UNIVERSITATEA TEHNICĂ “GH ASACHI” IAŞI

FACULTATEA AUTOMATICĂ ŞI CALCULATOARE

SPECIALIZAREA CALCULATOARE ŞI TEHNOLOGIA INFORMAŢIEI

DISCIPLINA BAZE DE DATE

Programare spălătoria auto Andor

Coordonator,

Prof. Avram Sorin

Studenți,

Mărgărit Andrada-Mihaela

Toporaș Tudor-Andrei

Grupa 1311A, respectiv 1310B

Iași, 2023

Titlul proiectului

Programare spălătoria auto Andor

Analiza, proiectarea și implementarea unei baze de date care să modeleze programul unei spălătorii de mașini.

Descrierea proiectului

Informațiile de care avem nevoie sunt despre mașinile care vin la spălătorie, angajații care se ocupă de spălarea mașinilor, boxele și stațiile la care se spală mașinile si de promoțiile aferente în intervalul orar stabilit pentru o programare.
Programarea este definită de data și intervalul orar în care a fost făcută. Pentru a face o programare este nevoie de o mașină, de un angajat, o boxa disponibilă și de o promoție care este valabilă cu un anumit discount.
Fiecare angajat are propriul său contract de muncă în care îi este specificat doar orașul în care lucrează. Clientul(reprezentat de mașină) poate vedea doar numele și prenumele angajatului.
Boxele din spălătorie sunt de două tipuri: interior și exterior. Tot acestea determină și prețul unui serviciu. Prețul este variabil de la o boxă la alta, iar durata este de 30 de minute.
Fiecare stație are un număr variabil de boxe de interior și exterior.
Spălătoria ofera și câteva intervale orare în care fiecarui serviciu i se aplica un anumit discount.
Mașinile sunt identificate prin numărul matricol unic și prin marca lor.

Descrierea funcțională a aplicației

Principalele funcții ale aplicației sunt:

- Crearea personalizată de programări
- Evidența programărilor efectuate de o mașină
- Modificarea programărilor existente
- Ștergerea programărilor

Structura și relațiile dintre tabele

Tabelele din această aplicație sunt:

- Mașini
- Stații
- Boxe
- Promoții
- Angajați
- Contracte
- Programări

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

1:1 (one-to-one), 1:n (one-to-many)

Între tabela **Angajați** și tabela **Contracte** se stabilește o relație de 1:1. Un angajat are propriul său contract de muncă și este doar un contract pentru fiecare angajat în parte. Legătura între cele două tabele se face prin câmpul **id_angajat.**

Între tabela **Stații** și tabela **Boxe** se realizează o relație de 1:n. O stație poate avea mai multe boxe, dar o boxă nu poate să aparțină mai multor stații. Legătura dintre cele două tabele se face prin câmpul **id_stație**.

Între tabela **Boxe** și tabela **Programări** se realizează o relație de 1:n. La o boxă se pot face mai multe programări, iar o programare nu poate fi făcută la mai multe boxe în același timp. Legătura dintre cele două tabele se face prin câmpul **id_boxa**.

Între tabela **Angajați** și tabela **Programări** se realizează o relație de 1:n. Un angajat poate lucra la mai multe programări, dar o programare este facută de un angajat. Legătura dintre cele două tabele se face prin câmpul **id_angajat**.

Între tabela **Mașină** și tabela **Programări** se stabilește o relație de 1:n. O mașină poate avea mai multe programări, dar la o programare poate fi doar o mașină. Legătura dintre cele două tabele se face prin câmpul **id_mașină**.

Între tabela **Promoții** și tabela **Programări** se realizează o relație de 1:n. La o programare se poate aplica doar o promoție, dar aceeași promoție poate fi aplicată la mai multe programări. Legătura dintre cele două tabele se face prin câmpul **id_promoție**.

Normalizare

Tabela **Contract** este în FN1(grupurile nu se repetă, există și o cheie externă spre tabela Angajați, realizând o relație one to one), FN2(cheia primară este id_contract, detaliile depind de angajat), FN3(există o singură cheie de care depind toate atributele).Join-urile se fac pe baza id_angajat, nu există nici dependență funcțională, nici dependență multi-vloare, deci se află în **FN5**.

