UNIVERSITATEA TEHNICĂ “GHEORGHE ASACHI” DIN IAȘI

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

SPECIALIZAREA CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI

DISCIPLINA BAZE DE DATE PROIECT

Baze de date pentru o organizație de voluntariat

Student: Băcălie Ioana-Roxana

Grupa: 1308A

Cadru didactic coordonator: Avram Sorin

Descrierea proiectului

Titlul proiectului este "**Baza de date pentru o organizație de voluntariat**". Acesta presupune analiza, proiectarea și implementarea unei baze de date care să stocheze și să modeleze informații despre voluntari, departamentele din care fac parte, evenimentele la care au fost asignați și dacă au fost sau nu prezenți la acestea, locațiile unde au avut loc evenimentele și diverse observații legate de ele. Se permit acțiuni precum adăugarea, modificarea și ștergerea informațiilor despre voluntari, departamente, evenimente și locațiile acestora. Atunci când apare o cerință pentru un eveniment de a participa un număr de voluntari dintr-un anumit departament, voluntarii pot fi programați pentru a participa și pot avea și diferite observații legate de prezența acestora.

Informațiile de care avem nevoie sunt legate de:

- Voluntari - ne interesează datele de contact ale acestora pentru a putea fi contactați în vederea participării la evenimente
- Departamente - trebuie să știm în ce departamente sunt împărțiți voluntarii din organizație pentru ca cererile care vin de a participa la evenimente să solicite un anumit număr de voluntari pe un anumit department
- Evenimente - fiecare eveniment se întâmplă la o anumită dată și oră, într-o anumită locație
- Locații - ne interesează orașul unde poate avea loc un eveniment, strada și numărul pentru a ști unde trebuie voluntarii să fie prezenți
- Cerințe pentru eveniment - pentru fiecare eveniment este nevoie de un anumit număr de voluntari de pe un anumit departament
- Observații legate de prezența/absența voluntarului la eveniment și felul în care s-a comportat

Entități

- **Volunteer** - entitate ce ține informațiile legate de voluntari
    - Volunteer_ID (NUMERIC(4)) - primary key
    - Last_Name (VARCHAR(20)) - mandatory
    - First_Name (VARCHAR(20)) - mandatory
    - Faculty (VARCHAR(30))
    - Email (VARCHAR(30)) - mandatory
    - Phone_Number (VARCHAR(15)) - mandatory
    - City (VARCHAR(20))
    - Department_ID (NUMERIC(3)) - mandatory, Relation UID
- **Department** - entitate care conține informații despre departamentele existente în organizație
    - Department_ID (NUMERIC(3)) - primary key

    - Department_Name (VARCHAR(20)) - mandatory, unique
- **Event**
    - Event_ID (NUMERIC) - primary key
    - StartTime (TIMESTAMP) - mandatory
    - EndTime (TIMESTAMP) - mandatory
    - Location_ID (NUMERIC(3)) - Relation UID, mandatory
- **Location**
    - Location_ID (NUMERIC(3)) - primary key
    - City (VARCHAR(20)) - mandatory, unique împreună cu county, street și number_of_street
    - County (VARCHAR(20)) - mandatory, unique împreună cu city, street și number_of_street
    - Street (VARCHAR(30)) - mandatory, unique împreună cu city, county și number_of_street
    - Number_of_street (NUMERIC(3)) - mandatory, unique împreună cu city, county și street
- **Requirement**
    - Requirement_ID (NUMERIC(4)) - primary key
    - Number_of_volunteers (NUMERIC(2)) - mandatory
    - Event_ID (NUMERIC(3)) - Relation UID, mandatory
    - Department_ID (NUMERIC(3)) - Relation UID, mandatory
- **Appointment**
    - Appointment_ID (NUMERIC(4)) - primary key
    - Attendance (VARCHAR(20))
    - Volunteer_ID (NUMERIC(4)) - Relation UID, mandatory
    - Requirement_ID (NUMERIC(4)) - Relation UID, mandatory
- **Observation**
    - Observation_ID (NUMERIC(4)) - primary key
    - Message (VARCHAR(30)) - mandatory
    - Appointment_ID (NUMBER(4)) - Relation UID, mandatory

Relații între entități

Între entitățile Department și Volunteer există o relație de one-to-many deoarece un voluntar poate face parte dintr-un singur departament, dar dintr-un departament pot face parte mai mulți voluntari.

