Universitatea tehnica “Gheorghe Asachi” Iași
Facultatea de Automatică și Calculatoare
Specializarea Calculatoare și Tehnologia Informației

Tema de casa
Disciplina : Baze de Date

Gestiunea produselor și vânzărilor unei cofetării

Coordonator,
Prof. Avram Sorin

Student,
Buțu Alexandra-Gabriela
Grupa 1306B

Iași, 2023

Titlul proiectului:

Gestiunea produselor și vânzărilor unei cofetării

Analiza, proiectarea și implementarea unei baze de date care să modeleze gestionarea produselor și vânzărilor unei cofetării.

Descrierea proiectului :

O cofetărie pune la dispoziție clienților o gamă largă de produse, cum ar fi: prăjituri, briose, fursecuri etc. . Aceste produse sunt pregătite după rețete autentice, din ingrediente atent selectate, fiecare produs având propria rețetă și de asemenea, un nume și un preț unic. Pe langa toate acestea, se adaugă și alte detalii cum ar fi alergeni, etichete (bio/vegan/de post etc.) si bineinteles, data producției și cea a expirarii.

Fiecare cumpărător se poate bucura de mai multe produse, plasând o comandă. Cofetăria tine evidenta preparării produselor din vitrină și a vânzărilor lor.

NU se ține cont de multiplele locații ale cofetăriei, doar de cea din orașul curent. NU se ține evidența produselor care au expirat sau care nu s-au vândut și trebuie retrase din stoc. NU se tine evidenta stocurilor în ceea ce privește produsele din vitrina și ingredientele.

În urma analizei cerințelor rezultă:

1. Entitatea **PRODUS** - această entitate reprezintă fiecare produs pe care cofetăria îl vinde
    - ID produs (valoare numerica, obligatorie, unică)
    - Denumire produs (valoare șir de caractere, obligatorie,unica)
    - Preț (valoare numerică, obligatorie)
    - Gramaj (valoare numerică)

2. Entitatea **DETALII PRODUS** - caracteristici ale produsului vandut
    - Alergeni (valoare sir de caractere, obligatorie)
    - Data fabricarii(valoare tip date, obligatorie)
    - Data expirarii (valoare tip date, obligatorie)
    - Eticheta (valoare sir de caractere, obligatorie)

3. Entitatea **CLIENT**- cumpărătorii care achiziționează produsele cofetăriei
    - ID client (valoare numerica, obligatorie, unica)
    - Nume (valoare șir de caractere, obligatorie)
    - Prenume (valoare șir de caractere, obligatorie)
    - Număr de telefon (valoare numerică,unica, obligatorie)
    - Email (valoare numerică, opțional)

4. Entitatea **COMANDĂ** - comenzi plasate de clienți
    - ID comandă (valoare numerica, obligatorie, unica)
    - Data comenzii (valoare tip date, obligatorie)
    - Status (valoare șir de caractere, obligatorie)
    - Metodă de plată (valoare șir de caractere)

5. Entitatea **INGREDIENT** materiile prime utilizate pentru a pregăti produsele
    - ID ingredient (valoare numerică, obligatorie, unică)
    - Nume ingredient (valoare șir de caractere, obligatorie,unic)
    - Unitate de măsură (valoare șir de caractere)
    - Cantitate(valoare numerică,obligatorie)

6. Entitatea **RETETA** - cum se prepara un anumit produs
    - ID_reteta (valoare numerica, obligatorie, unica)
    - Nume_reteta (valoare șir de caractere, obligatorie)
    - Timp_preparare_minute(valoare numerică)
    - Specificatii (valoare sir de caractere, obligatorie)

- Cantitate (valoare numerica, obligatorie)

În proiectarea acestei bazei de date s-au identificat următoarele relații:

1:1(one to one)

- Între Produs și Detalii Produs va exista o relație 1:1, deoarece un produs are propriile sale detalii, acestea fiindu-i asociate exclusiv lui. Relația de legătură se va realiza prin atributul ID_Produs (cheie externă în tabela Detalii_produs).

1:n(one to many)

