERATORE (**nome_utente**, password)

6.2 Riferimenti

- LEZIONE
  - professore $\longmapsto$ professore.professore_id
  - del_corso $\longmapsto$ corso.corso_id
- AREA_TEMATICA
  - nome_macroarea $\longmapsto$ macroarea.nome
- ISCRIZIONE
  - studente_iscritto $\longmapsto$ studente.studente_id
  - corso_scelto $\longmapsto$ corso.corso_id
- PARTECIPAZIONE
  - lezione_frequentata $\longmapsto$ lezione.lezione_id
  - studente_partecipante $\longmapsto$ studente.studente_id
- PAROLE_LEZIONE
  - parola $\longmapsto$ parole_chiave.parola
  - lezione $\longmapsto$ lezione.lezione_id
- APPARTENENZA_AREA
  - corso $\longmapsto$ corso.corso_id
  - tema_trattato $\longmapsto$ area_tematica.tema

Parte IV
Progettazione Fisica

7 Definizione delle Tabelle

7.1 CORSO

```
CREATE TABLE IF NOT EXISTS corso(
        corso_id INTEGER PRIMARY KEY,
        nome VARCHAR(50) NOT NULL UNIQUE,
        descrizione VARCHAR(500) NOT NULL,
        max_partecipanti INTEGER NOT NULL,
        percentuale_minima_presenze INTEGER NOT NULL,
        terminato BOOLEAN);

```

7.2 STUDENTE

```
CREATE TABLE IF NOT EXISTS studente(
        studente_id INTEGER PRIMARY KEY,
        nome VARCHAR(25) NOT NULL,
        cognome VARCHAR(25) NOT NULL,
        data_nascita DATE NOT NULL,
        email VARCHAR(50) UNIQUE NOT NULL,
        telefono VARCHAR(25) UNIQUE);
```

7.3 PROFESSORE

```sql
CREATE TABLE IF NOT EXISTS professore(
                professore_id INTEGER PRIMARY KEY,
                nome VARCHAR(50) NOT NULL,
                cognome VARCHAR(50) NOT NULL);
```

7.4 LEZIONE

```sql
CREATE TABLE IF NOT EXISTS lezione(
                lezione_id INTEGER PRIMARY KEY,
                titolo VARCHAR(50) UNIQUE NOT NULL,
                descrizione VARCHAR(500),
                data DATE NOT NULL,
                ora_inizio TIME NOT NULL,
                ora_fine TIME NOT NULL,
                professore INTEGER NOT NULL,
                del_corso INTEGER NOT NULL);


ALTER TABLE lezione
        ADD CONSTRAINT lezione_fk1 FOREIGN KEY (professore)
        REFERENCES professore(professore_id) ON DELETE CASCADE ON UPDATE CASCADE,
        ADD CONSTRAINT lezione_fk2 FOREIGN KEY (del_corso)
        REFERENCES corso(corso_id) ON DELETE CASCADE ON UPDATE CASCADE;
```

7.5 PAROLE_CHIAVE

```
CREATE TABLE IF NOT EXISTS parole_chiave(
        parola VARCHAR(25) PRIMARY KEY);
```

7.6 MACROAREA

```
CREATE TABLE IF NOT EXISTS macroarea(
        nome VARCHAR(50) PRIMARY KEY);
```

7.7 AREA_TEMATICA

```
CREATE TABLE IF NOT EXISTS area_tematica(
        tema VARCHAR(100) NOT NULL PRIMARY KEY,
        nome_macroarea VARCHAR(50) NOT NULL);


ALTER TABLE area_tematica
        ADD CONSTRAINT area_tematica_fk1 FOREIGN KEY (nome_macroarea)
        REFERENCES macroarea(nome) ON DELETE CASCADE ON UPDATE CASCADE;
```

7.8 ISCRIZIONE

```
CREATE TABLE iscrizione(
        superato BOOLEAN DEFAULT FALSE,
        studente_iscritto INTEGER NOT NULL,
        corso_scelto INTEGER NOT NULL);


ALTER TABLE iscrizione
    ADD CONSTRAINT iscrizione_fk1 FOREIGN KEY (studente_iscritto)
    REFERENCES studente(studente_id) ON DELETE CASCADE ON UPDATE CASCADE,

    ADD CONSTRAINT iscrizione_fk2 FOREIGN KEY (corso_scelto)
    REFERENCES corso(corso_id) ON DELETE CASCADE ON UPDATE CASCADE,
```

7.9 PARTECIPAZIONE

```
CREATE TABLE IF NOT EXISTS partecipazione(
        lezione_frequentata INTEGER NOT NULL,
        studente_partecipante INTEGER NOT NULL);


ALTER TABLE partecipazione
        ADD CONSTRAINT partecipazione_fk1 FOREIGN KEY (lezione_frequentata)
        REFERENCES lezione(lezione_id) ON DELETE CASCADE ON UPDATE CASCADE,
        ADD CONSTRAINT partecipazione_fk2 FOREIGN KEY (studente_partecipante)
        REFERENCES studente(studente_id) ON DELETE CASCADE ON UPDATE CASCADE;
```

7.10 PAROLE_LEZIONE

```
CREATE TABLE IF NOT EXISTS parole_lezione(
        parola VARCHAR(25) NOT NULL,
        lezione INTEGER NOT NULL);


ALTER TABLE parole_lezione
        ADD CONSTRAINT parole_lezione_fk1 FOREIGN KEY (parola)
        REFERENCES parole_chiave(parola) ON DELETE CASCADE ON UPDATE CASCADE,

        ADD CONSTRAINT parole_lezione_fk2 FOREIGN KEY (lezione)
        REFERENCES lezione(lezione_id) ON DELETE CASCADE ON UPDATE CASCADE,
```

7.11 APPARTENENZA_AREA

```
CREATE TABLE IF NOT EXISTS appartenenza_area(
        corso INTEGER NOT NULL,
        tema_trattato VARCHAR(100) NOT NULL);


ALTER TABLE appartenenza_area
        ADD CONSTRAINT appartenenza_area_fk1 FOREIGN KEY (corso)
        REFERENCES corso(corso_id) ON DELETE CASCADE,

        ADD CONSTRAINT appartenenza_area_fk2 FOREIGN KEY (tema_trattato)
        REFERENCES area_tematica(tema) ON DELETE CASCADE ON UPDATE CASCADE,
```

7.12 OPERATORE

```
CREATE TABLE IF NOT EXISTS operatore(
        nome_utente VARCHAR(50) PRIMARY KEY,
        password VARCHAR(50) NOT NULL);
```

8 Implementazione dei vincoli

8.1 Vincoli Intrarelazionali

8.1.1 ISCRIZIONE_UNIQUE

```
ALTER TABLE iscrizione
ADD CONSTRAINT iscrizione_unique
UNIQUE (studente_iscritto, corso_scelto);
```

8.1.2 PAROLE_LEZIONE_UNIQUE

```
ALTER TABLE parole_lezione
ADD CONSTRAINT parole_lezione_unique
UNIQUE (parola,lezione);
```

8.1.3 AREA_TEMATICA_CORSO_UNIQUE

```
ALTER TABLE appartenenza_area
ADD CONSTRAINT area_tematica_corso_unique
UNIQUE (corso,tema_trattato);
```

8.1.4 LEZIONE_STUDENTE_UNIQUE

```
ALTER TABLE partecipazione
ADD CONSTRAINT lezione_frequentata_studente_partecipante_unique
UNIQUE (lezione_frequentata, studente_partecipante)
```

8.1.5 NUMERO_PARTECIPANTI_MAGGIORE_0

```
ALTER TABLE corso
ADD CONSTRAINT numero_partecipanti_maggiore_di_0
CHECK (max_partecipanti>0);
```

8.1.6 SOLO_ORARI_CONSENTITI_LEZIONI_ON_INSERT

```
/* La seguente function restituisce un trigger che, al momento dell'inserimento
di una lezione, controlla che la lezione non inizi prima delle 8:00 e che
non finisca dopo le 20:00.
Qualora dovesse verificarsi questa illegalità, il trigger elimina la
lezione appena inserita. */


CREATE FUNCTION check_orario_valido_lezione_on_insert()
RETURNS TRIGGER AS
$$
DECLARE
BEGIN
 IF (NEW.ora_inizio < '8:00' OR NEW.ora_fine > '20:00') THEN
      DELETE FROM lezione WHERE (lezione_id = NEW.lezione_id);
      RAISE SQLSTATE '12311';
      RAISE EXCEPTION 'Impossibile creare una lezione che inizi
                       prima delle 8:00 o che finisca dopo le 20:00.';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER solo_orari_consentiti_lezioni_on_insert
AFTER INSERT ON lezione
FOR EACH ROW
EXECUTE PROCEDURE check_orario_valido_lezione_on_insert();
```

8.1.7 SOLO_ORARI_CONSENTITI_LEZIONI_ON_UPDATE

```
/* La seguente function restituisce un trigger che, al momento della modifica
di una lezione, controlla che la lezione non inizi prima delle 8:00 e che
non finisca dopo le 20:00.
Qualora dovesse verificarsi questa illegalità, il trigger annulla la modifica
effettuata. */


CREATE FUNCTION check_orario_valido_lezione_on_update()
RETURNS TRIGGER AS
$$
DECLARE
BEGIN
 IF (NEW.ora_inizio < '8:00' OR NEW.ora_fine > '20:00') THEN
      RAISE SQLSTATE '12312';
      ROLLBACK;
      RAISE EXCEPTION 'Impossibile creare una lezione che inizi
                       prima delle 8:00 o che finisca dopo le 20:00.';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER solo_orari_consentiti_lezioni_on_update
AFTER UPDATE ON lezione
FOR EACH ROW
EXECUTE PROCEDURE check_orario_valido_lezione_on_update();
```

