UNIVERSITATEA TEHNICĂ “GH ASACHI” IAŞI

FACULTATEA AUTOMATICĂ ŞI CALCULATOARE

DOMENIUL: CALCULATOARE ŞI TEHNOLOGIA INFORMAŢIEI

DISCIPLINA: BAZE DE DATE

Gestiunea unei pizzerii cu delivery (exemplu Mamma Mia)

Profesor coordonator:

Cristian Buțincu

Student:

Brezeanu Gabriela – 1306B

Capitolul 1: Introducere

Aplicatia este creata cu scopul de a gestiona baza de date a unei pizzerii care livreaza la domiciliu.

Nu se pune problema unui lant de pizzerii, afacerea fiind abia la inceput. Din acest motiv, nici meniul nu este unul foarte variat, iar personalizarea sortimentelor nu este acceptata.

Livrarea se va face pe toata raza orasului Iasi, in toate cartierele.

Fiecare client are posibilitatea de a obtine un card de fidelitate pe care sa acumuleze puncte pe care le va putea folosi in viitor.

Ne propunem:

- sa avem o evidenta a comenzilor cu toate detaliile aferente
- sa cunoastem valoarea totala a fiecarei comenzi (clasa bon: se extrag informatii din mai multe tabele)

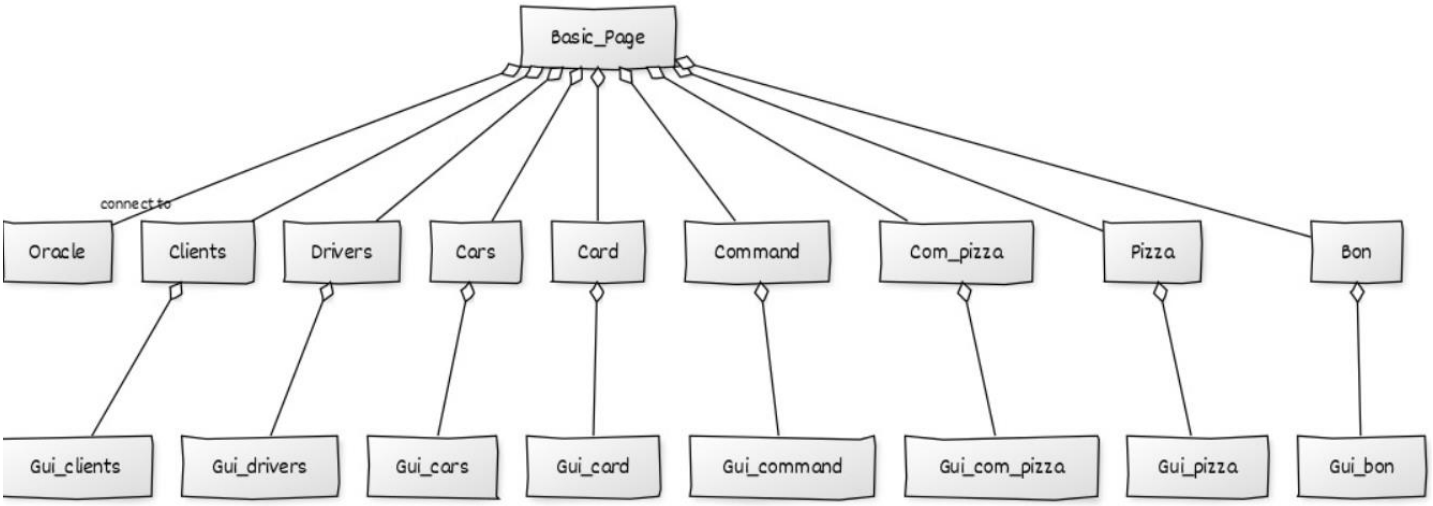
Informatiile de care vom avea nevoie sunt legate de:

- Client
- Card_fidelitate
- Masina
- Sofer
- Comanda
- Corespondenta dintre comanda si tipuri pizza, in sensul de numarul de bucati comandate din fiecare sortiment ales
- Tipuri_pizza

Capitolul 2: Tehnologii folosite pentru front-end si back-end

Baza de date folosita in aceasta aplicatie este Oracle.

Pentru interfata grafica s-a folosit libraria PyQt5 din Python.



Pentru introducerea datelor din interfata s-au folosit widget-uri precum Line Edit, Combo Box din libraria PyQt5.