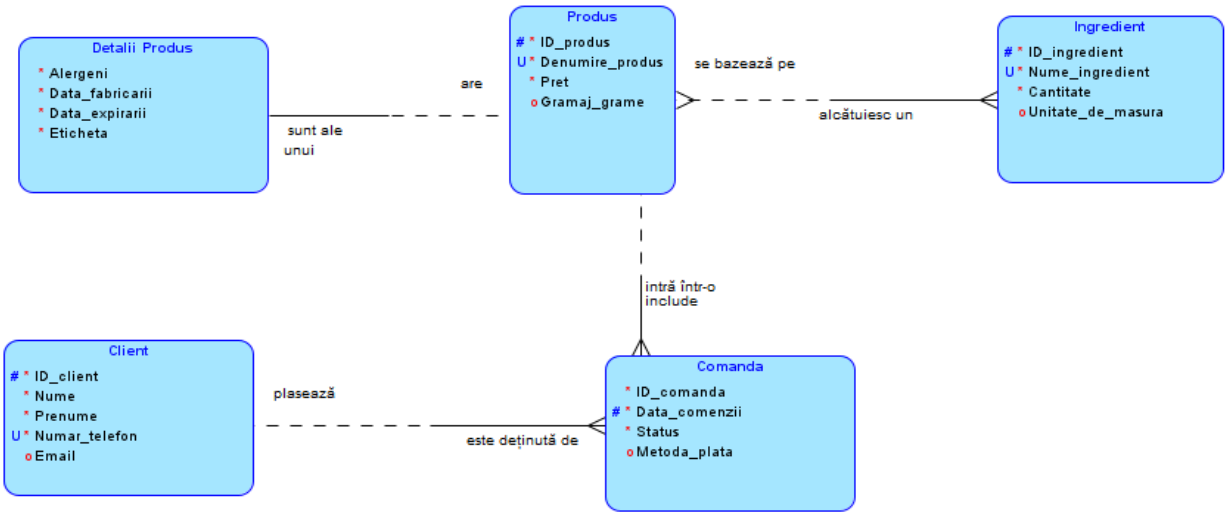
- Între Client și Comanda va exista o relație 1:n , întrucât un client poate plasa mai multe comenzi , însă o comanda este asociată unui singur client. Legătura dintre aceste tabele va avea loc prin ID_Client (cheie externă în tabela comanda).
- Între Produs și Comanda va exista o relație 1:n , deoarece într-o comanda se pot regasi mai multe produse, dar fiecare produs specific este regasit doar într-o comanda.Legătura dintre aceste tabele va avea loc prin ID_Produs(cheie externă în tabela comanda).

m:m (many to many)

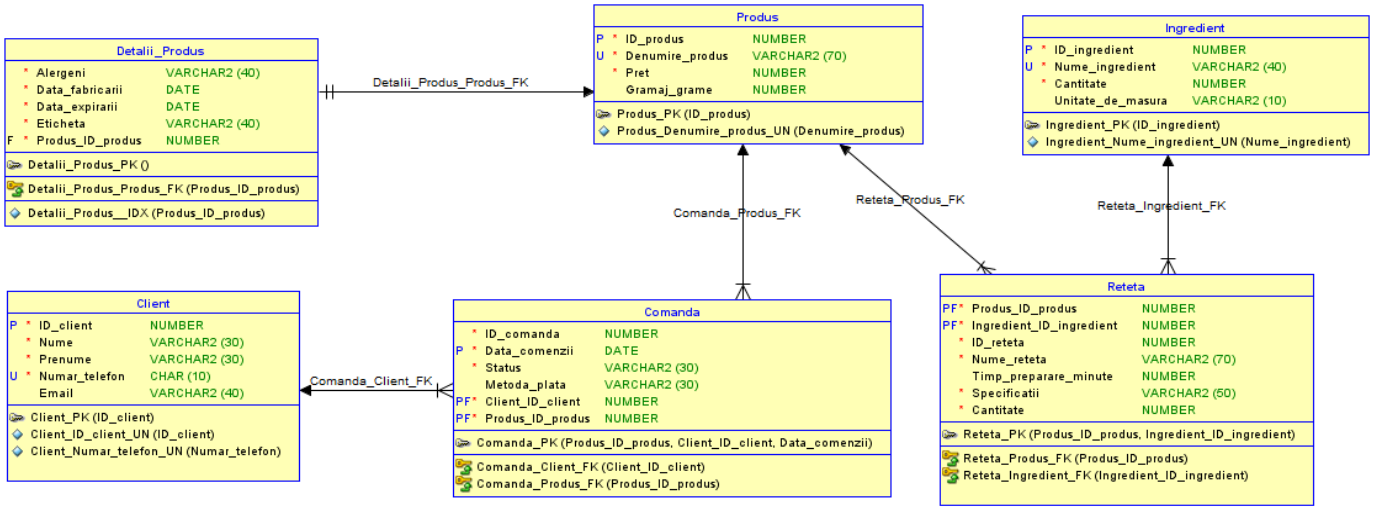
- Relația dintre Produs și Ingredient este inițial de tip mulți la mulți, deoarece un produs poate fi făcut din mai multe ingrediente, iar la randul sau, un ingredient poate fi utilizat în mai multe produse. Normalizarea implica transformarea acestei relații (spargerea) în relații de tip many to one și one to many, ceea ce se poate obține prin introducerea tabelei intermediare Reteta.
- Produs -> Reteta (1:m): Un produs poate avea mai multe rețete, deci se leaga prin prin ID_produs(cheie externa) in Reteta.

- Reteta -> Ingredient (m:1): O rețetă poate folosi mai multe ingrediente, deci se leaga prin ID_ingredient(cheie externa) in Reteta
  Aceasta normalizare optimizează structura bazei de date, eliminand redundantă și asigurand o structura mai clară și ușor de întreținut. De asemenea, facilitează interogarile și operațiile de modificare asupra datelor. Aceste relații ajută la conectarea și gestionarea datelor din baza de date, permitand urmărirea modului în care produsele, retetele, ingredientele, clienții și comenzile sunt interconectati în cadrul sistemului cofetariei.