8.1.8 NO_LEZIONI_PIU'_DI_2ORE_ON_INSERT

```
/* La seguente function restituisce un trigger che, al momento dell'inserimento
di una lezione, controlla che essa non duri più di 2 ore.
Qualora dovesse verificarsi questa illegalità, il trigger elimina
la lezione appena inserita. */


CREATE FUNCTION check_durata_massima_lezioni_on_insert()
RETURNS TRIGGER AS
$$
DECLARE
BEGIN
 IF ((SELECT calcolo_durata_lezione(NEW.ora_inizio, NEW.ora_fine)) > '2:00:00') THEN
      DELETE FROM lezione WHERE lezione.lezione_id = NEW.lezione_id;
      RAISE SQLSTATE '12313';
      RAISE EXCEPTION 'Una lezione non può durare più di due ore.';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER no_lezioni_più_di2ore_on_insert
AFTER INSERT ON lezione
FOR EACH ROW
EXECUTE PROCEDURE check_durata_massima_lezioni_on_insert();
```

8.1.9 NO_LEZIONI_PIU'_DI_2ORE_ON_UPDATE

```
/* La seguente function restituisce un trigger che, al momento della modifica
di una lezione, controlla che essa non duri più di 2 ore.
Qualora dovesse verificarsi questa illegalità, il trigger annulla
la modifica effettuata. */


CREATE FUNCTION check_durata_massima_lezioni_on_update()
RETURNS TRIGGER AS
$$
DECLARE
BEGIN
 IF ((SELECT calcolo_durata_lezione(NEW.ora_inizio, NEW.ora_fine)) > '2:00:00') THEN
      RAISE SQLSTATE '12314';
      ROLLBACK;
      RAISE EXCEPTION 'Una lezione non può durare più di due ore.';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER no_lezioni_più_di2ore_on_update
AFTER UPDATE ON lezione
FOR EACH ROW
EXECUTE PROCEDURE check_durata_massima_lezioni_on_update();
```

8.1.10 NO_ORARIO_ILLEGALE_LEZIONI_ON_INSERT

```
/* La seguente function restituisce un trigger che, al momento dell'inserimento
di una lezione, controlla che la sua ora di fine non sia precedente
all'ora di inizio.
Qualora dovesse verificarsi questa illegalità, il trigger elimina la lezione
appena inserita. */


CREATE FUNCTION check_correttezza_orario_lezioni_on_insert()
RETURNS TRIGGER AS
$$
DECLARE
BEGIN
 IF (NEW.ora_inizio>=NEW.ora_fine) THEN
      DELETE FROM lezione WHERE lezione.lezione_id = NEW.lezione_id;
      RAISE SQLSTATE '12308';
      RAISE EXCEPTION 'Una lezione non può avere orario
                       di inizio posteriore a quello di fine.';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER no_orario_illegale_lezioni_on_insert
AFTER INSERT ON lezione
FOR EACH ROW
EXECUTE PROCEDURE check_correttezza_orario_lezioni_on_insert();
```

8.1.11 NO_ORARIO_ILLEGALE_LEZIONI_ON_UPDATE

```
/* La seguente function restituisce un trigger che, al momento della modifica
di una lezione, controlla che la sua ora di fine non sia precedente
all'ora di inizio.
Qualora dovesse verificarsi questa illegalità, il trigger annulla la
modifica appena effettuata. */


CREATE FUNCTION check_correttezza_orario_lezioni_on_update()
RETURNS TRIGGER AS
$$
DECLARE
BEGIN
 IF (NEW.ora_inizio>=NEW.ora_fine) THEN
      RAISE SQLSTATE '12309';
      ROLLBACK;
      RAISE EXCEPTION 'Una lezione non può avere orario
                       di inizio posteriore a quello di fine.';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER no_orario_illegale_lezioni_on_update
AFTER UPDATE ON lezione
FOR EACH ROW
EXECUTE PROCEDURE check_correttezza_orario_lezioni_on_update();
```

8.2 Vincoli Interrelazionali

8.2.1 NO_LEZIONI_CORSO_TERMINATO

```
/* La seguente function restituisce un trigger che elimina l'aggiunta di una lezione
ad un corso terminato. */

CREATE FUNCTION check_corso_terminato_lezione()
RETURNS TRIGGER AS
$$
BEGIN
 IF ((SELECT c.terminato
      FROM lezione l JOIN corso c ON l.del_corso = c.corso_id
      WHERE l.lezione_id = NEW.lezione_id) = TRUE)
 THEN
      DELETE FROM lezione WHERE lezione.lezione_id = NEW.lezione_id;
      RAISE SQLSTATE '12300';
      RAISE EXCEPTION 'Non puoi inserire una lezione se il corso è terminato.';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER no_lezioni_corso_terminato
AFTER INSERT ON lezione
FOR EACH ROW
EXECUTE PROCEDURE check_corso_terminato_lezione();
```

8.2.2 NO_ISCRIZIONI_CORSO_TERMINATO

```
/* La seguente function restituisce un trigger che elimina l'iscrizione
ad un corso terminato da parte di uno studente */

CREATE FUNCTION check_corso_terminato_iscrizione()
RETURNS TRIGGER AS
$$
BEGIN
 IF ((SELECT c.terminato
      FROM iscrizione i JOIN corso c ON i.corso_scelto=c.corso_id
      WHERE i.corso_scelto = NEW.corso_scelto AND
            i.studente_iscritto=NEW.studente_iscritto) = TRUE)
 THEN
      DELETE FROM iscrizione WHERE iscrizione.corso_scelto=NEW.corso_scelto;
      RAISE SQLSTATE '12301';
      RAISE EXCEPTION 'Non puoi iscrivere uno studente ad un corso terminato';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER no_iscrizioni_corso_terminato
AFTER INSERT ON iscrizione
FOR EACH ROW
EXECUTE PROCEDURE check_corso_terminato_iscrizione();
```

8.2.3 NO_PARTECIPAZIONE_LEZIONE_CORSO_TERMINATO

```
/* La seguente function restituisce un trigger che elimina una
partecipazione ad una lezione da parte di uno studente
se il corso in questione è terminato */

CREATE FUNCTION check_corso_terminato_partecipazione()
RETURNS TRIGGER AS
$$
BEGIN
 IF ((SELECT c.terminato
      FROM lezione l JOIN corso c ON l.del_corso=c.corso_id
           JOIN partecipazione p ON l.lezione_id=p.lezione_frequentata
      WHERE p.lezione_frequentata=NEW.lezione_frequentata AND
            p.studente_partecipante=NEW.studente_partecipante) = TRUE)
 THEN
      DELETE FROM partecipazione p
             WHERE p.lezione_frequentata=NEW.lezione_frequentata;
      RAISE SQLSTATE '12302';
      RAISE EXCEPTION 'Non puoi partecipare ad una lezione di un corso terminato';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER no_partecipazioni_lezione_corso_terminato
AFTER INSERT ON partecipazione
FOR EACH ROW
EXECUTE PROCEDURE check_corso_terminato_partecipazione();
```

8.2.4 NO_PARTECIPAZIONI_NON_ISCRITTI_CORSO

```
/*La seguente function restituisce un trigger che elimina una
partecipazione ad una lezione da parte di uno studente
non iscritto al relativo corso. */

CREATE FUNCTION check_iscrizione()
RETURNS TRIGGER AS
$$
DECLARE
tot integer;
BEGIN
 SELECT count(*) INTO tot
 FROM iscrizione i
 WHERE i.studente_iscritto=NEW.studente_partecipante AND
       i.corso_scelto = (SELECT c.corso_id
                         FROM corso c JOIN lezione l ON c.corso_id=l.del_corso
                         WHERE NEW.lezione_frequentata = l.lezione_id);
 IF (tot <= 0) THEN
      DELETE FROM partecipazione
             WHERE partecipazione.lezione_frequentata = NEW.lezione_frequentata
        AND partecipazione.studente_partecipante = NEW.studente_partecipante;
      RAISE SQLSTATE '12303';
      RAISE EXCEPTION 'Uno studente non può partecipare ad una lezione
                       se non è iscritto al relativo corso.';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER no_partecipazioni_non_iscritti_corso
AFTER INSERT ON partecipazione
FOR EACH ROW
EXECUTE PROCEDURE check_iscrizione();
```

8.2.5 NO_ISCRIZIONI_CORSO_PIENO

```
/* La seguente function restituisce un trigger che elimina l'iscrizione
di uno studente ad un corso che ha già raggiunto il numero
massimo di iscrizioni. */

CREATE FUNCTION check_numero_massimo_partecipanti()
RETURNS TRIGGER AS
$$
DECLARE numero_iscritti INTEGER;
BEGIN
 SELECT COUNT(*) INTO numero_iscritti
 FROM iscrizione i
 WHERE i.corso_scelto=NEW.corso_scelto;

 IF (numero_iscritti > (SELECT c.max_partecipanti
                        FROM corso c JOIN iscrizione i
                             ON c.corso_id=i.corso_scelto
                        WHERE i.corso_scelto=NEW.corso_scelto AND
                              i.studente_iscritto=NEW.studente_iscritto))
 THEN
      DELETE FROM iscrizione i WHERE i.corso_scelto=NEW.corso_scelto AND
                                     i.studente_iscritto=NEW.studente_iscritto;
      RAISE SQLSTATE '12304';
      RAISE EXCEPTION 'Un corso non può avere più iscritti del
                       numero massimo di partecipanti';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER no_iscrizioni_corso_pieno
AFTER INSERT ON iscrizione
FOR EACH ROW
EXECUTE PROCEDURE check_numero_massimo_partecipanti();
```