Între entitățile Department și Requirement există o relație de one-to-many pentru că pot apărea mai multe cerințe pentru un anumit departament (de a-și aduce voluntari din departament la eveniment), însă fiecare cerință este pentru un singur departament. Dacă un eveniment are nevoie de voluntari din mai multe departamente, se fac mai multe cereri.

Între entitățile Location și Event este tot o relație de tip one to many pentru că un eveniment are loc la o singură locație, însă mai multe evenimente se pot desfășura în același loc.

Între entitățile Event și Requirement există, de asemenea, o relație one-to-many, întrucât un eveniment va avea mai multe cereri de voluntari, însă o astfel de cerere se referă la un singur eveniment.

O programare este pentru o singură cerință, însă o cerință este rezolvată prin mai multe programări. Prin urmare, și între entitățile Requirement și Appointment este o relație de unu la mulți.

O programare programează un singur voluntar, dar un voluntar poate fi programat la mai multe evenimente prin mai multe programări, deci și între Volunteer și Appointment este o relație one-to-many.

O relație de unu la unu se află între Observation și Appointment, deoarece o programare va avea o singură observație, iar o observație se va referi la o singură programare.

Aspecte legate de normalizare

O relație este în prima formă normală dacă un atribut conține valori atomice din domeniul său (și nu grupuri de astfel de valori) și nu conține grupuri care se repetă. Acest lucru se respectă deoarece niciun atribut nu poate fi descompus în unități mai mici. Chiar dacă voluntarul ar avea mai multe adrese de email sau mai multe numere de telefon, pe noi ne interesează doar unul dintre acestea și doar unul va fi stocat în tabelă. De aceea, nu apar probleme legate de respectarea primei forme normale.

O relație este în a doua formă normală dacă se respectă prima formă normală și toate dependețele între cheia primară și celelalte atribute ale sale sunt elementare (atributele nu depind de o parte din cheie). Deoarece prima formă normală se respectă și nu avem chei primare multiple, atunci se respectă în toate tabelele această formă.

O relație este în a treia formă normală dacă este în a doua formă normală și dacă sunt eliminate toate dependențele funcționale tranzitive (dacă nu există nicio dependență funcțională între atributele non-cheie). Dacă am fi pus numele departamentului tot în tabela Volunteer, această formă normală nu s-ar fi respectat, deoarece numele ar fi depins de id-ul departamentului. În forma actuală a tabelelor, această formă normală e respectată, întrucât nu avem dependențe între atributele care nu sunt primary-key.

Modul în care arată tabelele și cum sunt legate între ele poate fi observat mai jos, în cadrul diagramelor modelului logic și modelului fizic.