Modelul logic



Modelul relational



Descrierea constrângerilor :

În tabela **Client** este definită pe câmpul email (atribut opțional) o constrângere de tipul unique numită client_email_un (nu pot exista mai multe persoane care au aceeași adresă de mail). În același timp câmpul email mai are o constrângere de tip check care verifică dacă adresa introdusă respecta formatul standard.
Pe câmpul nume și prenume sunt definite constrângeri de tip check pentru a verifica că numele și prenumele au o lungime > 1 și conțin doar litere .
Atributul Numar_telefon va avea o constrângere unique (client_numar_telefon_un), deoarece 2 persoane nu pot avea același număr de telefon. Acesta mai are și o constrângere de tip check care verifică lungimea sa fie 10 și corectitudinea scrierii numărului.
Cheia primara este id-ul clientului care are **autoincrement**. Astfel, ne asigurăm că id_client primește valori unice și crescătoare(incepand de la 1) automat la fiecare inserare a unui nou rand in tabel, daca valoarea nu a fost furnizata explicit in instructiunea insert.

În tabela **Comanda** avem constrângeri de tip check pe campul status, acesta poate avea doar valorile: 'In curs de preparare'/ 'Livrata'/'Preluata'. La fel pentru campul comanda care poate lua valorile: 'Cash', 'Cu card'.
Cheia primara e formata din campurile produs_id_produs, client_id_client si data_comenzii , astfel ne permite sa introducem mai multe produse (la fel sau cu alt id), pe rand, in comanda unui client, fără a mai crea o tabela suplimentară. Nu avem nici autoincrement pentru a putea face inserari multiple pe randuri diferite.

În tabela **Detalii produs** exista un index unic Detalii_index_unic pentru a ne asigura ca fiecare valoare din coloana produs_id_produs din tabela detalii produs este unică. Aceasta constrângere se aplica pentru a preveni inserarea de valori duplicate în aceasta coloana.
Avem și un **trigger** Det_prod care are rolul de a preveni inserarea/actualizarea în tabelă a unor înregistrări ce conțin date de expirare în trecut sau date de fabricație în viitor, generand erori specifice in aceste situatii .

În tabela **Ingredient** avem ca primary key id-ul ingredientului care are autoincrement. Astfel, ne asigurăm că id_ingredient primește valori unice și crescătoare(incepand de la 1) automat la fiecare inserare a unui nou rand in tabel, daca valoarea nu a fost furnizata explicit in instructiunea insert.
Exista o constrângere de tip check care asigura ca lungimea numelui ingredientului sa fie minimum de 3 litere și o constrângere de tip unique pentru a ne asigura ca fiecare ingredient are denumire unică.

În tabela **Produs** avem ca primary key id-ul produsului care are autoincrement. Astfel, ne asigurăm că id_produs primește valori unice și crescătoare(incepand de la 1) automat la fiecare inserare a unui nou rand in tabel, daca valoarea nu a fost furnizata explicit in instructiunea insert.

Exista o constrângere de tip check care asigura ca lungimea denumirii produsului sa fie minimum de 5 litere și o constrângere de tip unique pentru a ne asigura ca fiecare produs are denumire unică.

În tabela **Reteta** primary key-ul e format din produs_id_produs, ingredient_id_ingredient pentru a putea introduce mai multe ingrediente, pe rand, în reteta corespunzătoare unui produs, fără a mai crea o tabela suplimentară.

Tehnologii folosite pentru Front-end și Back-end

Front-end:
Am implementat interfața prin care s-a făcut legătura cu baza de date cu ajutorul limbajelor de programare **python** (folosind cx_Oracle pentru conectarea cu baza de date),Flask, CSS (utilizat pentru a oferi estetică paginilor), HTML (utilizat pentru crearea structurii tuturor paginilor).

Back-end:
Am utilizat Oracle SQL Developer, Oracle SQL Developer Data Modeler pentru crearea tabelelor, realizarea modelului logic și modelului relațional, de unde a fost generat și scriptul de creare a tabelelor. Iar mai apoi,în Oracle SQL Developer s-a creat un fișier de populare a bazei de date