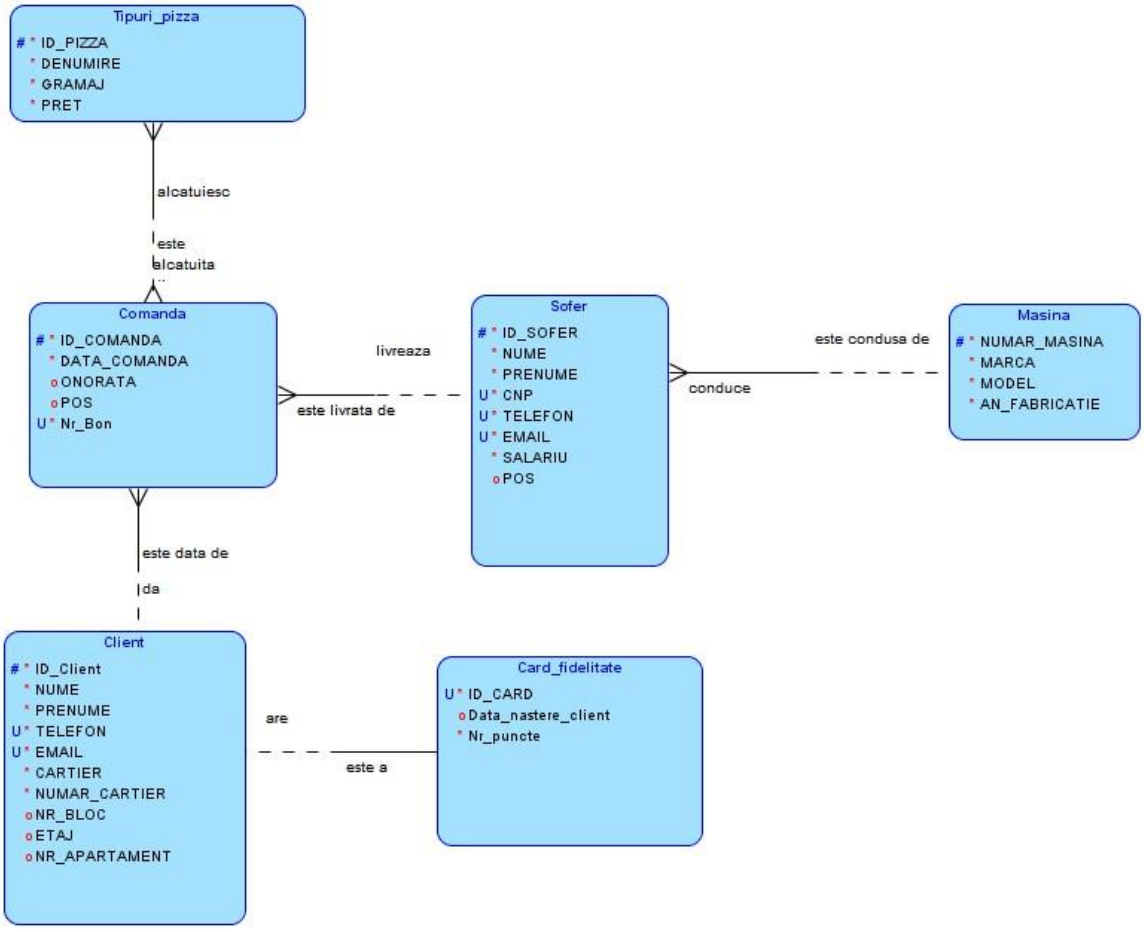
Pentru vizualizarea inregistrarior s-a folosit QtableWidget pentru fiecare tabel din baza de date.

Capitolul 3: Structura si inter-relationarea tabelelor

Descrierea entitatilor:

1. **Client:** pentru a putea identifica persoana care a plasat o comanda, este nevoie de datele sale de contact, mai exact nume, prenume, telefon, email (atribute obligatorii).
   Pentru livrare este necesar sa se cunoasca adresa fiecarui client. Indiferent daca locuieste la casa sau la bloc, va exista numele unui cartier si numarul acestuia(atribute obligatorii), insa nr_bloc, etaj si nr_apartament sunt specifice doar pentru cei care stau la bloc (atribute optionale).
2. **Card_fidelitate**: fiecare client va putea detine un card unde sa acumuleze puncte in urma comenzilor date.
3. **Masina**: aceasta entitate va salva numarul masinii, dar si marca, modelul si anul de fabricatie (atribute obligatorii).
4. **Sofer**: fiecare sofer va avea cateva atribute prin care poate fi identificat (id, nume, prenume, cnp, telefon, email). Se cunoaste si salariul fiecaruia, dar si daca are POS sau nu (atribut optional).
5. **Comanda:** fiecare comanda va avea un id unic prin care va fi identificata. Se va salva data la care a fost primita comanda, insa nu poate fi mai veche decat momentul prezent (comenzi programate in viitor se pot da). Clientul are si optiunea de a plati cu POS, iar pentru acest lucru, comanda va fi livrata de un sofer care va avea implit POS. Bineinteles ca pentru fiecare comanda, se va emite un bon fiscal unic.
6. **Tipuri_pizza:** aceasta entitate reprezinta meniul, in sensul ca aici se vor gasi sortimentele de pizza din care va putea alege clientul pentru care va sti si gramajul+pretul.

