UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” IAȘI

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

DOMENIUL: Calculatoare și tehnologia informației

Platformă de gestionare a inventarului și al angajaților unui spital

Temă la disciplina

Baze de date

Student: Diaconu Rareș-George - 1306A

Cadru didactic coordonator: Cătălin Mironeanu

Cuprins:

Titlu capitol **Pagina**

1. Introducere 2
2. Descrierea tehnologiilor folosite - Front-end 3
3. Descrierea tehnologiilor folosite - Back-end 3
4. Structura tabelelor 4
5. Conectarea la baza de date 6
6. Capturi de ecran și exemple de cod 7

1.Introducere

Proiectul se axează pe dezvoltarea unei platforme avansate de gestionare a inventarului și a angajaților într-un mediu spitalicesc, folosind puterea și eficiența bazelor de date relaționale. Scopul principal al acestei platforme este să ofere un instrument comprehensiv și intuitiv pentru administrarea eficientă a resurselor și personalului medical în cadrul spitalului.

Obiective principale:

1 .Gestionarea Departamentelor:

- Monitorizarea detaliată a fiecărui departament, inclusiv nume, manager și echipament disponibil.

2. Administrarea Cadrelor Medicale:

- Înregistrarea, monitorizarea și gestionarea personalului medical, inclusiv medici, asistente și personal administrativ.

3. Managementul Pacienților:

- Înregistrarea detaliată a pacienților, inclusiv informații personale, date de naștere și istoric medical.

4. Inventarul Echipamentelor Medicale:

- Urmărirea și gestionarea inventarului echipamentelor medicale în fiecare departament.

5. Tranzacții cu Medicamente:

- Gestionarea achizițiilor și vânzărilor de medicamente, inclusiv informații despre furnizori și tranzacții anterioare.

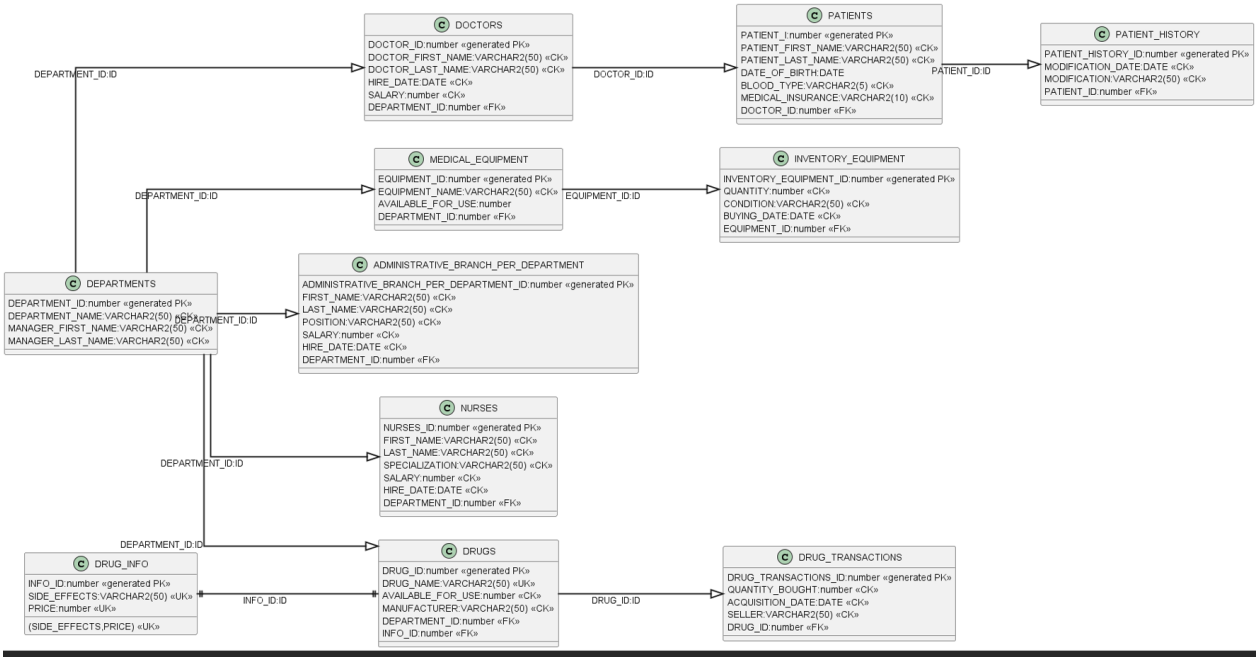
2. Descrierea tehnologiilor folosite - Front-end

