UNIVERSITATEA TEHNICĂ "GH ASACHI" IAŞI
FACULTATEA AUTOMATICĂ ŞI CALCULATOARE
SPECIALIZAREA CALCULATOARE ŞI TEHNOLOGIA INFORMAŢIEI

DISCIPLINA: BAZE DE DATE – Tema de casa

Aplicatie pentru gestiunea stocului si plasarea comenzilor in comercializarea de incaltaminte

Studenti:

Chirca Radu-Iulian 1310B

Ionescu Razvan 1310B

Coordonator:

Avram Sorin

1. Descrierea proiectului

Prin aplicatia noastra, ne propunem oferirea unei posibilitati de a urmari si modifica stocul de bunuri in cadrul unui site web ce ofera servicii de vanzare si livrare.

Cum bunurile comercializate pot include un numar mare de diferite tipuri de incaltaminte, am avut in vedere in principal monitorizarea aspectelor logistice. Aplicatia este impartita in trei functionalitati, dintre care una este un showcase destinat clientului.

Administratorul poate aduce modificari stocului de papuci prin intermediul primului tab, unde exista posibilitatea introducerii marcii (ID-ul) produsului, a numelui, marimii si cantitatii disponibile. De asemenea, se pot realiza stergeri, update-uri si operatii de get prin care se poate verifica, prin specificarea ID-ului, disponibilitatea unui produs. Monitorizarea se realizeaza printr-un tabel in care sunt afisate toate intrarile.

In cea de-a doua partea a aplicatiei, este oferita o interfata prin care un posibil client poate plasa comenzi, specificandu-se in functie de stocul disponibil numele, marimea si cantitatea dorita dintr-un anumit produs. Aici pot fi vizualizate toate comenzile plasate de-a lungul timpului, indiferent daca acestea au putut fi validate sau nu pentru livrare.

In final, o a treia functionalitate este reprezentata de posibilitatea verificarii comenzilor si ‘livrarea’ acestora de catre administrator. Toate comenzile livrate cu success pot fi vizualizate aici, iar stocul se modifica in mod specific cantitatilor si produselor livrate.

2. Tehnologiile folosite

- *Intregul proiect a fost realizat in Python 3.10, utilizand IDE-ul PyCharm.*

https://www.python.org/downloads/release/python-3100/

- *Pentru partea de back-end, conectivitatea cu baza de date a fost realizata utilizand biblioteca cx_Oracle.*

```
cursor = con.cursor()
cursor.execute("select * from snkrs")
```

Prin intermediul acestui cursor am efectuat diferitele operatii in SQLPlus in cadrul programului.

Cx_Oracle este un modul de extensie Python care permite accesul la Oracle Database.

https://cx-oracle.readthedocs.io/en/latest/

De asemenea, este nevoie de bibliotecile client Oracle. In cazul nostru am utilizat Oracle Instant Client.

```
python - m pip install cx_Oracle – upgrade
```