Modelul logic



Dupa cum se poate observa, sunt folosite 6 entitati cu diverse relatii intre ele: 1:1, 1:N, M:N.

Relatie 1:1

Aceasta relatie apare intre entitatea Client si Card_fidelitate pentru ca un client poate avea doar un card, iar cel din urma apartine unui singur client.

Relatie 1:N

-intre entitatea Client si Comanda: un client poate da mai multe comenzi, insa o comanda anume apartine unui singur client, nu poate aparea o comanda cu acelasi id la 2 clienti diferiti.

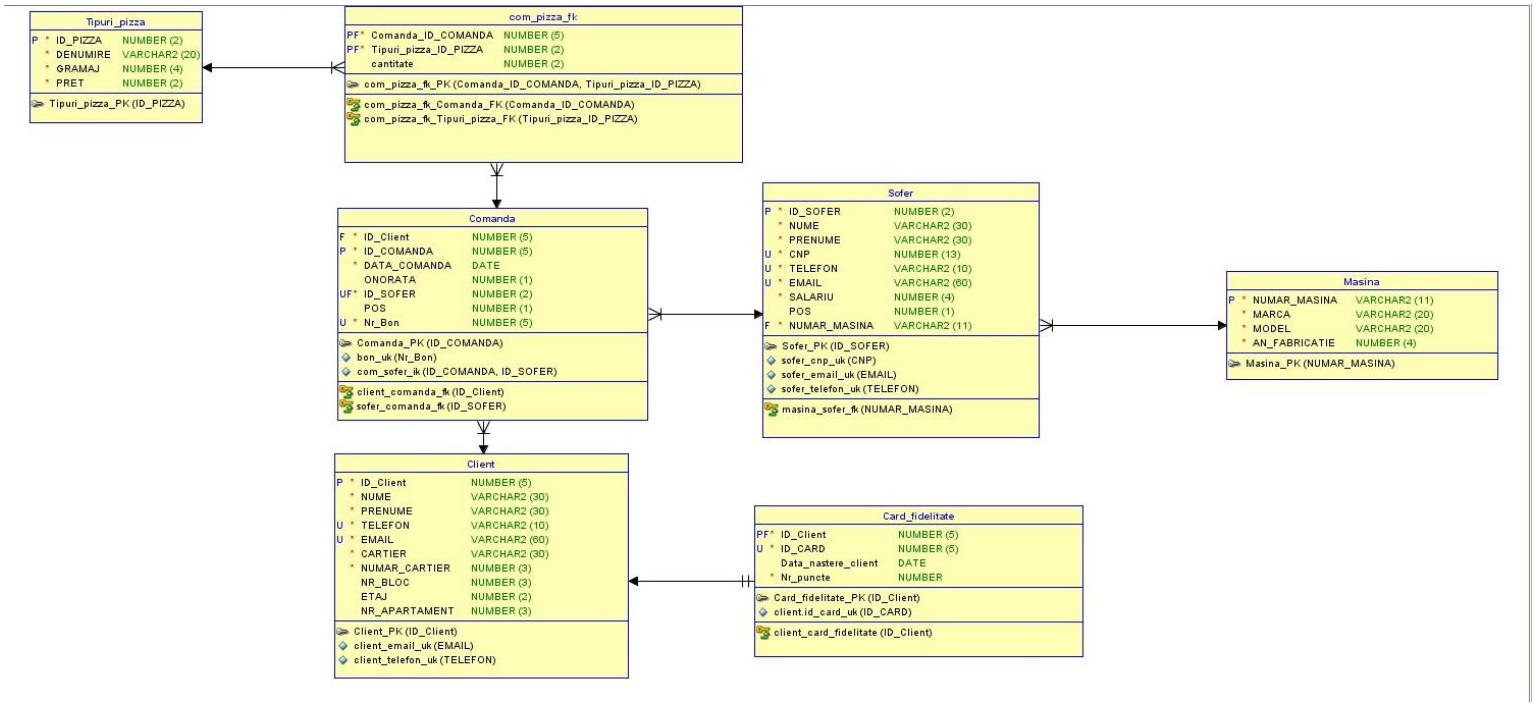
-intre entitatea Sofer si Comanda: o comanda este livrata de un singur sofer, insa cel din urma poate pleca cu mai multe comenzi simultan.

-intre entiatea Masina si Sofer: un sofer conduce doar o masina, insa aceeasi masina poate fi condusa de mai multi soferi.

Relatie M:N

-intre entitatea Comanda si Tipuri_pizza: o comanda poate fi alcatuita din mai multe tipuri de pizza, dar si acelasi sortiment poate aparea in mai multe comenzi.

