UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și tehnologia informației
SPECIALIZAREA: Tehnologia informației

Gestiune echipe de fotbal

Proiect la disciplina
Baze de date

Cadru didactic coordonator:
Ș.l.dr.ing. Cătălin Mironeanu
Student:
Fundueanu-Constantin Stefan
grupa 1310A

Cuprins

1. Introducere

Aplicația noastră este creată cu scopul de a ține gestiunea bazei de date aferentă echipelor de fotbal. Aceasta permite adăugarea si ștergerea echipelor de fotbal precum și a stadioanelor, managerilor, contractelor și a jucătorilor de fotbal corespunzători acestora.

Baza de date conține 5 tabele:

- ✓ Teams
- ✓ Stadium
- ✓ Manager
- ✓ Contracts
- ✓ Players

Prin această aplicație am încercat să implementăm vizual funcționalitatea principalelor funcții de interogare a unei baze de date. Astfel putem modifica/ accesa o bază de date cu ușurință datorită bibliotecilor din python.

Principalele funcții folosite sunt:

- ➢ Select
- ➢ Insert
- ➢ Delete

2. Instalare aplicații necesare

Se instalează python versiunea 3.10 care pote fi descarcată de pe site -ul oficial https://www.python.org/downloads/ .

Aplicația are nevoie de urmatoarele biblioteci:

2.1. Flask versiunea 2.2.2

Flask este un API (Application Programming Interface) Python ce permite construirea de aplicații web. A fost dezvoltat de Armin Ronacher (un programator austriac și un speaker cunoscut la conferințele în domeniul software).

Acest framework este mult mai ușor de înțeles decât framework-ul Django's și mult mai ușor de invățat deoarece are mai puțin cod de implementat în cazul unei aplicații web.

Comanda instalare Flask

```
pip install Flask
```

Un exemplu de folosire a Flask-ului:

```
from flask import Flask
app = Flask(__name__) # Flask constructor
# A decorator used to tells the application
# which URL is associated function
@app.route('/')
def hello():
return 'HELLO'
if __name__=='__main__':
app.run()
```

2.2. Jinja2

Jinja2 este o librărie pentru Python creată pentru a fi flexibilă, rapidă și sigură.
Poți instala ultimele versiuni de Jinja2 folosind `easy_install` sau `pip`.

```
easy_install Jinja2
pip install Jinja2
```

Caracteristici:

- Conține server de dezvoltare și depanator;
- Suport integrat pentru testarea unităților;
- Utilizează Jinja2 templating;
- Suport pentru cookie-urile securizate;
- Compatibilitatea Google App Engine.

```
from flask import Flask
app = Flask ( __name__ )

@app.route ( "/" )
def hello ():
return "Hello World!"

if __name__ == "__main__" :
app .  run ()
```

2.3. cx_Oracle

cx_Oracle este un modul de extensie Python care permite accesul la Oracle Database. Conformă cu specificația API 2.0 a bazei de date Python, cu un număr considerabil de adăugiri și câteva excluderi.

Pentru a utiliza cx_Oracle 8 cu Python și Oracle Database aveți nevoie de:

- Python 3.5 și mai mult. Versiunile mai vechi ale cx_Oracle pot funcționa cu versiuni mai vechi ale Python.
- Bibliotecile client Oracle. Acestea pot fi de la clientul Oracle Instant Client gratuit sau de la cele incluse în Oracle Database dacă Python se află pe aceeași mașină ca și baza de date. Bibliotecile clienților Oracle 21,19, 18, 12 și 11.2 sunt acceptate pe Linux, Windows și MacOS. Utilizatorii au raportat, de asemenea, succesul cu alte platforme.
- O bază de date Oracle. Standardul de interoperabilitate Oracle standard pentru client-server permite cx_Oracle să se conecteze la bazele de date mai vechi și mai noi.

Instalare

```
python - m pip install cx_Oracle – upgrade
```

3. Tehnologii folosite

3.1. SQL Developer

Oracle SQL Developer este un mediu de dezvoltare integrat pentru lucrul cu SQL în bazele de date Oracle.