Pentru partea de front-end, s-a folosit HTML și CSS. HTML este limbajul standard utilizat pentru structurarea și prezentarea conținutului pe paginile web. Am implementat elemente HTML pentru a defini structura și organizarea informațiilor afișate pe interfața utilizator. Etichetele HTML au fost utilizate pentru a crea formulare, liste, tabele și alte componente esențiale în interfața web. CSS a fost folosit pentru a gestiona aspectul și stilul paginilor web. Am aplicat reguli CSS pentru a controla aspecte precum culorile, fonturile, dimensiunile și alinierea elementelor HTML. Astfel, am asigurat o prezentare coerentă și atrăgătoare a informațiilor, facilitând navigarea și înțelegerea conținutului. Paginile sunt generate de server cu ajutorul procesorului de șabloane Jinja, inclus în Flask, ce permite crearea unei pagini sub formă de șablon HTML. Jinja2 este un motor de șabloane folosit în Flask pentru a facilita integrarea datelor din Python în șabloanele HTML. Acesta a permis generarea dinamică a conținutului paginilor web în funcție de datele extrase din baza de date.

3.Descrierea tehnologiilor folosite - Back-end

Pentru partea de back-end, s-a folosit Flask. Flask, un framework web pentru limbajul de programare Python, a fost ales pentru a dezvolta partea de backend a platformei. Acesta oferă o abordare simplă și modulară pentru construirea aplicațiilor web, facilitând gestionarea rutelor, cererilor HTTP și interacțiunii cu baza de date. Flask se evidențiază prin flexibilitate și ușurință în implementare, făcându-l ideal pentru acest proiect. Am implementat conexiunea la baza de date Oracle prin intermediul OracleDB. OracleDB reprezintă un modul Python destinat conectării la baza de date Oracle. A fost utilizat pentru a facilita interacțiunea cu sistemul de gestiune a bazelor de date Oracle, permițând astfel manipularea eficientă a datelor stocate în baza de date a platformei noastre. Prin intermediul OracleDB, am putut executa interogări SQL, actualizări și inserări de date într-un mod securizat și eficient.

4. Structura tabelelor



S-a ales folosirea unei coloane de tip autoincrement drept cheie primară pentru toate tabelele din mai multe motive:

- Simplificarea tabelelor și a lucrului cu acestea prin eliminarea cheilor primare compuse acolo unde ar fi apărut
- Simplificarea aplicației prin existența unei chei primare de un singur tip
- Unicitatea cheilor primare: Prin utilizarea auto-incrementării, fiecare nouă înregistrare va primi automat o valoare de cheie primară unică, evitând astfel riscul de conflicte sau duplicări ale cheilor primare.
- Eficiența în execuția operațiilor de inserare: Auto-incrementarea poate contribui la optimizarea performanței în cazul operațiilor de inserare, deoarece nu mai este nevoie să furnizați explicit valori pentru cheia primară și să vă asigurați că acestea sunt unice. Baza de date gestionează acest aspect în mod automat.

Pentru toate tabelele s-au folosit constrângeri CHECK pentru a verifica că valorile sunt corect introduse. Toate atributele de tip _id au constrângeri de tip FOREIGN KEY către tabela numită.

1. Departamente

În cadrul acestei tabele sunt definite departamentele(managerul, numele) spitalului.