8.2.6 NO_LEZIONI_CONTEMPORANEE_STESSO_CORSO_ON_INSERT

```
/* La seguente function restituisce un trigger che elimina l'inserimento
di una lezione la cui programmazione coincide o si accavalla con
una lezione dello stesso corso. */

CREATE FUNCTION check_orario_lezioni_on_insert()
RETURNS TRIGGER AS
$$
DECLARE
lezioni_sovrapposte INTEGER;
BEGIN
 SELECT count(*) INTO lezioni_sovrapposte
 FROM lezione l JOIN corso c ON l.del_corso = c.corso_id
 WHERE c.corso_id = NEW.del_corso AND
       l.data = NEW.data AND
       (NEW.ora_inizio BETWEEN l.ora_inizio AND l.ora_fine OR
        NEW.ora_fine BETWEEN l.ora_inizio AND l.ora_fine OR
       (NEW.ora_inizio < l.ora_inizio AND NEW.ora_fine > l.ora_fine));

 IF (lezioni_sovrapposte > 1) THEN
      DELETE FROM lezione WHERE lezione.lezione_id = NEW.lezione_id;
      RAISE SQLSTATE '12305';
      RAISE EXCEPTION 'Due lezioni dello stesso corso non
                       possono svolgersi contemporaneamente';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER no_lezioni_contemporanee_stesso_corso_on_insert
AFTER INSERT ON lezione
FOR EACH ROW
EXECUTE PROCEDURE check_orario_lezioni_on_insert();
```

8.2.7 NO_LEZIONI_CONTEMPORANEE_STESSO_CORSO_ON_UPDATE

```
/* La seguente function restituisce un trigger che blocca la modifica di una
lezione qualora la nuova programmazione coincida o si accavalli
con quella di una lezione già esistente dello stesso corso. */

CREATE FUNCTION check_orario_lezioni_on_update()
RETURNS TRIGGER AS
$$
DECLARE
lezioni_sovrapposte INTEGER;
BEGIN
 SELECT count(*) INTO lezioni_sovrapposte
 FROM lezione l JOIN corso c ON l.del_corso = c.corso_id
 WHERE c.corso_id = NEW.del_corso AND
       l.data = NEW.data AND
       (NEW.ora_inizio BETWEEN l.ora_inizio AND l.ora_fine OR
        NEW.ora_fine BETWEEN l.ora_inizio AND l.ora_fine OR
       (NEW.ora_inizio < l.ora_inizio AND NEW.ora_fine > l.ora_fine));

 IF (lezioni_sovrapposte > 1) THEN
      RAISE SQLSTATE '12306';
      ROLLBACK;
      RAISE EXCEPTION 'Due lezioni dello stesso corso non
                       possono svolgersi contemporaneamente';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER no_lezioni_contemporanee_stesso_corso_on_update
AFTER UPDATE ON lezione
FOR EACH ROW
EXECUTE PROCEDURE check_orario_lezioni_on_update();
```

8.2.8 NO_PARTECIPAZIONI_CONTEMPORANEE

```
/* La seguente function restituisce un trigger che elimina l'inserimento
di una partecipazione di uno studente ad una lezione qualora lo studente
sia stato già registrato come presente ad una lezione di un altro corso
che si svolge in contemporanea alla lezione stessa. */

CREATE FUNCTION check_orario_partecipazioni()
RETURNS TRIGGER AS
$$
DECLARE

lezioni_stesso_intervallo INTEGER;
new_ora_inizio TIME;
new_ora_fine TIME;
new_data DATE;

BEGIN

 SELECT l1.data, l1.ora_inizio, l1.ora_fine INTO new_data, new_ora_inizio, new_ora_fine
 FROM lezione l1
 WHERE l1.lezione_id = NEW.lezione_frequentata;

 SELECT count(*) INTO lezioni_stesso_intervallo
 FROM partecipazione p
 WHERE p.studente_partecipante = NEW.studente_partecipante AND
       p.lezione_frequentata IN (SELECT l.lezione_id
                                 FROM lezione l JOIN partecipazione p1
                                      ON l.lezione_id = p1.lezione_frequentata
                                 WHERE l.data = new_data AND
                                       (new_ora_inizio BETWEEN l.ora_inizio AND
                                        l.ora_fine OR new_ora_fine BETWEEN
                                        l.ora_inizio AND l.ora_fine OR
                                        (new_ora_inizio < l.ora_inizio AND
                                        new_ora_fine > l.ora_fine)));
```

```
38  IF (lezioni_stesso_intervallo > 1) THEN
39       DELETE FROM partecipazione p
40              WHERE p.lezione_frequentata = NEW.lezione_frequentata AND
41                    p.studente_partecipante = NEW.studente_partecipante;
42       RAISE SQLSTATE '12307';
43       RAISE EXCEPTION 'Uno studente non può partecipare a due lezioni (di corsi diversi)
44                        che si svolgono nello stesso intervallo di tempo';
45
46  END IF;
47  RETURN NEW;
48 END;
49 $$ LANGUAGE plpgsql;
50
51 CREATE TRIGGER no_partecipazioni_contemporanee
52 AFTER INSERT ON partecipazione
53 FOR EACH ROW
54 EXECUTE PROCEDURE check_orario_partecipazioni();
```

8.2.9 NO_DECREMENTO_NUM_MAX_PARTECIPANTI

```
/* Quando il numero massimo di studenti che possono iscriversi ad un corso
viene aggiornato, questa function restituisce un trigger che controlla
se il numero attuale di studenti iscritti supera il nuovo limite inserito
dall'operatore. In tal caso, il trigger annulla la modifica. */

CREATE FUNCTION check_nuovo_num_max_partecipanti()
RETURNS TRIGGER AS
$$
DECLARE
BEGIN
 IF (NEW.max_partecipanti < (SELECT count(*)
                             FROM iscrizione i
                             WHERE i.corso_scelto = NEW.corso_id))
 THEN
      RAISE SQLSTATE '12310';
      ROLLBACK;
      RAISE EXCEPTION 'Impossibile diminuire il numero massimo di
                       iscritti al di sotto del numero di iscritti attuale.';
 END IF;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER no_decremento_num_max_partecipanti
AFTER UPDATE ON corso
FOR EACH ROW
EXECUTE PROCEDURE check_nuovo_num_max_partecipanti();
$$ LANGUAGE plpgsql;
```

9 Definizione delle Funzioni e delle Procedure

9.1 CALCOLO_DURATA_LEZIONE

```
/* Function che calcola la durata di una lezione date l'ORA_INIZIO e l'ORA_FINE. */

CREATE FUNCTION calcolo_durata_lezione(inizio TIME, fine TIME)
RETURNS INTERVAL AS
$$
BEGIN
        RETURN fine - inizio;
END;
$$ LANGUAGE plpgsql;
```

9.2 CALCOLO_STATISTICHE_MIN_PRESENZE

```
/* Function che calcola il numero minimo di presenze registrate in un corso. */

CREATE FUNCTION calcolo_statistiche_min_presenze(nome_corso VARCHAR)
RETURNS INTEGER AS
$$
DECLARE
        ret INTEGER;
BEGIN
        SELECT min(p1.conteggio) INTO ret
        FROM presenze p1 JOIN corso c1 ON p1.corso_id = c1.corso_id
        WHERE c1.nome = nome_corso
        GROUP BY c1.corso_id;

        RETURN ret;
END;
$$ LANGUAGE plpgsql;
```

9.3 CALCOLO_STATISTICHE_MAX_PRESENZE

```
/* Function che calcola il numero massimo di presenze registrate in un corso. */

CREATE FUNCTION calcolo_statistiche_max_presenze(nome_corso VARCHAR)
RETURNS INTEGER AS
$$
DECLARE
        ret INTEGER;
BEGIN
        SELECT max(p1.conteggio) INTO ret
        FROM presenze p1 JOIN corso c1 ON p1.corso_id = c1.corso_id
        WHERE c1.nome = nome_corso
        GROUP BY c1.corso_id;

        RETURN ret;
END;
$$ LANGUAGE plpgsql;
```

9.4 CALCOLO_STATISTICHE_MEDIA_PRESENZE

```
/* Function che calcola il numero medio di presenze registrate in un corso.
Si è reso necessario calcolare esplicitamente la media non utilizzando
AVG poichè era necessario effettuare il conteggio del numero di lezioni
sulla tabella lezioni, e non su presenze, in modo da includere anche le
lezioni senza partecipanti. */

CREATE FUNCTION calcolo_statistiche_media_presenze(nome_corso VARCHAR)
RETURNS INTEGER AS
$$

DECLARE
num_lezioni INTEGER;
somma_presenze_corso INTEGER;
risultato INTEGER;

BEGIN

 SELECT SUM(conteggio) INTO somma_presenze_corso
 FROM presenze p
 WHERE p.corso_id = (SELECT c.corso_id
                     FROM corso c
                     WHERE c.nome = nome_corso);

 SELECT COUNT(*) INTO num_lezioni
 FROM lezione l
 WHERE l.del_corso = (SELECT c.corso_id
                      FROM corso c
                      WHERE c.nome = nome_corso);

 risultato := (somma_presenze_corso/num_lezioni);

RETURN risultato;
END;
$$ LANGUAGE plpgsql;
```

9.5 CALCOLO_PERCENTUALE_RIEMPIMENTO_MEDIA

```
/* Function che calcola la percentuale media di riempimento di un corso. */

CREATE FUNCTION calcolo_percentuale_riempimento_media(nome_corso VARCHAR)
RETURNS INTEGER AS
$$
DECLARE
        max_partecipanti INTEGER;
        ret INTEGER;
BEGIN

        SELECT c.max_partecipanti INTO max_partecipanti
        FROM corso c
        WHERE c.nome = nome_corso;

        IF max_partecipanti= '0' THEN
            ret := '100';
        ELSE
            ret := calcolo_statistiche_media_presenze(nome_corso);
            ret := ret*'100';
            ret := ret/max_partecipanti;

        END IF;
        RETURN ret;
END;
$$ LANGUAGE plpgsql;
```

9.6 CALCOLO_SUPERAMENTO_CORSO

```
/* Function che restituisce un trigger.
Il trigger in questione, nel momento in cui un corso viene terminato dall'operatore
stabilisce quali sono gli studenti che hanno ottenuto il numero minimo di
presenze necessarie al superamento del corso (in base all'attributo
"percentuale_minima_presenze" della tabella CORSO).
L'attributo "superato" della tabella ISCRIZIONE verrà aggiornato quindi
da 'FALSE' a 'TRUE' nelle righe corrispondenti agli studenti idonei.  */

CREATE FUNCTION calcolo_superamento_corso()
RETURNS TRIGGER AS
$$
DECLARE
lezioni_totali INTEGER;
presenze INTEGER;
percentuale FLOAT;
cursore CURSOR IS (SELECT *
                   FROM iscrizione i
                  WHERE i.corso_scelto = NEW.corso_id);
BEGIN

        FOR S IN cursore
        LOOP

        lezioni_totali := '0';
        presenze := '0';
        percentuale := '0';

        SELECT count(*) INTO presenze
        FROM partecipazione p JOIN lezione l
                ON p.lezione_frequentata = l.lezione_id
        WHERE l.del_corso = S.corso_scelto AND
                p.studente_partecipante = S.studente_iscritto;

        SELECT count(*) INTO lezioni_totali
        FROM lezione l
        WHERE l.del_corso = S.corso_scelto;

```

```
38          IF(lezioni_totali<>0) THEN
39                  percentuale := FLOOR((presenze*100)/lezioni_totali);
40          END IF;
41
42
43          UPDATE iscrizione
44          SET superato = TRUE
45          WHERE studente_iscritto = S.studente_iscritto AND
46                  corso_scelto = S.corso_scelto AND
47                  percentuale >= (SELECT c.percentuale_minima_presenze
48                                  FROM corso c
49                                  WHERE c.corso_id = S.corso_scelto);
50          END LOOP;
51          RETURN NEW;
52  END;
53
54  $$ LANGUAGE plpgsql;
55
56
57  CREATE TRIGGER su_terminazione_corso
58  AFTER UPDATE ON corso
59  FOR EACH ROW
60  WHEN (OLD.terminato = FALSE AND NEW.terminato = TRUE)
61  EXECUTE PROCEDURE calcolo_superamento_corso();
```

---

10 Definizione delle Viste

10.1 PRESENZE

```
CREATE OR REPLACE VIEW presenze(corso_id,lezione_id,titolo_lezione,conteggio) AS(
        SELECT c.corso_id, l.lezione_id, l.titolo, count(*)
        FROM corso c JOIN lezione l ON c.corso_id = l.del_corso JOIN
            partecipazione p ON l.lezione_id = p.lezione_frequentata
        GROUP BY c.corso_id, l.lezione_id
        ORDER BY c.corso_id);
```

10.2 INSEGNAMENTI

```
CREATE OR REPLACE VIEW insegnamenti(professore_id,corso_id) AS(
        SELECT DISTINCT p.professore_id, c.corso_id
        FROM professore p JOIN lezione l ON l.professore=p.professore_id JOIN
            corso c ON l.del_corso=c.corso_id
        ORDER BY professore_id);
```

11 Definizione del Datestyle

```
    SET DATESTYLE TO Postgres, DMY;
```

12 Definizione delle Sequenze

12.1 Sequenza CORSO_ID

```
CREATE SEQUENCE sequenza_corso_id
    START 1
    INCREMENT 1
    OWNED BY corso.corso_id;
```

12.2 Sequenza STUDENTE_ID

```
CREATE SEQUENCE sequenza_studente_id
    START 1
    INCREMENT 1
    OWNED BY studente.studente_id;
```

12.3 Sequenza PROFESSORE_ID

```
CREATE SEQUENCE sequenza_professore_id
    START 1
    INCREMENT 1
    OWNED BY professore.professore_id;
```

12.4 Sequenza LEZIONE_ID

```
CREATE SEQUENCE sequenza_lezione_id
    START 1
    INCREMENT 1
    OWNED BY lezione.lezione_id;
```

Parte V
Popolamento del DataBase

13 Tabelle

13.1 CORSO

```
/* Corso di Public Speaking */
INSERT INTO corso VALUES(nextval('sequenza_corso_id'), 'public speaking',
        'Impara a parlare in pubblico con sicurezza e spigliatezza.',
        '25', '20', 'false');
/* Corso di Chitarra Classica */
INSERT INTO corso VALUES(nextval('sequenza_corso_id'), 'chitarra classica',
        'Impara a suonare le tue canzoni preferite.',
        '50', '30', 'false');
/* Corso di Finanza Personale */
INSERT INTO corso VALUES(nextval('sequenza_corso_id'), 'finanza personale',
        'Impara a gestire la tua disponibilità economica.',
        '75', '30', 'false');
/* Corso di Java per Tutti */
INSERT INTO corso VALUES(nextval('sequenza_corso_id'), 'java per tutti',
        'Le basi del linguaggio e della programmazione o-o.',
        '100', '90', 'false');
/* Corso di Produzione Musicale */
INSERT INTO corso VALUES(nextval('sequenza_corso_id'), 'produzione musicale',
        'Imparare a produrre tracce audio con software di produzione musicale.',
        '50', '75', 'false');
/* Corso di Tedesco Base */
INSERT INTO corso VALUES(nextval('sequenza_corso_id'), 'tedesco base',
        'Introduzione alla lingua e alla cultura tedesca.',
        '150', '60', 'false');
/* Corso di C for Dummies */
INSERT INTO corso VALUES(nextval('sequenza_corso_id'), 'c for dummies',
        'Introduzione al linguaggio c.',
        '200', '80', 'false');
```

13.2 STUDENTE

```
INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
        'Francesco', 'Pio', '05/05/2001', 'francescopio@gmail.com', '081258784');
INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
        'Luigi', 'Avezzano', '03/03/2001', 'luiave@libero.it', '081778465');
INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
        'Miriam', 'Giacobelli', '07/08/2000', 'mirigiac@gmail.com', '089654784');
INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
        'Luca', 'Mirabile', '04/11/2002', 'lucamirabile@libero.it', '081235432');
INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
        'Marco', 'Nappa', '12/05/2000', 'marknapp@hotmail.com', '081512577');
INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
        'Umberto', 'Guerra', '11/05/2002', 'ginnywar@gmail.com', '081561291');
INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
        'Mario', 'Vasaturo', '12/09/2000', 'mavmav@libero.it', '081235684');
INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
        'Luca', 'Zampino', '29/03/1999', 'zampluk@hotmail.com', '089448478');
INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
        'Valentina', 'Corvino', '12/05/2000', 'corvinovale@gmail.com', '02213595');
INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
        'Sofia', 'Carini', '01/08/2003', 'sofiacar@hotmail.com', '02111717');
INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
        'Camilla', 'Russo', '30/01/2002', 'camirusso@gmail.com', '081113222');
INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
        'Ana', 'Diaz', '15/10/2003', 'anaziad@gmail.com', '345165852');
INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
        'Maria', 'Picon', '23/12/2000', 'maripicon@gmail.com', '341002131');
INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
        'Luca', 'Salzano', '14/08/1999', 'luksalz@gmail.com', '081114765');
INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
        'Sabrina', 'Romano', '12/11/2000', 'sabbrr@gmail.com', '081424212');
INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
        'Simone', 'Albano', '30/03/1998', 'simmaf@libero.it', '342154575');
INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
        'Salvatore', 'Marciano', '07/07/2001', 'sasimar@gmail.com', '345768668');
INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
        'Pino', 'Esposito', '14/02/2000', 'pinesps@gmail.com', '081456999');
INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
```

```
38          'Viviana', 'Daniele', '03/07/1999', 'vividannn@libero.it', '089112435');
39  INSERT INTO studente VALUES (nextval('sequenza_studente_id'),
40          'Alessandro', 'Insegno', '10/02/1980', 'asmaf@gmail.com', '089974511');
```

13.3 PROFESSORE

```
INSERT INTO professore VALUES(nextval('sequenza_professore_id'),
        'Ambrogio', 'Cristoforetti');
INSERT INTO professore VALUES(nextval('sequenza_professore_id'),
        'Laura', 'De Luca');
INSERT INTO professore VALUES(nextval('sequenza_professore_id'),
        'Tommaso', 'Amodio');
INSERT INTO professore VALUES(nextval('sequenza_professore_id'),
        'Melania', 'Castronuovo');
INSERT INTO professore VALUES(nextval('sequenza_professore_id'),
        'Rossella', 'De Simone');
INSERT INTO professore VALUES(nextval('sequenza_professore_id'),
        'Filippo', 'Secondo');
INSERT INTO professore VALUES(nextval('sequenza_professore_id'),
        'Salvatore', 'Orecchio');
INSERT INTO professore VALUES(nextval('sequenza_professore_id'),
        'Francesca', 'Pardini');
```

13.4 LEZIONE

```
/* Lezioni di Public Speaking */
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'la giusta postura', 'Una buona postura è necessaria per dare
        una buona impressione.', '05/01/2022', '13:00','14:30', '1', '1');
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'la giusta dizione', 'Imparare a esprimersi correttamente.',
                '07/01/2022', '12:00', '14:00', '1', '1');
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'storytelling', 'Costruire un racconto nel tuo discorso.',
        '09/01/2022', '17:00', '19:00', '1', '1');
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'tecniche di persuasione', 'I segreti della persuasione
        durante un discorso.' , '11/01/2022', '13:00', '14:00', '1', '1');
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'lezione conclusiva', 'Esercitazione pratica.',
        '13/01/2022', '18:00', '20:00', '1', '1');
/* Lezioni di Chitarra Classica */
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'introduzione allo strumento', 'I primi passi.',
        '02/02/2022', '13:00', '15:00', '2', '2');
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'lettura dello spartito', 'Impara a leggere la musica.',
        '03/02/2022', '14:00', '16:00', '2', '2');
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'solfeggio', 'Il solfeggio è una parte fondamentale della formazione
        di uno stumentista', '05/02/2022', '13:00', '15:00', '2', '2');
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'primi accordi', 'Impara a suonare le tue prime canzoni.',
        '07/02/2022', '9:30', '11:00', '2', '2');
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'le scale', 'Impara ad eseguire la scala di do.',
        '09/02/2022', '10:30', '12:30', '2', '2');
/* Lezioni di Finanza Personale */
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'fondamenti di economia', 'Comprendere le nozioni base.',
        '05/03/2022', '14:00', '16:00', '3', '3');
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
```

```
        'risparmiare', 'Come e perchè risparmiare.',
        '07/03/2022', '14:00', '16:00', '3', '3');
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'gestione delle spese', 'Riconoscere i motivi delle proprie
        spese e come tracciarle.','10/03/2022', '13:30', '15:00', '3', '3');
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'investire', 'Nozioni di base per investire passivamente.',
        '11/03/2022', '10:30', '12:00', '3', '3');
/* Lezioni di Java per Tutti */
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'introduzione alla programmazione ad oggetti',
        'Classi, oggetti, incapsulamento.',
        '05/03/2022', '15:00', '16:00', '4', '4');
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'gerarchie', 'Specializzazioni e polimorfismo.',
        '10/03/2022', '14:00', '16:00', '4', '4');
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'classi astratte ed interfacce', 'Parole chiave abstract ed interface.',
        '14/03/2022', '14:00', '15:00', '4', '4');
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'gui', 'Costruire interfacce grafiche in Java.',
        '17/03/2022', '11:30', '12:30', '4', '4');
/* Lezioni di Produzione Musicale */
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'introduzione alle daw', 'Primi passi nelle Digital Audio Workstation.',
        '02/05/2022', '14:00', '16:00', '5', '5');
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'teoria musicale', 'Elementi di teoria musicale per produzioni audio.',
        '07/05/2022', '14:00', '16:00', '5', '5');
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'mixing e mastering', 'Nozioni di tecnica del suono.',
        '10/05/2022', '12:00', '14:00', '5', '5');
/* Lezioni di Tedesco Base */
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'presentarsi', 'Saluti e presentazioni in tedesco.',
        '25/06/2022', '11:00', '12:20', '6', '6');
INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
        'la struttura della frase', 'Struttura di frasi affermative,negative
        e interrogative in tedesco.', '26/06/2022', '14:00', '16:00', '6', '6');
```

```
77 INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
78         'generi e articoli', 'I 3 generi, gli articoli e i pronomi possessivi.',
79         '27/06/2022', '13:20', '15:10', '6', '6');
80 INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
81         'i casi', 'Le declinazioni e i casi.',
82         '28/06/2022', '11:30', '12:30', '6', '6');
83 /* Lezioni di C for Dummies */
84 INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
85         'funzioni e variabili', 'Primi passi nel linguaggio C.',
86         '02/05/2021', '14:00', '16:00', '7', '7');
87 INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
88         'puntatori', 'Fondamenti sui puntatori.',
89         '07/05/2021', '14:00', '16:00', '7', '7');
90 INSERT INTO lezione VALUES (nextval('sequenza_lezione_id'),
91         'puntatori', 'Sempre puntatori.',
92         '10/05/2021', '12:00', '14:00', '7', '7');
```

13.5 PAROLE_CHIAVE

```
1  INSERT INTO parole_chiave VALUES('oratoria');
2  INSERT INTO parole_chiave VALUES('eloquenza');
3  INSERT INTO parole_chiave VALUES('convincimento');
4  INSERT INTO parole_chiave VALUES('parlare');
5  INSERT INTO parole_chiave VALUES('note');
6  INSERT INTO parole_chiave VALUES('spartito');
7  INSERT INTO parole_chiave VALUES('performance');
8  INSERT INTO parole_chiave VALUES('ritmo');
9  INSERT INTO parole_chiave VALUES('investimenti');
10 INSERT INTO parole_chiave VALUES('risparmio');
11 INSERT INTO parole_chiave VALUES('denaro');
12 INSERT INTO parole_chiave VALUES('programmazione');
13 INSERT INTO parole_chiave VALUES('sviluppo');
14 INSERT INTO parole_chiave VALUES('mixing');
15 INSERT INTO parole_chiave VALUES('mastering');
16 INSERT INTO parole_chiave VALUES('composizione');
17 INSERT INTO parole_chiave VALUES('software');
18 INSERT INTO parole_chiave VALUES('grammatica');
19 INSERT INTO parole_chiave VALUES('lessico');
```

13.6 MACROAREA

```
INSERT INTO macroarea VALUES ('crescita personale');
INSERT INTO macroarea VALUES ('musica');
INSERT INTO macroarea VALUES ('economia e finanza');
INSERT INTO macroarea VALUES ('informatica');
INSERT INTO macroarea VALUES ('lingue');
```

13.7 AREA_TEMATICA

```
INSERT INTO area_tematica VALUES ('dialettica', 'crescita personale');
INSERT INTO area_tematica VALUES ('strumenti musicali', 'musica');
INSERT INTO area_tematica VALUES ('personal economy', 'economia e finanza');
INSERT INTO area_tematica VALUES ('linguaggi di programmazione', 'informatica');
INSERT INTO area_tematica VALUES ('music production', 'musica');
INSERT INTO area_tematica VALUES ('lingua e cultura tedesca', 'lingue');
```

13.8 APPARTENENZA_AREA

```
INSERT INTO appartenenza_area VALUES ('1', 'dialettica');
INSERT INTO appartenenza_area VALUES ('2', 'strumenti musicali');
INSERT INTO appartenenza_area VALUES ('3', 'personal economy');
INSERT INTO appartenenza_area VALUES ('4', 'linguaggi di programmazione');
INSERT INTO appartenenza_area VALUES ('5', 'music production');
INSERT INTO appartenenza_area VALUES ('5', 'strumenti musicali');
INSERT INTO appartenenza_area VALUES ('6', 'lingua e cultura tedesca');
INSERT INTO appartenenza_area VALUES ('7', 'linguaggi di programmazione');
```

13.9 ISCRIZIONE

```
INSERT INTO iscrizione VALUES ('false', '1', '1');
INSERT INTO iscrizione VALUES ('false', '2', '1');
INSERT INTO iscrizione VALUES ('false', '3', '1');
INSERT INTO iscrizione VALUES ('false', '5', '1');
INSERT INTO iscrizione VALUES ('false', '2', '2');
INSERT INTO iscrizione VALUES ('false', '4', '2');
INSERT INTO iscrizione VALUES ('false', '5', '2');
INSERT INTO iscrizione VALUES ('false', '5', '3');
INSERT INTO iscrizione VALUES ('false', '9', '3');
INSERT INTO iscrizione VALUES ('false', '10', '3');
INSERT INTO iscrizione VALUES ('false', '11', '3');
INSERT INTO iscrizione VALUES ('false', '12', '4');
INSERT INTO iscrizione VALUES ('false', '13', '4');
INSERT INTO iscrizione VALUES ('false', '14', '4');
INSERT INTO iscrizione VALUES ('false', '12', '5');
INSERT INTO iscrizione VALUES ('false', '15', '5');
INSERT INTO iscrizione VALUES ('false', '10', '5');
INSERT INTO iscrizione VALUES ('false', '5', '5');
INSERT INTO iscrizione VALUES ('false', '16', '6');
INSERT INTO iscrizione VALUES ('false', '17', '6');
INSERT INTO iscrizione VALUES ('false', '10', '6');
INSERT INTO iscrizione VALUES ('false', '18', '6');
INSERT INTO iscrizione VALUES ('false', '6', '7');
INSERT INTO iscrizione VALUES ('false', '7', '7');
INSERT INTO iscrizione VALUES ('false', '19', '7');
```

13.10 PARTECIPAZIONE

```
1  INSERT INTO partecipazione VALUES ('1','1');
2  INSERT INTO partecipazione VALUES ('1','2');
3  INSERT INTO partecipazione VALUES ('1','3');
4  INSERT INTO partecipazione VALUES ('1','5');
5  INSERT INTO partecipazione VALUES ('2','1');
6  INSERT INTO partecipazione VALUES ('2','2');
7  INSERT INTO partecipazione VALUES ('2','3');
8  INSERT INTO partecipazione VALUES ('3','1');
9  INSERT INTO partecipazione VALUES ('3','2');
10 INSERT INTO partecipazione VALUES ('3','3');
11 INSERT INTO partecipazione VALUES ('4','1');
12 INSERT INTO partecipazione VALUES ('4','2');
13 INSERT INTO partecipazione VALUES ('5','1');
14 INSERT INTO partecipazione VALUES ('5','2');
15 INSERT INTO partecipazione VALUES ('6','2');
16 INSERT INTO partecipazione VALUES ('6','4');
17 INSERT INTO partecipazione VALUES ('6','5');
18 INSERT INTO partecipazione VALUES ('7','2');
19 INSERT INTO partecipazione VALUES ('7','4');
20 INSERT INTO partecipazione VALUES ('7','5');
21 INSERT INTO partecipazione VALUES ('8','4');
22 INSERT INTO partecipazione VALUES ('9','2');
23 INSERT INTO partecipazione VALUES ('9','4');
24 INSERT INTO partecipazione VALUES ('10','4');
25 INSERT INTO partecipazione VALUES ('11','5');
26 INSERT INTO partecipazione VALUES ('11','9');
27 INSERT INTO partecipazione VALUES ('11','10');
28 INSERT INTO partecipazione VALUES ('11','11');
29 INSERT INTO partecipazione VALUES ('12','9');
30 INSERT INTO partecipazione VALUES ('12','10');
31 INSERT INTO partecipazione VALUES ('12','11');
32 INSERT INTO partecipazione VALUES ('13','9');
33 INSERT INTO partecipazione VALUES ('13','10');
34 INSERT INTO partecipazione VALUES ('13','11');
35 INSERT INTO partecipazione VALUES ('14','9');
36 INSERT INTO partecipazione VALUES ('14','10');
37 INSERT INTO partecipazione VALUES ('15','12');
```

```
38 INSERT INTO partecipazione VALUES ('15','13');
39 INSERT INTO partecipazione VALUES ('15','14');
40 INSERT INTO partecipazione VALUES ('16','12');
41 INSERT INTO partecipazione VALUES ('16','14');
42 INSERT INTO partecipazione VALUES ('17','12');
43 INSERT INTO partecipazione VALUES ('17','13');
44 INSERT INTO partecipazione VALUES ('17','14');
45 INSERT INTO partecipazione VALUES ('18','13');
46 INSERT INTO partecipazione VALUES ('19','5');
47 INSERT INTO partecipazione VALUES ('19','12');
48 INSERT INTO partecipazione VALUES ('19','10');
49 INSERT INTO partecipazione VALUES ('19','15');
50 INSERT INTO partecipazione VALUES ('20','5');
51 INSERT INTO partecipazione VALUES ('20','12');
52 INSERT INTO partecipazione VALUES ('20','15');
53 INSERT INTO partecipazione VALUES ('21','12');
54 INSERT INTO partecipazione VALUES ('21','10');
55 INSERT INTO partecipazione VALUES ('21','15');
56 INSERT INTO partecipazione VALUES ('22','17');
57 INSERT INTO partecipazione VALUES ('22','18');
58 INSERT INTO partecipazione VALUES ('22','10');
59 INSERT INTO partecipazione VALUES ('22','16');
60 INSERT INTO partecipazione VALUES ('23','16');
61 INSERT INTO partecipazione VALUES ('23','18');
62 INSERT INTO partecipazione VALUES ('23','10');
63 INSERT INTO partecipazione VALUES ('24','17');
64 INSERT INTO partecipazione VALUES ('24','10');
65 INSERT INTO partecipazione VALUES ('24','16');
66 INSERT INTO partecipazione VALUES ('25','18');
67 INSERT INTO partecipazione VALUES ('25','17');
68 INSERT INTO partecipazione VALUES ('26','6');
69 INSERT INTO partecipazione VALUES ('26','7');
70 INSERT INTO partecipazione VALUES ('26','19');
71 INSERT INTO partecipazione VALUES ('27','6');
72 INSERT INTO partecipazione VALUES ('27','7');
73 INSERT INTO partecipazione VALUES ('27','19');
74 INSERT INTO partecipazione VALUES ('28','19');
```

13.11 PAROLE_LEZIONE

```
INSERT INTO parole_lezione VALUES('oratoria', '2');
INSERT INTO parole_lezione VALUES('oratoria', '3');
INSERT INTO parole_lezione VALUES('oratoria', '4');
INSERT INTO parole_lezione VALUES('oratoria', '5');
INSERT INTO parole_lezione VALUES('eloquenza','3');
INSERT INTO parole_lezione VALUES('convincimento','4');
INSERT INTO parole_lezione VALUES('parlare','5');
INSERT INTO parole_lezione VALUES('parlare','2');
INSERT INTO parole_lezione VALUES('performance', '5');
INSERT INTO parole_lezione VALUES('note', '7');
INSERT INTO parole_lezione VALUES('note', '8');
INSERT INTO parole_lezione VALUES('note', '9');
INSERT INTO parole_lezione VALUES('note', '10');
INSERT INTO parole_lezione VALUES('spartito', '7');
INSERT INTO parole_lezione VALUES('performance', '8');
INSERT INTO parole_lezione VALUES('ritmo', '10');
INSERT INTO parole_lezione VALUES('denaro', '11');
INSERT INTO parole_lezione VALUES('denaro', '12');
INSERT INTO parole_lezione VALUES('risparmio', '12');
INSERT INTO parole_lezione VALUES('denaro', '13');
INSERT INTO parole_lezione VALUES('risparmio', '13');
INSERT INTO parole_lezione VALUES('denaro', '14');
INSERT INTO parole_lezione VALUES('investimenti', '14');
INSERT INTO parole_lezione VALUES('programmazione', '15');
INSERT INTO parole_lezione VALUES('programmazione', '16');
INSERT INTO parole_lezione VALUES('programmazione', '17');
INSERT INTO parole_lezione VALUES('sviluppo', '17');
INSERT INTO parole_lezione VALUES('programmazione', '18');
INSERT INTO parole_lezione VALUES('sviluppo', '18');
INSERT INTO parole_lezione VALUES('software', '18');
INSERT INTO parole_lezione VALUES('software', '19');
INSERT INTO parole_lezione VALUES('note', '20');
INSERT INTO parole_lezione VALUES('spartito', '20');
INSERT INTO parole_lezione VALUES('ritmo', '20');
INSERT INTO parole_lezione VALUES('composizione', '20');
INSERT INTO parole_lezione VALUES('mixing', '21');
INSERT INTO parole_lezione VALUES('mastering', '21');
```

```
38 INSERT INTO parole_lezione VALUES('parlare', '22');
39 INSERT INTO parole_lezione VALUES('lessico', '22');
40 INSERT INTO parole_lezione VALUES('grammatica', '23');
41 INSERT INTO parole_lezione VALUES('grammatica', '24');
42 INSERT INTO parole_lezione VALUES('grammatica', '25');
43 INSERT INTO parole_lezione VALUES('lessico', '25');
44 INSERT INTO parole_lezione VALUES('programmazione', '26');
45 INSERT INTO parole_lezione VALUES('programmazione', '27');
46 INSERT INTO parole_lezione VALUES('programmazione', '28');
47 INSERT INTO parole_lezione VALUES('sviluppo', '27');
48 INSERT INTO parole_lezione VALUES('sviluppo', '28');
```

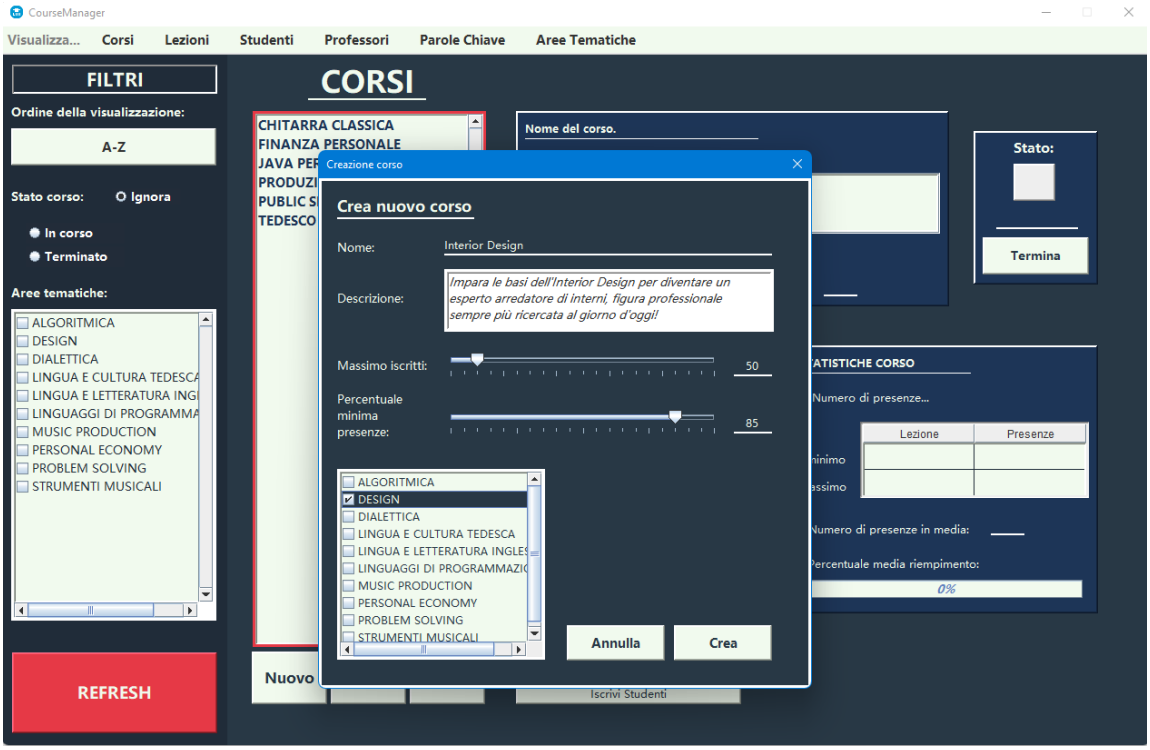
13.12 OPERATORI

```
1 INSERT INTO operatore VALUES ('matteo_giordano', 'password1');
2 INSERT INTO operatore VALUES ('gian_marco_addati', 'password2');
3 INSERT INTO operatore VALUES ('admin', 'admin);
```

Parte VI
Guida per l'utente

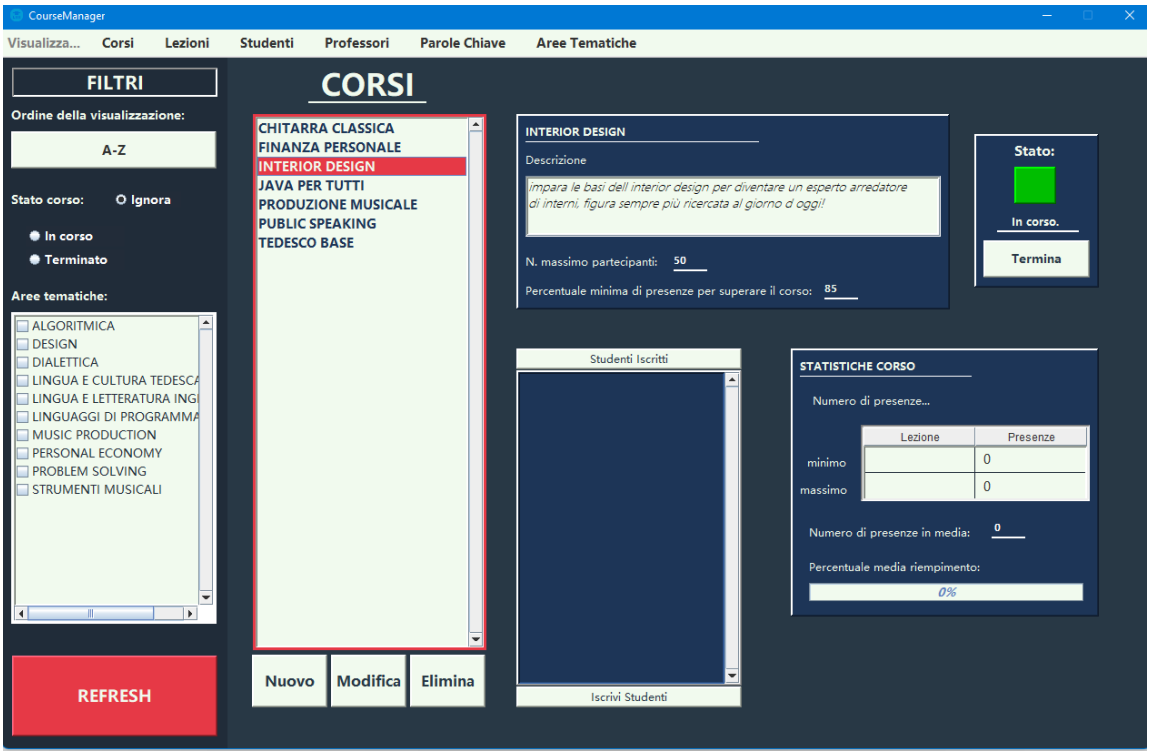
14 Esempi d'uso

14.1 Inserimento di un corso



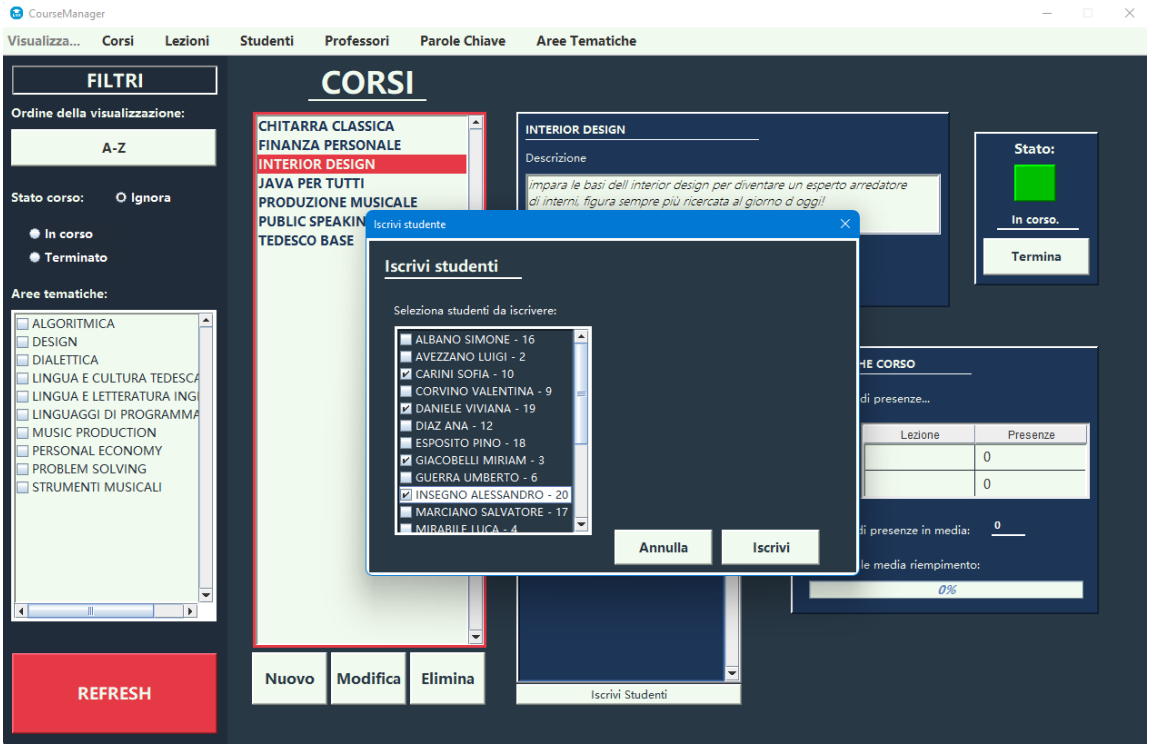
- Vai sulla schermata dei Corsi.
- Fai click su "Nuovo".
- Inserisci il titolo, la descrizione, il numero massimo di iscritti, la percentuale minima di presenze e una o più aree tematiche di riferimento.
- Fai click su "Crea".
- Il corso è stato creato con successo!

14.2 Informazioni del corso appena inserito



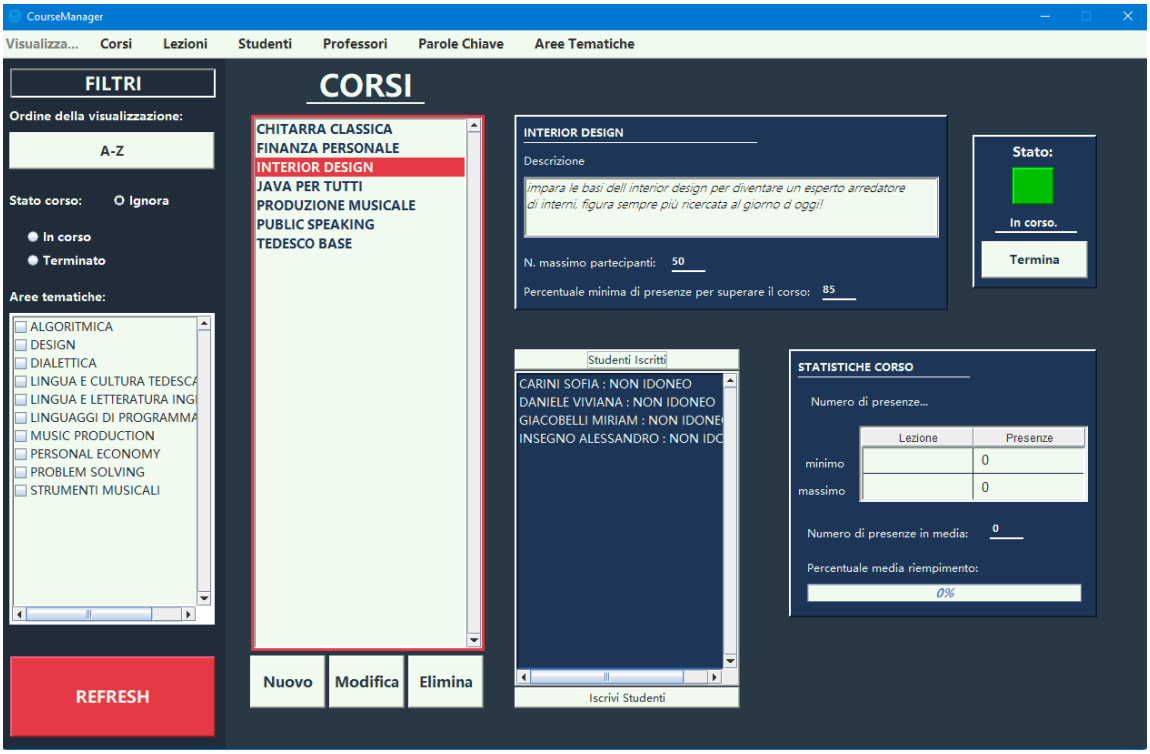
- Vai sulla schermata dei Corsi.
- Fai click sul corso al quale sei interessato.
- Sulla destra avrai tutte le informazioni relative al corso, come il titolo, la descrizione, le statistiche, gli studenti iscritti e le lezioni ad esso associate.

14.3 Iscrizione di studenti ad un corso



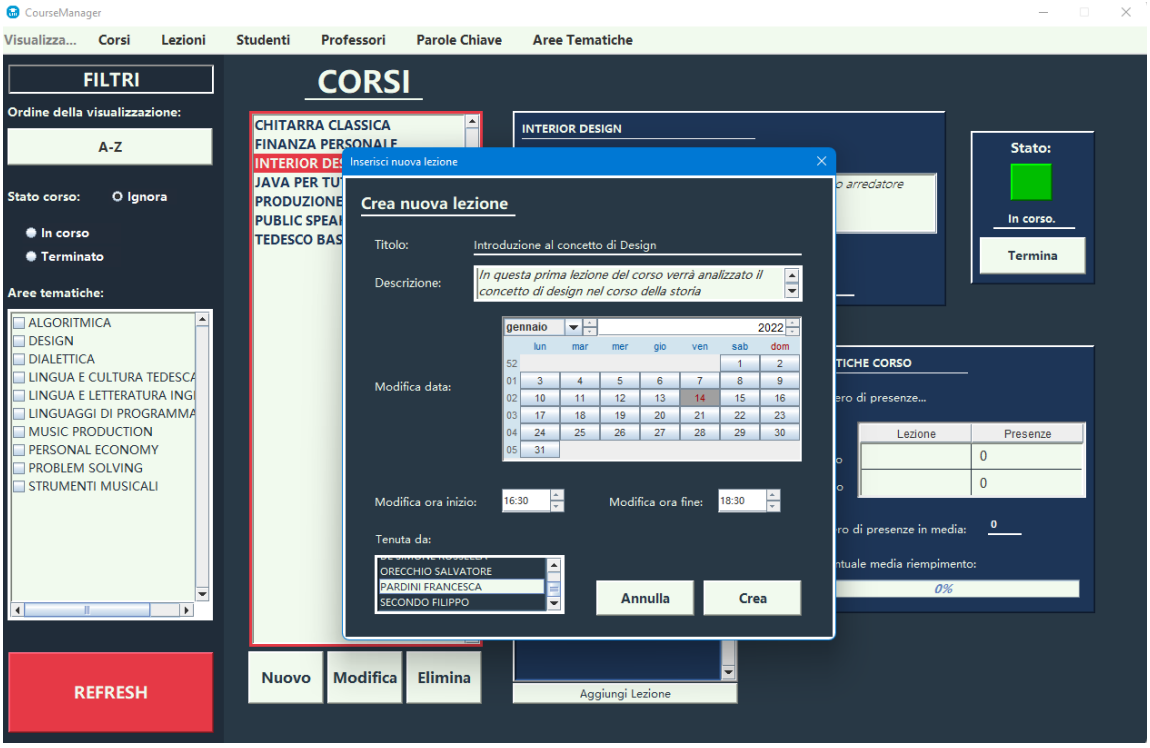
- Vai sulla schermata dei Corsi
- Fai click su "Iscrivi Studenti"
- Qui avrai visibilità di tutti gli studenti NON ancora iscritti al corso
- Seleziona uno o più studenti da iscrivere al corso
- Fai click su "Iscrivi"
- Hai iscritto gli studenti al corso!

14.4 Visualizzazione studenti iscritti al corso



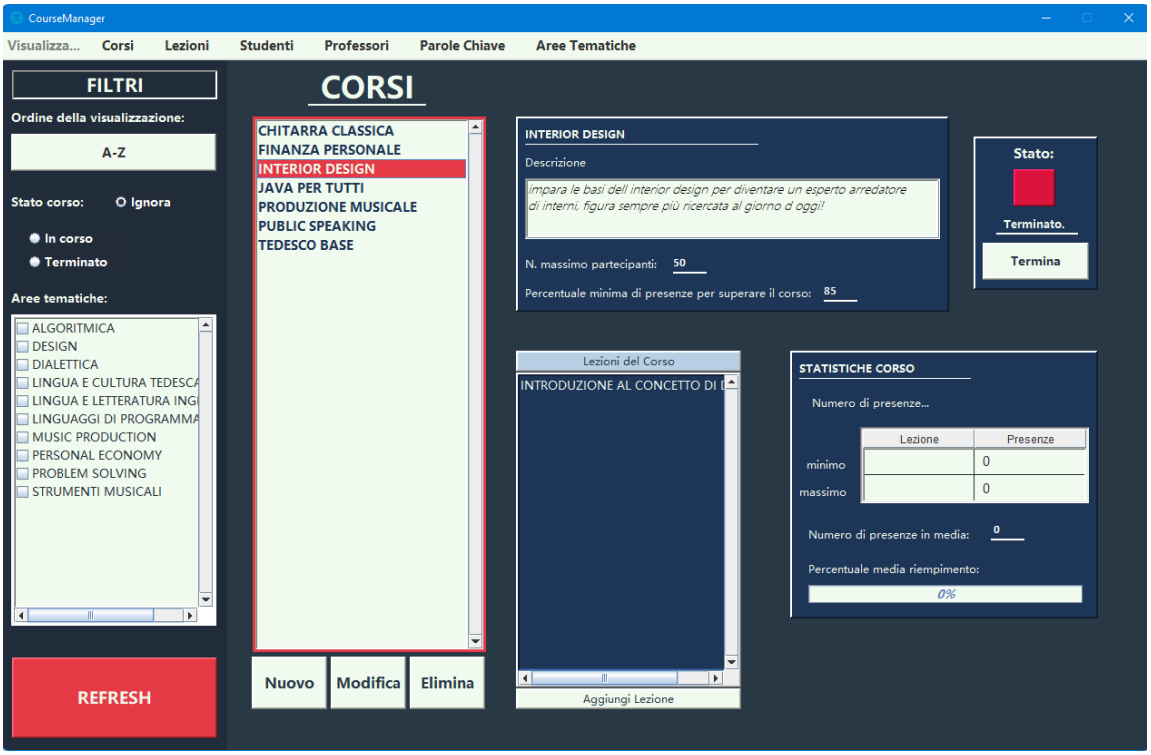
- Vai sulla schermata dei Corsi.
- Dopo aver iscritto degli studenti ad un corso, troverai i loro nomi nelle informazioni del corso
- Il nome di ciascuno studente sarà seguito dallo stato della sua iscrizione. N.B: All'interno dei corsi non ancora terminati, tutti gli studenti risulteranno NON idonei.

14.5 Inserimento di una lezione ad un corso



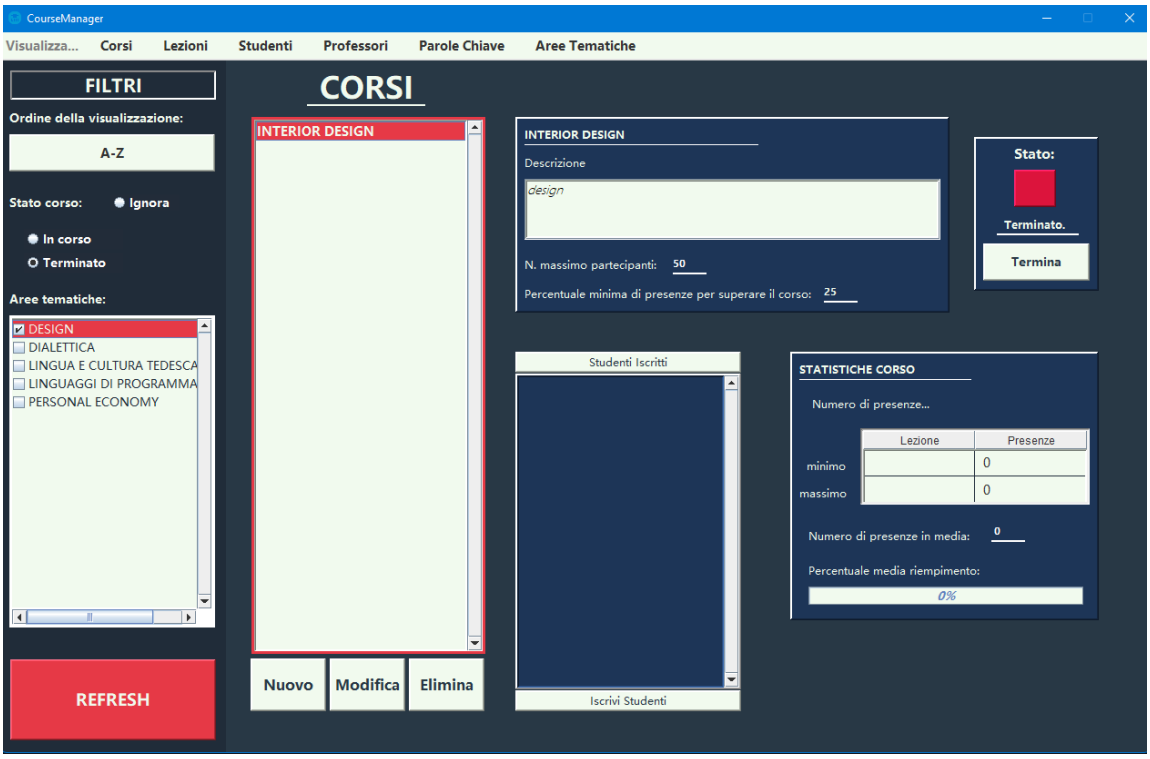
- Vai sulla schermata dei Corsi.
- Fai click su "Aggiungi Lezione" sotto la lista blu al centro dello schermo. Se non dovesse risultare disponibile, premere sul bottone in cima alla lista. Esso permette di passare dalla visualizzazione degli iscritti al corso a quella delle lezioni del corso, e viceversa.
- Inserire titolo e descrizione della lezione. Scegli una data e un'ora per la lezione. Ricorda che la durata massima per una lezione è **2 ore** e una lezione si può svolgere **dalle 8:00 alle 20:00**. Infine scegli il professore che tiene la lezione.
- Premi su "Crea".

14.6 Terminazione di un corso



- Vai sulla schermata dei Corsi.
- Fai click su "Termina" in alto a destra. Si aprirà una schermata di conferma.
- Fai click su OK. Ricorda che l'operazione di terminazione di un corso è irreversibile.
- Le statistiche sugli studenti si sono aggiornate: ora gli studenti con un numero di presenze sufficiente risultano idonei.

14.7 Filtraggio



- Vai sulla schermata dei Corsi.
- Sulla sinistra, nel pannello "FILTRI" sono elencati i possibili filtri selezionabili per questo pannello.
- Scegli i filtri da applicare, ad esempio solo corsi terminati e appartenenti all'area tematica Design.
- Fai click sul bottone rosso "REFRESH". La nuova lista di corsi filtrati appare nella colonna bianca sotto "CORSI".
- Ogni schermata (Corsi, Lezioni, Studenti...) ha a disposizione filtri diversi.