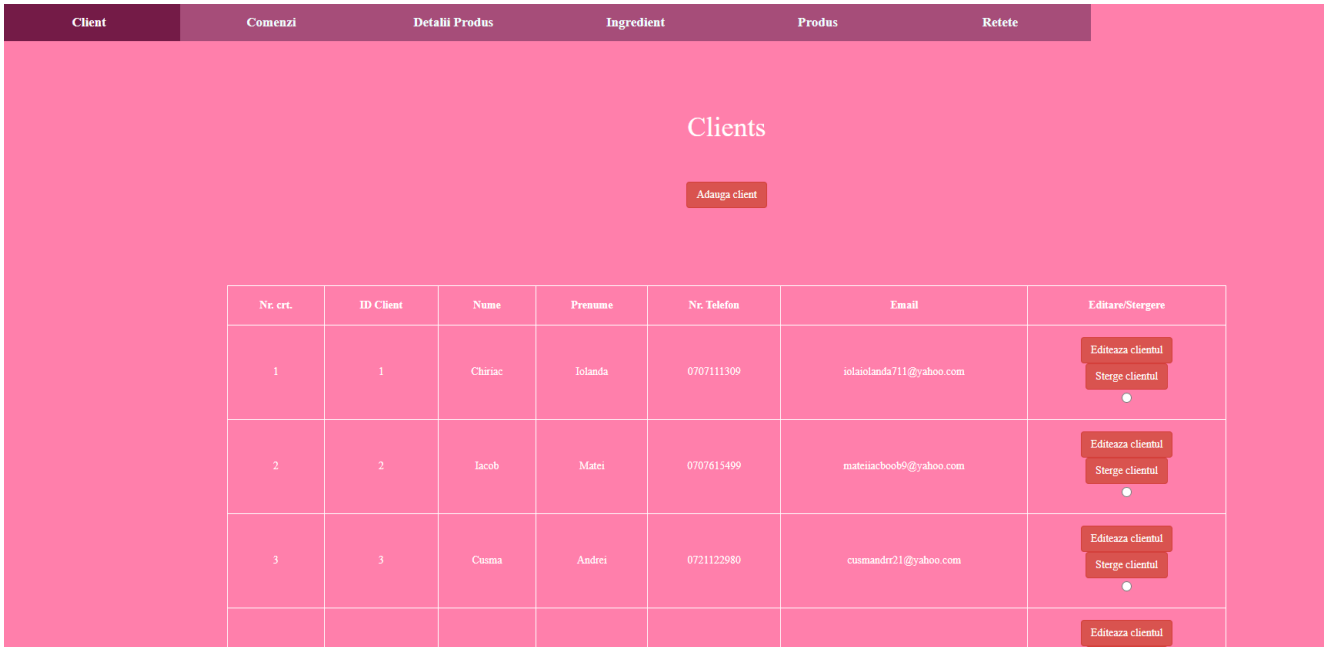
Descrierea modalității de conectare la baza de date din aplicație

Aplicația utilizează pachetul cx-Oracle și Oracle client pentru conectarea la baza de date. Pentru a realiza această conexiune, a fost esențial să descărcăm un folder de la companie, care conține diverse fișiere necesare, și să-l integrăm în aplicație. De

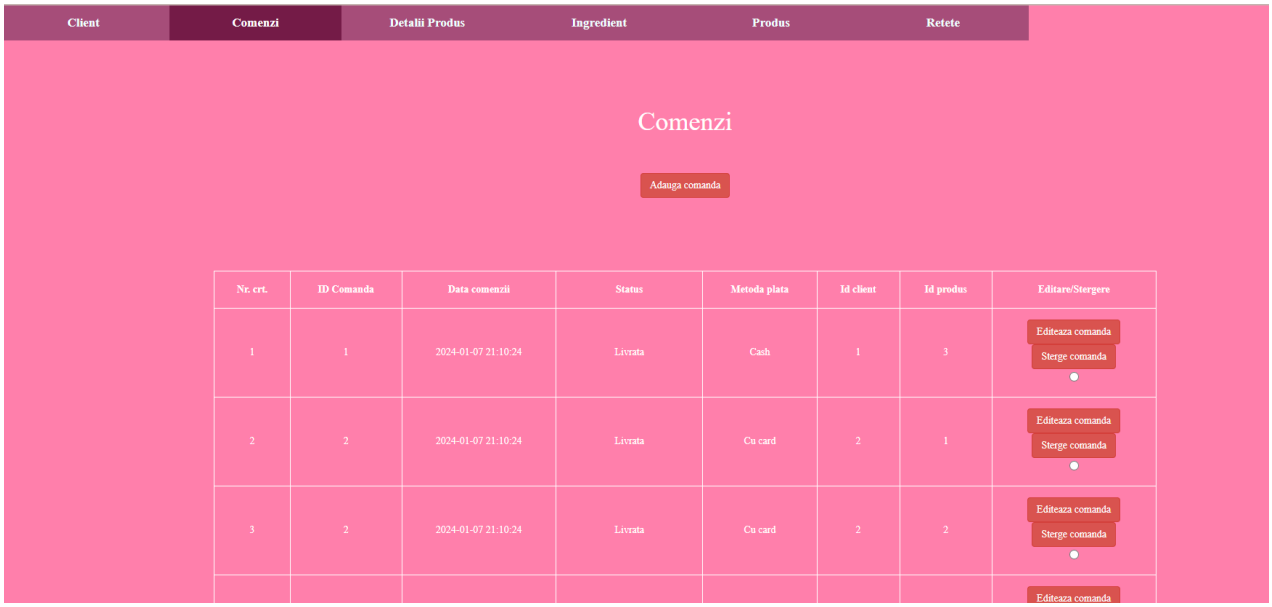
asemenea, au fost necesare informații specifice privind datele de autentificare pentru a facilita conectarea corectă la baza de date Oracle.

```python
from flask import Flask, render_template, request, redirect
import cx_Oracle
import os
from datetime import datetime

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

cx_Oracle.init_oracle_client(os.path.join(os.getcwd(), "instantclient_21_7"))
dsn_tns = cx_Oracle.makedsn( *args: 'bd-dc.cs.tuiasi.ro', '1539', 'orcl')
con = cx_Oracle.connect("bd015", "bd015", dsn_tns)
```