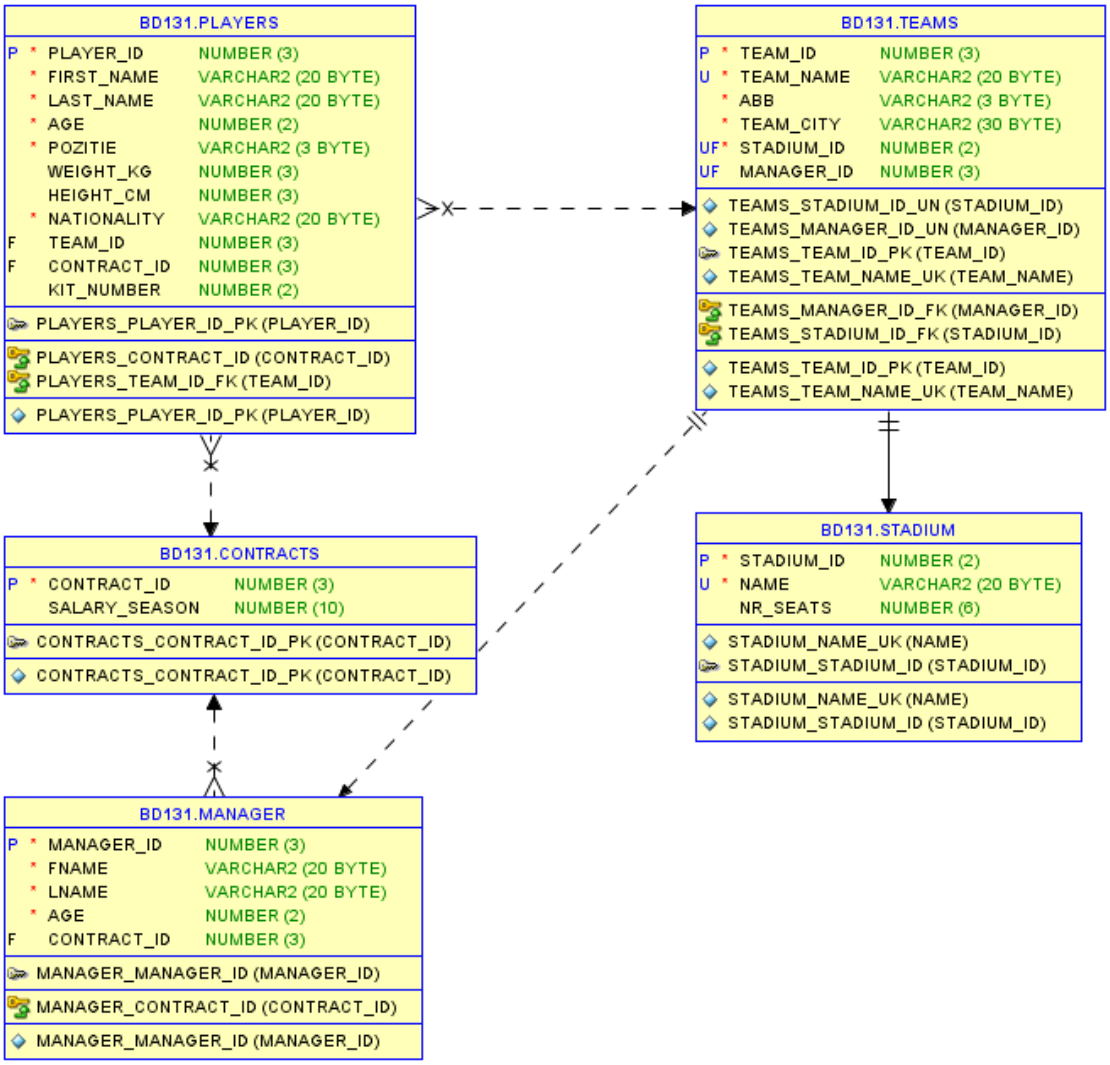
3.2. SQL Data Modeler

Oracle SQL Developer Data Modeler este un instrument grafic gratuit care îmbunătățește productivitatea și simplifică sarcinile de modelare a datelor. Data Modeler poate fi utilizat atât în medii tradiționale, cât și în medii Cloud.

3.3. PyCharm

PyCharm este un mediu de dezvoltare integrat folosit pentru programare în Python. Oferă analiză de cod, un depanator grafic, un tester unitar integrat, integrare cu sisteme de control al versiunilor și sprijină dezvoltarea web cu Django. PyCharm este dezvoltat de compania cehă JetBrains.