Tabela departamente este în 3NF deoarece fiecare valoare este atomică, foreign key-ul către nume este unic (este cheie candidat), atributele non-cheie (numele) depind de cheia primară și nu depind de atribute non-cheie.

Constrângeri:

- DEPARTMENT_NAME_CK pentru a preveni existența literelor în nume;
- MANAGER_FIRST_NAME_CK pentru a preveni existența literelor în nume;
- MANAGER_LAST_NAME_CK pentru a preveni existența literelor în nume;

2. Doctori

În cadrul acestei tabele sunt definiți doctorii(data angajării, salariul, departamentul, numele). Tabela doctori este în 3NF analog tabelului departamente.

Constrângeri:

- DOCTOR _LAST_NAME_CK pentru a preveni existența literelor în nume;
- DOCTOR_LAST_NAME_CK pentru a preveni existența literelor în nume;
- HIRE_DATE_CK pentru a preveni introducerea unei date din viitor;
- SALARY_CK pentru a preveni introducerea unui salariu negativ;
- DEPARTMENT_DOCTOR_FK un foreign key către tabelul departamente.

3. Pacienți

În cadrul acestei tabele sunt definiți pacienții(numele, data nașterii, grupa de sânge, asigurarea medicală,doctorul). Tabela pacienți este în 3NF analog tabelului departamente.

Constrângeri:

- PATIENT _LAST_NAME_CK pentru a preveni existența literelor în nume;
- PATIENT _FIRST _NAME_CK pentru a preveni existența literelor în nume;
- BLOOD_TYPE_CK pentru a preveni introducerea unei grupe de sânge care nu există;
- MEDICAL_INSURANCE_CK pentru a preveni introducerea unei asigurări medicale care nu există
- DOCTOR_ID_FK un foreign key către tabelul doctori.

4. Echipament medical

În cadrul acestei tabele sunt definite echipamentele medicale(numele, câte pot fi folosite).Tabela echipament medical este în 3NF analog tabelului departamente.

Constrângeri:

- EQUIPMENT_NAME_CK pentru a preveni existența literelor în nume;
- AVAILABLE_FOR_USE_CK pentru a preveni introducerea unui cantități negative;
- DEPARTMENT_ID_ME_FK un foreign key către tabelul departamente.

5. Inventar echipamente

În cadrul acestei tabele este definit inventarul echipamentelor(numele, etc.).Tabela echipament medical este în 3NF analog departamente.

Constrângeri:

- QUANTITY_FOR_USE_CK pentru a preveni introducerea unui salariu negativ;
- CONDITION_CK pentru a preveni introducerea unei condiții care nu există.
- BUYING_DATE_CK pentru a preveni introducerea unei date din viitor;
- EQUIPMENT_ID_FK un foreign key către tabelul echipamente medicale.

6. ADMINISTRATIVE_BRANCH_PER_DEPARTMENT

În cadrul acestei tabele este definită conducerea unui departament(numele, salariu , etc.).Tabela echipament medical este în 3NF analog tabelului departamente.

Constrângeri:

- FIRST_NAME_A_CK pentru a preveni existența literelor în prenume;
- LAST_NAME_A_CK pentru a preveni existența literelor în nume;
- POSITION_CK pentru a preveni existența literelor în poziția angajatului;
- SALARY_P_CK pentru a preveni introducerea unui salariu negativ;
- ADMINISTRATIVE_BRANCH_PER_DEPARTMENT_DEPARTMENTS__FK un foreign key către tabelul departamente.

7. ASISTENTE

În cadrul acestei tabele sunt definite asistentele medicale(numele, salariu , etc.).Tabela asistente este în 3NF analog tabelului departamente.

Constrângeri:

- FIRST_NAME_N_CK pentru a preveni existența literelor în prenume;
- LAST_NAME_N_CK pentru a preveni existența literelor în nume;
- SPECIALIZATION_CK pentru a preveni existența literelor în specializare;
- SALARY_N_CK pentru a preveni introducerea unui salariu negativ;
- NURSES_DEPARTMENTS__FK un foreign key către tabelul departamente.