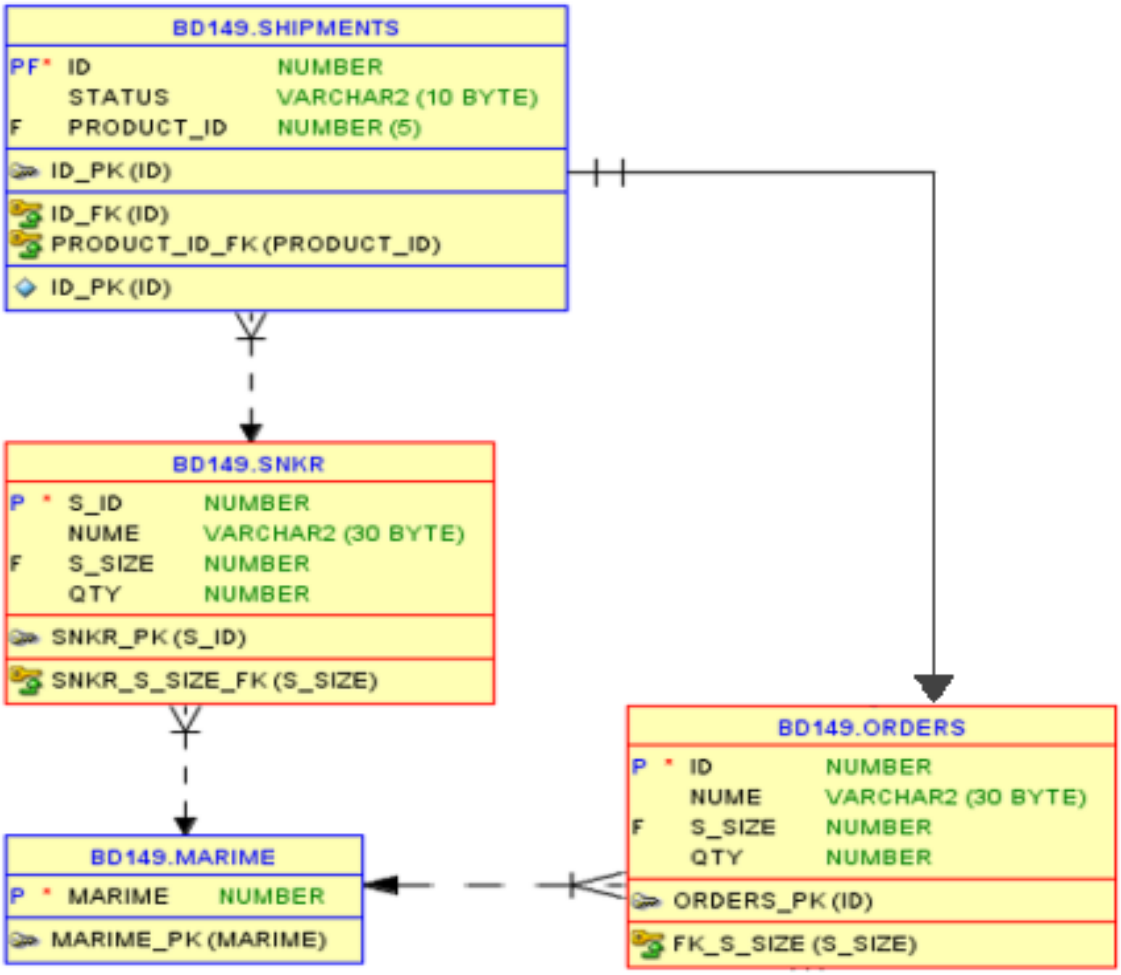
- *Interfata a fost realizata folosind customtkinter, o extensie a cunoscutei biblioteci tkinter pentru Python, care fata de aceasta adauga elemente UI extra si ofera un aspect mai placut.*

https://pypi.org/project/customtkinter/0.3/

Conectare la baza de date:

```
connection = cx_Oracle.connect("bd149/bd149@bd-dc.cs.tuiasi.ro:1539/orcl")
```

3. Data Modeler



4. Descrierea tabelelor

Tabelul SNKR

Pentru evidenta stocului, tabelul SNKR contine 4 campuri: S_ID, NUME, S_SIZE si QTY. Evident, cel dintai este primary key.

La crearea tabelelor acesta este populat cu diferite tipuri de papuci, iar pe parcurs se pot aduce modificari cum ar fi stergerea intrarilor, adaugarea, etc.

Pot exista mai multe intrari cu acelasi nume, insa acestea trebuie sa aiba neaparat marimi diferite, astfel incat vom avea un ID specific pentru fiecare pereche nume-marime.

Marimea prezinta un constraint de tip foreign key care face referinta la campul marime din tabelul cu acelasi nume; in acest fel putem stii strict ce marime putem introduce atat pentru stoc cat si pentru comanda, deoarece marimile sunt prestabilite la crearea tabelelor.

Tabelul Marimi

Scopul sau este descris mai sus, astfel evitam introducerea foreign key-urilor direct ca si valori, ele fiind selectabile prin intermediul unui combobox. Este populat la creare si ramane neschimbat.

Tabelul Orders

Campurile sunt similare cu cele din tabelul snkr, reflectand produsul comandat. ID-ul este generat automat prin intermediul unui contor si este primary key.

Asemanator tabelului snkr, s_size este preluat ca foreign key din tabela cu marimi.

La creare, tabelul orders este gol, urmand a fi create intrari odata cu plasarea de comenzi din interfata.

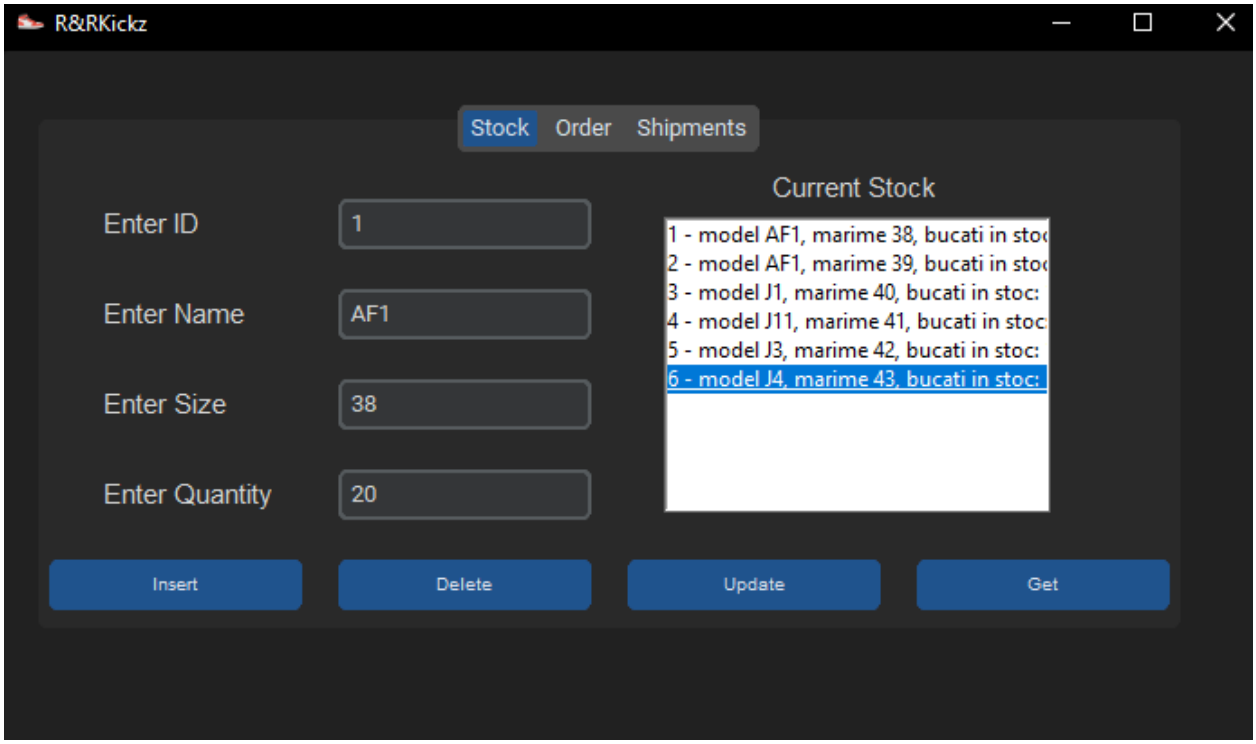
Tabelul Shipments

Prezinta un ID, status de livrare si un product_id care reflecta prin foreign key ID-ul produsului livrat din tabelul snkr. ID-ul livrarii corespunde unui ID din tabela orders, indicand mecanismul prin care se urmaresc comenzile.

O comanda nu poate fi livrata daca cantitatea din tabelul snkr nu corespunde cerintelor comenzii, adica nu avem destul stoc. (CHECK).

In table se introduc toate comenzile care au fost efectuate cu success si au trecut in statusul ‘shipped’.

5. Functionalitatea aplicatiei



Se pot observa cele 3 tab-uri diferite si butoanele pentru functiile de modificare a tabelei snkr, prezentata in lista din stanga.

Functia get permite furnizarea unui ID pentru umplerea casutelor cu datele intrarii cu ID-ul respective.

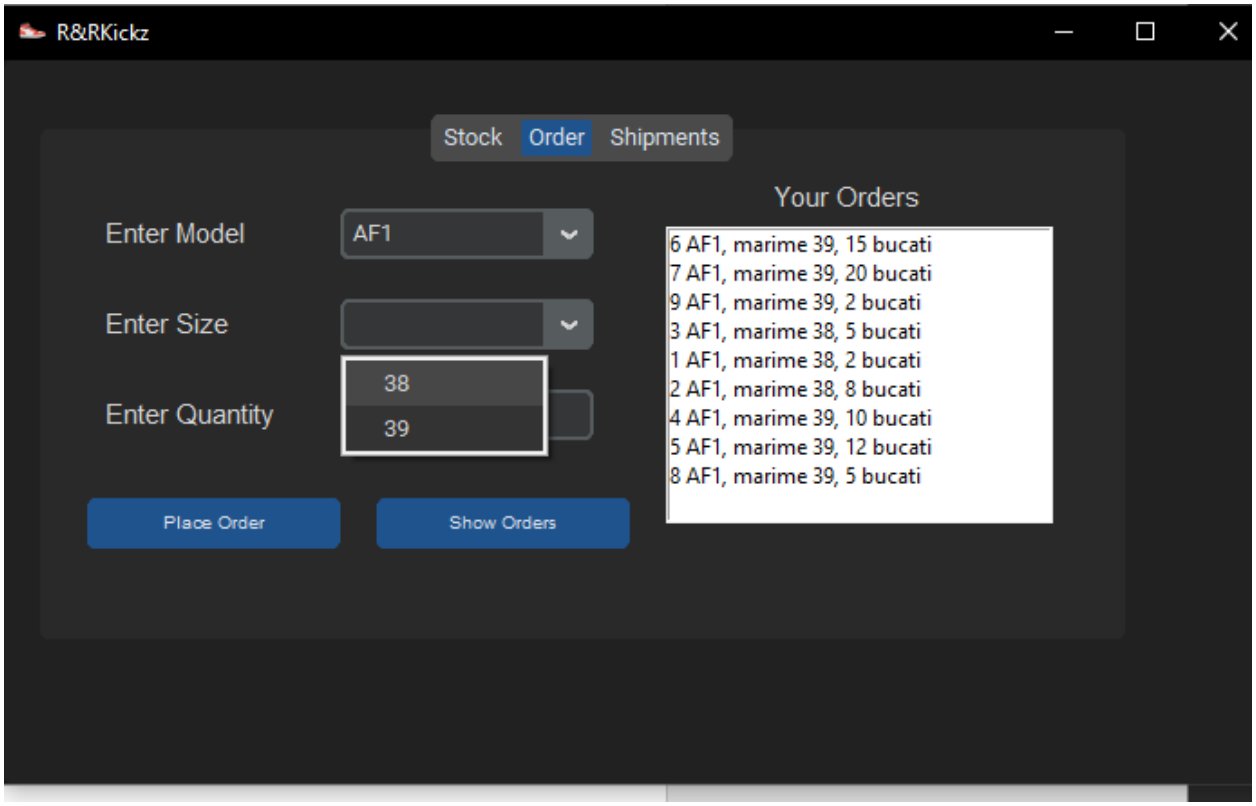
```
def insert():
    id = e_id.get()
    name = e_name.get()
    size = e_size.get()
    qty = e_qty.get()

    con = cx_Oracle.connect("bd149/bd149@bd-dc.cs.tuiasi.ro:1539/orcl")
    cursor = con.cursor()
    cursor.execute("select * from snkrs")
    rows = cursor.fetchall()
    for row in rows:...
    if id == "" or name == "" or size == "" or qty == "":
        tkinter.messagebox.showinfo("Insert Status", "All fields are required")
    if int(qty) < 0:
        tkinter.messagebox.showinfo("Insert Status", "Quantity must be greater or equal to 0")

    else:
        cursor.execute("insert into snkrs values('" + id + "','" + name + "','" + size + " ','" + qty + "')")
        cursor.execute("commit")

        e_id.delete(0, 'end')
        e_name.delete(0, 'end')
        e_size.delete(0, 'end')
        e_qty.delete(0, 'end')
        show()
        tkinter.messagebox.showinfo("Insert Status", "Inserted successfully")
        con.close()
```

Ex. pentru una din cele 4 functii.