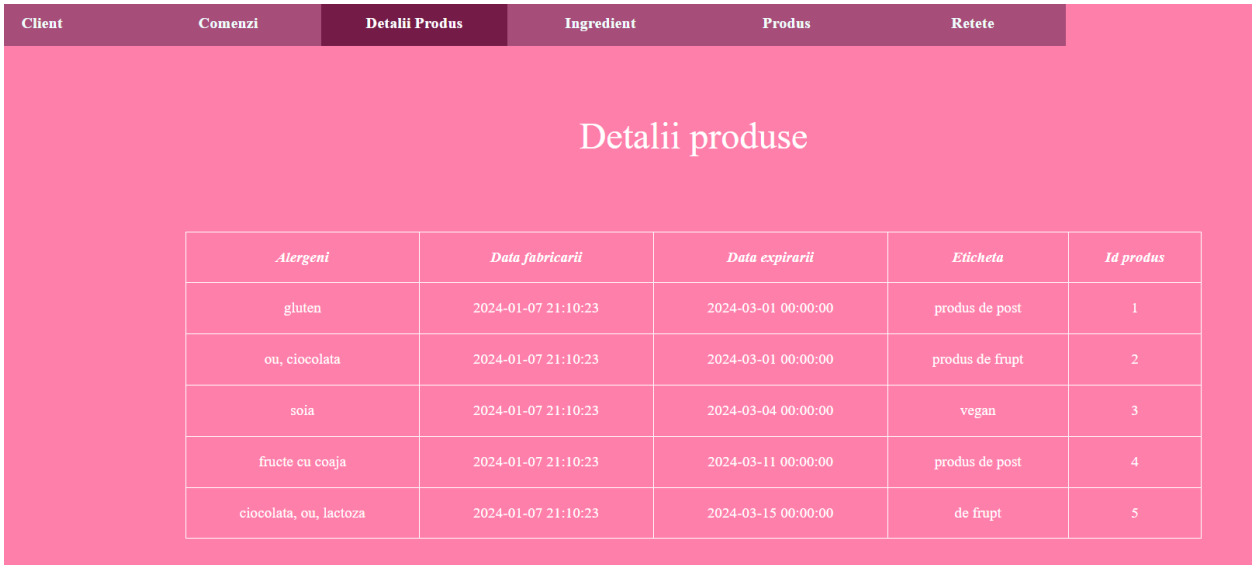
Vizualizare interfața pentru tabela CLIENT



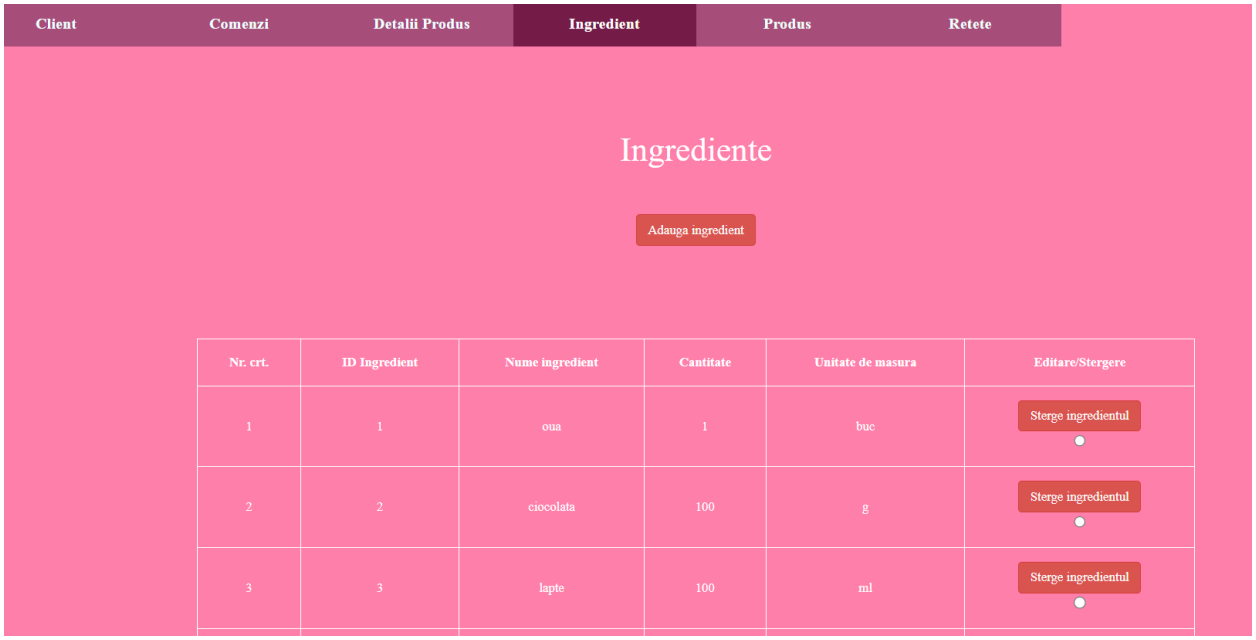
Vizualizare interfața pentru tabela COMANDA