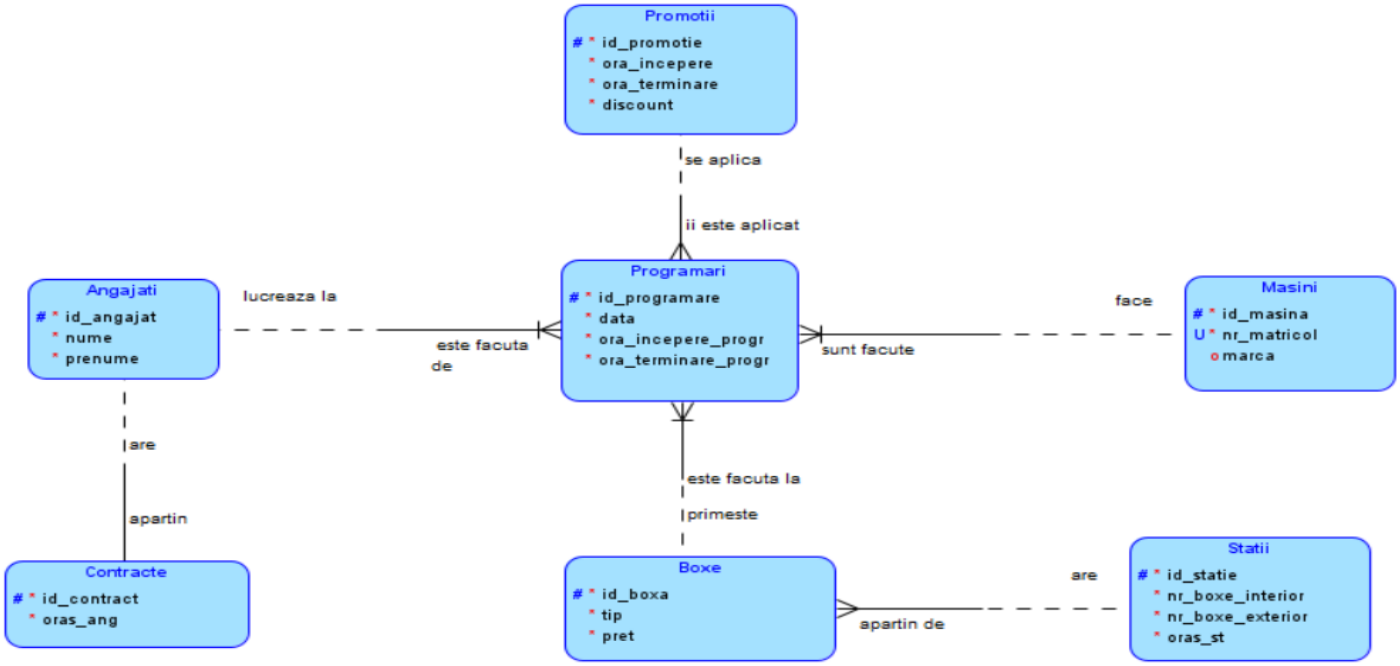
Tabela **Promoții** este tot în **FN5**. FN1(are atribute atomice, grupurile nu se repetă), FN2(atributele depind în totalitate de cheia primară), FN3(este doar o singură cheie candidat), FN3.5(nu există dependența funcțională), FN4(nu este dependență multi-valoare), FN5(joi-urile se fac pe baza id_promoție care este super cheie).

Tabela **Mașini** este in forma normală 5, deoarece este în FN1(are valori atomice și grupurile nu se repetă), este în FN2(atributele depind în totalitate de cheia primară ID), este și în FN3(atributele depind direct de toate cheile candidat), este și în Boyce-Codd – FN3.5(nu există dependență funcțională). Satisface si condiția de a fi în FN4 – nu există dependența multi-valoare, este și în **FN5** pentru că dependența de tip join se face având la bază o super cheie(join-urile se fac cu tabela programări pe baza ID).

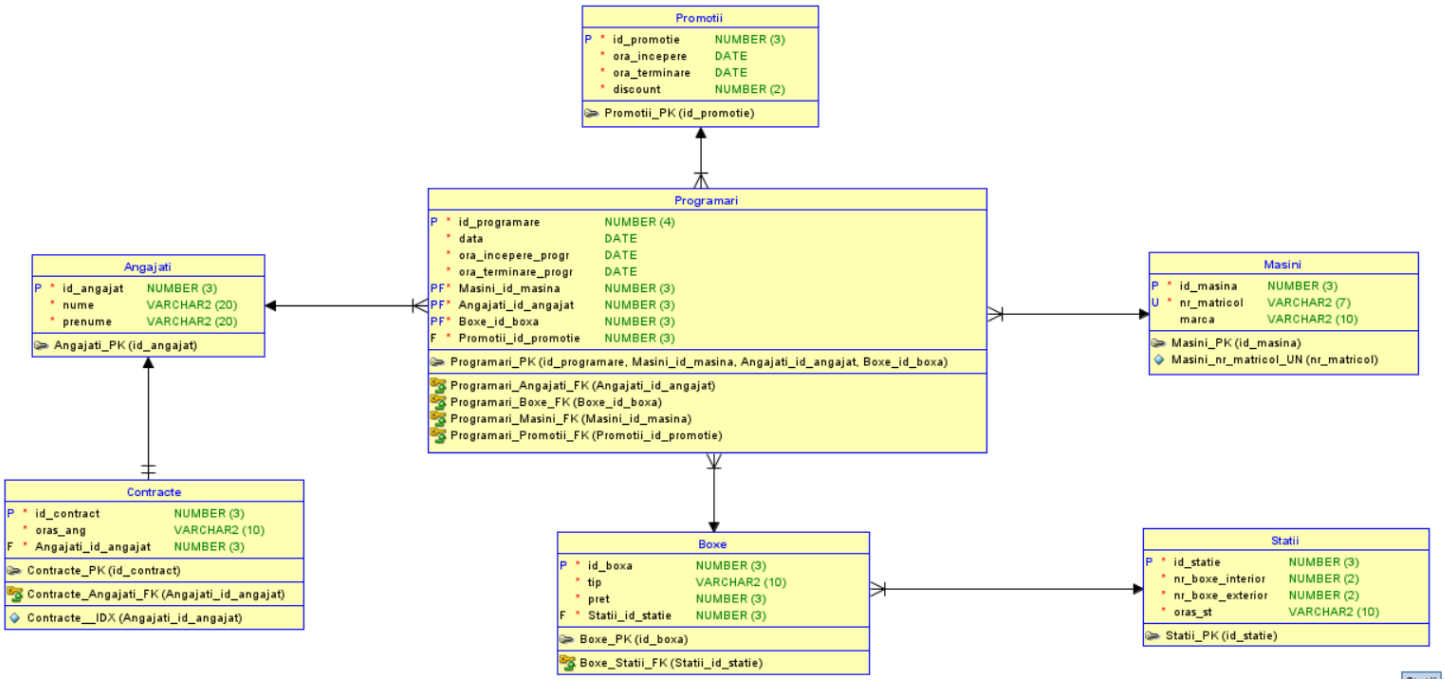
Tabela **Programări** este în FN1(are valori atomice și grupurile sunt unice pe baza ID). Este în FN2 întrucât atributele depind în totalitate de cheia primară, este și în FN3 pentru că este o singură cheie candidat. Este și în FN3.5, deoarece nu există o dependență funcțională, nu există nici dependențe multi-valoare(este în **FN4**), nu este în FN5, deoarece join-urile pentru mașină, promoție nu se fac pe baza unei super-chei.

Tabela **Boxe** este în FN1(grupurile nu se repetă, există și o cheie externă spre tabela Angajați, realizând o relație one to one), FN2(cheia primară este id_contract, detaliile depind de angajat), **FN3**(există o singură cheie de care depind toate atributele).

Modelul logic



Modelul relațional



Descrierea constrângerilor

Constrângerile de tip check se găsesc în aproape toate tabelele. Are loc verificarea dimensiunii valorilor introduse pentru nume(minim 3 caractere), prenume(minim 3 caractere), nr_matricol(6 sau 7 caractere).

Constrângerea de tip check mai apare și pentru a nu putea introduce cifre în câmpurile nume și prenume.

Constrângerile de tip check sunt folosite și pentru verificarea formatului numărului matricol: primele două și ultimele trei caractere să fie litere, iar al treilea și al patrulea să fie cifre sau primul și ultimele trei caractere să fie litere, iar al doilea,al treilea și al patrulea să fie cifre sau primul și ultimele trei caractere să fie litere, iar al doilea și al treilea să fie cifre.

Constrângerea de tip check mai este folosită si atunci când introducem tipul boxei. Aceasta poate fi doar de tip interior sau exterior.

Constrângerea de tip check este folosita si pentru a valida daca ora_terminare este mai mare decat ora_terminare (din tabela Promoții) sau dacă diferența dintre ora_terminare_progr și ora_incepere_progr (din tabela Programări) este de 30 de minute.

Am folosit constrângerea de tip check și pentru a verifica dacă prețul este mai mare decât 20.

Sunt folosite și constrângeri de tip unic pentru atributul nr_matricol deoarece două mașini diferite nu pot avea același număr matricol.

Constrângerile de tip not null se găsesc la toate atributele din tabele, excepție făcând doar marca(din tabela Mașini).

Primary key-urile sunt generate de baza de date printr-un mecanism de tip autoincrement(id_contract, id_angajat, id_programare, id_promotie, id_boxa, id_masina, id_statie).

Foreign key-urile au fost create automat o dată cu crearea relațiilor din tabele.

Tehnologii folosite

Qt Creator este un mediu de dezvoltare integrat care oferă tot ceea ce este nevoie pentru a crea aplicații Qt. Acesta include un editor de cod sursă avansat, un designer de interfață grafică de utilizator (GUI) pentru a ajuta la crearea de formulare și widget-uri, un debugger pentru depanarea aplicațiilor.

cx_Oracle este o librărie Python care permite programelor Python să se conecteze și să interacționeze cu bazele de date Oracle. Utilizând această librărie, se pot executa comenzi SQL și se pot primi rezultatele în aplicația Python. De asemenea, se poate folosi cx_Oracle pentru a crea, modifica și șterge obiecte din baza de date Oracle, precum tabele, indici și proceduri stocate.

PyQt5 este o librărie Python care oferă suport pentru crearea de aplicații grafice folosind Qt, o colecție de instrumente de dezvoltare a interfețelor grafice de utilizator (GUI) și alte utilități de dezvoltare. PyQt5 include module pentru ferestre, widget-uri de bază (cum ar fi butoanele, listele etc.), widget-uri de formulare și de afișare de date (cum ar fi tabele) și multe altele. De asemenea, oferă suport pentru conectarea la semnale, permițând să se scrie aplicații care reacționează la evenimente specifice.

Conectare la baza de date

Această metodă se conecteză la o bază de date Oracle folosind datele de conectare furnizate de utilizator (nume de utilizator și parolă). Dacă conectarea reușește, se închide fereastra actuală și se setează variabila "self.success" la "True". Dacă conectarea nu reușește, se afișează o eroare într-un "label" (probabil o etichetă de text într-o fereastră de interfață grafică) și se resetează câmpul de parolă.

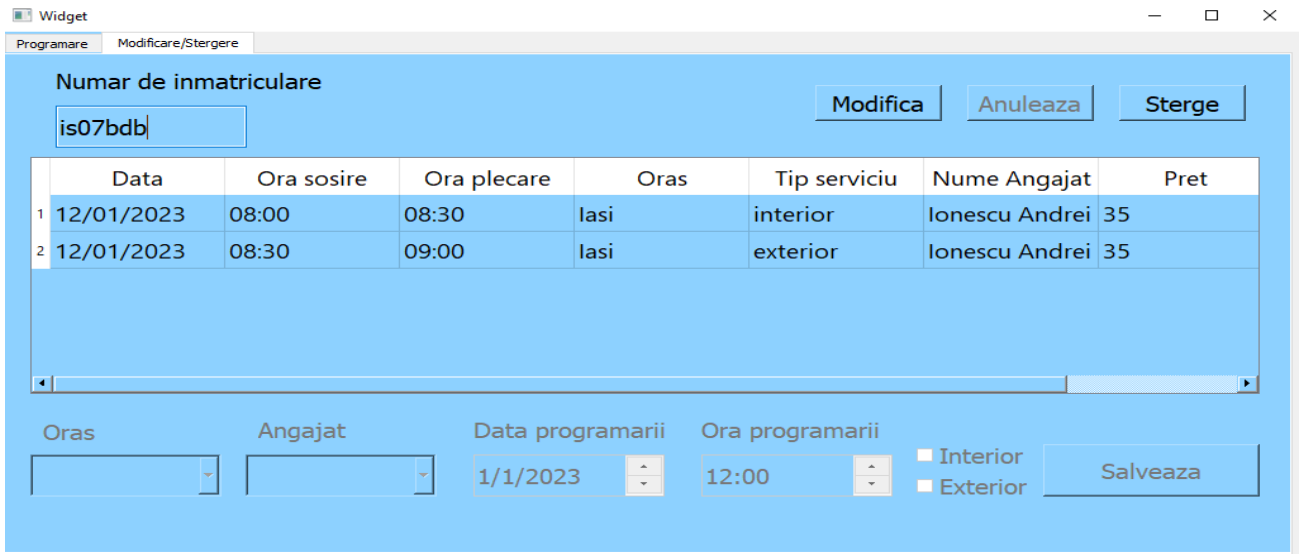
Înainte de a încerca să se conecteze la baza de date, se verifică dacă "self.sq_init" este zero. Dacă este zero, înseamnă că nu s-a inițializat clientul Oracle încă și se inițializează clientul Oracle cu o cale specificată către o librărie de client Oracle. Valoarea lui "self.sq_init" este apoi setată la 1 pentru a indica faptul că clientul Oracle a fost inițializat.

```python
def connect(self):
    """bd154"""
    self.errorLabel.setText(" ")
    if self.sq_init == 0:
        sq.init_oracle_client(lib_dir=r"D:\Different_tools_and_programs\OracleInstantClient\instantclient_21_8")
        self.sq_init = 1
    try:
        self.connection = sq.connect(self.user.text(), self.password.text(), "bd-dc.cs.tuiasi.ro:1539/orcl")
        self.success = True
        self.close()
    except sq.DatabaseError as e:
        err, = e.args
        self.errorLabel.setText(err.message)
        self.password.setText("")
```

Instrucțiuni SQL folosite

Cel mai des a fost folosită instrucțiunea **SELECT** pentru a căuta în baza de date câmpurile/datele necesare. Ca exemplu, utilizarea acestei instrucțiuni a ajutat la afișarea programărilor în tabelă.

```python
def tableUpdate(self):
    self.cursor.execute(f"select p.data,to_char(p.ora_incepere_progr,'hh24:mi')sosire,to_char(p.ora_terminare_progr,'hh24:mi')plecare,\
                          s.oras_st, b.tip,a.nume||' '||a.prenume nume,b.pret*(1-prom.discount*0.01) pret\
                        from angajati a, contracte c,masini m, boxe b,statii s, programari p,promotii prom\
                        where b.statii_id_statie=s.id_statie\
                        and b.id_boxa=p.boxe_id_boxa\
                        and a.id_angajat=c.angajati_id_angajat\
                        and a.id_angajat=p.angajati_id_angajat\
                        and m.id_masina=p.masini_id_masina\
                        and prom.id_promotie=p.promotii_id_promotie\
                        and c.oras_ang=s.oras_st\
                        and m.nr_matricol=upper('{self.matricolUD.text()}')\
                        order by p.data, p.ora_incepere_progr")
    rows = self.cursor.fetchall()
```



