

UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și tehnologia informației

Parc Auto App

Tema de casa la disciplina Baze de Date

Studenti:

Davidovaia Alexandru - Ionuț

Grupa: 1307B

1. Descrierea proiectului si scopul aplicatiei

Proiectul constă în dezvoltarea unei aplicații desktop utilizând limbajul de programare Python si biblioteca TKinter. Această aplicație are ca scop gestionarea și monitorizarea activităților specifice unui parc auto. Platforma TKinter a fost aleasă pentru a beneficia de o interfață grafică modernă și pentru a asigura o experiență utilizator coerentă.

Scopul principal al acestei aplicații, intitulată "Parc Auto", este de a

gestiona informațiile legate de înregistrări ale clienților, mașinile disponibile în parc, închirierile efectuate și locațiile asociate. Aceasta oferă o interfață grafică utilizatorului pentru a realiza acțiuni precum înregistrarea în baza de date, vizualizarea mașinilor disponibile, realizarea închirierilor și accesarea informațiilor despre locații.

2. Tehnologii folosite pentru front-end si back-end

Aplicația pe care am creat-o este o aplicație desktop dezvoltată în Python și Tkinter, reprezentând un sistem simplist de gestionare a unui parc auto, oferind o interfață intuitivă, funcționalități eficiente de înregistrare a clienților, gestionare simplificată a închirierilor, vizualizare rapidă a mașinilor disponibile la vânzare, toate susținute de o bază de date Oracle pentru o stocare eficientă și sigură a informațiilor.

Front-end:

Tkinter

Tkinter este un modul în limbajul de programare Python, destinat creării interfețelor grafice (GUI - Graphical User Interface). Numele "Tkinter" provine de la "Tk interface", deoarece modulul este bazat pe toolkit-ul Tk, o bibliotecă de interfețe grafice originară din limbajul de programare Tcl (Tool Command Language).

Tkinter oferă programatorilor Python posibilitatea de a crea ferestre, butoane, casete de text, etichete și alte elemente de interfață grafică, facilitând dezvoltarea aplicațiilor cu o interfață utilizator intuitivă și atrăgătoare. Este inclus implicit în distribuția standard a limbajului Python.

Prin intermediul Tkinter, poți gestiona evenimente, interacțiuni cu utilizatorul și actualizarea dinamică a elementelor în cadrul aplicației tale.

Back-end:

- Python

Descriere: Python este un limbaj de programare de înalt nivel, interpretat și generalist. Este cunoscut pentru sintaxa sa clară, ușurința învățării și versatilitatea, fiind potrivit pentru dezvoltarea rapidă a aplicațiilor.

Motivul Utilizării: Python este ales pentru simplitatea sa și comunitatea sa activă, oferind numeroase biblioteci și module care facilitează dezvoltarea.

- Cx_oracle

Descriere: cx_Oracle este un modul Python pentru conectarea la baze de date Oracle. Furnizează un API pentru a interacționa cu baza de date Oracle, inclusiv pentru execuția de comenzi SQL și manipularea datelor.

Motivul Utilizării: Având în vedere că se lucrează cu o bază de date Oracle, cx_Oracle oferă o interfață eficientă pentru a realiza operațiuni de bază de date, precum inserarea și interogarea datelor.

3. Structura si inter-relationarea tabelor din baza de date

În baza de date la care am conectat aplicația am creat 5 tabele: inregistrari (contine toti userii care se inregistreaza prin aplicatie), masini(contine masinile disponibile pentru vanzare si informatii despre acestea), locatii(contine locatiile in care exista o reprezentanta a parcului auto), inchirieri(contine masinile disponibile exclusiv pentru inchiriere), istoric_inchirieri(contine istoricul inchirierilor realizate).

Tabela „inregistrari” este alcătuită din coloanele id de tip number, nume, prenume, numar de telefon si cnp toate 4 de tip VARCHAR2.

Constrangerile din aceasta tabela sunt:

- id_pk: Cheia primară asigură unicitatea fiecărui utilizator în tabel.
- Numar_telefon check verifica sa fie de tipul "07*****"
- Cnp check verifica sa aiba exact 13 caractere.

Tabela masini este alcatuita din coloanele marca,model,culoare,kilometraj de tip VARCHAR2 si pret,an_fabricatie de tip NUMERIC.

Tabela locatii este alcatuita din coloanele oras, strada, email, numar_telefon, program toate de tip VARCHAR2.

Constrangerile din aceasta tabela sunt:

- chk_telefon care verifica ca toate inregistrarile din coloana numar_telefon sa fie de tipul "07*****"
- chk_email care verifica ca toate inregistrarile din coloana email sa aiba un format valid.
- chk_oras care verifica ca toate inregistrarile din coloana oras sa contina doar litere si caracterul „-”.

Tabela inchirieri este alcatuita din coloanele marca,model,locatie de tip VARCHAR2 si an_fabricatie,pret_zi de tip NUMERIC.

Tabela istoric_inregistrari este alcatuita din coloanele nume,prenume,marca,model,locatie de tip VARCHAR2 si an_fabricatie si pret de tip NUMERIC.

| BD050.ISTORIC_INCHIRIERI | |
|-------------------------------|--------------------|
| P * ID | NUMBER (3) |
| NUME | VARCHAR2 (20 BYTE) |
| PRENUME | VARCHAR2 (20 BYTE) |
| MARCA | VARCHAR2 (20 BYTE) |
| MODEL | VARCHAR2 (20 BYTE) |
| AN_FABRICATIE | NUMBER (5) |
| PRET | NUMBER (5) |
| LOCATIE | VARCHAR2 (20 BYTE) |
| F ID_MASINA | NUMBER (5) |
| ID_ISTORIC_INCHIRIERI_PK (ID) | |
| ID_MASINA_FK (ID_MASINA) | |
| ID_ISTORIC_INCHIRIERI_PK (ID) | |



| BD050.INCHIRIERI | |
|------------------|--------------------|
| P * ID | NUMBER (5) |
| MARCA | VARCHAR2 (20 BYTE) |
| MODEL | VARCHAR2 (20 BYTE) |
| AN_FABRICATIE | NUMBER (5) |
| PRET_ZI | NUMBER (5) |
| LOCATIE | VARCHAR2 (20 BYTE) |
| ID_RENT_PK (ID) | |
| ID_RENT_PK (ID) | |

| BD050.INREGISTRARI | |
|--------------------|--------------------|
| P * ID | NUMBER (5) |
| NUME | VARCHAR2 (20 BYTE) |
| PRENUME | VARCHAR2 (20 BYTE) |
| NUMAR_TELEFON | VARCHAR2 (10 BYTE) |
| CNP | VARCHAR2 (13 BYTE) |
| ID_PK (ID) | |
| ID_PK (ID) | |

| BD050.LOCATII | |
|---------------|--------------------|
| ORAS | VARCHAR2 (40 BYTE) |
| STRADA | VARCHAR2 (40 BYTE) |
| EMAIL | VARCHAR2 (40 BYTE) |
| NUMAR_TELEFON | VARCHAR2 (40 BYTE) |
| PROGRAM | VARCHAR2 (40 BYTE) |

| BD050.MASINI | |
|-------------------|--------------------|
| P * ID | NUMBER (5) |
| MARCA | VARCHAR2 (20 BYTE) |
| MODEL | VARCHAR2 (20 BYTE) |
| CULOARE | VARCHAR2 (10 BYTE) |
| PRET | NUMBER (15) |
| AN_FABRICATIE | NUMBER (10) |
| KILOMETRAJ | NUMBER (10) |
| ID_MASINI_PK (ID) | |
| ID_MASINI_PK (ID) | |

4. Conectarea la baza de date din aplicație

Conectarea la baza de date Oracle din aplicație implică utilizarea unui obiect de conexiune specific pentru Oracle și gestionarea corespunzătoare a conexiunii.

Biblioteca necesară pe care trebuie să ne asigurăm că am instalat-o este `cx_oracle`.

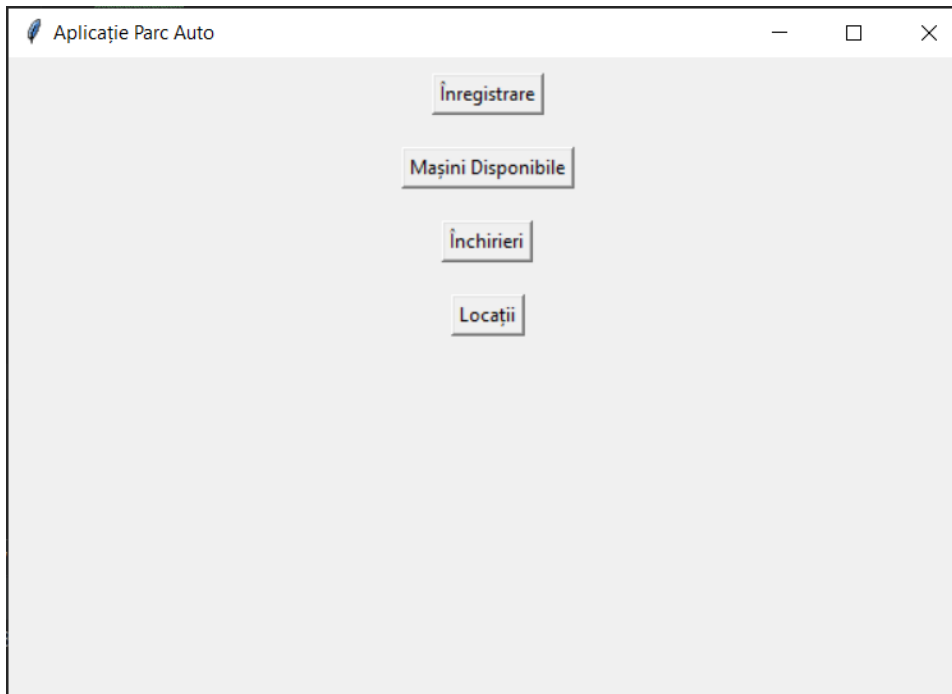
Se poate descărca și instala de pe site-ul oficial *Python* sau direct din managerul *Python Packages* din PyCharm.

Pentru a stabili conexiunea, definim un string de conexiune care conține informațiile necesare pentru conectarea la baza de date:

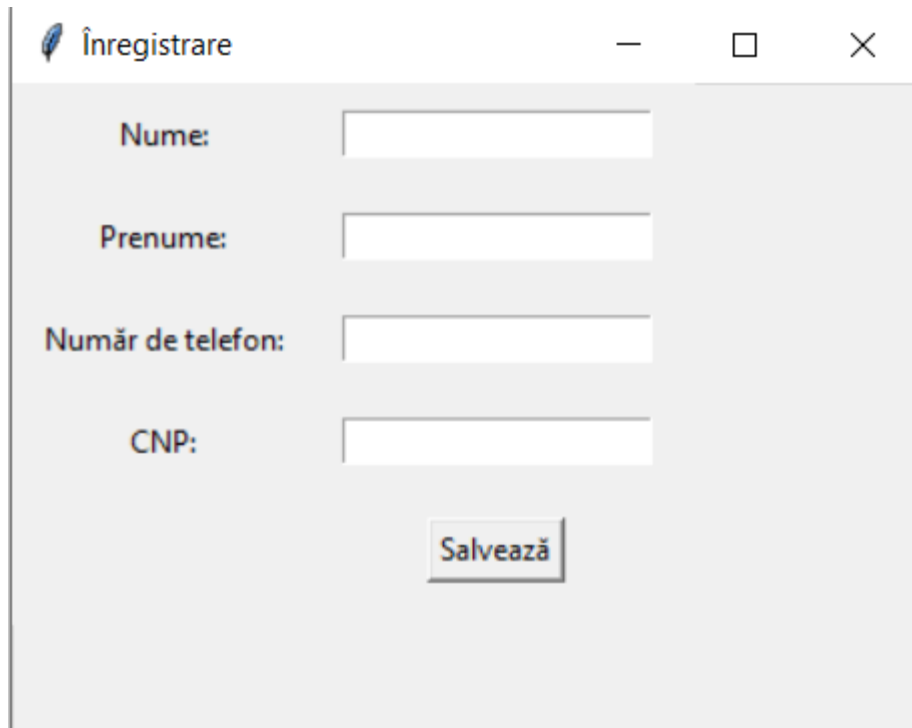
```
connection = cx_Oracle.connect("bd050", "bd050", "81.180.214.85:1539/orcl")
cursor = connection.cursor()
```

5. Interfata aplicatiei si exemple de cod

Imaginea de mai jos reprezintă fereastra principală a aplicației. Prin intermediul acesteia putem folosi toate atributele aplicației.



Un exemplu de utilizare a aplicației împreună cu baza de date Oracle ar fi fereastra de înregistrare a utilizatorilor. În această fereastră se află câteva câmpuri în care persoanele își vor introduce datele, iar acestea vor fi adăugate în tabelele "înregistrari". Voi lăsa mai jos imaginea ferestrei și codul care face posibilă această situație.



Înregistrare

Nume:

Prenume:

Număr de telefon:

CNP:

Salvează

```
def salveaza_inregistrare(self):  
  
    nume = self.entry_nume.get()  
    prenume = self.entry_prenume.get()  
    telefon = self.entry_telefon.get()  
    cnp = self.entry_cnp.get()  
  
    # Validare simplă  
    if not nume or not prenume or not telefon or not cnp:  
        messagebox.showwarning("Avertisment", "Toate câmpurile trebuie completate.")  
        return
```

```
        cursor.execute("INSERT INTO inregistrari (nume, prenume, numar_telefon, cnp) VALUES (:1, :2, :3, :4)",  
                        (nume, prenume, telefon, cnp))  
  
    connection.commit()  
    messagebox.showinfo("Succes", "Înregistrarea a fost salvată cu succes în baza de date.")  
  
    # Închide fereastra de înregistrare după salvare  
    self.root.destroy()
```

Imaginile de mai jos prezintă vizualizarea mașinilor disponibile pentru vânzare în aplicație și codul care asigură funcționalitatea acestei funcții.

| ID | Marca | Model | Culoare | Pret | An Fabricatie | Kilome |
|----|------------|---------|----------|-------|---------------|--------|
| 1 | Toyota | Corolla | alb | 25000 | 2020 | 50000 |
| 2 | Ford | Focus | albastru | 20000 | 2019 | 40000 |
| 3 | Honda | Civic | rosu | 22000 | 2021 | 30000 |
| 4 | Chevrolet | Malibu | negru | 28000 | 2018 | 60000 |
| 5 | Volkswagen | Golf | verde | 23000 | 2022 | 10000 |

```
self.tree = ttk.Treeview(root, columns=("id", "marca", "model", "culoare", "pret", "an-fabricatie", "kilometraj"))
self.tree.heading("id", text="ID")
self.tree.heading("marca", text="Marca")
self.tree.heading("model", text="Model")
self.tree.heading("culoare", text="Culoare")
self.tree.heading("pret", text="Pret")
self.tree.heading("an-fabricatie", text="An Fabricatie")
self.tree.heading("kilometraj", text="Kilometraj")

self.tree.pack(expand=True, fill=tk.BOTH)
```

```
# Populează tabelul cu datele din baza de date Oracle
self.incarca_masini_disponibile()

def incarca_masini_disponibile(self):
    try:
        # Conectare la baza de date Oracle
        connection = cx_Oracle.connect("bd050", "bd050", "81.180.214.85:1539/orcl")
        cursor = connection.cursor()

        # Execută interogarea pentru a obține mașinile disponibile
        cursor.execute("SELECT id,marca, model, culoare, pret, an-fabricatie, kilometraj FROM Masini")

        rows = cursor.fetchall()

        if not rows:
            print("No data returned from the query.")
        else:
            for row in rows:
                self.tree.insert("", "end", values=row)
```