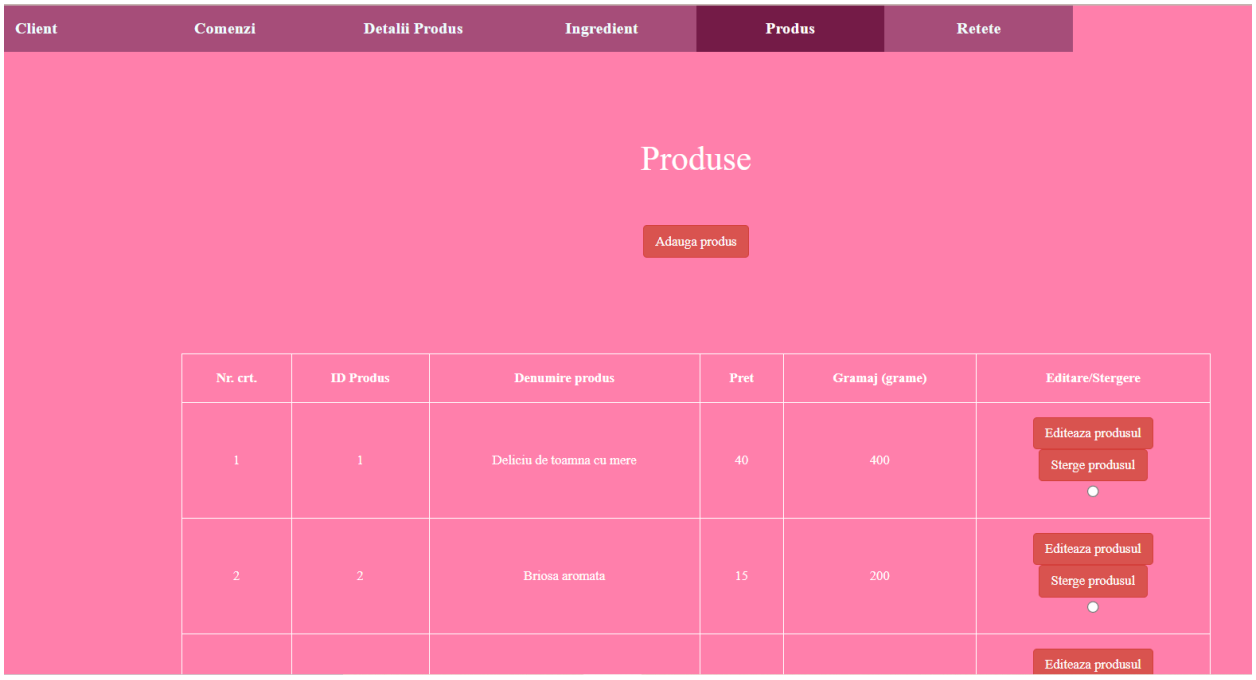
Vizualizare interfața pentru tabela DETALII PRODUS



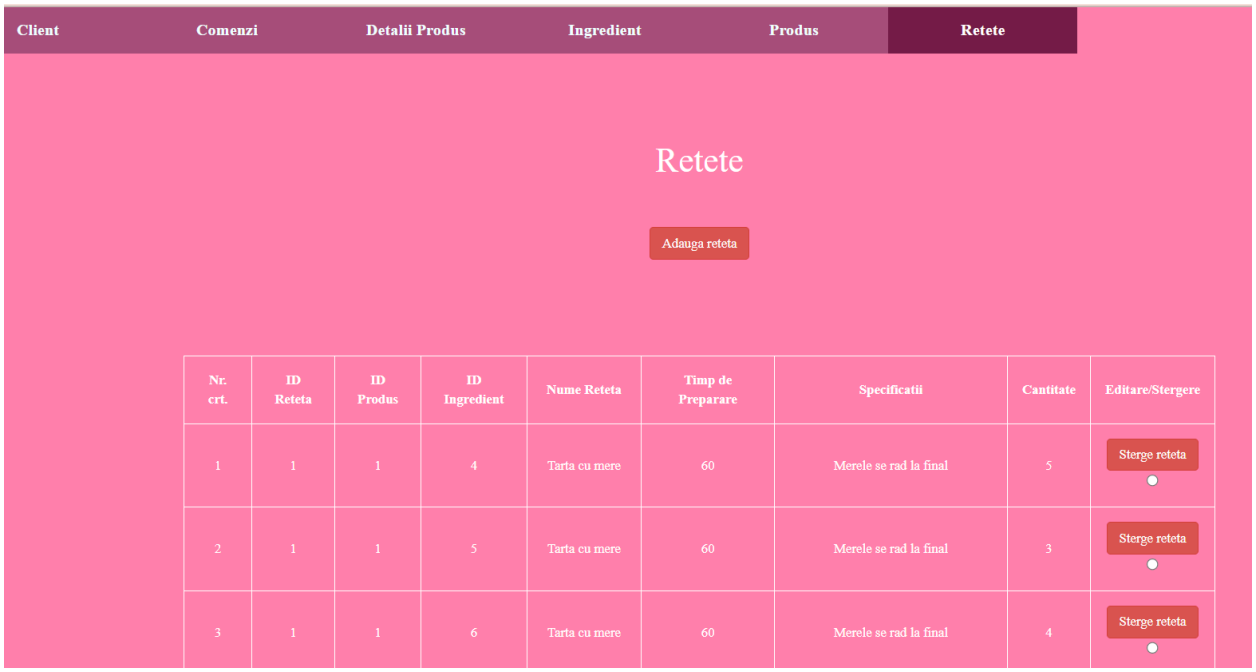
Vizualizare interfața pentru tabela INGREDIENT