Funcția **INSERT** a fost utilizată pentru a insera în tabela Mașini o mașină nouă pe baza numărului de înmatriculre citit din interfață sau pentru a insera o nouă programare în tabela de Programări pe baza numărului de înmatriculare, a orașului, a datei, a orei, a angajatului și a tipului de serviciu, toate fiind selectate din interfață.

```
elif cond == 1:
    self.cursor.execute(
        f"insert into Programari values(NULL,TO_DATE('{self.date.text()}','DD/MM/YYYY'),TO_DATE('{self.time.text()}', 'HH24:MI'),\
                                    TO_DATE(to_char(to_date('{self.time.text()}','hh24:mi')+30/24/60,'hh24:mi'), 'HH24:MI'),{carId},{angIdin},{boxaIdin},{promId1})")
    self.cursor.execute(
        f"insert into Programari values(NULL,TO_DATE('{self.date.text()}','DD/MM/YYYY'),TO_DATE('{self.timetravel()}', 'HH24:MI'),\
                                    TO_DATE(to_char(to_date('{self.timetravel()}','hh24:mi')+30/24/60,'hh24:mi'), 'HH24:MI'),{carId},{angIdex},{boxaIdex},{promId2})")
    self.cursor.execute("commit")
    pret = self.getPrice(self.date.text(), self.time.text(), angIdin)
    pret += self.getPrice(self.date.text(), self.timetravel(), angIdex)
    self.price_label.setText(f"Pretul: {pret} lei")
elif cond == 0:
    self.cursor.execute(
        f"insert into Programari values(NULL,TO_DATE('{self.date.text()}','DD/MM/YYYY'),TO_DATE('{self.time.text()}', 'HH24:MI'),\
                                    TO_DATE(to_char(to_date('{self.time.text()}','hh24:mi')+30/24/60,'hh24:mi'), 'HH24:MI'),{carId},{angIdex},{boxaIdex},{promId1})")
    self.cursor.execute(
        f"insert into Programari values(NULL,TO_DATE('{self.date.text()}','DD/MM/YYYY'),TO_DATE('{self.timetravel()}', 'HH24:MI'),\
                                    TO_DATE(to_char(to_date('{self.timetravel()}','hh24:mi')+30/24/60,'hh24:mi'), 'HH24:MI'),{carId},{angIdin},{boxaIdin},{promId2})")
    self.cursor.execute("commit")
```

O altă funcție utilizată este **UPDATE**. Cu ajutorul acesteia s-au făcut actualizări în tabela de Programări, utilizatorul reușind să schimbe orașul, angajatul, data, ora, tipul de serviciu dorit sau orice combinație între acestea.

```
self.cursor.execute(f"update programari\
                      set data=to_date('{self.date_modify.text()}','DD/MM/YYYY'),\
                          ora_incepere_progr=to_date('{self.time_modify.text()}','HH24:MI'),\
                          ora_terminare_progr=to_date('{self.time_modify.text()}','hh24:mi')+30/24/60,\
                          angajati_id_angajat={angId},\
                          boxe_id_boxa={boxaId},\
                          promotii_id_promotie={promId}\
                              where data=to_date('{item_data}','DD/MM/YYYY')\
                              and ora_incepere_progr=to_date('{item_ora_s}','HH24:MI')\
                              and angajati_id_angajat=(select a.id_angajat from angajati a where a.nume||' '||a.prenume like '{item_nume_angajat}')")
self.cursor.execute("commit")
```

Funcția **DELETE** a ajutat la ștergerea din tabela Programări a unei programări, știindu-se doar numărul de înmatriculare al mașinii. Datorită interfeței folosite utilizatorul poate șterge mai multe programări în același timp.

```
    try:
        self.cursor.execute(f"delete from programari where data=to_date('{item_data}','DD/MM/YYYY')\
                              and ora_incepere_progr=to_date('{item_ora_s}','HH24:MI')\
                              and angajati_id_angajat=(select c.id_angajat from angajati c where c.nume||' '||c.prenume like '{item_nume_angajat}')")
    except Exception as e:
        self.modify_err_label.setText(str(e))
    self.tabelP.removeRow(row)
self.cursor.execute("commit")
```

Contribuția fiecărui student

Proiectul a fost realizat online. S-a lucrat simultan la proiect, fiecare student ajutându-și colegul oferind feedback constant. Toporaș Tudor s-a focusat pe implementarea în python a aplicației, iar Mărgărit Andrada s-a concentrat asupra scrierii interogărilor și comenzilor în baza de date. A fost adus un raport egal de contribuție și în impactul vizual al interfeței.