Modelul Relational



In figura se pot observa 7 tabele si nu 6 (numarul entitatilor din modelul logic). Acest lucru se datoreaza relatiei M:N care prin normalizare se imparte in 2 relatii 1:N, aparand astfel o noua tabela intermediara (com_pizza_fk).

In tabela com_pizza_fk se va crea legatura dintre comanda si tipuri_pizza, in sensul ca vom dori sa stim exact ce sortimente alcatuiesc o comanda, dar si numarul de bucati din fiecare tip de pizza ales(cantitatea).

Comanda_id_comanda si tipuri_pizza_id_pizza vor avea aplicate constrangerile foreign key pentru a apartine valorilor din cele 2 tabele, dar vor constitui impreuna si cheia primara a tabelei (putem avea mai multe inregistrari cu aceeasi valoare in id_comanda atunci cand o comanda contine mai multe sortimente de pizza, deci valorea din tipuri_pizza_id_pizza va fi diferita).

Capitolul 4: Constrangerile utilizate in baza de date

Primary Key - Acest tip de constrangere identifica in mod unic o inregistrare din tabel si nu poate avea valoarea null ori o valoare care sa se repete.

Fiecare atribut de tipul id_x din fiecare tabela are aceasta constrangere aplicata, dar si perechea (comanda_id_comanda, tipuri_pizza_id_pizza) din tabelul com_pizza_fk.

Pentru tabelul Masina, atributul care va fi primary key este NUMAR_MASINA, exact ca si in viata reala unde fiecare autovehicul detine un numar de inmatriculare unic.

Foreign Key - Acest tip de constrangere este folosit pentru a lega coloanele din doua tabele intre ele.

In tabelul Client: id_client este foreign key catre id_client din tabelul Client, astfel nu se va putea introduce un id gresit, adica un client care sa nu existe efectiv inregistrat inainte

In tabelul Card_fidelitate: id_client este foreign key catre id_client din tabelul Client pentru a sti exact carui client apartine acel card

In tabelul Com_pizza_fk: comanda_id_comanda este foreign key catre id_comanda din tabelul Comanda, iar tipuri_pizza_id_pizza indica spre id_pizza din tabelul Tipuri_pizza.

In tabelul Sofer: numar_masina este foreign key catre NUMAR_MASINA din tabelul Masina,in acest mod masinile pe care le vor conduce fac parte strict din cele care apartine afacerii.

Unique Key - Aceasta constrangere se asigura ca intr-o coloana toate valorile sunt diferite.

In tabelul Client: telefon, email.

In tabelul Card_fidelitate: id_card

In tabelul Comanda: nr_bon

In tabelul Sofer: cnp, telefon, email

Not null - Aceasta constrangere obliga un atribut sa nu primeasca valori nule.

In tabelul Client doar Nr_bloc, Etaj si Nr_apartament nu au aceasta constrangere in cazul in care clientul locuieste la casa.

In tabelul Comanda doar POS nu este obligatoriu si poate fi Null pentru ca atunci plata va fi cash, valabil si in tabelul Sofer unde nu fiecare are un POS alocat.

Check - Acest tip de constrangere este folosit pentru a limita valorile din campuri.

Pentru fiecare atribut de tipul nume, prenume exista constrangeri regex care sa permita introducerea doar a caracterelor(literelor), dar si o verificare a lungimii, sa existe macar 2 caractere in fiecare camp.

Pentru telefon s-a impus tot un format anume, si anume 07..., cu o lungime totala de 10 cifre.

Se verifica si formatul adresei de email tot utilizand regex.

De asemenea s-au impus niste intervale de valori si pentru campurile ce tin de adresa, mai exact nr_cartier, etaj, nr_apartament.

Pentru a verifica varsta soferilor ca au impliniti minim 18 ani, se face o verificare pe baza cnp-ului (implicit si a formatului acestuia), extragandu-se de acolo anul.

Pretul, gramajul unei pizza trebuie sa fie valori pozitive, la fel ca si salariul unui sofer.

Numarul de puncte de pe un card de fidelitate poate avea valori nule sau pozitive, dar nu negative.

Pentru Numar_Masina se verifica cu un regex formatul clasic al unui numar clasic de inmatriculare, si anume 'AA 11 SSS' (2_litere 2_cifre 3_litere).

Tot pentru Masina pe langa verificarile de caractere din Marca si Model, An_fabricatie trebuie sa se incadreze intre 2010 si 2021.

Alte explicatii:

Pentru **primary key** din tabele s-a folosit un **mecanism de tip autoincrement** astfel incat id-urile sa nu fie introduse manual. Autoincrement-ul este implementat atat prin utilizarea lui identity, dar si prin metoda alcatuita din secventa si trigger.

In tabela Comanda exista doua mecanisme de autoincrement, atat pentru id, cat si pentru nr_bon, pentru amandoua folosindu-se varianta cu secventa si trigger. Secventa de la id_comanda va fi utilizata la inserarea in tabela com_pizza_fk pentru a putea face legatura intre comanda curenta si tipul de pizza continut.

S-a mai folosit un trigger pentru verificarea datei unei comenzi astfel incat sa nu fie inregistrate comenzi mai vechi, ci doar din ziua curenta si din viitor pentru cele programate.

Capitolul 5: Descrierea modalitatii de conectare la baza de date

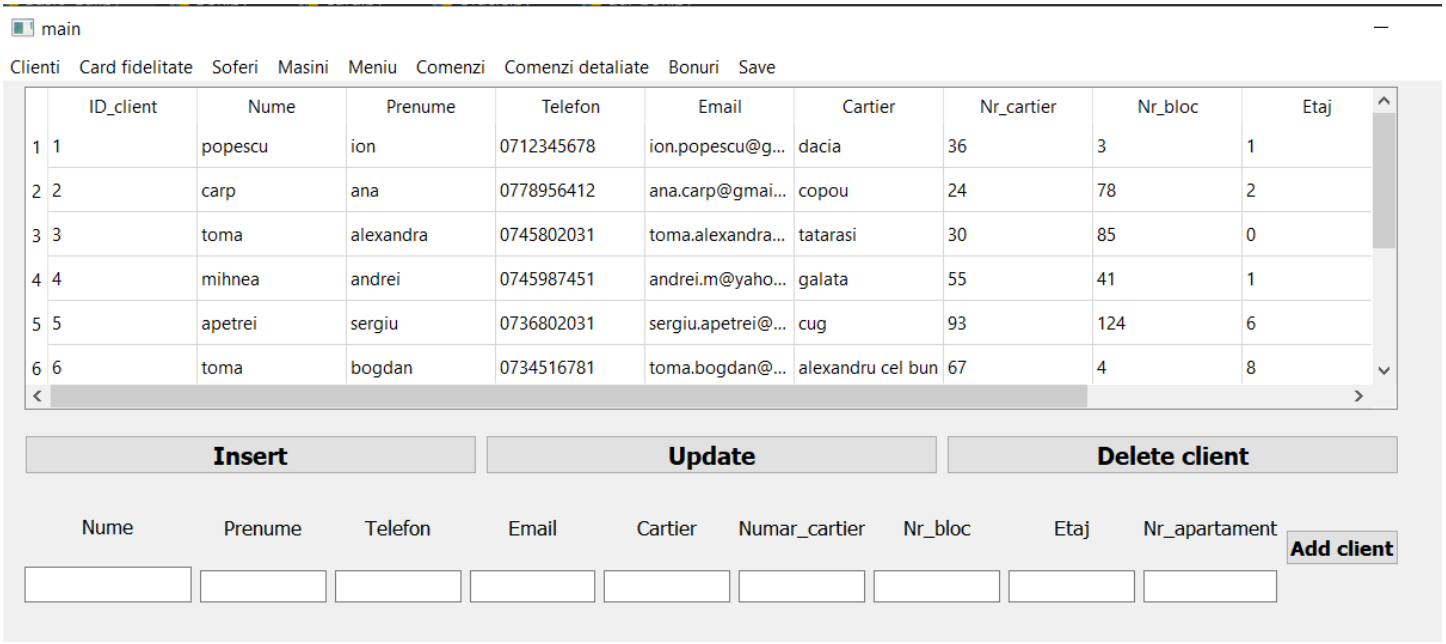
Aplicatia se conecteaza la baza de date prin intermediul librariei cx_Oracle.

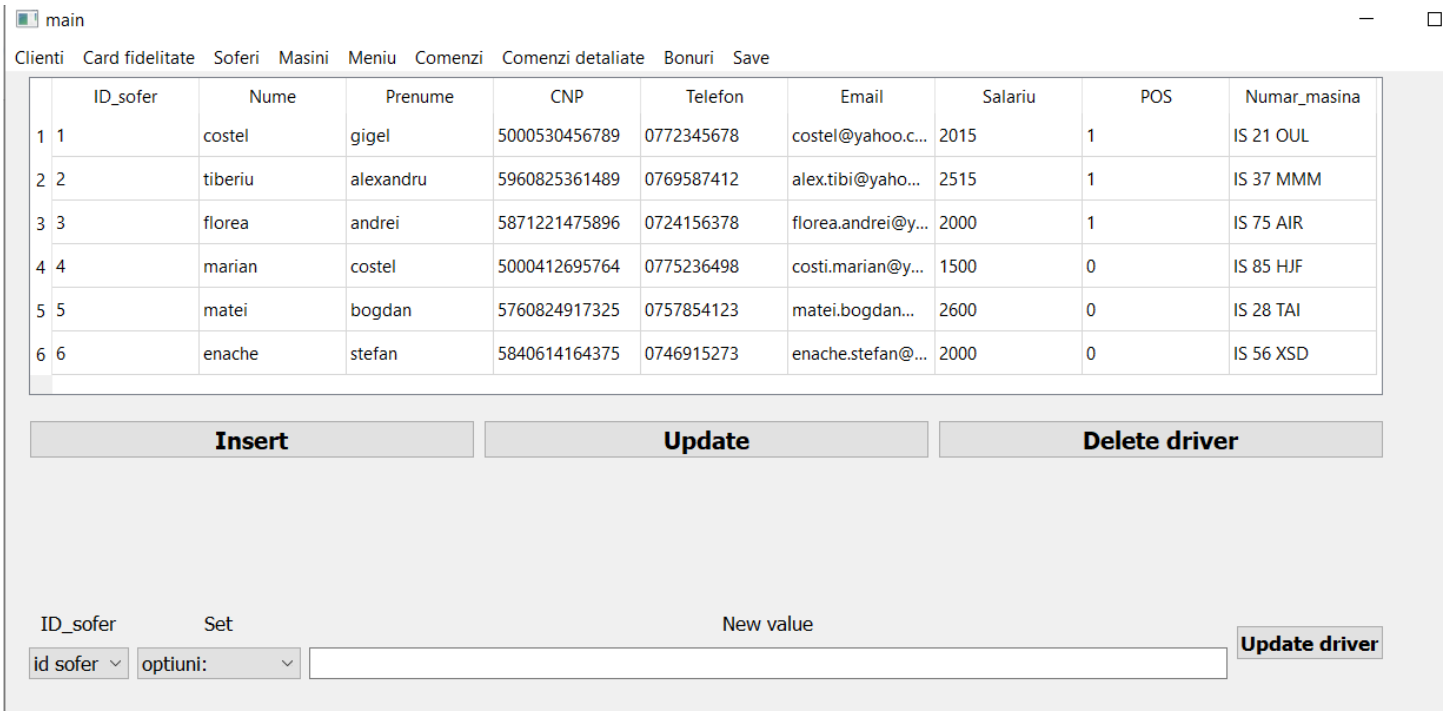
```
def connect_to_db(self):
    try:
        print("Se conecteaza la Oracle....")
        con = cx_Oracle.connect(self.user, self.password, self.hostname, encoding="UTF-8")
        print("S-a realizat conexiunea la baza de date Oracle cu succes")
        print(con.version)
        self.cursor = con.cursor()
    except Exception as err:
        print(err)
```

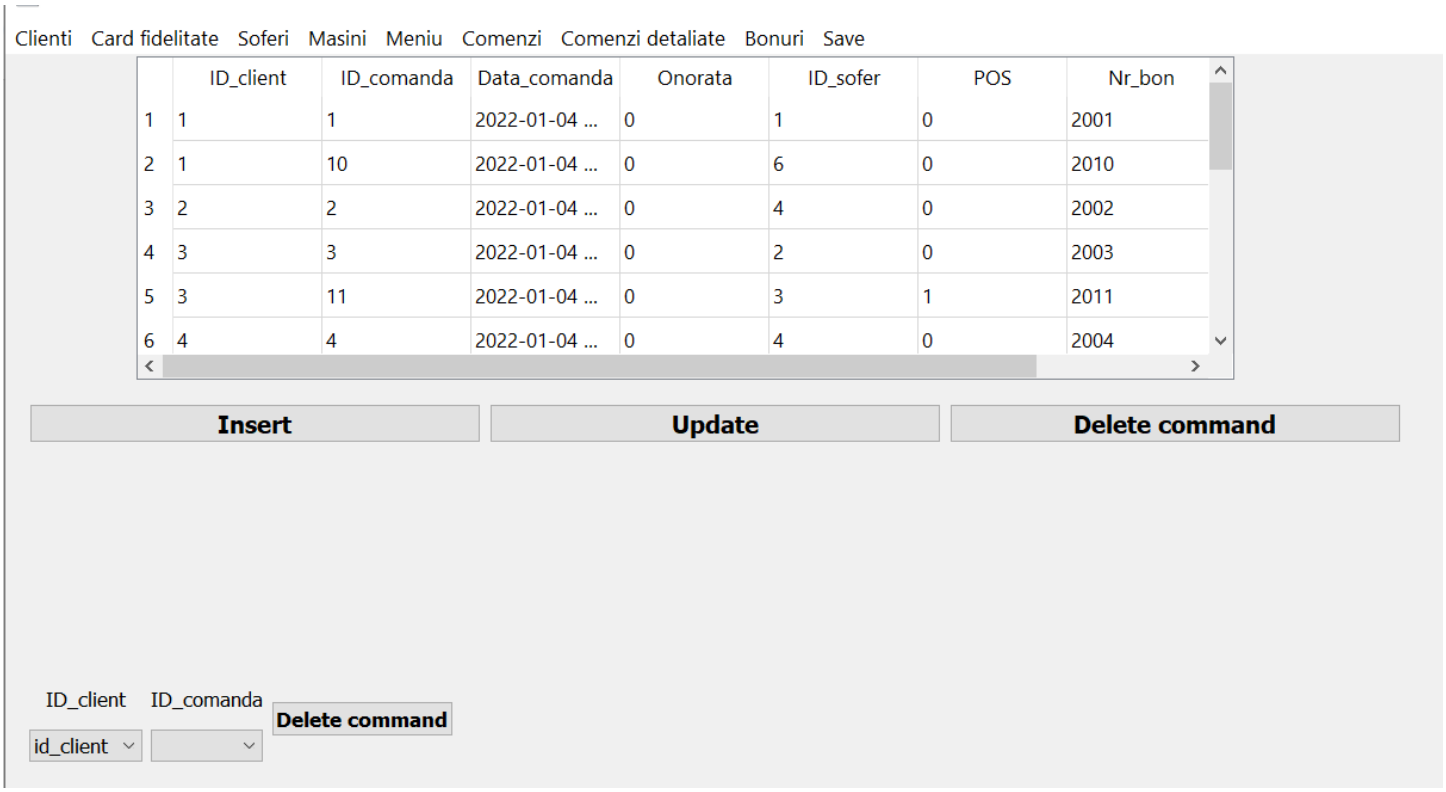
```
def execute_query(self,query):
    answer = self.cursor.execute(query)
    return answer
```