Diagrama modelului logic

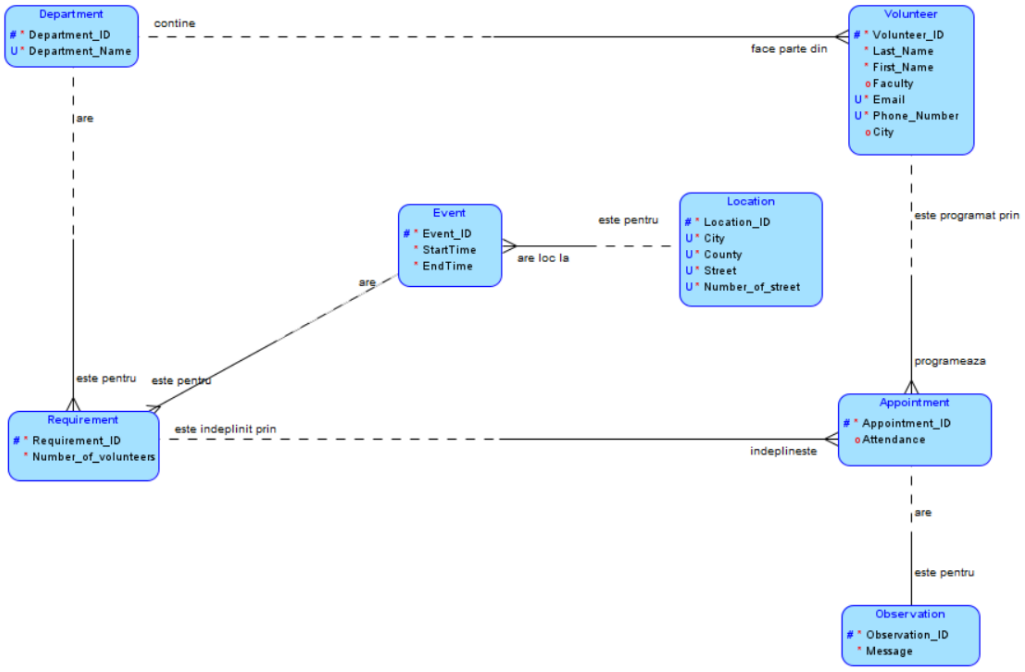
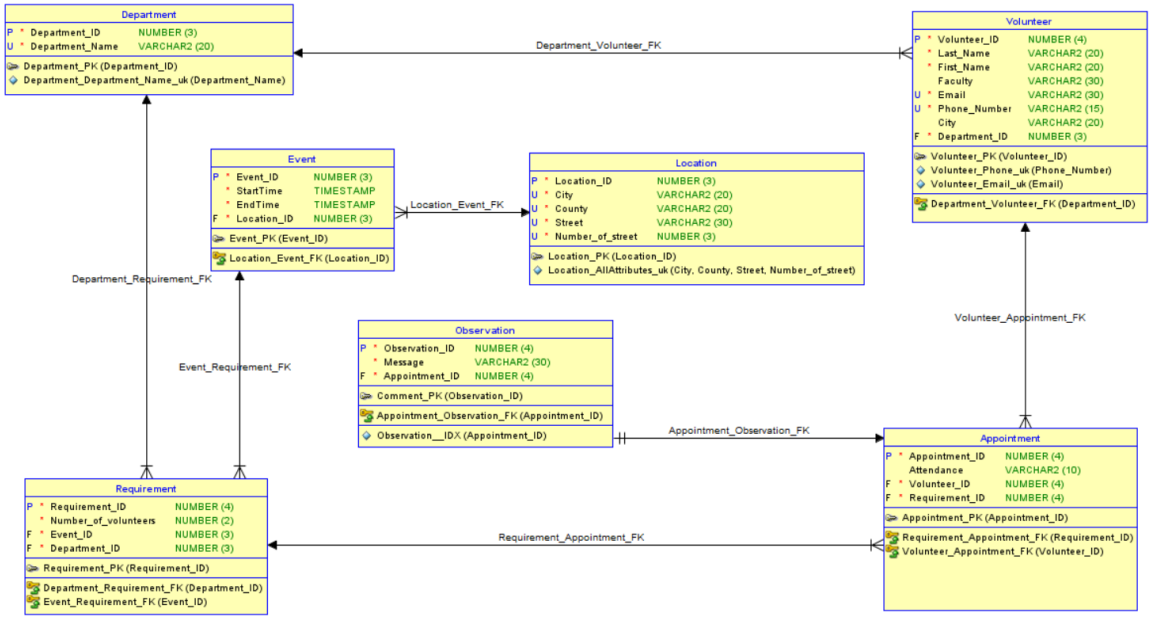


Diagrama modelului fizic

În modelul fizic am implementat autoincrementări pentru cheile primare și, totodată, am implementat un trigger pentru a verifica dacă Starttime este dupa data și ora curentă, iar Endtime este după Starttime.