Vizualizare interfața pentru tabela PRODUS

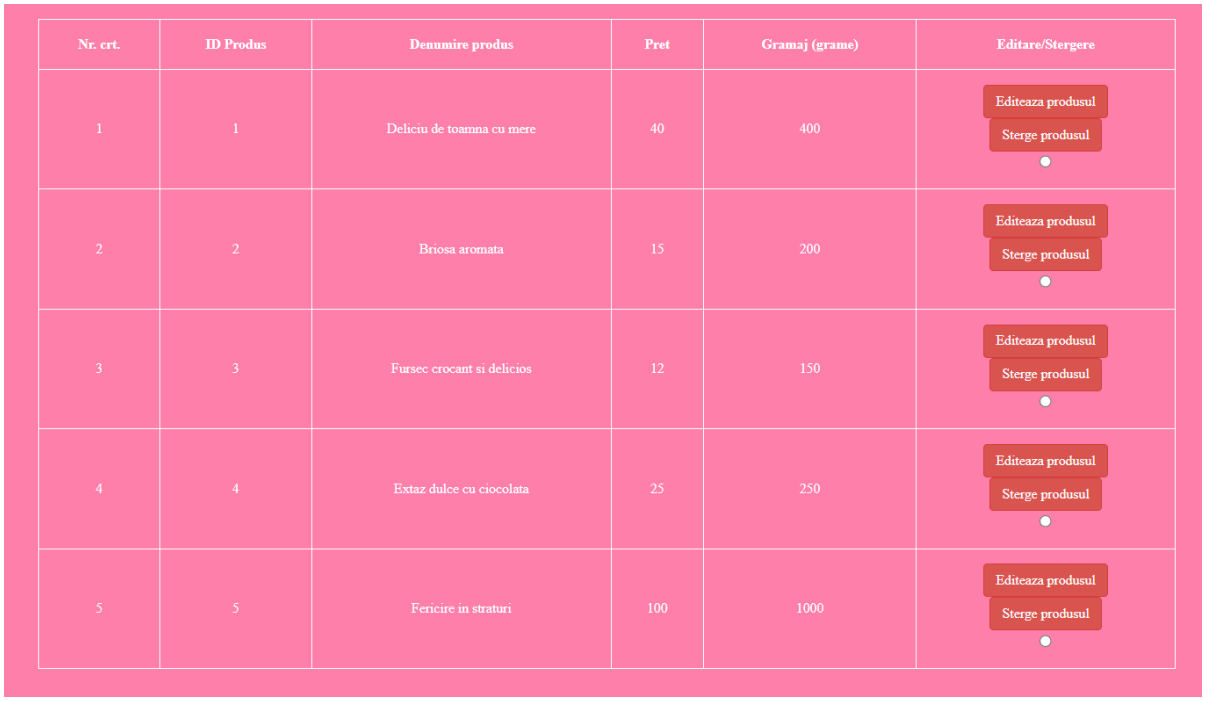


Vizualizare interfața pentru tabela RETETA



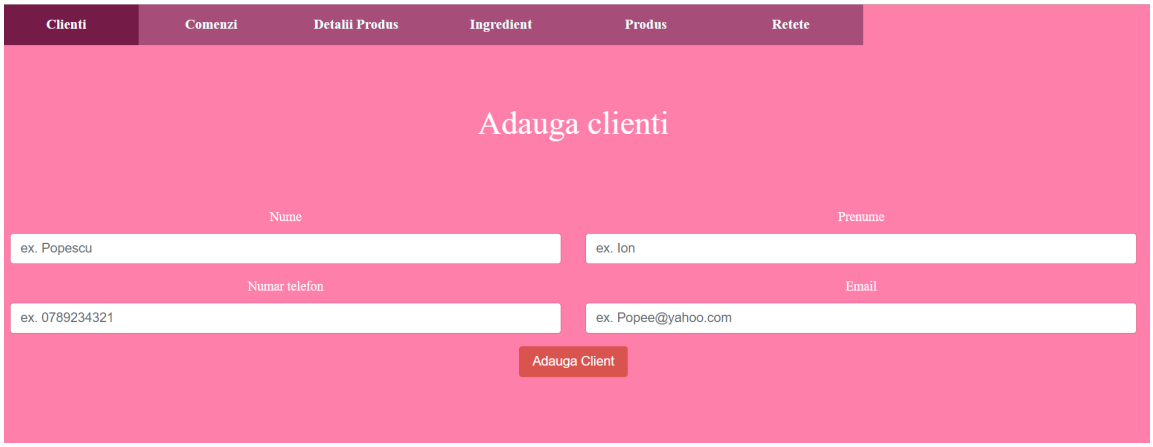
Instrucțiuni SQL folosite + exemple din aplicație :