8. ISTORIC PACIENȚI

În cadrul acestei tabele este definit istoricul unui pacient(modificarea istoricului, data modificării).Tabela istoric pacienți este în 3NF analog tabelului departamente.

Constrângeri:

- MOD_CK pentru a preveni existența literelor în modificare;
- PATIENT_HISTORY_PATIENTS__FK un foreign key către tabelul pacienți.

9. MEDICAMENTE

În cadrul acestei tabele sunt definite medicamentele (numele, cantitatea, etc.). Tabela medicamente este în 3NF analog tabelului departamente.

Constrângeri:

- DRUG_NAME_UK de tip unique;
- DRUG_NAME_CK pentru a preveni existența literelor în nume;
- AVAILABLE_FOR_USE_CK pentru a preveni introducerea unui cantități negative;
- MAN_CK pentru a preveni existența literelor în numele producătorului;
- INFO_ID_UK de tip unique;
- DEPARTMENT_ID_DRUGS_FK un foreign key către tabelul departamente.

10.INFORMAȚII MEDICAMENT

În cadrul acestei tabele sunt definite toate informațiile unui medicament (efecte secundare, preț). Tabela informații medicament este în 3NF analog tabelului departamente.

Constrângeri:

- SIDE_EFFECTS_CK pentru a preveni existența literelor în efectele secundare;
- PRICE_CK pentru a preveni introducerea unui preț negativ;
- DRUG_UK de tip unique.

11.TRANZACȚII MEDICAMENTE

În cadrul acestei tabele sunt definite toate informațiile despre tranzacțiile de medicamente (cantitate, vânzător). Tabela tranzacții medicamente este în 3NF analog tabelului departamente.

Constrângeri:

- QUANTITY_BOUGHT_CK pentru a preveni introducerea unui cantități negative;
- SELLER_CK pentru a preveni existența literelor în numele vânzătorului;
- DRUG_TRANSACTIONS_DRUGS__FK un foreign key către tabelul medicamente.

6. Conectarea la baza de date

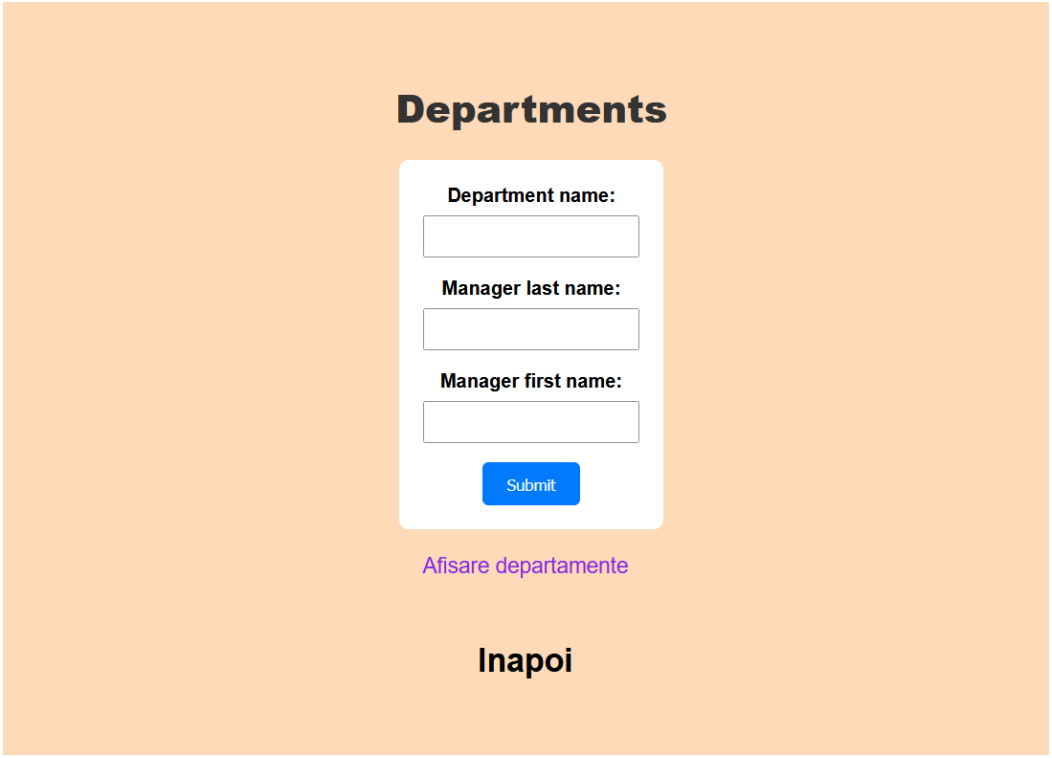
```
connection = oracledb.connect(user="bd006", password="ms006", host="81.180.214.85", port=1539, service_name="orcl")
```

Pentru conectarea la baza de date Oracle, s-a folosit driver-ul python-oracledb. Funcția `connect()` din modulul util apelează funcția corespunzătoare driver-ului. Pentru controlul tranzacțiilor dispunem de metodele commit() și rollback(). O tranzacție:

```
    cursor = connection.cursor()

    cursor.execute( statement: "INSERT INTO DRUG_INFO (SIDE_EFFECTS,PRICE) VALUES (:side_effects,:price)",
                    parameters: {'side_effects': side_effects, 'price': price})
    info_id = cursor.execute(
         statement: "SELECT * FROM DRUG_INFO where side_effects = :side_effects and price = :price",
         parameters: {'side_effects': side_effects, 'price': price}).fetchone()[0]
    cursor.execute( statement: "INSERT INTO DRUGS(DRUG_NAME,AVAILABLE_FOR_USE,DEPARTMENT_ID,MANUFACTURER,INFO_ID)"
                   "VALUES (:DRUG_NAME,:AVAILABLE_FOR_USE,:DEPARTMENT_ID,:MANUFACTURER,:INFO_ID)",
                    parameters: {'DRUG_NAME': name, 'AVAILABLE_FOR_USE': afu, 'DEPARTMENT_ID': id_dept,
                     'MANUFACTURER': man, 'INFO_ID': info_id})

    connection.commit()

except oracledb.DatabaseError as e:
    connection.rollback()
    print(f"Eroare la inserare in baza de date: {str(e)}")
```

