UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

Calculatoare și tehnologia informației

**Administrare de restaurant cu livrari**

Tema de casa la disciplina Baze de date

Coordonator, Avram Sorin

Student, Prelipcean Dragos-Iulian

Grupa, 1307A

Introducere

Aplicația are ca scop gestionarea modului în care se poate comanda mâncare de la un restaurant urmând ca aceasta să fie livrata.

Baza de date cuprinde 7 tabele:

- Restaurant

- Angajat

- Detalii\_angajat

- Client

- Comanda

- Comanda\_detalii

- Produs

**Descrierea functionala a aplicației:**

Principalele operațiuni care se întalnesc într-un restaurant cu livrări sunt:

➢ Preluarea comenzilor

➢ Gestionarea comenzilor in vederea livrării

Pentru preluarea comenzilor aplicația dispune de o interfata in care se parcurg o serie de pași pana la finalizarea acesteia de către client. Acești pași sunt după cum urmează:

- se alege orașul restaurantului de la care se dorește a fi data comanda

- se adauga produsele dorite

- la ultimul pas se completeaza datele clientului într-un formular și se finalizeaza comanda

Deasemenea clientul poate vizualiza produsele adaugate în comanda și poate face o serie de modificari dacă dorește.

Pentru partea de gestionare a comenzilor de către angajații restaurantului, interfata pune la dispoziție o serie de comenzi si functionalitati cum ar fi: actualizarea statusului unei comenzi(gata de livrare), adaugarea unui curier responsabil de livrare sau marcarea acesteia ca fiind livrata. Se pot adauga noi produse dar exista și posibilitatea de modificare sau stergere a celor deja existente. Deasemenea, se pot adauga, modifica sau sterge informațiile despre angajați și restaurante.

Descrierea detaliată a entitaților si a relațiilor dintre tabele

In proiectarea acestei baze de date s-au definit tipurile de relatii 1:1 si 1:n.

Între tabelele Restaurant si Angajat se stabilește o relație de tip one-to-

many deoarece o persoană poate fi angajată la un singur restaurant din cadru

acestui lanț, iar un restaurant poate avea mai mulți angajați. Legatura dintre cele

două tablele se realizează prin câmpul id\_restaurant.

Între tabelele Angajat si Detalii\_angajat se stabilește o relație de tip one-to-

one, fiecare angajat având propriile informații, iar aceste informații, evident sunt

unice pentru fiecare anagajat. Legatura dintre tabele se stabilește prin câmpul

id\_detalii\_angajat.

Între Angajat si Comanda există o legatură de tip one-to-many. Unui

angajat i se vor atribui una sau mai multe comenzi pentru a fi livrate sau preparate,

insă o comandă nu poate fi asignată mai multor angajați care livrează. Legătura

dintre aceste tabele se realizează prin câmpul id\_angajat.

Între Client si Comanda se stabilește o legătură de tip one-to-many. Un

client poate face mai multe comenzi, insă o comandă va fi atribuită unui singur

client. Legătura dintre cele două tablele este realizată prin câmpul id\_client.

Între Comanda și Comanda\_detalii se stabilește o legătură de tip one-to-

many. Tabela Detalii\_comanda a fost creată pentru a se evita o relatie de tip many-

to-many, astfel această tabelă mai este implicată intr-o relație de tip one-to-many și

anume intre Produs si Comanda\_detalii. Avându-se în vedere aceste aspecte

tabela Comanda\_detalii conține cantitatea pentru un produs din tabela Produs si

totodata id-ul comenzii care conține aceste produse. Legătura dintre Comanda si

Comanda\_detalii se realizează prin câmpul id\_client, iar intre Produs și

Comanda\_detalii prin câmpul id\_produs.

Între Comanda si Plata se stabilește o legătură de tip one-to-one deoarece

fiecărei comenzi îi corespunde valoarea de achitat. Această sumă se calculează

inmulțind cantitatea cu prețul produsului si apoi insumând toate articolele care

aparțin comenzii respective. Legătura dintre aceste două tablele se realizează cu

ajutorul câmpului id\_plata.