SELECT - pentru a extrage datele din baza de date (exemplu : afisare produse)



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

    cur = con.cursor()
    cur.execute('SELECT * FROM produs')  # Modificati cu numele real al tabelei
    for result in cur:
        product = {
            'id_produs': result[0],
            'denumire_produs': result[1],
            'pret': result[2],
            'gramaj_grame': result[3]
        }
        products.append(product)
    cur.close()
    return render_template( template_name_or_list: 'Produs.html', products=products)
```

INSERT - pentru a insera date în tabele (de exemplu adaugare client in tabela CLIENT)




```python
@app.route( rule: '/addClient', methods=['GET', 'POST'])
def add_cl():
    error = None
    if request.method == 'POST':
        # Conectarea la baza de date și codul pentru inserarea datelor
        emp = 0
        cur = con.cursor()
        cur.execute('SELECT MAX(id_client) FROM client')
        for result in cur:
            emp = result[0]
        cur.close()

        emp += 1
        cur = con.cursor()
        values = []
        values.append("'" + str(emp) + "'")
        values.append("'" + request.form['nume'] + "'")
        values.append("'" + request.form['prenume'] + "'")
        values.append("'" + request.form['numar_telefon'] + "'")
        values.append("'" + request.form['email'] + "'")
```

```
    fields = ['id_client', 'nume', 'prenume', 'numar_telefon', 'email']
    query = 'INSERT INTO client (%s) VALUES (%s)' % (', '.join(fields), ', '.join(values))


    cur.execute(query)
    con.commit()
    return redirect('/clients')

# Redirectionează către pagina de clienti după adăugare
else:
    return render_template('addClient.html')
 # Returnează șablonul HTML pentru adăugarea unui client
```

DELETE - pentru a șterge o înregistrare dintr-o tabela prin intermediul butonului

```
@app.route( rule: '/deleteClient', methods=['POST'])
def del_cl():
    emp = request.form['id_client']

    cur = con.cursor()
    cur.execute('delete from comanda where client_id_client=' + emp)
    cur.execute('delete from client where id_client=' + emp)
    cur.execute('commit')
    return redirect('/clients')
```

Stergere in cascada - de ex: pentru a șterge un produs , produsul fiind inclus într-o comanda, având anumite detalii și o rețetă proprie (legatura intre mai multe tabele)

```
@app.route( rule: '/deleteProdus', methods=['POST'])
def del_prod():
    emp = request.form['id_produs']
    cur = con.cursor()
    cur.execute('delete from comanda where produs_id_produs=' + emp)
    cur.execute('delete from detalii_produs where produs_id_produs=' + emp)
    cur.execute('delete from reteta where produs_id_produs=' + emp)
    cur.execute('delete from produs where id_produs=' + emp)
    cur.execute('commit')
    return redirect('/products')
```

UPDATE - se face prin intermediul butonului de EDIT

De exemplu, pentru a modifica datele unui client

```python
@app.route( rule: '/editClient', methods=['POST'])
def edit_cl():
    try:
        id_client = "'" + request.form['id_client'] + "'"
        nume = "'" + request.form['nume'] + "'"
        prenume = "'" + request.form['prenume'] + "'"
        numar_telefon = "'" + request.form['numar_telefon'] + "'"
        email = "'" + request.form['email'] + "'"
        cur = con.cursor()

        query = "UPDATE client SET nume=%s, prenume=%s, numar_telefon=%s, email=%s WHERE id_client=%s" % (nume, prenum

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

    except Exception as e:
        error_message = str(e).splitlines()[0]
        return render_template( template_name_or_list: 'eroare.html', error_message=error_message, back_url='/clients')
```

```python
@app.route( rule: '/getClient', methods=['POST'])
def get_cl():
    emp = request.form['id_client']
    cur = con.cursor()
    cur.execute('select * from client where id_client=' + emp)

    emps = cur.fetchone()
    id_client= emps[0]
    nume = emps[1]
    prenume = emps[2]
    numar_telefon = emps[3]
    email = emps[4]
    cur.close()

    return render_template( template_name_or_list: 'editClient.html', nume=nume, prenume=prenume, numar_telefon=numar_telefon
```

În cazul în care apare o eroare în aplicație: