


UNIVERSITATEA TEHNICĂ “GH ASACHI” IAŞI

FACULTATEA AUTOMATICĂ ŞI CALCULATOARE

SPECIALIZAREA CALCULATOARE ŞI TEHNOLOGIA INFORMAŢIEI

DISCIPLINA: BAZE DE DATE

Documentatie Baze de date-proiect
Tema de casă
Gestionarea unei săli de dans

Coordonator,

Avram Sorin

Student,

Rotaru Irina

Grupa: 1308A

Iași,2022

Descrierea proiectului și scopul aplicației:

Aplicația gestionează funcționarea unei săli de dans. Se vor modela informații despre cursanti, cursuri, stilurile aferente acestora, antrenorii și orarul după care se intampla. Problema in cauza prezinta o singura sala in care se pot petrece cursurile de dans. Una sau mai multe persoane poate participa la un curs, ele fiind privite drept ședințe private. Durata cursului poate fi variabila. Ședința privată se petrece o singura data, pentru o nouă ședința trebuie ca persoanele sa se înscrie și să achite prețul din nou.
La începutul înscrierii unei persoane se achita prețul pe ședința la care merge. În înscrieri se introduce persoana și cursul la care s-a înscris. Ea poate sa se înscrie la mai multe cursuri separat.
Un curs este descris prin nume, prețul ce îi corespunde și dificultatea. Lui îi mai corespunde și un antrenor, un stil specific, un orar și un număr maxim de cursanti ce se pot înscrie la el. Prețul trebuie sa fie între 30 și 300 de lei. Fiecare curs are asignat un preț specific. Dificultate poate fi de trei tipuri: ușor, mediu sau greu și se referă la complexitatea mișcărilor, ritmul de predare, ritmicitate/muzicalitate, viteza coregrafiilor. Se recomandă înscrierea în funcție de nivelul actual al persoanei, dar nu este restricționată. Dacă o persoana se înscrie, ea participă la ședința pe care a platit-o. Cursului îi corespunde un singur antrenor. Antrenorul poate preda însă mai multe cursuri și are asignat un număr maxim de cursanti la care poate preda. Numele unui stil poate sa nu fie menționat, deși este de preferat. Orarul este alcatuit dintr-un final și un început, adică 2 date specifice cu tot cu ora care sunt unice, fiind o singura sală. Intervalul respectiv corespunde unei ședințe. Cursantii sunt identificați în baza de date prin numele lor și un id specific. Un antrenor poate să nu predea la niciun curs, dar un curs nu poate sa nu aiba antenor. Cursantii pot sa ramana în baza de data chiar dacă ședința lor s-a terminat.

Structura și inter-relationarea tabelelor

Aplicația contine următoarele entitati:

- Cursuri_Dans
- Cursanti
- Antrenori
- Stiluri
- Orar
- Inscrieri

Cursuri_Dans prezinta descrierea cursului ce se petrece, are o denumire și caracteristici specifice. Cursantii sunt persoanele înscrise la sala de dans și care au participat, participa sau vor participa. Antrenorii reprezinta toate persoanele angajate la sala de dans pentru a preda un curs. Stiluri monitorizează toate stiluri ce sunt, au fost sau vor fi predate, iar orarul reprezinta intervalul orar alocat pentru o anumita sedinta. Fiecare sedinta are un interval orar unic atribut.

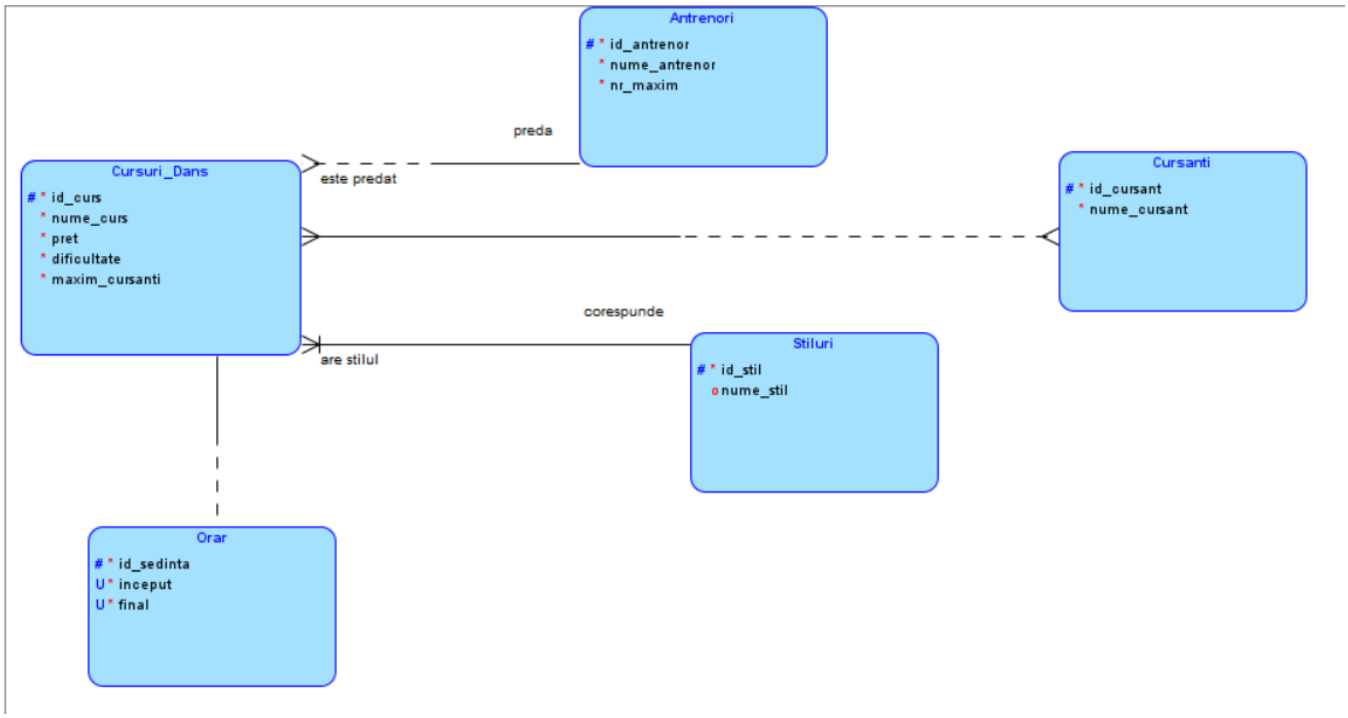
Aplicatia contine urmatoarele relatii intre tabele:

- Relație de 1 la n (one to many) între Antrenori și Cursuri_Dans
  Un antrenor poate preda mai mult cursuri de dans, însă un curs are asignat un singur antrenor.

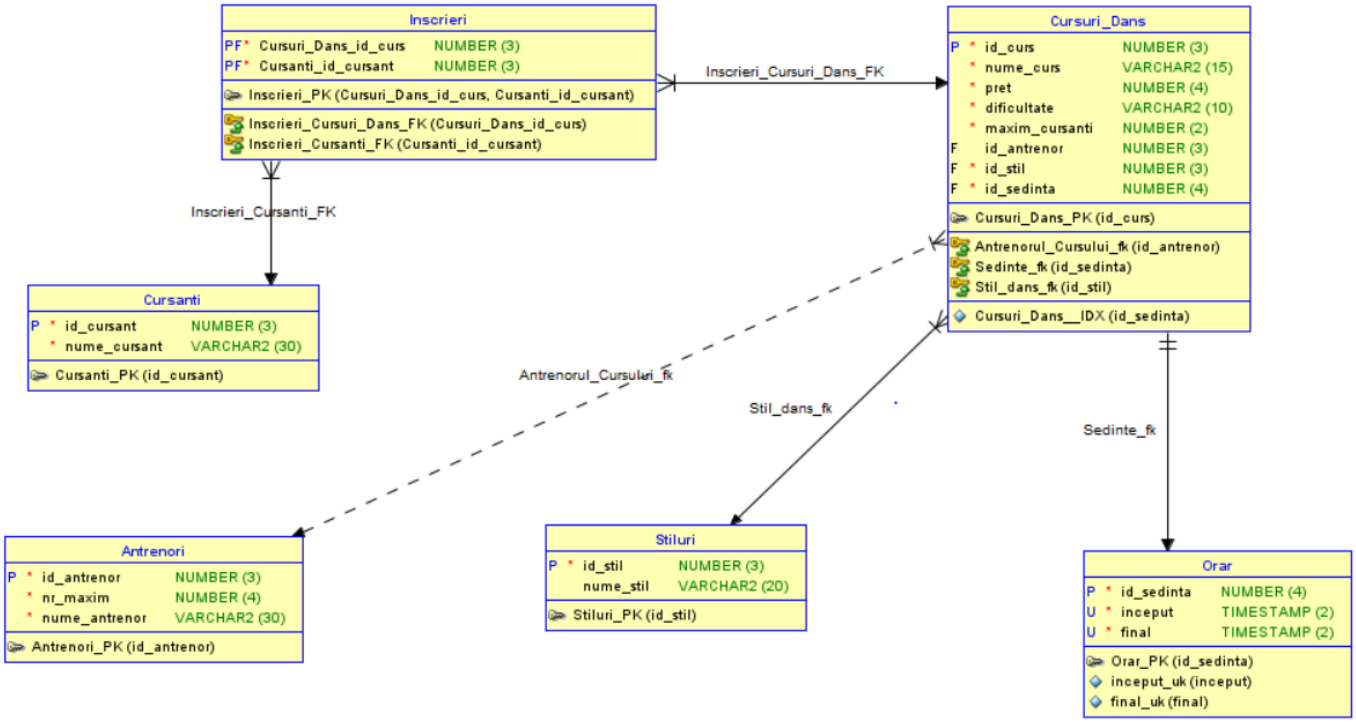
- Relație de 1 la n (one to many) între Stiluri și Cursuri_Dans
  Pot exista mai multe cursuri de dans ce au același stil, dar un curs nu poate avea mai multe stiluri.
- Relație de 1 la 1 (one to one) între Cursuri_Dans și Orar
  O sedinta se petrece doar într-un interval orar. Pentru o nouă ședința trebuie un interval orar nou.
- Relație de n la n (many to many) între Cursanti si Cursuri_Dans
  La un curs pot participa unul sau mai multi cursanti, iar acesta se poate înscrie la mai multe cursuri dacă dorește.

S-a dorit gestionarea unei probleme în care persoanele înscrise pot participa la cate cursuri doresc și în care o sedinta de dans e privită ca un antrenament privat cu un anumit număr maxim de persoane. Aplicația nu permite adaugarea cursurilor, înscrierea persoanelor sau modificarea numărului de cursanți al antrenorilor dacă nu se respecta numărul maxim al cursului sau al antrenorului.

Modelul logic:



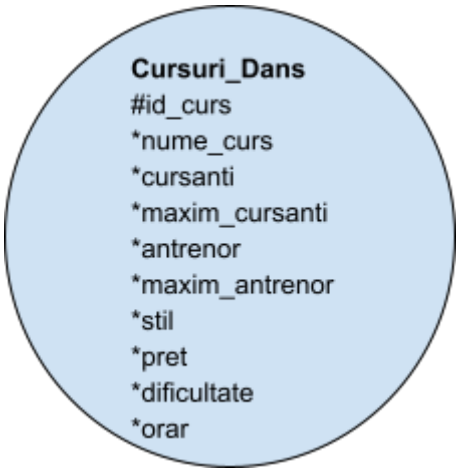
Modelul Relational:



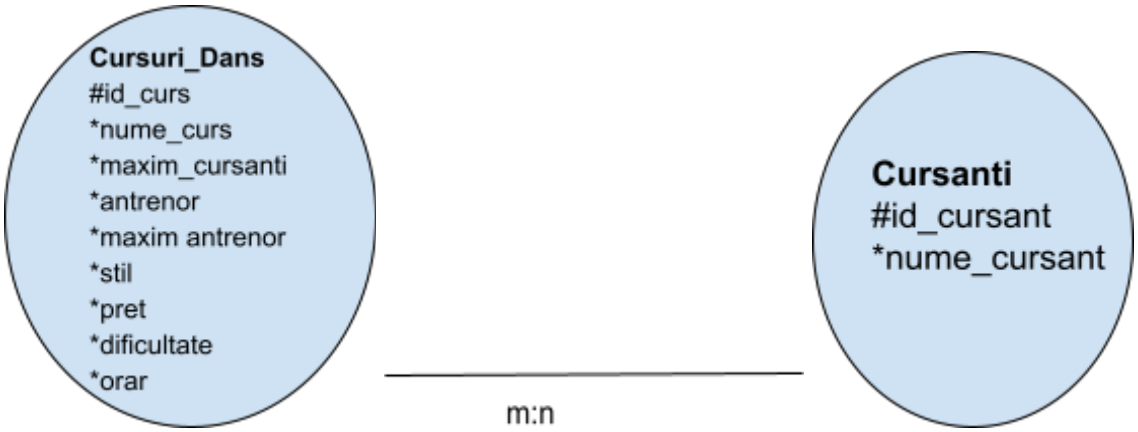
Aspecte legate de normalizare și explicații

Folosim normalizarea pentru a scapa de anomalii și inconsistente la nivel de date.

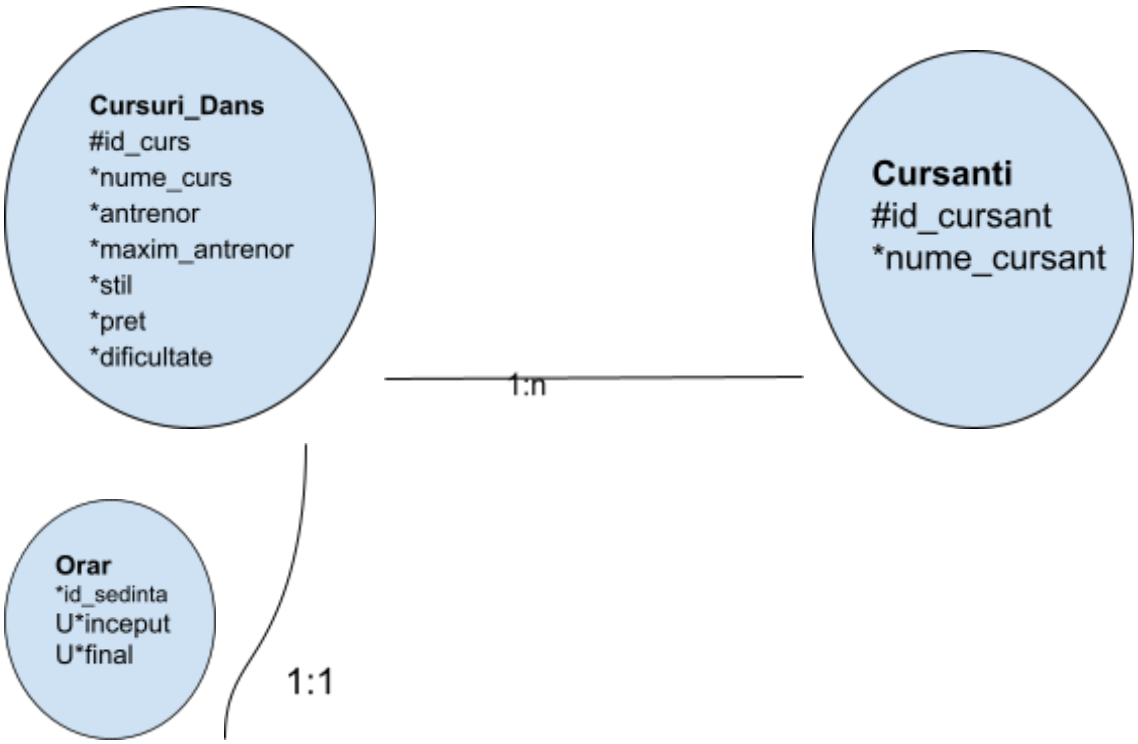
Inițial, trebuie memorate toate datele referitoare la un curs de dans. Cream o entitate cu toate informațiile necesare



Prima forma normala
La curs pot participa unul sau mai mulți oameni , iar un om poate merge la cate cursuri vrea. Facem o nouă entitate numită Cursanti si o legăm printr-o relație de m la n.

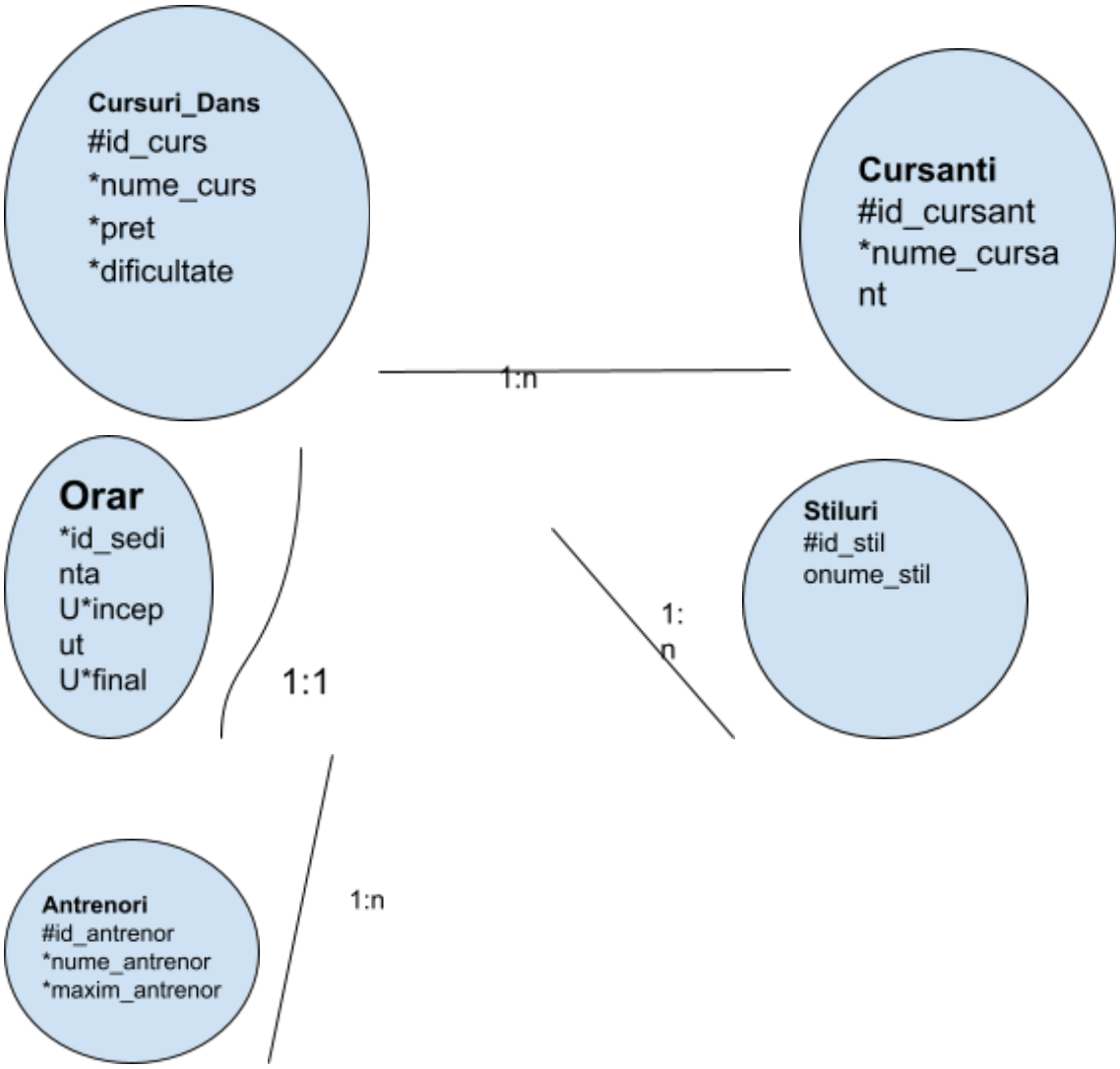


Un curs poate avea loc la o singura data și într-un singur interval orar, acesta fiind unic pentru fiecare curs. Mai facem o entitate Orar pe care o legăm printr-o relație de 1 la 1.



A doua forma normala
Trebuie sa rezolvăm probleme care implica că anumite atribute nu depinde de tot UID. Antrenorul și stilul nu depinde de toate informațiile de acolo. Vom crea 2 entități noi. Din

moment ce un antrenor poate preda mai multe cursuri și pot exista mai multe cursuri cu același stil, vom avea relații de 1 la n.



A treia forma normala
Nu mai avem atribute ce sa depindă de alte informații mai specifice, adică nu de curs în sine sau care sunt specifice unui alt atribut din tabela și nu tuturor, deci ele sunt bune în tabelul în care sunt. Forma a treia normala este deja prezenta.
Obs: toate liniile de relații sunt cu tabela Cursuri_Dans

Descrierea constrângerilor folosite și a necesității lor
Tabela Cursuri_Dans*
id_curs -> constrângere primary key, vrem sa fie unica fiecare înregistrare
nume_curs -> constrângere de tip check, trebuie să aibă lungimea mai mare ca 1 și să fie format din litere, fără a contine cifre, poate avea un singur cuvânt
pret-> constrângere de tip check, trebuie ca prețul să fie între 30 și 300

dificultate-> constrângere de tip check, poate lua valori doar dintre usor/mediu/greu
maxim_cursanti -> constrângere de tip check, trebuie sa fie mai mare ca 0

Tabela Antrenori
id_antrenor -> constrângere primary key, vrem sa fie unica fiecare înregistrare
nume_antrenor -> constrângere de tip check, trebuie să aibă lungimea mai mare ca 1 și să fie format din litere, fără a contine cifre, poate fi format din 2 nume
nr_maxim -> constrângere de tip check, trebuie sa fie mai mare ca 0

Tabela Orar
inceput -> constrângere de tip unique key, avem o singura sală
final -> constrângere de tip unique key, avem o singura sală
final>incept ->constrangere de tip check, sa respecte ordinea naturala a timpului

Tabela Stiluri
id_stil -> constrângere primary key, vrem sa fie unica fiecare înregistrare
nume_stil -> constrângere de tip check, trebuie să aibă lungimea mai mare ca 1 și să fie format din litere, fără a contine cifre, poate avea un singur cuvânt

Tabela Cursanti
id_cursant -> constrângere primary key, vrem sa fie unica fiecare înregistrare
nume_cursant -> constrângere de tip check, trebuie să aibă lungimea mai mare ca 1 și să fie format din litere, fără a contine cifre, poate fi format din 2 nume

Constrângere de tip Not Null -> am pus toate atributele mandatory, în afara de numele stilului, consider ca toate restul informațiilor sunt necesare și nu pot lipsi
Constrângere de tip foreign key -> pentru a stabili relații dintre coloanele a 2 tabele, stabilesc informatii intre curs si antrenor,cursanti, stil si orar
Antrenorul_Cursului_fk
Stilul_dans_fk
Sedinte_fk
Inscrieri :Inscrieri_Cursuri_Dans_fk, Inscrieri_Cursanti_fk

Necesitatea lor este următoarea:
În cazul numelor, am vrut sa verific ca nu se introduc nume ireale, adică ce conțin cifre și că respecta regule de prenume nume, nu sa fie porecle sau doar un nume.
Numele de stiluri și de cursuri am vrut sa fie limitate la un singur cuvânt și din nou, sa fie valide, adică sa nu contina numere în ele sau alte caractere speciale.
În ceea ce privește prețul, am considerat ca este un interval realist, sunt ședințe foarte ușoare și antrenori la început care vor costa mai puțin și ședințe mai grele cu antrenori mai experimentați, mai scumpe.
Dificultatea este necesar sa fie contorizata, dacă erau termeni diferiți era greu de înțeles și urmărit nivelul, așa se vad clar 3 nivele și scrie la fiecare curs ce grad are.

În ceea ce privește orarul, ședințele se desfășoară cu un inceput si un final care reprezinta o data si o ora de început și o data si o ora de final. Ambele trebuie sa fie unice fiind doar o sală și ședințele fiind private, iar finalul trebuie sa fie după început.

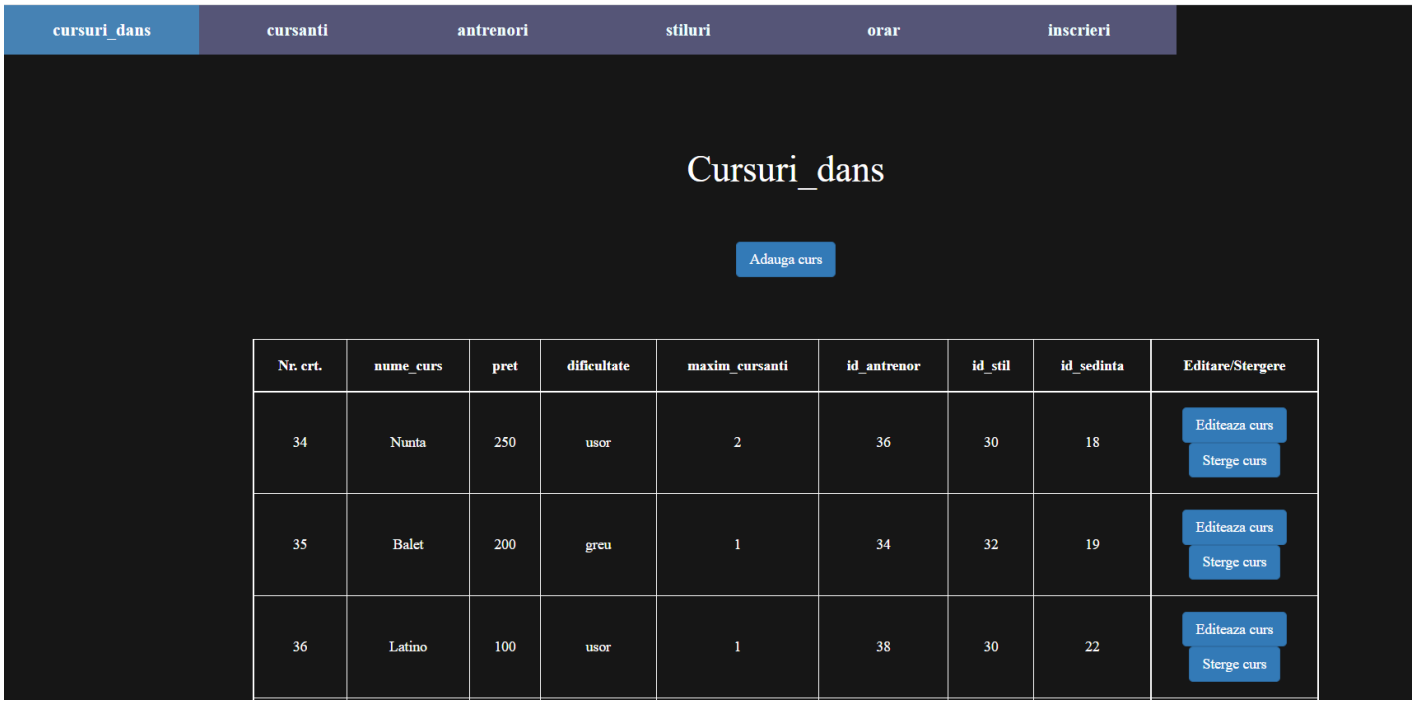
Tehnologii folosite:
(backend si frontend)
Ca și tehnologii am folosit Python (partea de backend) pentru a realiza conexiunea la baza de date (cu ajutorul cx_Oracle) și pentru a interactiona cu aceasta, iar pentru interfața grafica am folosit html (partea de frontend) pentru a crea aplicația prin care se gestionează baza de date. În python am folosit si API-ul numit Flask pentru a construi aplicația web.

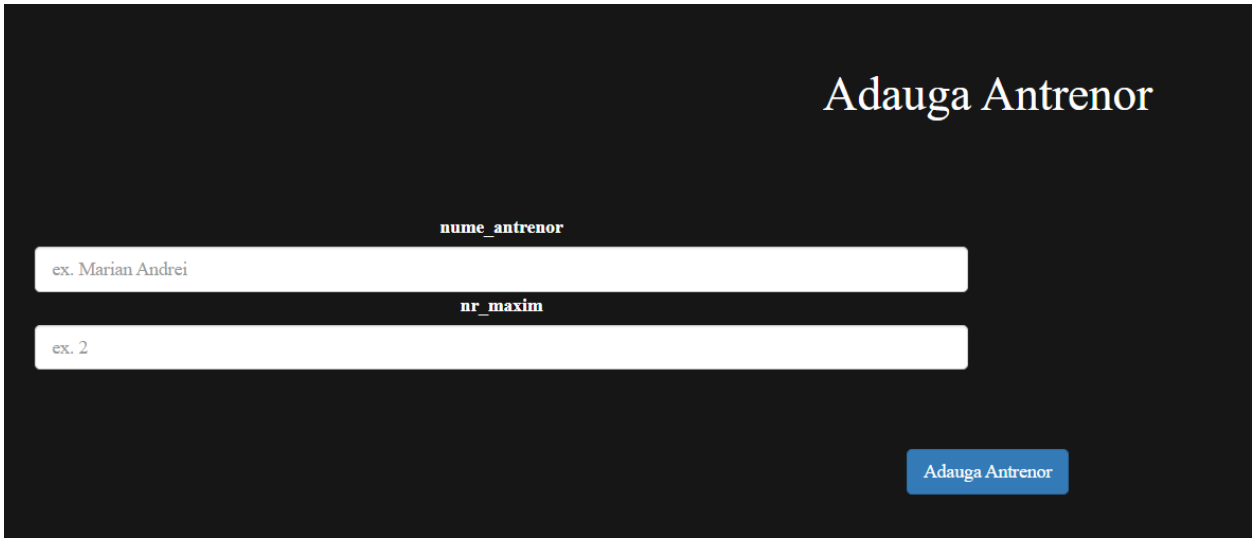
Conectare la baza de date:

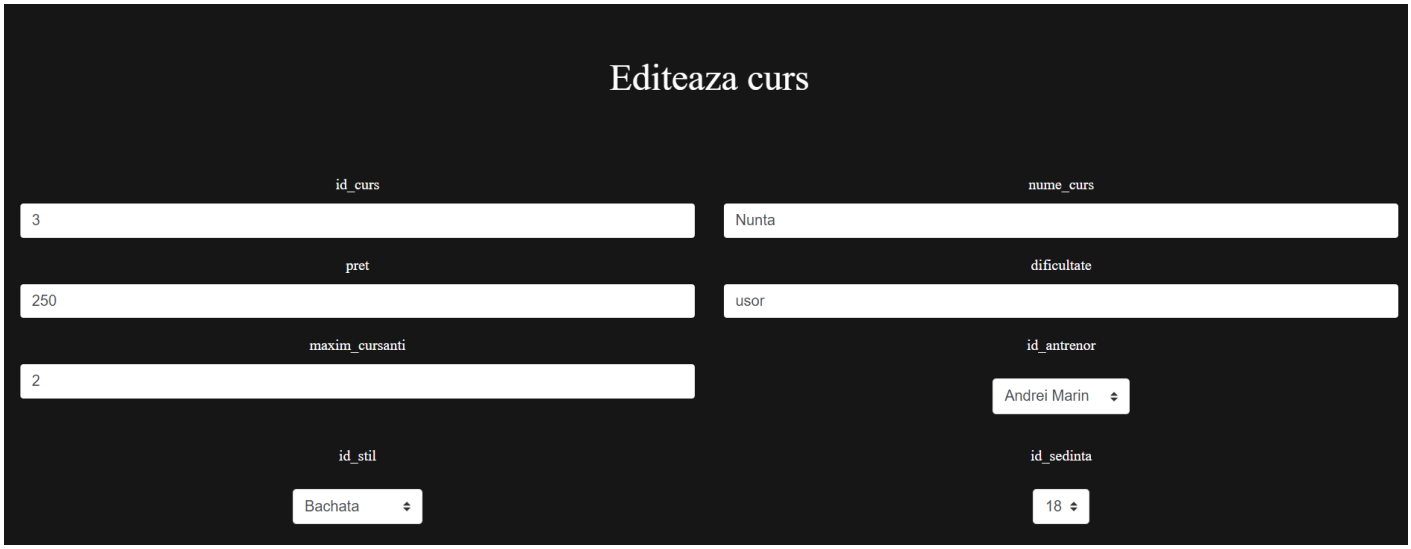
```
con = cx_Oracle.connect("bd076", "bd076", "bd-dc.cs.tuiasi.ro:1539/orcl")
```

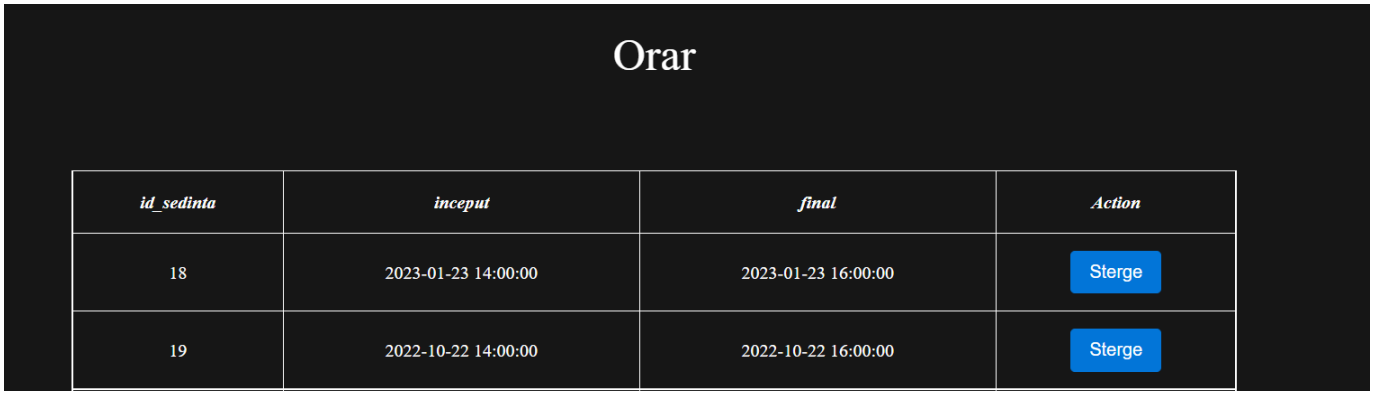
Pentru conectarea la baza de date am folosit extensia Python cx_Oracle ce permite accesul la informațiile de care am nevoie din baza de date. Am folosit contul și parola mea de student și am specificat caracteristicile conexiunii din documentul în care sunt asignate conturile.

Interfața aplicației și exemple de cod:









Ca și cod consider următoarele secvențe relevante, scrise in python și în html pentru functionalitati precum selectare, adăugare, ștergere și editare.
Aici este si bara de sus ce prezinta toate tabelele si navigheaza prin ele.

```
</head>
<body>

 <a href="cursuri_dans"><button class="tablink" style="background: rgb(70,130,180); color: white">cursuri_dans</button></a>
 <a href="cursanti"><button class="tablink">cursanti</button></a>
 <a href="antrenori"><button class="tablink">antrenori</button></a>
 <a href="stiluri"><button class="tablink">stiluri</button></a>
 <a href="orar"><button class="tablink">orar</button></a>
 <a href="inscrieri"><button class="tablink">inscrieri</button></a>
```

Vizualizare:

```
<center>
    <br>
    <h1 id="h3">Cursanti</h1>
    <br>
    <table style="width:70%">
  <br><br>
      <tr>
        <th><i><b>id_cursant</b></i> </th>
        <th><i><b>nume_cursant</b></i></th>
      <th><i><b>Action</b></i></th>
    </tr>
    {% for cursant in cursanti %}
      <tr>
        <td>{{cursant["id_cursant"]}}</td>
        <td>{{cursant["nume_cursant"]}}</td>
      <td>
```

```
@app.route('/cursanti')
def cursanti():
    cursanti = []

    cur = con.cursor()
    cur.execute('select * from cursanti')
    for result in cur:
        cursant = {}
        cursant['id_cursant'] = result[0]
        cursant['nume_cursant'] = result[1]

        cursanti.append(cursant)
    cur.close()
    return render_template('cursanti.html', cursanti=cursanti)
```

Adăugare:

```
    if (suma+int(maxim))<=maxant:
        cur=con.cursor()
        fields = ['id_curs', 'nume_curs', 'pret', 'dificultate', 'maxim_cursanti','id_antrenor', 'id_stil','id_sedinta']
        query = 'INSERT INTO %s (%s) VALUES (%s)' % ('cursuri_dans', ', '.join(fields), ', '.join(values))

        cur.execute(query)
        cur.execute('commit')
        return redirect('/cursuri_dans')
    else:
        return redirect('/cursuri_dans')
else:
    ant=[]
    sed=[]
    stil=[]
    cur = con.cursor()
    cur.execute('select nume_antrenor from antrenori')
    for result in cur:
        print(result[0])
        ant.append(result[0])
    cur.close()
```

```
<div class="col-md-6 mb-3">
      <label for="validationServer01">dificultate</label>
      <input name="dificultate" type="text" class="form-control is-valid" id="validationServer01" placeholder="ex. usor" required>
    </div>
<div class="col-md-6 mb-3">
      <label for="validationServer01">maxim_cursanti</label>
      <input name="maxim_cursanti" type="number" class="form-control is-valid" id="validationServer01" placeholder="ex. 2" required>
    </div>
<div class="col-md-6 mb-3">
    <label for="validationServer01">id_antrenor</label>
    <select name="nume_antrenor" class="custom-select d-block my-3">
      <option value="">Alege antrenor</option>
      {% for com in antrenor %}
        <option value={{com}}>{{com}}</option>
      {% endfor %}
    </select>
  </div>
  <div class="col-md-6 mb-3">
    <label for="validationServer01">id_stil</label>
    <select name="nume_stil" class="custom-select d-block my-3">
      <option value="">Alege stil</option>
      {% for com in stil %}
        <option value={{com}}>{{com}}</option>
      {% endfor %}
    </select>
  </div>
```

Stergere:

```
<td>
        <form class="was-validated" method="POST" action="/delSedinta">
          <button type="submit" name="id_sedinta" value="{{sedinta['id_sedinta']}}" class="btn btn-primary">Sterge</button>
        </form>
  </td>
```

```python
@app.route('/delSedinta', methods=['POST'])
def del_orar():
    sedinta = request.form['id_sedinta']
    cur = con.cursor()
    cur.execute('delete from orar where id_sedinta=' + sedinta)
    cur.execute('commit')
    return redirect('/orar')
```

Editare:

```html
<br>
<h1 id="h3">Editeaza antrenor</h1>
<br><br><br>
<form class="was-validated" method="POST" action="/editAntrenor">
  <div class="row">
      <div class="col-md-6 mb-3">
          <label for="validationServer01">id_antrenor</label>
          <input name="id_antrenor" type="text" class="form-control is-valid" id="validationServer01" value="{{id_antrenor}}" required>
        </div>
        <div class="col-md-6 mb-3">
          <label for="validationServer01">nr_maxim</label>
          <input name="nr_maxim" type="text" class="form-control is-valid" id="validationServer01" value="{{nr_maxim}}" required>
        </div>
    <div class="col-md-6 mb-3">
      <label for="validationServer01">nume_antrenor</label>
      <input name="nume_antrenor" type="text" class="form-control is-valid" id="validationServer01" value="{{nume_antrenor}}" required>
    </div>
  </div>
```

```python
    if suma <= int(nr_maxim):
        cur=con.cursor()
        query = "UPDATE antrenori SET nume_antrenor=%s, nr_maxim=%s where id_antrenor=%s" % (
        nume_antrenor, nr_maxim, id_antrenor)
        cur.execute(query)
        cur.execute('commit')
        return redirect('/antrenori')
    else:
        return redirect('/antrenori')


@app.route('/getAntrenor',methods=['POST'])
def get_antrenor():
    ant=request.form['id_antrenor']
    cur=con.cursor()
    cur.execute('select * from antrenori where id_antrenor='+ant)
    c=cur.fetchone()
    id_antrenor=c[0]
    nr_maxim=c[1]
    nume_antrenor=c[2]
    cur.close()
    return render_template('editAntrenor.html',id_antrenor=id_antrenor,nr_maxim=nr_maxim,nume_antrenor=nume_antrenor)
```