4. Structura și relațiile tabelelor



4.1. Tabela Contracts

Tabela contracts conține câmpurile contract_id și salary_season. Primary Key-ul tabelei este contract_id. Totodată, tabela conține o constrângere de tip CHECK ce impune ca salary_season să fie între două valori.

4.2. Tabela Manager

Tabela manager conține câmpurile manager_id, fname, lname, age, contract_id. Primary Key-ul tabelei este manager_id iar câmpul contract_id este Foreign Key ce referențiază tabela Contracts.

4.3. Tabela Stadium

Tabela stadium conține câmpurile stadium_id, name, nr_seats. Primary Key-ul tabelei este stadium_id. Tabela conține și o constrângere de tip CHECK ce impune ca nr_seats (nr. de locuri) să fie mai mare decât 1000 și mai mic decât 150000.

4.4. Tabela Teams

Tabela teams conține câmpurile team_id, Team_name, ABB, team_city, stadium_id, manager_id. Primary Key-ul tabelei este team_id în timp ce stadium_id și manager_id prezintă constrângeri de tip Foreign Key ce referențiază tabelele stadium și manager. Totodată, stadium_id si manager_id au constrângeri de tip UNIQUE astfel realizându-se legături de tip One-to-One între tabelele teams si manager precum și între teams și stadium. Câmpul Team_name are și el o constrângere de tip UNIQUE pentru a nu exista două echipe cu același nume.

4.5. Tabela Players

Tabela players conține câmpurile player_id, , first_name, last_name, age, pozitie, weight_kg, height_cm, nationality, team_id, contract_id, kit_number. Primary Key-ul tabelei este player_id în timp ce team_id și contract_id prezintă constrângeri de tip Foreign Key ce referențiază tabelele teams și contracts. Se realizează astfel legături de tip One-to-Many între tabelele teams și players (o echipă poate avea mai mulți jucători) precum și între players și contracts (mai mulți jucători pot avea același contract). Câmpul weight_kg are o constrângere de tip CHECK ce se asigură că data introdusă este mai mare decât 60. Câmpul height_cm are o constrângere de tip CHECK ce se asigură că data introdusă este mai mare decât 155. Câmpul kit_number are o constrângere de tip CHECK ce se asigură că data introdusă este situată între valorile 1 si 99.

5. Functionalitatea aplicației

5.1. Conectarea la baza de date

```
app = Flask(__name__)
with open(app.root_path + '\config.cfg', 'r') as f:
    app.config['ORACLE_URI'] = f.readline()

con = cx_Oracle.connect("sys", "tiger", "localhost/orclpdb",mode = cx_Oracle.SYSDBA)
```

Conectarea la baza de date s-a făcut cu ajutorul extensiei Python cx_Oracle ce a fost importată la începutul proiectului.

Principalele funcții folosite sunt:

5.1.1. Funcția Select

Fiecare din următoarele declarații sunt valide:

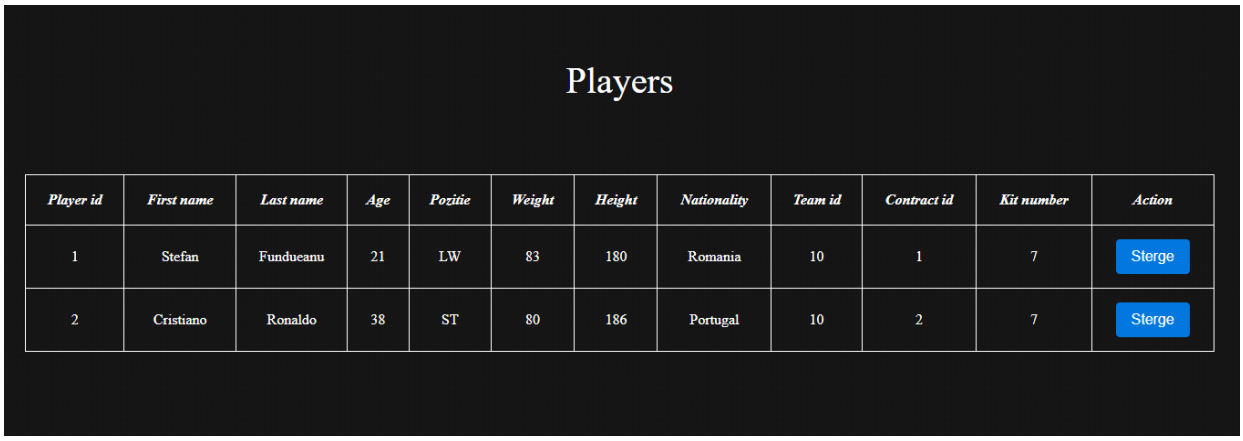
```
SELECT * FROM EMPLOYEES;

SELECT
*
FROM
EMPLOYEES
;

SELECT *
FROM  EMPLOYEES;
```