**Descrierea constrângerilor**

În vederea validării datelor de intrare am aplicat următoarele constrângeri:

În tabela **client** avem:

- nume: să conțină doar litere.

- telefon: prima cifră să fie 0 și următoarea 2 sau 7 (2 pentru telefoane fixe si 7

pentru mobile), fie aibă o lungime de 10 cifre si să fie unic.

- email: să fie de forma x@y.z și să fie unic.

Tehnologi folosite:

Pe partea de back-end s-a folosit Flask, care este un API (Application Programming Interface) Python ce permite construirea de aplicații web.

Pentru front-end s-a utilizat HTML, CSS și framework-ul Materialize care permite crearea unei interfete bazandu-se pe Material Design by Google.

Pentru baza de date s-a folosit Mysql.

Conectarea la baza de date:

Pentru conectare se folosește Mysql Connector, care permite unui program Python sa acceseze o baza de date Mysql.

Modul în care se poate realizeaza conexiunea este prezentat mai jos:

mydb = mysql.connector.connect(

host="localhost",

user="root",

passwd=”12345”,

database="restaurant\_db"

)

Prezentarea principalelor functionalitati ale aplicației

1. **Functia Select**

Este utilizata pentru obtinerea datelor din baza de date.

Un exemplu de utilizare pentru extragerea datelor dintr-o tabela.

@bp.route('/meniu', methods=['POST', 'GET'])

def show():

mycursor = mydb.cursor()

sql = "SELECT \* FROM produs where stoc > 0

and restaurant\_id\_restaurant=

(select restaurant\_id\_restaurant from comanda

order by id\_comanda desc limit 1)"

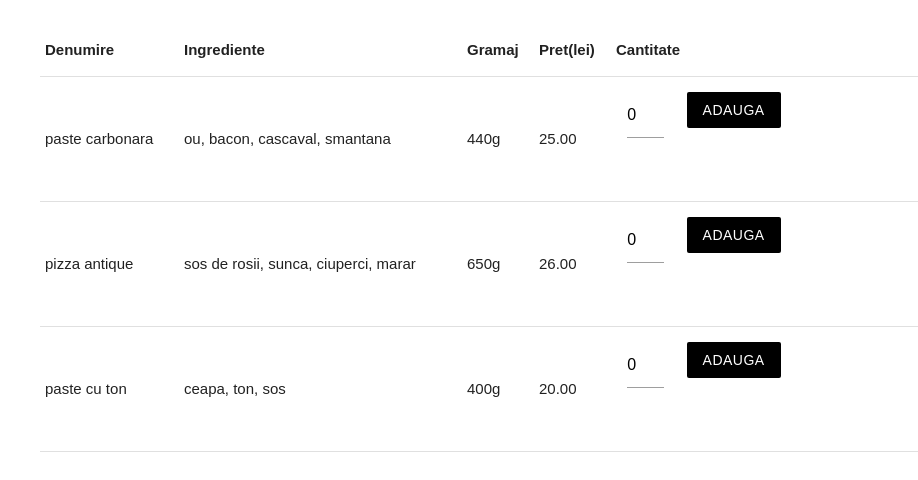
val = (id\_produs, )

mycursor.execute(sql)

produse = mycursor.fetchall()

return render\_template('meniu.html', produse=produse, detalii\_resturant=detalii\_resturant, errors=errors)

Rezultatul rularii este:



**2. Functia Update și Delete**

Datele pot fi modificate sau sterse iar informațiile din campuri sunt citite din tabele fără a fi nevoie sa introducem și informațiile nemodificate.

Exemplu de cod pentru update:

status\_incheiat = 0;

sql = "UPDATE comanda SET status=%s where id\_comanda=%s;"

val = (status\_incheiat, id\_comanda, )

mycursor.execute(sql, val)

mydb.commit()

Exemplu de cod pentru Delete:

sql = "DELETE from comanda where id\_comanda=%s;"

val = (id\_comanda, )

mycursor.execute(sql, val)

mydb.commit()

Pagina de Update sau Delete:

