MANAGEMENTUL UNUI LANȚ FARMACEUTIC

PHARMAPLUS

-proiect la disciplina Baze de Date-

Student: Stratulat Ștefănel

Grupa: 1306B

Profesor coordinator: Cătălin Mironeanu

Descrierea proiectului

Farmacia este un spațiu în care se desfășoară activități comerciale, principalul obiectiv fiind însănătoșirea pacientului. Astfel, obiectul tranzacției dintre vânzător și client este medicamentul. Fiind o entitate ce are ca obiectiv sănătatea omului, instituțiile guvernamentale au grijă ca farmaciile să comercializeze anumite medicamente, cu rețetă sau liber, unele putând fi preparate chiar și intern, nefiind nevoie de un distribuitor general.

În esență, farmacia este o entitate comercială, motiv pentru care, poate face parte dintr-un lanț extins pe plan național, sau chiar internațional, are un manager și mai mulți angajați cu diferite specializări. Totodată, farmaciile suportă reglementările impuse de lege, trebuind să plătească TVA și să impună un adaos propriu care va reprezenta profitul.

Proiectul "Managementul unui lanț farmaceutic – PharmaPlus" își propune coordonarea activității unui lanț farmaceutic, ținând cont de toate cele prezentate mai sus. Astfel, este prezentat un lanț farmaceutic alcătuit din trei farmacii, fiecare având o anumită locație în orașe diferite. Fiecare farmacie are câte un manager, câte doi farmaciști și trei asistenți, fiecare având salariul în funcție de complexitatea activității prestate. Existența diferențierii dintre farmacist și asistent are la bază ideea diversității tipurilor de medicamente vândute. Cum s-a prezentat mai sus, farmaciile pot vinde medicamente disponibile de la un distribuitor sau pot prepara intern. Astfel, farmaciștii se ocupă de prepararea medicamentelor intern (ceaiuri, creme, tincturi, etc) iar asistenții se ocupă doar de vânzare medicamentelor de la distribuitor. (cu sau fără rețetă)

De asemenea, fiecare farmacie are un anumit stoc de medicamente, cu un preț ce prezintă valorea impusă de stat, la care se adaugă adaosul impus de farmacia respectivă. În cazul epuizării stocurilor, există posibilitatea alimentării de la distribuitori. Fiecare farmacie are un anumit distribuitor, care are un anumit depozit, în care sunt disponibile, în cantități mari, toate medicamentele de care are nevoie o farmacie, medicamente pentru care fiecare farmacie trebuie să plătească o anumită valoare, pentru care trebuie să se asigure că nu depășește valoarea de vânzare.

Tranzacționarea este principala activitate realizată în farmacii, aceasta fiind implementată în cadrul proiectului. Un client merge la farmacie, alege un ghișeu (un farmacist sau un asistent) și solicită un anumit medicament. Dacă există stoc suficient să satisfacă nevoia clientului, acesta este servit, altfel este redirecționat către o altă farmacie din lanț sau i se comunică faptul că nu se găsește nicăieri pe piață cantitatea dorită. Fiecare tranzacție este tratară intern, fiind actualizate tabelele de stocuri și de vânzări.

Ca orice entitate comercială, farmacia trebuie să realizeze contabilitatea. Sunt astfel contorizate : numărul de medicamente vândute, valoarea vânzării, valoarea profitului și valoarea cheltuită pe actualizările de stocuri.

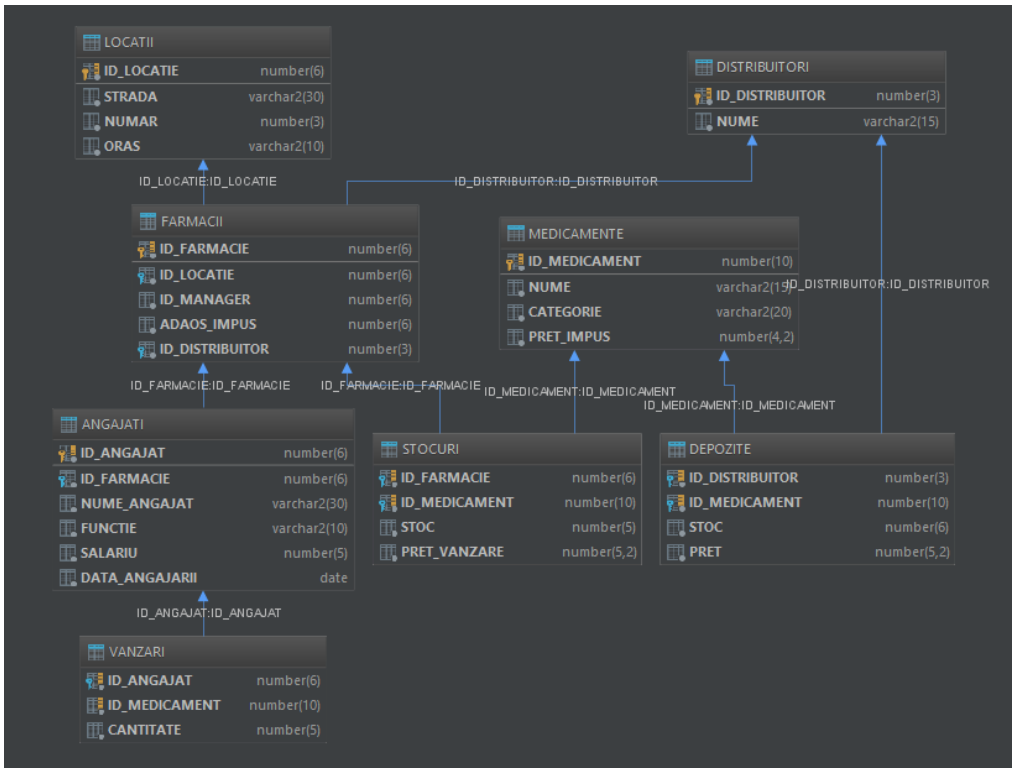
Astfel, proiectul realizează managementul unui lanț farmaceutic, ținând cont, atât de obiectul tranzacției și de nevoile oamenilor, dar și de farmacie ca și entitate comercială.

Tehnologii folosite

Proiectul se bazează pe folosirea bazelor de date relaționale Oracle. Serverul de baze de date Oracle rulează intern pe mașină (localhost). Aplicația a fost dezvoltată folosind Python 3. Pentru toate operatiile de gestiune a bazei de date, s-a folosit librăria disponibilă pentru Python, cx_Oracle. S-a realizat astfel conectarea la baza de date și executarea tuturor prelucrărilor necesare (descriere, afisare, modificare).

Pentru partea de frond-end, s-a folosit tot Python 3, librăria tkinter, care a oferit suport pentru butoane, label-uri, combobox-uri, tabele, entry-uri, etc.

Structura și inter-relaționarea bazelor de date



Între tabele există următoarele relații :

- LOCAȚII – FARMACII : 1 – 1 , deoarece fiecare farmacie are o singură locație
- FARMACII – ANGAJAȚI : 1 - * , deoarece fiecare farmacie are mai mulți angajați (câte 5)
- ANGAJAȚI – VÂNZĂRI : 1 - *, deoarece fiecare angajat a vândut mai multe tipuri de medicamente
- DISTRIBUITOR – FARMACII : 1 - *, deoarece mai multe farmacii pot avea același distribuitor (în cazul nostru, 2 farmacii au același distribuitor, Farmexpert)
- MEDICAMENTE – STOCURI : 1 - *, deoarece un medicament este disponibil în stocul mai multor farmacii.
- MEDICAMENTE – DEPOZITE : 1 - *, deoarece un medicament este disponibil în depozitul mai multor distribuitori.
- DEPOZITE – DISTRIBUITORI : 1 – 1 , deoarece fiecare distribuitor are propriul depozit
- FARMACII – STOCURI : 1 – 1, deoarece fiecare farmacie are un singur stoc.

Descrierea constrângerilor și a modulului de folosire a structurii de tabele

Toate câmpurile din tabela LOCAȚII au constrângerea NOT NULL, iar câmpul ID_LOCATIE are constrângerea PRIMARY KEY, deoarece trebuie să existe câte o locație pentru fiecare farmacie și este câmpul prin care tabela FARMACII identifică locația.

Toate câmpurile din tabela ANGAJAȚI au constrângerea NOT NULL. Câmpul ID_ANGAJAT este PRIMARY KEY, deoarece fiecare angajat este unic și acest câmp se foloseste la identificarea angajatului în tabela VÂNZĂRI. Câmpul ID_FARMACIE este FOREIGN KEY, făcând referire la câmpul ID_FARMACIE din tabela FARMACII, pentru a evidenția faptul că fiecare angajat trebuie să facă parte doar dintr-una dintre farmaciile existente.

Toate câmpurile din tabela VÂNZĂRI au constrângerea NOT NULL. Câmpul ID_ANGAJAT este FOREIGN KEY, făcând referire la câmpul ID_ANGAJAT din tabela ANGAJAT, pentru a evidenția faptul că în tabela VÂNZĂRI, există intrări doar pentru angajații disponibili în tabela ANGAJAȚI. Tupla de câmpuri (ID_ANGAJAT, ID_MEDICAMENT) este UNIQUE KEY pentru a nu exista două câmpuri de același fel, adică, specificarea vânzării unui anumit medicament de către un anumit angajat, să nu fie duplicată, ci actualizată.

Toate câmpurile din tabela DISTRIBUITORI au constrângerea NOT NULL. Câmpul ID_DISTRIBUITOR este PRIMARY KEY, deoarece fiecare distribuitor este unic și este câmpul prin care tabela FARMACII identifică distribuitorul.

Toate câmpurile din tabela MEDICAMENTE au constrângerea NOT NULL. Câmpul ID_MEDICAMENT este PRIMARY KEY, deoarece fiecare medicament este unic și este câmpul prin care tabelele STOCURI și DEPOZITE identifică medicamentul.

Toate câmpurile din tabela STOCURI au constrângerea NOT NULL. Tupla de câmpuri (ID_FARMACIE, ID_MEDICAMENT) este UNIQUE KEY, pentru a nu exista corespondența dintre o farmacie și un medicament de mai multe ori. Câmpul ID_FARMACIE este FOREIGN KEY, referind câmpul ID_FARMACIE din tabela FARMACII, pentru a evidenția faptul că există un stoc, adică o corespondență cu toate medicamente, pentru toate farmaciile, doar dintre cele disponibile in tabela FARMACII. Totodată, câmpul ID_MEDICAMENT este FOREIGN_KEY, referind la câmpul

ID_MEDICAMENT din MEDICAMENTE pentru a evidenția, corespondența farmaciilor doar cu medicamentele disponibile.

Toate câmpurile din tabela DEPOZITE au constrângerea NOT NULL. Tupla de câmpuri (ID_DISTRIBUITOR,ID_MEDICAMENT) este UNIQUE KEY, din aceleași considerente, prezentate anterior. Totodată, câmpurile ID_DISTRIBUITOR și ID_MEDICAMENT sunt FOREIGN KEY, din aceleași considerente, pentru a exista corespondențe doar cu distribuitori dintre cei disponibili în tabela DISTRIBUITORI și corespondențe doar cu medicamente dintre cele disponibile în tabela MEDICAMENTE.

Toate câmpurile din tabela FARMACII au constrângerea NOT NULL. Câmpul ID_FARMACIE este PRIMARY KEY, deoarece fiecare farmacie este unică și este modul în care farmacia este identificată. ID_LOCAȚIE este FOREIGN KEY, făcând referire la câmpul ID_LOCAȚIE din tabela LOCAȚII, sugerând faptul că fiecare farmacie are o locație identificată prin ID, doar dintre cele disponibile în tabela LOCAȚII. ID_DISTRIBUITOR este FOREIGN KEY, referind câmpul din tabela DISTRIBUITORI, sugerând unui distribuitor pentru fiecare farmacie, din tabela DISTRIBUITORI.

Prin existența acestor legături între tabele, aplicația poate extrage anumite informații necesare în anumite prelucrări. De exemplu, pentru a stabili prețul de vânzare al unui medicament pentru o anumită farmacie, vom accesa tupla căutată pe baza ID-ului în tabela MEDICAMENTE și vom extrage prețul pe care îl vom prelucra alături de câmpul ADAOS din tupla din tabela FARMACII, pe baza câmpului ID_FARMACIE.

Conectarea la baza de date

Conectarea la baza de date se face pe baza suportului oferit de librăria cx_Oracle pentru Python 3. Astfel, prin specificarea username-ului, a parolei și a adresei serverului, ca parametrii ai funcției “connect”, se face conectarea. În cazul nostru, funcția va fi apelată astfel :

Connection = cx_Oracle.connect(user =“PHARMA”,password=”8246”,address = ”localhost/xe”).

Aplicația vine și cu suport pentru logare. Astfel, pentru conectarea la baza de date dorită, trebuie ca username-ul și parola să corespundă. În cazul în care acestea corespund, se realizează lansarea meniului aplicației, în caz contrar, se afișează mesajul de eroare corespunzător.

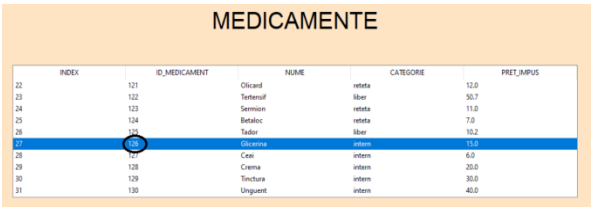
```
def authentification(self,user,passw):
    username = "pharma"
    password = "8246"
    if user.lower() == username.lower():
        self.username = user
    else:
        return False, "Username gresit!\n"
    if passw.lower() == password.lower():
        self.password = passw
    else:
        return False, "Parola gresita!\n"

    self.connection = Oracle.connect(self.username,self.password,self.server_address)
    return True, "Conectare realizata cu succes!\n"
```

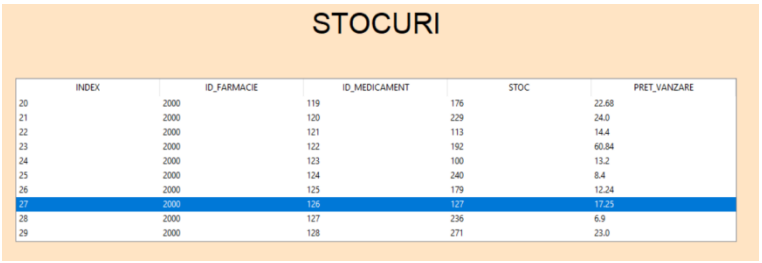
Exemple din implementare

Vom prezenta modul de implementare a operației de efectuare a unei tranzacții.

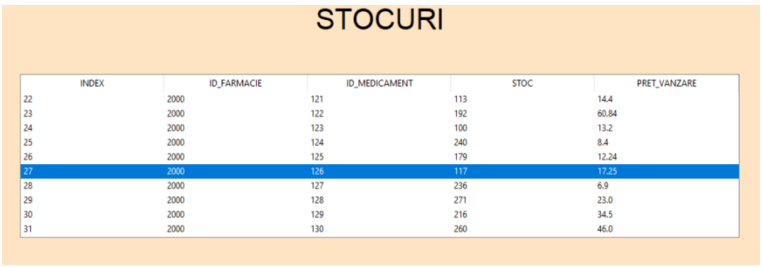
După ce au fost selectate datele necesare, și anume farmacia, ghișeul (angajatul sau farmacistul), medicamentul și cantitatea dorită, se încearcă executarea tranzacției. Dacă cantitatea dorită este disponibilă pe stoc, se realizează tranzacția, altfel se recomandă farmacia care are disponibilă cantitatea dorită sau se afișează mesaj de eroare.

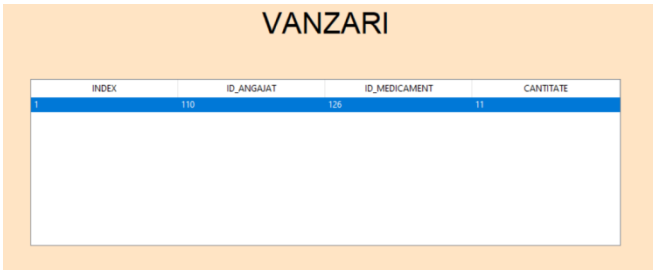


Înainte



După





Implementare

```
def Tranzactie(self):
    if int(self.cantitate.get()) < int(self.stoc[0]):
        nume_medicament = DataBaseOperations.ExtractColumnsWithCondition(database.cursor,
                                                                         "MEDICAMENTE",["NUME"],
                                                                         "ID_MEDICAMENT = {}".format(self.medicament['selectie']))[0][0]
        self.status_tranzactie_var.set("Ati cumparat {} cutii de {} pentru {} lei.".format(self.cantitate.get(),
                                                                                           nume_medicament,
                                                                                           round(self.stoc[1] * float(self.cantitate.get()),2)))
        self.stoc = self.UpdateStoc(int(self.cantitate.get()))
        self.stoc_var.set(self.stoc[0])
        self.UpdateTabelaVanzari(int(self.cantitate.get()))
    else:
        self.status_tranzactie_var.set("Stoc insuficient de medicamente")
        stocuri_medicament_x = []
        farmacii = DataBaseOperations.ExtractColumnsWithCondition(database.cursor,"FARMACII",["ID_FARMACIE"],0)
        for farma in farmacii:
            if farma[0] != self.farmacie['selectie_id']:
                stoc_aux = DataBaseOperations.ExtractDrugsInventory(database.cursor,
                                                                    farma[0],
                                                                    self.medicament['selectie'])[0][0]
                stocuri_medicament_x.append([farma[0],stoc_aux])
        found = 0
        for stoc in stocuri_medicament_x:
            if stoc[1] > int(self.cantitate.get()):
                found = 1
                self.status_tranzactie_var.set("Stoc insuficient de medicamente. Puteti incerca la farmacia {}.".format(stoc[0]))
                break
        if found == 0:
            self.status_tranzactie_var.set("Stoc insuficient de medicamente. Nu gasiti la nicio farmacie cantitatea dorita.")
```

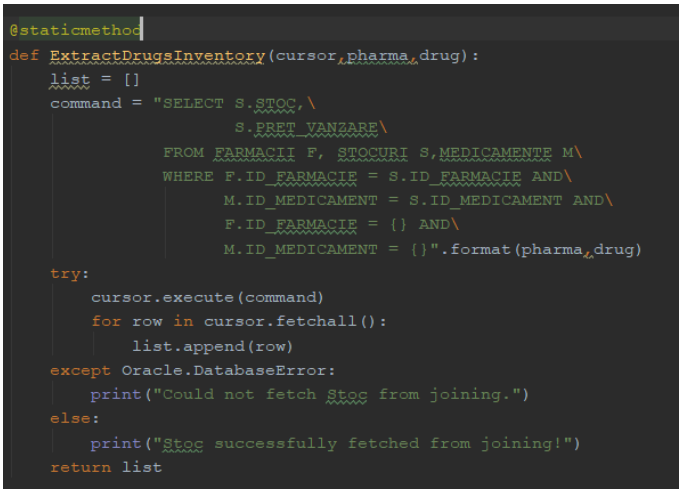
```
def UpdateStoc(self,cantitate):
    DataBaseOperations.AlterInventory(database.connection,
                                      cantitate,
                                      self.farmacie['selectie_id'],
                                      self.medicament['selectie'])


    self.stoc = DataBaseOperations.ExtractDrugsInventory(database.cursor,
                                                         self.farmacie['selectie_id'],
                                                         self.medicament['selectie'])
    return [self.stoc[0][0],self.stoc[0][1]]


def UpdateTabelaVanzari(self,cantitate):
    result = DataBaseOperations.InsertCommand(database.connection,"INSERT INTO VANZARI VALUES({},{},{})".format(self.angajat['selectie_id'],
                                                                                                                 self.medicament['selectie'],
                                                                                                                 cantitate))
    print("***")
    print(result[1])
    print("***")
    if result[0] == False:
        DataBaseOperations.AlterSales(database.connection,
                                      cantitate,
                                      self.angajat['selectie_id'],
                                      self.medicament['selectie'])
```

```
@staticmethod
def AlterSales(connection, quantity, employee, drug):
    cursor = connection.cursor()
    command = "UPDATE VANZARI\
              SET CANTITATE = CANTITATE + ()\
              WHERE ID_ANGAJAT = {} AND ID_MEDICAMENT = {}".format(quantity, employee, drug)
    try:
        cursor.execute(command)
        connection.commit()
    except Oracle.DatabaseError:
        print("Could not alter table VANZARI")
    else:
        print("VANZARI successfully altered.")

@staticmethod
def AlterInventory(connection, quantity, pharma, drug):
    cursor = connection.cursor()
    command = "UPDATE STOCURI\
              SET STOC = STOC - ({})\
              WHERE ID_FARMACIE = {} AND ID_MEDICAMENT = {}".format(quantity, pharma, drug)
    try:
        cursor.execute(command)
        connection.commit()
    except Oracle.DatabaseError:
        print("Could not alter table STOCURI")
    else:
        print("STOCURI successfully altered.")
```



```
@staticmethod
def ExtractDrugsInventory(cursor, pharma, drug):
    list = []
    command = "SELECT S.STOC,\
                      S.PRET_VANZARE\
               FROM FARMACII F, STOCURI S,MEDICAMENTE M\
               WHERE F.ID_FARMACIE = S.ID_FARMACIE AND\
                     M.ID_MEDICAMENT = S.ID_MEDICAMENT AND\
                     F.ID_FARMACIE = {} AND\
                     M.ID_MEDICAMENT = {}".format(pharma, drug)
    try:
        cursor.execute(command)
        for row in cursor.fetchall():
            list.append(row)
    except Oracle.DatabaseError:
        print("Could not fetch Stoc from joining.")
    else:
        print("Stoc successfully fetched from joining!")
    return list
```

Prin utilizarea sistemului de gestiune a bazelor de date Oracle, prin utilizarea limbajului Python 3, împreună cu librăriile corespunzătoare (cx_Oracle și tkinter), proiectul realizează managementul unui lanț farmaceutic.