Este comanda cea mai utilizată. Este folosită pentru obținerea datelor din bazele de date. Exemplu de utilizare a funcției select pentru afișarea datelor dintr-o tabelă în python.

```
@app.route('/players')
def ply():
    counselors = []

    cur = con.cursor()
    cur.execute('select * from players')
    for result in cur:
        counselor = {}
        counselor['player_id'] = result[0]
        counselor['first_name'] = result[1]
        counselor['last_name'] = result[2]
        counselor['age'] = result[3]
        counselor['pozitie'] = result[4]
        counselor['weight_kg'] = result[5]
        counselor['height_cm'] = result[6]
        counselor['nationality'] = result[7]
        counselor['team_id'] = result[8]
        counselor['contract_id'] = result[9]
        counselor['kit_number'] = result[10]
        counselors.append(counselor)
    cur.close()
    com = []
    cur = con.cursor()
    cur.execute('select contract_id from contracts')
    # import pdb;pdb.set_trace()
    for result in cur:
        com.append(result[0])
    cur.close()

    com2=[]
    cur = con.cursor()
    cur.execute('select Team_name from teams')
    # import pdb;pdb.set_trace()
```

```
    for result in cur:
        com2.append(result[0])
    cur.close()

    return render_template('players.html', counselors=counselors, player=com,player2=com2)
```

Rezultatul acestui cod este următorul:



5.1.2. Funcția Insert

Adaugă o nouă înregistrare.

Sintaxa este :

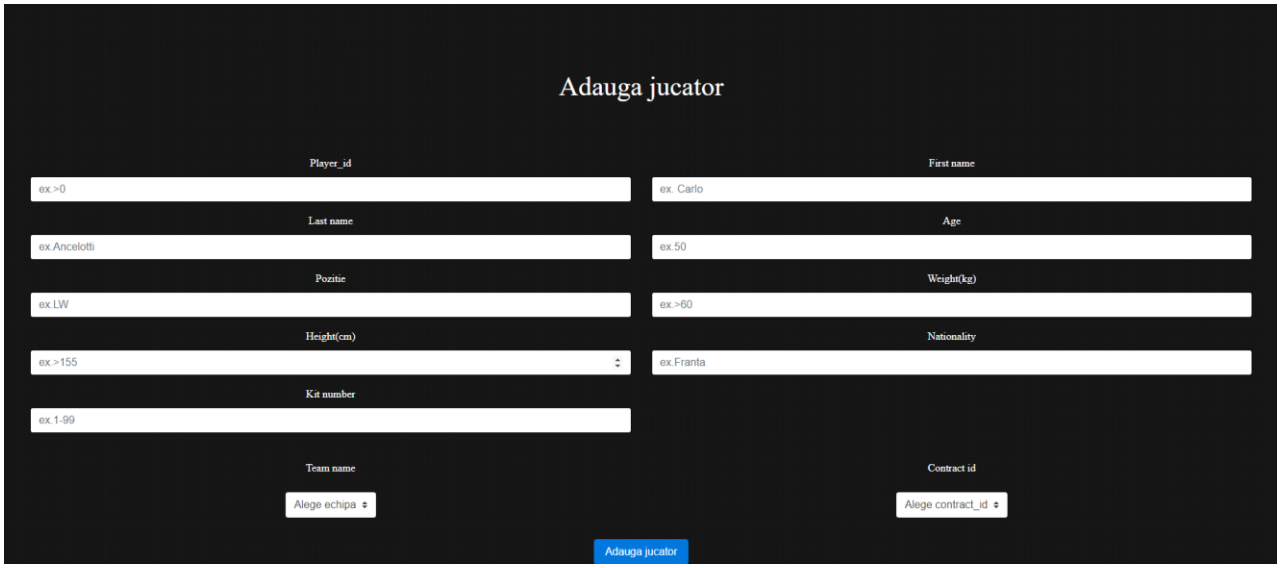
```
INSERT INTO tabela [ ( coloana [ , coloana . . . ])]
VALUES ( valoare [, valoare . . . ]);
```