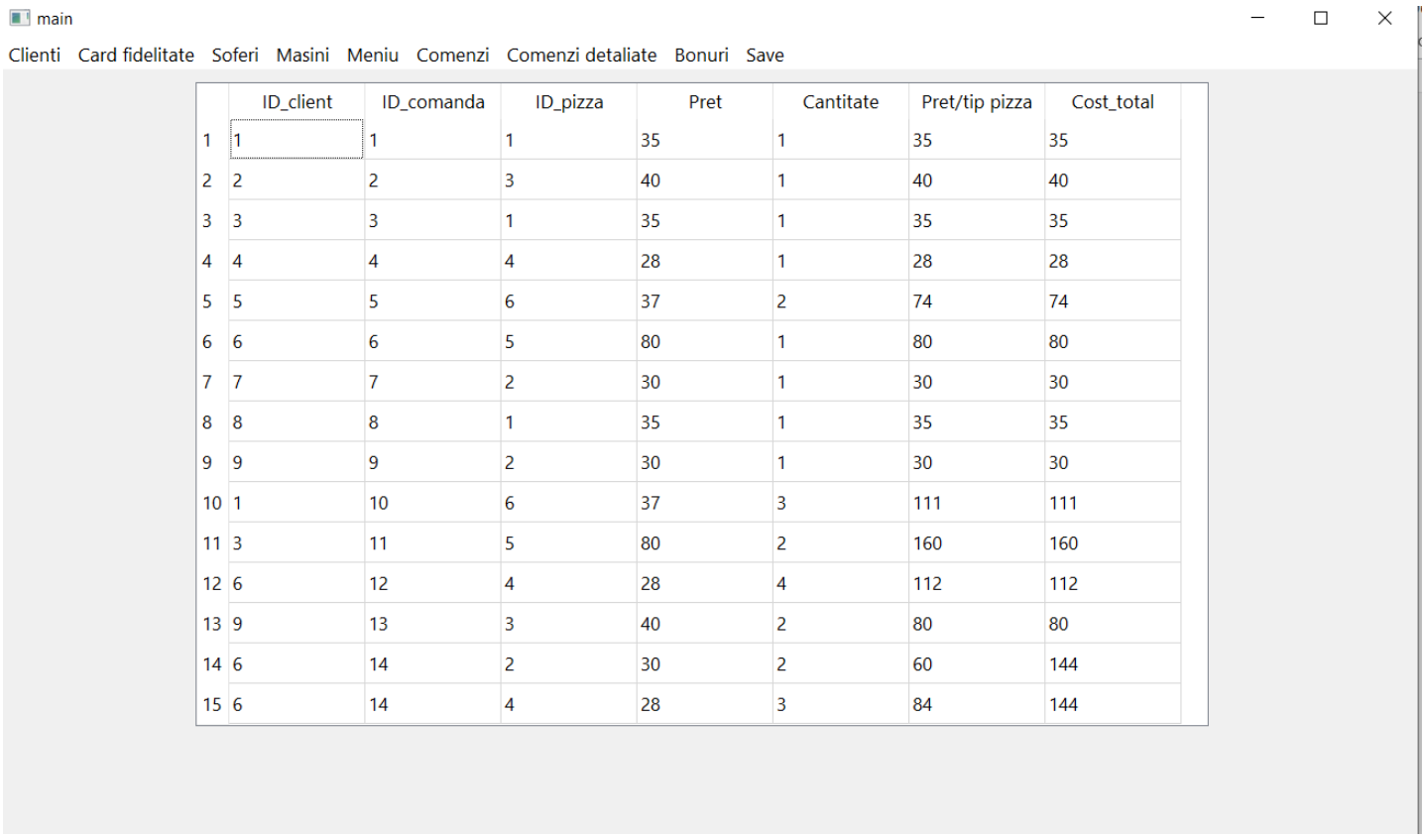
Functia execute_query executa o interogare si returneaza rezultatul. Aceste functii se gasesc in clasa Oracle.