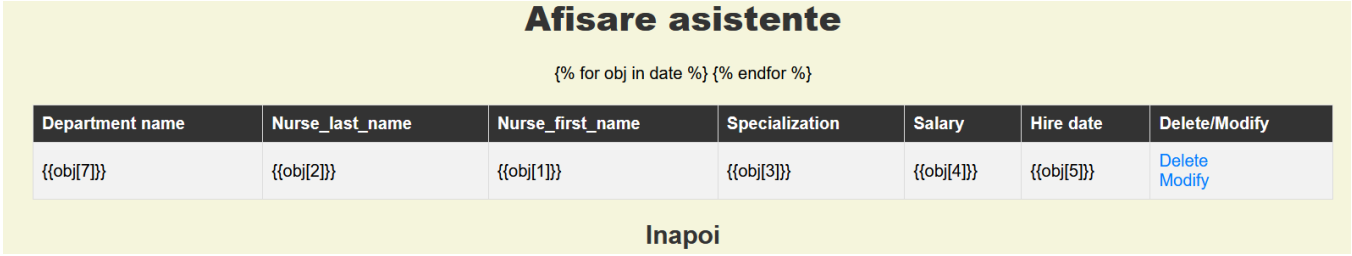
7. Capturi de ecran și exemple de cod

Inserarea într-un departament:




```python
@app.route( rule: "/departments/index_dept", methods=['GET', 'POST'])
def department():
    if request.method == 'POST':
        try:
            cursor = connection.cursor()
            dname = request.form['dname']
            manager_last = request.form['manager_last']
            manager_first = request.form['manager_first']

            if re.search( pattern: r'\d', dname):
                raise ValueError("Eroare: Numele nu poate contine cifre")
            if re.search( pattern: r'\d', manager_first):
                raise ValueError("Eroare: Numele nu poate contine cifre")
            if re.search( pattern: r'\d', manager_last):
                raise ValueError("Eroare: Numele nu poate contine cifre")

            cursor.execute( statement: "INSERT INTO DEPARTMENTS (department_name,manager_first_name,manager_last_name) "
                            "VALUES (:department_name,:manager_first_name,:manager_last_name)",
                            parameters: {'department_name': dname, 'manager_first_name': manager_first,
                             'manager_last_name': manager_last})
            connection.commit()
            return render_template('/departments/succes_inserare.html')
        except oracledb.DatabaseError as e:
            print(f"Eroare la inserare in baza de date:{str(e)}")
    return render_template('/departments/index_dept.html')
```

Afișarea unui asistente:



Afisare asistente

{% for obj in date %} {% endfor %}

Department name	Nurse_last_name	Nurse_first_name	Specialization	Salary	Hire date	Delete/Modify
{{obj[7]}}	{{obj[2]}}	{{obj[1]}}	{{obj[3]}}	{{obj[4]}}	{{obj[5]}}	Delete Modify

Inapoi

```python
@app.route('/nurses/afisare_nurses')
def afisare_nurses():
    cursor = connection.cursor()
    date: list[tuple[...]]
    date = cursor.execute(
        """SELECT N.*,DP.DEPARTMENT_NAME
    FROM NURSES N
    JOIN DEPARTMENTS DP ON N.DEPARTMENT_ID = DP.DEPARTMENT_ID
    ORDER BY N.NURSES_ID""").fetchall()
    return render_template( template_name_or_list: '/nurses/afisare_nurses.html', date=date)
```

Modificarea unui echipament medical:

Modify medical equipment

Choose a department:

{{dept[1]}}

Equipment name:

Available for use:

Modify

Inapoi

```python
@app.route( rule: '/medical/modify_med/<int:equipment_id>', methods=["GET", "POST"])
def modify_med(equipment_id: int):
    cursor = connection.cursor()
    departments = cursor.execute("SELECT * FROM DEPARTMENTS").fetchall()
    if request.method == 'POST':
        try:
            cursor = connection.cursor()
            name = request.form['name']
            afu = int(request.form['afu'])

            if re.search( pattern: r'\d', name):
                raise ValueError("Eroare: Numele nu poate contine cifre")
            if afu < 0:
                raise ValueError("Eroare:Numarul disponibil nu poate fi negativ")

            query1 = """
                     UPDATE medical_equipment
                     SET equipment_name = :equipment_name,
                         available_for_use = :available_for_use
                     WHERE equipment_id = :equipment_id
                     """

            cursor.execute(query1,  parameters: {
                'equipment_name': name,
                'available_for_use': afu,
                'equipment_id': equipment_id
            })
```

```python
            connection.commit()

            return render_template('/medical/succes_modificare_med.html')
        except oracledb.DatabaseError as e:
            print(f"Eroare la inserare in baza de date:{str(e)}")
    return render_template( template_name_or_list: '/medical/modify_med.html', departments=departments)
```

Ștergerea unui doctor:

```python
@app.route('/doctors/delete_doc/<int:doctor_id>')
def delete_doc(doctor_id: int):
    cursor = connection.cursor()
    cursor.execute( statement: "DELETE FROM doctors WHERE doctor_id = :doctor_id",  parameters: {'doctor_id': doctor_id})
    connection.commit()
    return render_template('/doctors/succes_stergere_doc.html')
```