Exemplu de cod pentru introducerea unor date noi în tabelă:

```
@app.route('/ADDplayer', methods=['POST'])
def ad_ply():
    error = None
    if request.method == 'POST':
        team = []
        c = "'" + request.form['Team_name'] + "'"

        cur = con.cursor()
        cur.execute('select team_id from teams where Team_name=' + c)
        for result in cur:
            team = result[0]
        cur.close()

        cur = con.cursor()
        values = []
        values.append("'" + request.form['player_id'] + "'")
        values.append("'" + request.form['first_name'] + "'")
        values.append("'" + request.form['last_name'] + "'")
        values.append("'" + request.form['age'] + "'")
        values.append("'" + request.form['pozitie'] + "'")
```

```
        values.append("'" + request.form['weight_kg'] + "'")
        values.append("'" + request.form['height_cm'] + "'")
        values.append("'" + request.form['nationality'] + "'")
        values.append("'" + str(team) + "'")
        values.append("'" + request.form['contract_id'] + "'")
        values.append("'" + request.form['kit_number'] + "'")
        fields = ['player_id', 'first_name', 'last_name',
'age','pozitie','weight_kg','height_cm','nationality','team_id','contract_id','kit_number']
        query = 'INSERT INTO %s (%s) VALUES (%s)' % (
            'players',
            ', '.join(fields),
            ', '.join(values)
        )

        cur.execute(query)
        cur.execute('commit')
        return redirect('/players')
```

Pagina pentru insert este următoarea:

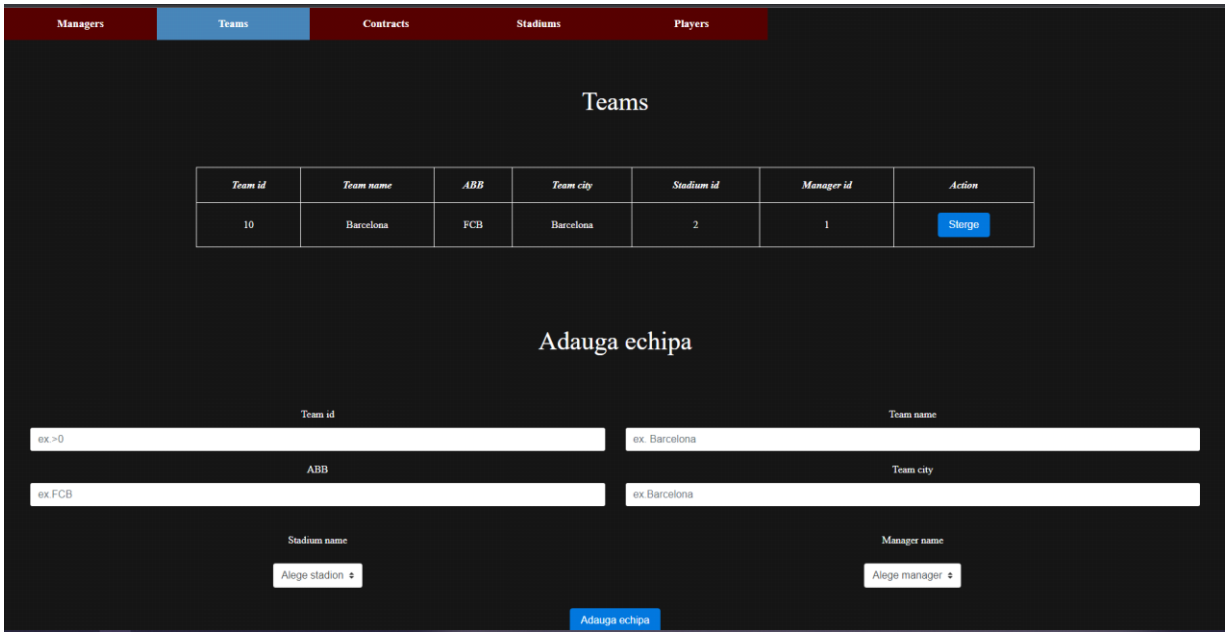


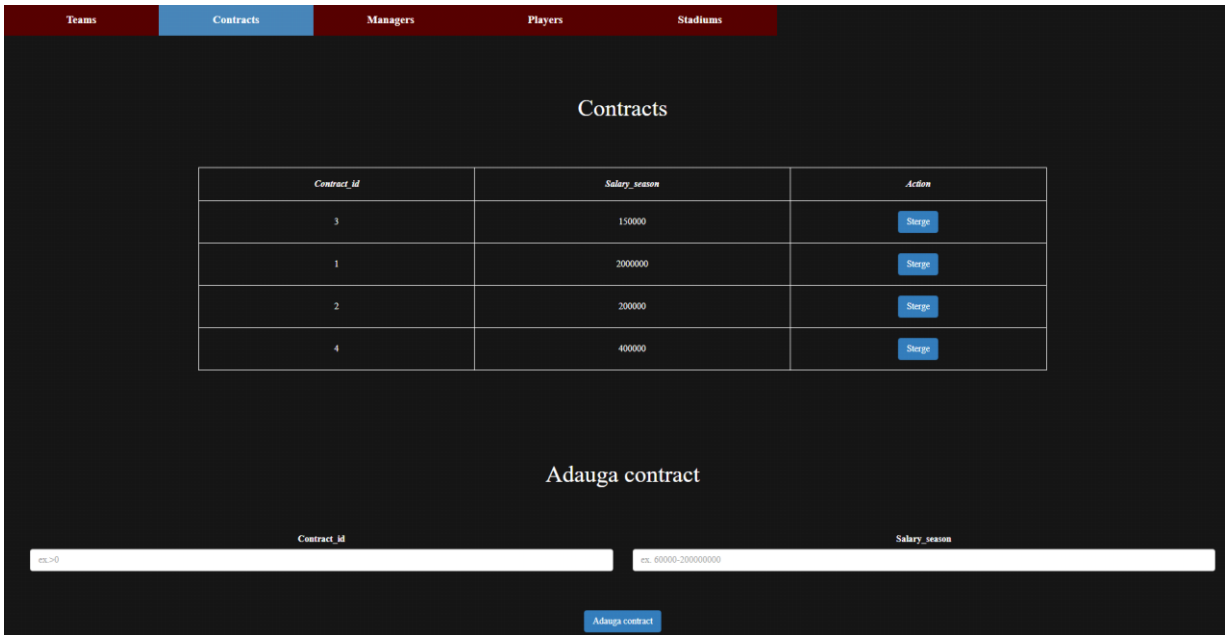
5.1.3. Funcția Delete

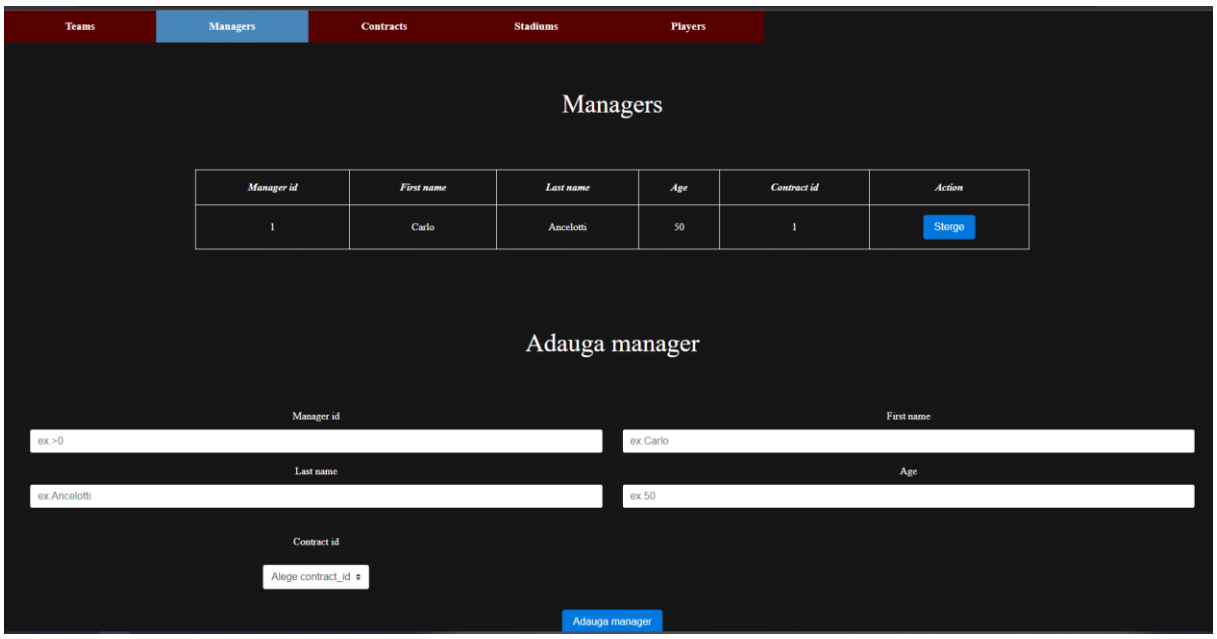
Șterge o înregistrare din tabelă.
Exemplu de cod utilizat în aplicația noastră:

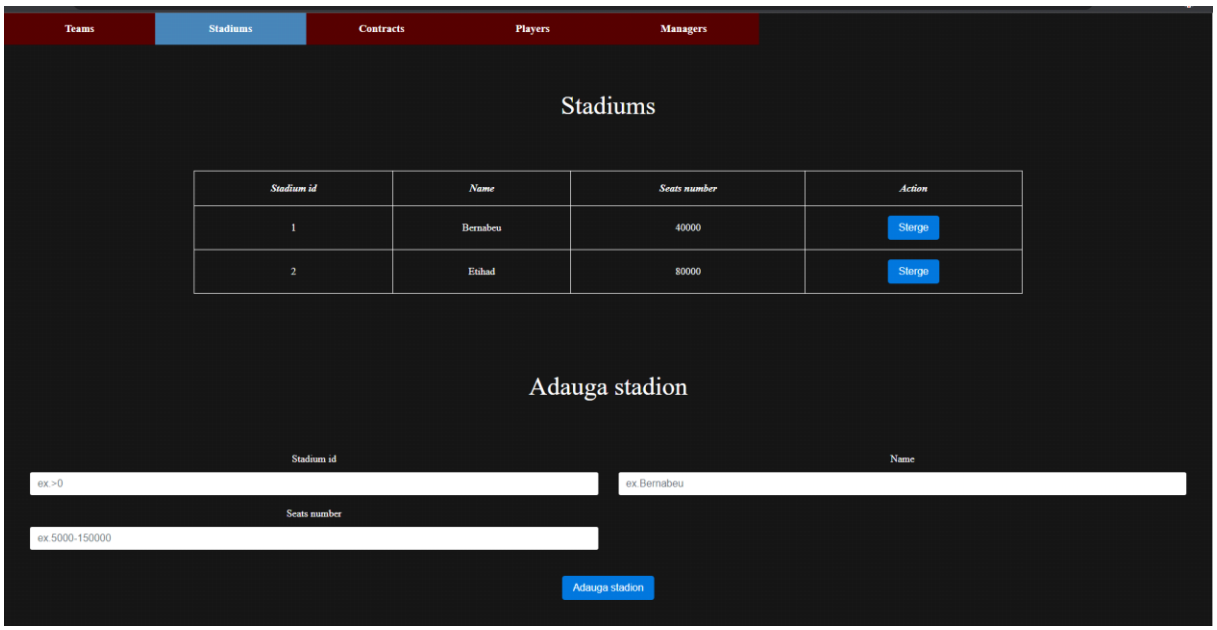
```
@app.route('/delPlayer', methods=['POST'])
def del_ply():
    cnp = request.form['player_id']
    cur = con.cursor()
    cur.execute('delete from players where player_id=' + cnp)
    cur.execute('commit')
    return redirect('/players')
```

5.2. Capturi de ecran interfață grafică






Teams
Managers
Contracts
Stadiums
Players
Managers
Manager id
First name
Last name
Age
Contract id
Action
1
Carlo
Ancelotti
50
1
Sterge
Adauga manager
Manager id
ex.>0
First name
ex.Carlo
Last name
ex.Ancelotti
Age
ex.50
Contract id
Alege contract_id
Adauga manager


Teams
Stadiums
Contracts
Players
Managers
Stadiums
Stadium id
Name
Seats number
Action
1
Bernabeu
40000
Sterge
2
Etihad
80000
Sterge
Adauga stadion
Stadium id
ex.>0
Name
ex.Bernabeu
Seats number
ex.5000-150000
Adauga stadion