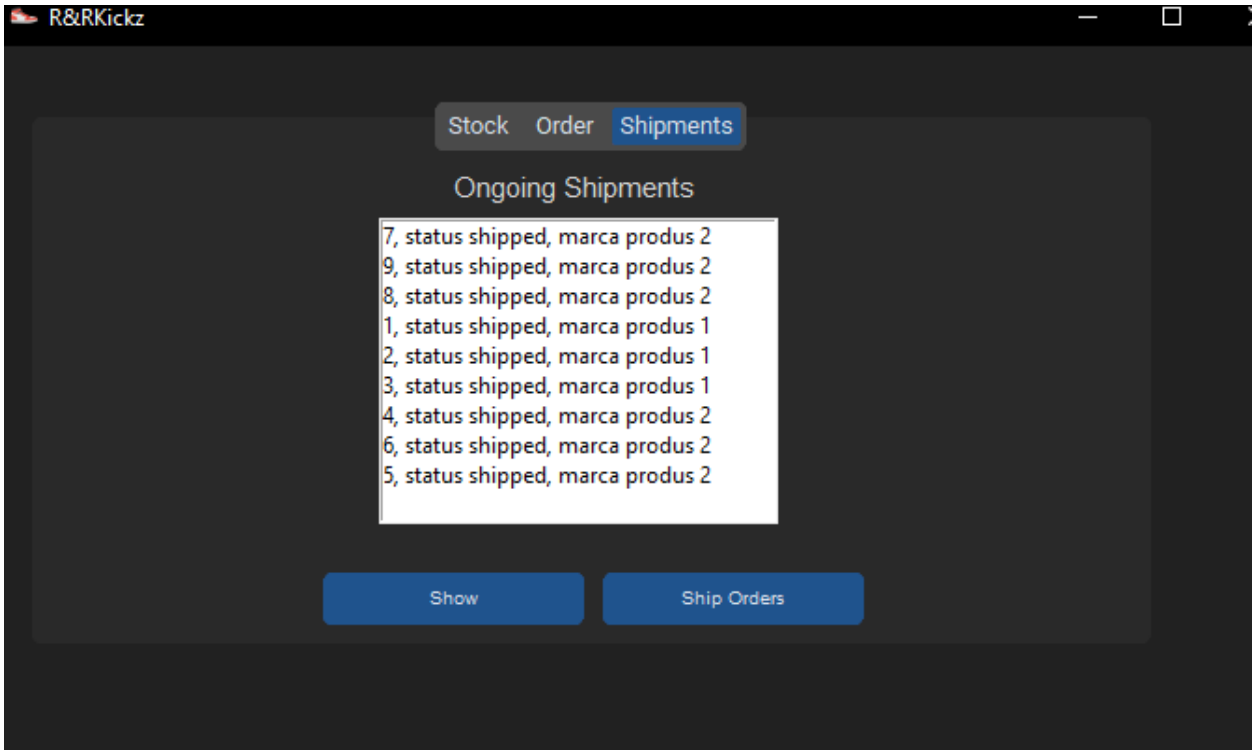


Fereastra pentru plasarea comenzilor. Se pot observa comboboxurile care preiau intrarile din tabele si permit selectarea exclusiv doar a modelelor si marimilor disponibile pt. fiecare.

```
def show():
    con = cx_Oracle.connect("bd149/bd149@bd-dc.cs.tuiasi.ro:1539/orcl")
    cursor = con.cursor()
    cursor.execute("select * from snkrs")
    rows = cursor.fetchall()
    list.delete(0, list.size())

    for row in rows:
        insertData = str(row[0]) + " - model " + row[1] + ", marime " + str(row[2]) + ", bucati in stoc: " + str(row[3])
        list.insert(list.size() + 1, insertData)
    con.close()
```

In majoritatea butoanelor sunt implementate functii de afisare precum aceasta, care permit updatarea listei specifice fiecarui tab pentru o vizualizare mai usoara a datelor.



Butonul ship orders realizeaza parcurgerea tabelelor orders si shipments, ignorand comenzile deja livrate si livrandu-le pe cele care au stoc valabil in momentul de fata. Se salveaza cele neefectuate si se revine la ele cand butonul mai este apasat daca stocul se modifica in viitor. Tabela snkrs este modificata in mod corespunzator.

```
for row1 in rows1:
    for row2 in rows2:
        if str(row1[1]) == str(row2[1]) and str(row1[2]) == str(row2[2]) and row1[0] not in ship_id:
            if int(row1[3]) <= int(row2[3]):
                cursor.execute("update snkrs set qty = qty - '" + str(row1[3]) + "' where nume = '"
                               + str(row1[1]) + "' and s_size = '" + str(row1[2]) + "'")
                cursor.execute("commit")
                print(row2[0])
                cursor.execute("insert into shipmentss values('" + str(row1[0]) + "','" + "shipped" + "','" + str(
                    row2[0]) + "')")
                cursor.execute("commit")
            else:
                tkinter.messagebox.showinfo("Insert Status", "Not enough shoes in stock")
```