Constrângeri

Constrângeri de tip unique

- **Entitatea Department**: atributul Department_Name este unic deoarece nu pot exista două departamente cu același nume
- **Entitatea Volunteer**: atributele Email și Phone_Number sunt unice deoarece fiecare persoană are propriul email și propriul număr de telefon
- **Entitatea Location**: atributele City, County, Street, Number_of_street formează împreună o cheie unică deoarece nu pot exista două adrese în același oraș, pe aceeași stradă și la același număr

Constrângeri de tip not null

- **Entitatea Department**: Department_Name nu poate fi null deoarece e nevoie să știm numele departamentului
- **Entitatea Volunteer**: atributele First_Name, Last_Name, Email și Phone_Number nu pot fi null deoarece sunt detalii care trebuie neapărat știute despre voluntari pentru a putea fi contactați, dar atributele Faculty și City pot fi null deoarece nu sunt neapărat necesare
- **Entitatea Event**: StartTime și EndTime nu pot fi null, deoarece sunt detalii care trebuie neapărat cunoscute despre un eveniment, pentru a ști dacă se suprapune sau nu cu altul
- **Entitatea Location**: atributele City, County, Street, Number_of_street nu pot fi null deoarece dacă una din acestea nu ar fi cunoscute, atunci adresa nu ar putea fi identificată corect
- **Entitatea Requirement**: atributul Number_of_volunteers nu poate fi null, e un detaliu care trebuie știut neapărat pentru a putea îndeplini cerința respectivă
- **Entitatea Appointment**: atributul Attendance nu este mandatory deoarece acesta este completat abia când are loc evenimentul, pentru a informa asupra prezenței/absenței voluntarului
- **Entitatea Observation**: atributul Message este mandatory

Constrângeri de tip check

- **Entitatea Volunteer**
  - **Atributul Last_Name:** lungimea trebuie să fie mai mare decât 2, trebuie să conțină doar litere
  - **Atributul First_Name:** lungimea trebuie să fie mai mare decât 2, trebuie să conțină doar litere
  - **Atributul Faculty:** lungimea trebuie să fie mai mare decât 1, numele trebuie să conțină doar litere
  - **Email:** mail-ul trebuie sa inceapa cu minim o literă/cifra, urmat de ‘@’, urmat de minim o literă, urmat de ‘.’, urmat de minim două litere
  - **Phone_Number:** numarul de telefon trebuie sa fie alcatuit doar din cifre
  - **City:** lungimea trebuie să fie mai mare decât 2
- **Entitatea Department**
  - **Atributul Department_Name:** lungimea trebuie să fie mai mare decât 1
- **Entitatea Event**
  - **Atributul StartTime:** timpul trebuie să fie după timpul curent
  - **Atributul EndTime:** timpul trebuie să fie dupa StartTime

- **Entitatea Location**
  - **Atributul City:** lungimea trebuie să fie mai mare decât 2
  - **Atributul County:** lungimea trebuie să fie mai mare decât 2
  - **Atributul Street:** lungimea trebuie să fie mai mare decât 2
- **Entitatea Requirement**
  - **Atributul Number_of_volunteers:** trebuie să fie mai mare ca 0
- **Entitatea Appointment**
  - **Atributul Attendance:** valorile introduse trebuie să fie din lista de valori 'prezent' sau 'absent'
- **Entitatea Observation:**
  - **Atributul Message:** lungimea trebuie să fie mai mare decât 2

În mod normal, ar trebui implementate constrângeri pentru a limita inserările atunci când s-au atins numărul de voluntari necesari pentru un anumit eveniment, însă există anumite limitări în SQL.

Tehnologii folosite

Am folosit limbajul de programare Python, HTML și CSS pentru interfața grafică.

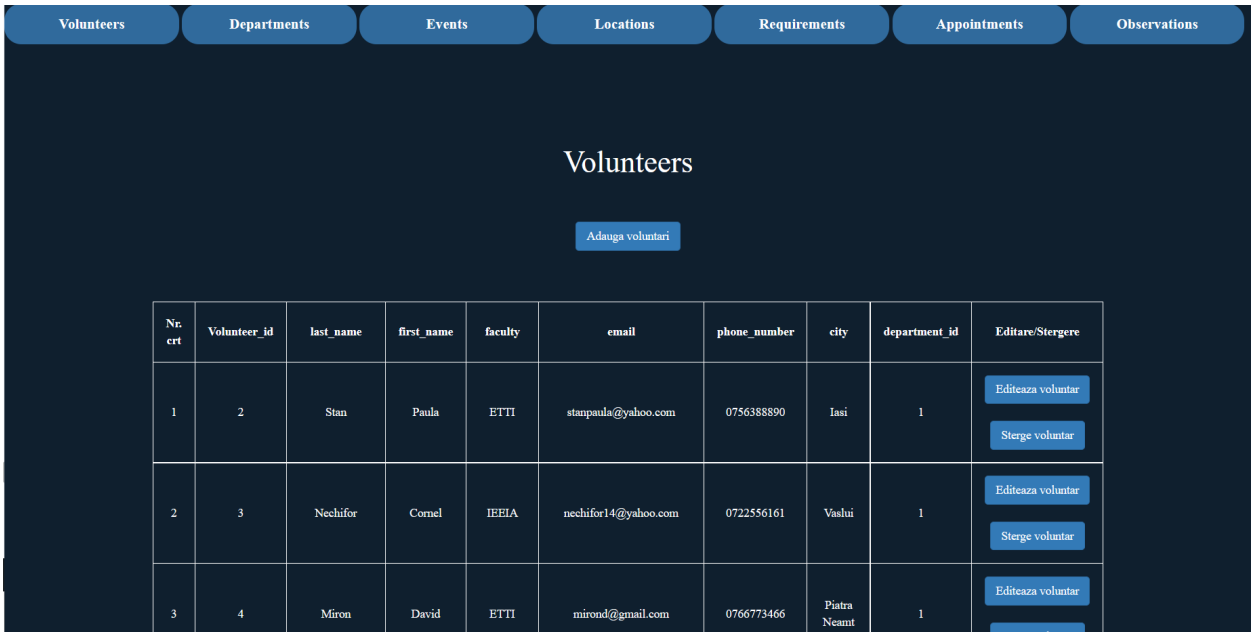
Conectarea la baza de date

Conexiunea cu baza de date și efectuarea schimbărilor din aceasta s-a realizat cu cx_Oracle, modul de extensie Python care permite accesul la Oracle Database și a fost necesară descărcarea Oracle Instant Client.