Capitolul 6: Capturi de ecran din interfata grafica






Clienti Card fidelitate Soferi Masini Meniu Comenzi Comenzi detaliate Bonuri Save
ID_client ID_comanda Data_comanda Onorata ID_sofer POS Nr_bon
1 1 1 2022-01-04 ... 0 1 0 2001
2 1 10 2022-01-04 ... 0 6 0 2010
3 2 2 2022-01-04 ... 0 4 0 2002
4 3 3 2022-01-04 ... 0 2 0 2003
5 3 11 2022-01-04 ... 0 3 1 2011
6 4 4 2022-01-04 ... 0 4 0 2004
Insert
Update
Delete command
ID_client
ID_comanda
id_client
Delete command


main
Clienti Card fidelitate Soferi Masini Meniu Comenzi Comenzi detaliate Bonuri Save
ID_client ID_comanda ID_pizza Pret Cantitate Pret/tip pizza Cost_total
1 1 1 1 35 1 35 35
2 2 2 3 40 1 40 40
3 3 3 1 35 1 35 35
4 4 4 4 28 1 28 28
5 5 5 6 37 2 74 74
6 6 6 5 80 1 80 80
7 7 7 2 30 1 30 30
8 8 8 1 35 1 35 35
9 9 9 2 30 1 30 30
10 1 10 6 37 3 111 111
11 3 11 5 80 2 160 160
12 6 12 4 28 4 112 112
13 9 13 3 40 2 80 80
14 6 14 2 30 2 60 144
15 6 14 4 28 3 84 144

Capitolul 7: Secvente de cod SQL

- Creare tabele cu constrangeri

```
CREATE TABLE masina (
    numar_masina  VARCHAR2(11) NOT NULL,
    marca         VARCHAR2(20) NOT NULL,
    model         VARCHAR2(20) NOT NULL,
    an_fabricatie NUMBER(4) NOT NULL
)
LOGGING;

ALTER TABLE masina
    ADD CONSTRAINT masina_numar_mas_ck CHECK ( REGEXP_LIKE ( numar_masina,
                                                             '^[A-Z]{2}\s[0-9]{2}\s[A-Z]{3}$' ) );

ALTER TABLE masina
    ADD CONSTRAINT masina_marca_ck CHECK ( REGEXP_LIKE ( marca,
                                                         '^[a-zA-Z]+$' )
                                           AND length(marca) > 1 );

ALTER TABLE masina
    ADD CONSTRAINT masina_model_ck CHECK ( REGEXP_LIKE ( model,
                                                         '^[a-zA-Z]+\s+[0-9]+$' )
                                           AND length(model) > 1 );

ALTER TABLE masina
    ADD CONSTRAINT masina_an_fab_ck CHECK ( REGEXP_LIKE ( an_fabricatie,
                                                          '^[0-9]{4}$' )
                                            OR an_fabricatie BETWEEN 2010 AND 2021 );

ALTER TABLE masina ADD CONSTRAINT masina_pk PRIMARY KEY ( numar_masina );
```

- Inserare in tabel

```
insert into masina values('IS 28 TAI', 'VW', 'UP 1', 2014);

insert into sofer (nume, prenume, cnp, telefon, email, salariu, pos,numar_masina) values ('costel','gigel',5000530456789,'0772345678','costel@yahoo.com',2000,1,'IS 21 OUL');
```

```
    def get_numar_masina(self):
        self.numar_masina = self.numar_masina_line.text()

    def get_an_masina(self):
        self.an_masina=self.an_masina_line.text()

    def insert(self):
        self.numar_masina_line.returnPressed.connect(lambda: self.get_numar_masina())
        self.an_masina_line.returnPressed.connect(lambda: self.get_an_masina())

    def add_new_car(self):
        try:
            query = "insert into masina (numar_masina, marca, model, an_fabricatie) " \
                    "values ('{}','{}','{}',{})"\
                .format(self.numar_masina, self.marca,self.model_masina,int(self.an_masina))
            print(query)
        except Exception as err:
            print(err)

        ok = 1
        try:
            self.database.execute_query(query)
        except Exception as err:
            print(err)
            ok = 0
        if ok:
            self.no_rows += 1
            self.cars.set_no_rows(self.no_rows)
        self.populate_cars()
```