TEMA DE CASA

BAZE DE DATE

Gestiunea unei scoli de balet

Nume: Iorga Elena

Grupa: 1310B

Profesor coordonator: dr. Cătălin Mironeanu

Cuprins

1. Descrierea proiectului

O scoala de balet are la dispozitie 6 sali, instructori si pianisti care ii vor ajuta pe elevi sa evolueze, iar rezultatele lor se vor vedea atunci cand participa la spectacole. Daca un elev participa la mai mult de 4 spectacole (>=5), va evolua la o grupa mai avansata (o grupa total diferita care va avea un alt pianist si un alt instructor) , iar intrarile care s-au luat in considerare la numaratoare vor fi sterse. Acestia vor intra in scoala intr-o grupa de incepatori, by default va fi grupa 1, scoala avand 2 grupe de acest gen. Intr-o grupa pot fi maxim 10 elevi. Daca un elev care deja este profesionist are numarul suficient de spectacole, atunci el nu mai poate avansa. Daca se incearca avansarea lui, acesta va ramane exact unde este, iar spectacolele la care a participat nu vor fi sterse din tabela. Interfata permite adaugare, editare, stergere si chiar avansare a unor noi elevi, adaugarea de noi spectacole, grupe, pianisti si stergerea lor.
O grupa nu poate avea decat maxim 10 elevi, deci daca se incearca inserarea unui nou elev in acea grupa, aceasta nu se va executa.

2. Tehnologii folosite (pentru front-end si back-end)

A fost folosit in prima faza Data Modeler pentru a crea diagrama logica si cea relationala, de unde a fost generat mai apoi script-ul pentru generarea tabelelor. In continuare, baza de date a fost creata in SQL Developer .
Interfata (butoanele, culorile, etc) a fost implementata in HTML, iar pentru a realiza functionalitatea ei (adaugarea elevilor in baza de date, stergerea, editarea, avansarea, etc) a fost folosit Python, Flask.

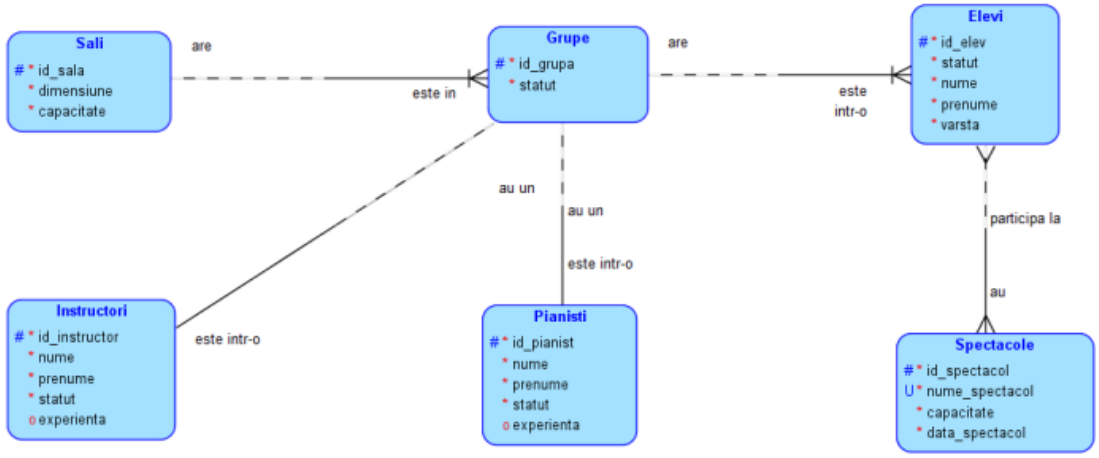
3. Structura si inter-relationarea tabelelor

Tabelele din aceasta aplicatie sunt:

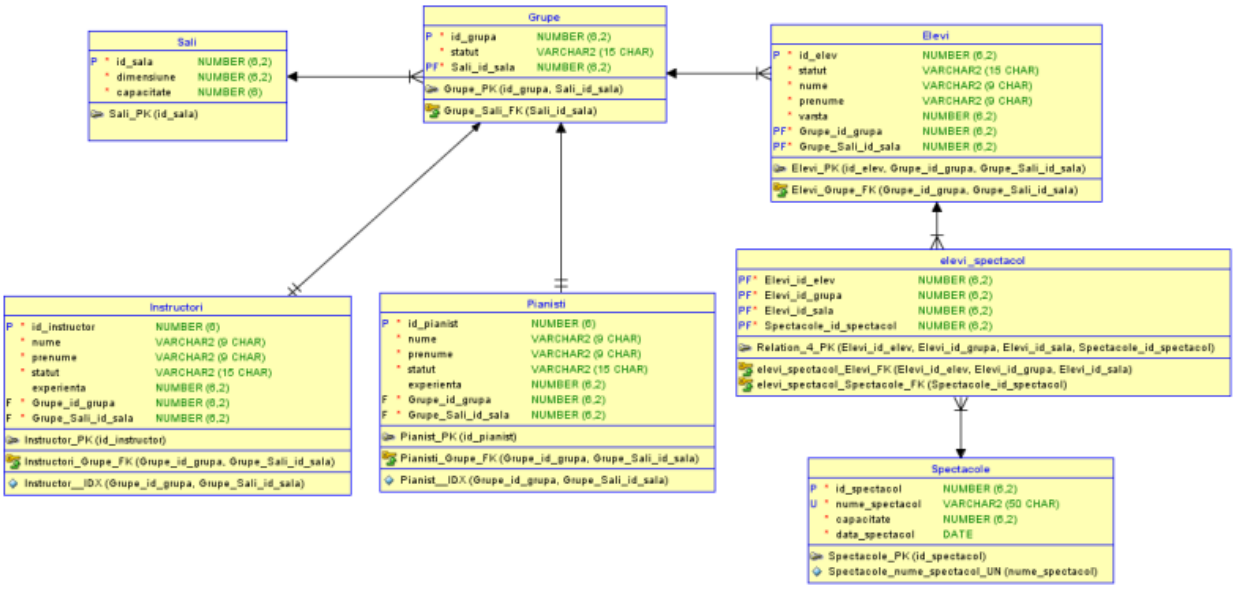
- Sali
- Grupe
- Instructori
- Pianisti
- Elevi
- Spectacole
- ‘elevi_spectacol’

  Intre tabelele Sali si Grupe, exista o relatie one to many (1:n). La fel si intre Grupe si Elevi.
  Avem si doua relatii one-to-one intre Grupe si Instructori si Grupe si Pianisti.
  Relatia many to many poate fi observata intre tabelele Elevi si Spectacole.

Modelul logic



Modelul relational



Observam ca relatia de many to many dintre tabelele Elevi si Spectacole a fost sparta in doua relatii one to many (1:n) , iar intre cele doua tabele a fost introdus un nou tabel elevi_spectacol care monitorizeaza la ce spectacole a participat un anumit elev. Sunt 2 relatii de 1:1 intre Grupe:Instructori si Grupe:Pianisti, restul relatiilor fiind de 1:n.

Fiecare id din fiecare tabela este introdus cu ajutorul autoincrementului : id_sala,id_grupa, id_elev, id_instructor, id_pianist, id_spectacol. Acest lucru ne ajuta sa nu mai introducem manual id-ul, asigurandu-ne ca fiecare id va avea o valoare diferita.

4. Constrangerile folosite

4.1 Constrangeri de integritate referentiala (primary key si foreign key)

- In tabela **Sali:**
  - O constrangere de tip **primary key** pentru id_sala pentru a ii atribui fiecareisali un id unic si pentru a fi preluat in alte tabele
- In tabela **Grupe**:
  - O constrangere de tip **primary key** pentru coloanal id_grupa pentru a ii atribui fiecarei grupe un id unic, pentru a fi preluat in alte tabele
  - O constrangere de tip **P Foreign Key** pentru Sali_id_sala care arata ca Sali_id_sala este extras din tabela Sali (fiind o relatie de 1:n intre Grupe:Sali) unde id_sala este de tip primary
- In tabela Instructori:
  - O constrangere de tip **primary key** pentru id_instructor intrucat fiecare instructor trebuie sa aiba un id unic si trebuie sa fie preluat in alte tabele
  - O constrangere de tip **foreign key** pentru Grupe_id_grupa care se datoreazarelatiei de 1:1 intre instructori:grupe, id-ul grupei fiind preluat astfel in aceastacoloana
  - Inca o constrangere de tip **foreign key** pentru Grupe_Sali_id_sala care se datoreaza faptului ca in tabela grupe cu care are relatie de 1:1 este un primary (foreign) key pe Sali_id_sala
- In tabela **Pianisti**:
  - O constrangere de tip **primary key** pentru id_pianist intrucat fiecare pianist trebuie sa aiba un id unic si trebuie sa fie preluat in alte tabele
  - O constrangere de tip **foreign key** pentru Grupe_id_grupa care se datoreaza relatiei de 1:1 intre pianisti:grupe, id-ul grupei fiind preluat astfel in aceasta coloana
  - Inca o constrangere de tip **foreign key** pentru Grupe_Sali_id_sala care se datoreaza faptului ca in tabela grupe cu care are relatie de 1:1 este un primary (foreign) key pe Sali_id_sala
- In tabela **Elevi**:
  - O constrangere de tip **primary key** pentru id_elev (fiecare elev trebuie saaiba un id unic pentru a fi preluat de alte tabele)
  - O constrangere de tip **foreign key** (care este si primary) pentru coloana Grupe_id_grupa care se datoreaza legaturii de 1:n dintre Elevi:Grupe, fiind preluat id-ul grupei pentru a putea stii in ce grupa se afla elevul
  - O constrangere de tip **foreign key** (care este si primary) pentru coloana Grupe_Sali_id_sala care se datoreaza relatiei dintre elevi si grupe , grupe avand ca primary key(foreign) care provine din tabela Sali atributul Sali_id_sala care a fost preluat de coloana noastra ca Grupe_Sali_id_sala
- In tabela **elevi_spectacol**:
  - Avem 4 constrangeri de tip **Primary Foreign Key** care sunt preluate din tabelele Elevi si Spectacole deoarece aceasta tabela este cea care face legatura dintre Elevi si Spectacole. Ea a rezultat in urma spargerii legaturii dem:n, fiind astfel in legaturi de 1:n cu 1:Elevi si 1:n cu 1:Spectacole, avand ca coloane: Elevi_id_elev, Elevi_id_grupa, Elevi_id_sala, Spectacole_id_spectacol, toate fiind PF-uri.
- In tabela **Spectacole**:
  - O constrangere de tip **primary key** pentru id_spectacol care diferentiaza fiecare

intrare in tabela si care este preluata de tabelele care sunt legate deacesta.

- Mai exista o tabela separata de celelalte, tabela **users**. Acesta va avea campurile id_user si parola, dintre care id_user va fi un primary key care diferentiaza fiecare intrare in tabela.

4.2 Constrangeri de integritate de tip check, unique, not null

- In tabela **Sali**:
  - Toate coloanele au constrangeri de tip **NOT NULL** care marcheaza faptul casunt obligatorii
  - Coloana dimensiune are o constrangere de tip **CHECK** pentru a marca faptulca dimensiunea trebuie sa fie mai mare decat 10
  - Coloana capacitate are o constrangere de tip **CHECK** care verifica valoarea introdusa intrucat aceasta trebuie sa fie intre 0 si 10 (intr-o sala nu pot incapea maim ult de 10 elevi cum nici intr-o grupa nu pot fi mai mult de 10 elevi)
- In tabela **Spectacole**:
  - Toate coloanele au constrangeri de tip **NOT NULL** care marcheaza faptul casunt obligatorii
  - Coloana capacitate are o constrangere de tip **CHECK** pentru a nu permite participarea a mai mult de 10 elevi la un spectacol , dar si pentru evitarea introducerii unui numar eronat de exemplu -1 asa ca avem capacitate BETWEEN 0 AND 10
  - Coloana nume_spectacol are o constrangere de tip **UNIQUE** pentru a ne asigura ca nu vor fi introduse doua spectacole cu acelasi nume
  - Pentru coloana data_spectacol a fost introdus un **trigger** deoarece nu pot introduce un spectacol care nu a avut loc inca, iar in acest trigger se va face comparatie cu data curenta
- In tabela **Elevi**:
  - Toate coloanele au constrangeri de tip **NOT NULL** care marcheaza faptul casunt obligatorii
  - Coloana statut are ca default valoarea ‘incepator’
  - Coloana statut are o constrangere de tip **CHECK** pentru ca statutul sa nu ia valori decat in lista : 'avansat', 'incepator', 'mediu', 'profesionist' cu numele de statut_elev_check
  - Coloana varsta are o constrangere de tip **CHECK** pentru ca elevii introdusisa nu fie mai mici de 4 ani sau mai mari de 30 cu numele de varsta_check
  - Numele si prenumele nu trebuie sa aiba o lungime mai mica decat 1 si nu trebuie sa contina cifre. Numele constrangerilor **CHECK** pentru acestea sunt:elevi_nume_ck, elevi_prenume_ck, elevi_prenume2_ck, elevi_nume2_ck
- In tabela **Elevi_spectacol**:
  - Toate coloanele au constrangeri de tip **NOT NULL** care marcheaza faptul ca sunt obligatorii
- In tabela **Grupe**
  - Toate coloanele au constrangeri de tip **NOT NULL** care marcheaza faptul ca sunt obligatorii
  - Coloana statut are o constrangere de tip **CHECK** pentru ca statutul sa nu iavalori decat in lista : 'avansat', 'incepator', 'mediu', 'profesionist' si are valoarea default

'incepator' care se numeste statut_grupa_check

- In tabela **Instructori**:
  - In afara de coloana experienta (deoarece doar aceasta nu este obligatorie),toate coloanele sunt de tip **NOT NULL**
  - Coloana statut are o constrangere de tip **CHECK** pentru ca statutul sa nu iavalori decat in lista : 'avansat', 'incepator', 'mediu', 'profesionist' si are valoarea default 'incepator' care se numeste statut_check_instructor
  - Coloanele de nume si prenume nu pot avea o lungime mai mica decat 1, asaca au constrangeri de tip **CHECK**, si nici nu pot contine cifre (pentru asta a fost introdusa o alta constrangere de tip **CHECK**), numele acestor constrangeri fiind: instructori_nume_ck, instructori_prenume_ck, instructori_prenume2_ck, instructori_nume2_ck
  - Coloana experienta daca este introdusa trebuie sa fie mai mare decat 1 , verificarea se face printr-o constrangere de tip **CHECK** care se numeste: experienta_instructrori_check
- In tabela **Pianisti**:
  - In afara de coloana experienta (deoarece doar aceasta nu este obligatorie),toate coloanele sunt de tip **NOT NULL**
  - Coloana statut are o constrangere de tip **CHECK** pentru ca statutul sa nu iavalori decat in lista : 'avansat', 'incepator', 'mediu', 'profesionist' si are valoarea default 'incepator' care se numeste: statut_pianist_check
  - Coloanele de nume si prenume nu pot avea o lungime mai mica decat 1, asaca au constrangeri de tip **CHECK**, si nici nu pot contine cifre (pentru asta a fost introdusa o alta constrangere de tip **CHECK**), constrangerile numindu-se : pianisti_nume_ck, pianisti_prenume_ck, pianisti_prenume2_ck, pianisti_nume2_ck
  - Coloana experienta daca este introdusa trebuie sa fie mai mare decat 1 , verificarea se face printr-o constrangere de tip **CHECK** , constrangerea numindu-se: experienta_pianisti_check

5. Modalitate de conectare la baza de date din aplicatie

A fost folosita din libraria cx_Oracle functia cx_Oracle.connect(...) folosind informatiile de conectare la baza de date.

```
con = cx_Oracle.connect(user='bd148', password="danuandsa",dsn="bd
dc.cs.tuiasi.ro:1539/orcl")
```

Apoi, cu ajutorul obiectului cursor din Python, executam cererile pe care le avem la baza de date si astfel facem rost de inregistrarile din baza de date.

6. Capturi de ecran si exemple de cod

Odata cu deschiderea interfetei, suntem intampinati de pagina de login unde se pot loga doar cei care au id_user-ul si parola stocate in baza de date (pentru a evita accesul oricui la baza de date) :

Parola introdusa nu este vizibila. Daca utilizatorul e correct si corespune cu datele din baza de date, atunci acesta va fi redirectionat catre interfata cu baza de date a scolii de balet. Daca nu, va fi redirectionat din nou catre pagina de Login pentru a mai incerca o data. Codul din spatele butonului de logare este:

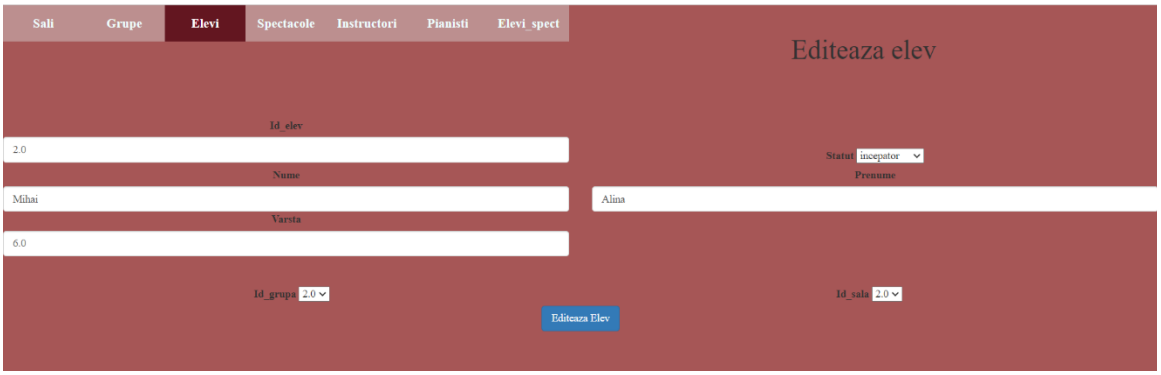
```
@app.route('/logare', methods=['POST'])
def login():
   if request.method == 'POST':
      cur = con.cursor()
      id = request.form['id_user']
      pas=request.form['parola']
      query='select parola from users where id_user=%s'%("'"+id+"'")
      cur.execute(query)
      vals=cur.fetchall()
      print(vals)
      if(len(vals)==0):
         val=None
      else:
         val=vals[0][0]
      if(val==pas):
         return redirect('/sali')
      else:
         return redirect('/')
```

In primul rand, interfata afiseaza intrarile din baza de date, iar in meniul de sus putem selecta tabela pe care vrem sa o vizionam sau schimbam. De exemplu, pentru tabela Elevi in care este selectata in bara de sus tabela, avem in html-ul tabelei :

```
<a href="sali"><button class="tablink">Sali</button></a>
<a href="grupe"><button class="tablink">Grupe</button></a>
<a href="elevi"><button class="tablink" style="background:rgb(99,22,32); color:
white">Elevi</button></a>
<a href="spectacole"><button class="tablink">Spectacole</button></a>
<a href="instructori"><button class="tablink">Instructori</button></a>
<a href="pianisti"><button class="tablink">Pianisti</button></a>
<a href="elevi_spect"><button class="tablink">Elevi_spect</button></a>
```

Apoi, se observa ca fiecare inregistrare din aceasta tabela are optiunea de a edita elevul, de a-l sterge si de a-l avansa.

Atunci cand se apasa butonul de **editare**, suntem trimisi pe aceasta pagina in care putem edita inregistrarea, dar ne sunt afisate si valorile anterioare pe care le avea:



Codul din spatele acestui buton este:

```
@app.route('/getElev', methods=['POST'])
def get_elev():
```

```
    id = request.form['id_elev']
    cur = con.cursor()
    cur.execute('select * from elevi where id_elev=' + id)

    ids = cur.fetchone()
    id_elev = ids[0]
    statut = ids[1]
    nume = ids[2]
    prenume = ids[3]
    varsta = ids[4]
    grupe_id_grupa = ids[5]
    grupe_sali_id_sala = ids[6]

    grupe = []
    cur.execute('select id_grupa from grupe')
    for result in cur:
        grupe.append(result[0])
    sali = []
    cur.execute('select id_sala from sali')
    for result in cur:
        sali.append(result[0])

    cur.close()
    return render_template('editElevi.html', id_elev=id_elev, statut=statut,
nume=nume,prenume=prenume, varsta=varsta, grupe_id_grupa=grupe_id_grupa,
grupe_sali_id_sala=grupe_sali_id_sala, grupe=grupe, sali=sali)


@app.route('/editElev', methods=['POST'])
def edit_elev():
    els=0
    cur = con.cursor()

    statut = "'" + request.form['statut'] + "'"
    nume = "'" + request.form['nume'] + "'"
    cur.execute('select id_elev from elevi where nume=' + nume)
    for result in cur:
        els = result[0]
    cur.close()

    prenume= "'" + request.form['prenume'] + "'"
    statut = "'" + request.form['statut']+ "'"
    varsta= request.form['varsta']
    grupe_id_grupa=request.form['grupe_id_grupa']
    grupe_sali_id_sala=request.form['grupe_sali_id_sala']

    cur = con.cursor()
    query = "UPDATE elevi SET statut=%s, nume=%s, prenume=%s, varsta=%s,
grupe_id_grupa=%s, grupe_sali_id_sala=%s  where id_elev=%s" % (
    statut, nume, prenume, varsta, grupe_id_grupa, grupe_sali_id_sala, els)
    cur.execute(query)

    return redirect('/elevi')
```

Functia get_elev este folosita pentru intrarea in pagina pentru editarea elevilor, pentru a pastra valorile pe care le avea inregistrarea de dinainte. Functia edit_elev este cea care editeaza elevul, punand noi valori daca au fost schimbate in locul celor vechi, sau doar pastreaza valorile care erau.

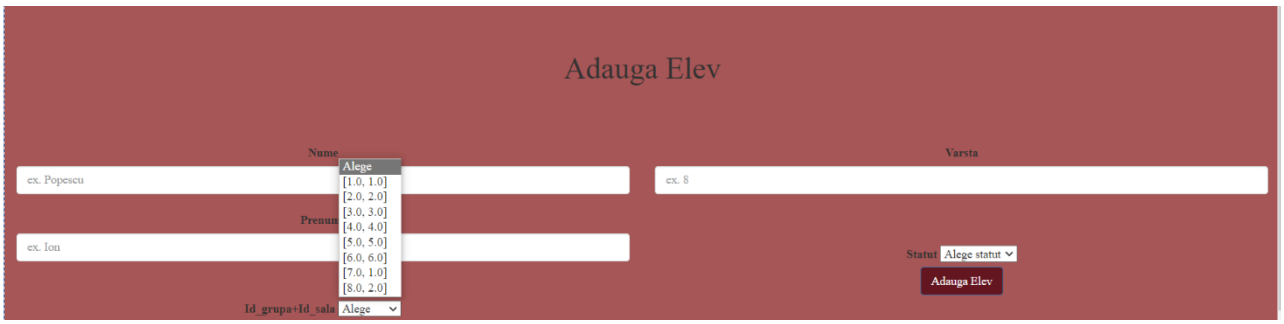
Butonul de stergere are in spate codul:

```
@app.route('/delElev', methods=['POST'])
def del_elev():
    id = "'"+request.form['id_elev']+"'"
    cur = con.cursor()
    cur.execute('delete from elevi where id_elev=' + id)
    cur.execute('commit')
    return redirect('/elevi')
```

El doar sterge din tabela elevul cu respectivul id (care este unic).
Codul din spatele butonului de editare este:

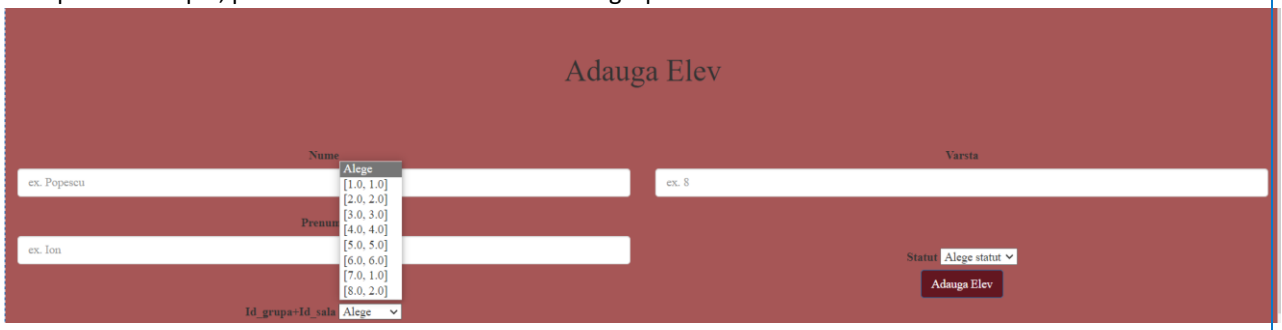
```
@app.route('/editElev', methods=['POST'])
def edit_elev():
    els=0
    cur = con.cursor()

    statut = "'" + request.form['statut'] + "'"
    nume = "'" + request.form['nume'] + "'"
    cur.execute('select id_elev from elevi where nume=' + nume)
    for result in cur:
        els = result[0]
    cur.close()

    prenume= "'" + request.form['prenume'] + "'"
    statut = "'" + request.form['statut']+ "'"
    varsta= request.form['varsta']
    grupe_id_grupa=request.form['grupe_id_grupa']
    grupe_sali_id_sala=request.form['grupe_sali_id_sala']

    cur = con.cursor()
    query = "UPDATE elevi SET statut=%s, nume=%s, prenume=%s, varsta=%s,
grupe_id_grupa=%s, grupe_sali_id_sala=%s  where id_elev=%s" % (
    statut, nume, prenume, varsta, grupe_id_grupa, grupe_sali_id_sala, els)
    cur.execute(query)

    return redirect('/elevi')
```

Acesta face un update in tabela acolo unde facem schimbarile.
Butonul de avansare este cel care va avansa in functie de numarul de spectacole la care a participat elevul. Daca este un incepator si a participat la >=5 spectacole, atunci acesta va fi avansat la o grupa de incepatori in care incape, inregistrarile din tabela elevi_spectacol cu acest elev vor fi sterse. Daca nu mai sunt locuri in aceasta, el va ramane in grupa lui si inregistrarile din tabela elevi_spectacol vor ramane neschimbate.
Acest lucru s-a facut verificand mai intai la cate spectacole a participat, apoi in ce grupa are loc, stergandu-se ceea ce trebuie si dupa readaugarea elevului intr-o alta grupa, cu un alt status. Daca nu are loc, se commite un rollback pentru a nu se sterge ceea ce trebuie.
Adaugarea in tabela elevi intr-o anumita grupa are loc doar daca acea grupa are mai putin de 10 elevi.



TEMA DE CASA

Se observa ca foreign keys-urile nu pot fi adaugate decat prin alegerea lor din cele existente din tabelele din care provin ca tuple, pentru a nu cauta in ce sala este si grupa elevului.



Codul din spatele acestei functionalitati este:

```
@app.route('/addElev', methods=['POST'])
def ad_elev():
   error = None
   if request.method == 'POST':
      cur = con.cursor()
      values = []

      values.append("'" + request.form['nume'] + "'")
      values.append("'" + request.form['varsta'] + "'")
      values.append("'" + request.form['prenume'] + "'")
      values.append("'" + request.form['statut'] + "'")
      values.append("'" + request.form['grupe_id_grupa'] + "'")
      values.append("'" + request.form['grupe_sali_id_sala'] + "'")
      fields = [ 'nume', 'varsta',
'prenume','statut','grupe_id_grupa','grupe_sali_id_sala']
      flag=0
      query1='SELECT COUNT(id_elev) FROM elevi WHERE grupe_id_grupa=%s'% ("'" +
request.form['grupe_id_grupa'] + "'")
      cur.execute(query1)
      val=cur.fetchall()
      if(val[0][0]<10):
         query = 'INSERT INTO %s (%s) VALUES (%s)' % (
         'elevi',
         ', '.join(fields),
         ', '.join(values)
         )
         cur.execute(query)
         cur.execute('commit')
         return redirect('/elevi')
      else:
         flash(f"Sunt deja 10 elevi in grupa!","info")
         return redirect('/elevi')
```

TEMA DE CASA