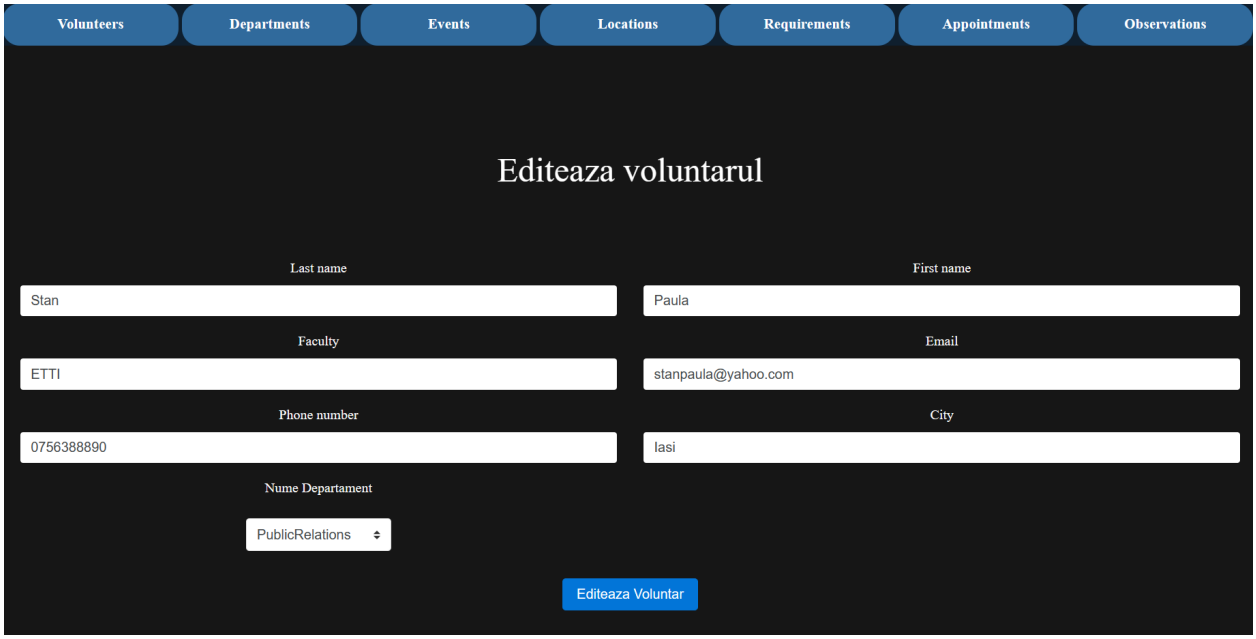
Interfața și exemple de cod

Pentru a ușura manipularea datelor din tabele, a fost construită o interfață grafică, utilizând Python și HTML.

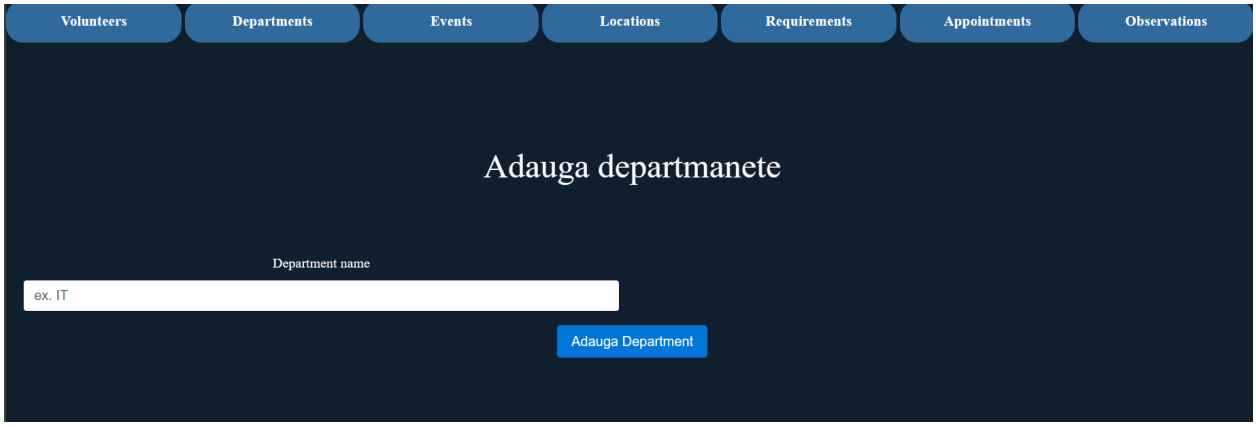
Interfața face posibilă interacțiunea cu fiecare tabelă (modificarea, ștergerea, adăugarea de noi valori). Există câte un tab corespunzător fiecărei tabele din baza de date.



Așa arată interfața când este deschisă



Pagina pentru editarea datelor despre voluntar



Pagina pentru adăugarea unei intrări în tabela department

```python
@app.route('/volunteers')
def vol():
    volunteers = []

    cur = con.cursor()
    cur.execute('select * from volunteer')

    for result in cur:
        volunteer = {}
        volunteer['volunteer_id'] = result[0]
        volunteer['last_name'] = result[1]
        volunteer['first_name'] = result[2]
        volunteer['faculty'] = result[3]
        volunteer['email'] = result[4]
        volunteer['phone_number'] = result[5]
        volunteer['city'] = result[6]
        volunteer['department_id'] = result[7]
        volunteers.append(volunteer)
    cur.close()
    return render_template('volunteers.html', volunteers=volunteers)
```

Exemplu de cod pentru vizualizarea datelor din tabela volunteer - cu utiilizarea funcției SELECT

```python
@app.route('/editVolunteer', methods=['POST'])
def edit_vol():
    vol = 0
    dep = 0

    cur = con.cursor()
    last_name = request.form['last_name']
    first_name = request.form['first_name']
    cur.execute('select volunteer_id from volunteer where last_name=' + "'" + last_name + "'" +'and first_name= '+ "'" + first_name + "'")
    for result in cur:
        vol = result[0]
    cur.close()

    faculty = request.form['faculty']
    email = request.form['email']
    phone_number = request.form['phone_number']
    city = request.form['city']
    department_name = "'" + request.form['department_name'] + "'"
    cur = con.cursor()
    cur.execute('select department_id from department where department_name=' + department_name)
    for result in cur:
        dep = result[0]
    cur.close()
    cur = con.cursor()
    query = """UPDATE volunteer SET last_name=:ln, first_name=:fn, faculty=:f, email=:em, phone_number=:pn, city=:c, department_id=:d where volunteer_id=:v"""

    cur.execute(query, [last_name, first_name, faculty, email, phone_number, city, dep, vol])
    cur.execute('commit')
    return redirect('/volunteers')
```

```python
@app.route('/getVolunteer', methods=['POST'])
def get_vol():
    vol = request.form['volunteer_id']
    cur = con.cursor()
    cur.execute('select * from volunteer where volunteer_id=' + "'" + vol + "'")

    vols = cur.fetchone()
    volunteer_id = vols[0]
    last_name = vols[1]
    first_name = vols[2]
    faculty = vols[3]
    email = vols[4]
    phone_number = vols[5]
    city = vols[6]
    department_id = vols[7]
    cur.close()
    department_name = ''
    cur = con.cursor()
    cur.execute('select department_name from department where department_id=' + str(department_id))
    for result in cur:
        department_name = result[0]
    cur.close()

    dep = []
    cur = con.cursor()
    cur.execute('select department_name from department')
    for result in cur:
        dep.append(result[0])
    cur.close()

    return render_template('editVolunteer.html', department=dep, department_name=department_name, last_name=last_name,
                           first_name=first_name, faculty=faculty, email=email, phone_number=phone_number,
                           city=city)
```

Exemplu de cod pentru editarea unei valori din tabela volunteer - cu utilizarea funcției UPDATE

```python
@app.route('/addDepartment', methods=['GET', 'POST'])
def add_dep():
    error = None
    if request.method == 'POST':
        dep = 0
        cur = con.cursor()
        cur.execute('select max(department_id) from department')
        for result in cur:
            dep = result[0]
        cur.close()
        dep += 1
        cur = con.cursor()
        values = []
        values.append("'" + str(dep) + "'")
        values.append("'" + request.form['department_name'] + "'")

        fields = ['department_id', 'department_name']
        query = 'INSERT INTO %s (%s) VALUES (%s)' % ('department', ', '.join(fields), ', '.join(values))

        cur.execute(query)
        cur.execute('commit')
        return redirect('/departments')

    else:
        dep = []
        cur = con.cursor()
        cur.execute('select department_name from department')
        for result in cur:
            dep.append(result[0])
        cur.close()

        return render_template('addDepartment.html', department=dep)
```

Exemplu de cod din python pentru adăugarea unei intrări în tabela department - cu utilizarea funcției INSERT

```python
@app.route('/delDepartment', methods=['POST'])
def del_dep():
    dep = request.form['department_id']
    cur = con.cursor()
    cur.execute('delete from department where department_id=' + dep)
    cur.execute('commit')
    return redirect('/departments')
```

Exemplu de cod din python pentru ștergerea unei intrări din tabela department - cu utilizarea funcției DELETE