UNIVERSITATEA TEHNICĂ „GHEORGHE ASACHI” IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
SPECIALIZARE CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI
DISCIPLINA BAZE DE DATE PROIECT

RhythmiCore, Where Music Meets Magic

Coordonator: Buțincu Cristian-Nicolae

Student: Alexandra Dabija

Grupa: 1307A

Tema proiectului: Music Library Manager

Descrierea proiectului (non-tehnică): Baza de date aleasă va modela modul de funcționare a unei librării muzicale. Ideea principală a afacerii este aceea a unei aplicații special creată pentru pasionații de muzică, care își doresc un mod facil de a-și organiza și reda colecțiile de melodii preferate. Cu ajutorul acestei aplicații, se pot adăuga și gestiona melodiile, artiștii și albumele preferate ale utilizatorului, dar și să se creeze propria bibliotecă personalizată de melodii, perfect adaptate stării de spirit sau evenimentului. Cu RhythmiCore orice utilizator poate să aibă la îndemână mereu melodiile preferate și să își creeze propriul spațiu muzical personalizat în funcție de genul cu care rezonează, artist, melodie sau album, în final având posibilitatea să acorde un rating în urma audiției.

Descrierea tehnică: În urma analizei cerinței pentru realizarea afacerii, baza de date își propune să gestioneze din punct de vedere tehnic și al analizei tehnice entitățile astfel:

1) **Entitatea Utilizator**:
   - **ID_utilizator** (cheie primară): valoarea numerică, obligatorie, unică pentru fiecare utilizator
   - **nume**: valoare șir de caractere, obligatorie, unică
   - **parola**: valoare șir de caractere, obligatorie
   - **email**: valoare șir de caractere, obligatorie, unică
2) **Entitatea Artist**:
   - **ID_artist** (cheie primară): valoarea numerică, obligatorie, unică pentru fiecare artist
   - **nume**: valoare șir de caractere, obligatorie, unică

3) **Entitatea Album**:
    - **ID_album** (cheie primară): valoarea numerică, obligatorie, unică pentru fiecare album
    - **nume**: valoare șir de caractere, obligatorie
4) **Entitatea Melodie**:
    - **ID_melodie** (cheie primară): valoarea numerică, obligatorie, unică pentru fiecare album
    - **nume**: valoare șir de caractere, obligatorie
5) **Entitatea BibliotecaUtilizator**:
    - **ID Bibliotecă Utilizator** (cheie primară): valoarea numerică, obligatorie, unică pentru fiecare bibliotecă a unui utilizator
    - **notă**: valoare numerică, obligatorie
    - **ID_utilizator** (cheie străină): cheie străină care leagă bibliotecă utilizator de utilizator
    - **ID_melodie** (cheie străină): cheie străină care leagă bibliotecă utilizator de melodie
6) **Entitatea GenMuzical**:
    - **ID_gen_muzical** (cheie primară): valoarea numerică, obligatorie, unică pentru fiecare gen muzical
    - **nume**: valoare șir de caractere, obligatorie
    - **detalii**: valoare șir de caractere, opțional
    - **ID_rating** (cheie străină): cheie străină care leagă genul muzical de rating
7) **Entitatea RatingMelodie**:
    - **rating**: valoarea numerică, obligatorie, în intervalul 1-5
    - **număr_voturi**: valoarea numerică, opțională
    - **ID_melodie** (cheie străină): cheie străină care leagă melodia de rating melodie
8) **Entitate DetaliiMelodie**:
    - **ID_melodie** (cheie străină): cheie străină care leagă albumul de detalii melodie
    - **ID_artist** (cheie străină): atribut folosit pentru a rezolva relația many to many dintre entitățile Melodie și Artist
    - **ID_album (cheie străină)**: cheie străină care leagă albumul de detalii melodie
    - **detalii**: valoare șir de caractere, opțional
9) **Entitatea DetaliiGenuri** -entitate menită să rezolve relația many to many dintre entitățile Melodie și GenMuzical:
    - **ID_melodie** (cheie străină): atribut folosit pentru a rezolva relația many to many dintre entitățile Melodie și GenMuzical
    - **ID_gen_muzical** (cheie străină): atribut folosit pentru a rezolva relația many to many dintre entitățile Melodie și GenMuzical
    - **descriere**: valoare șir de caractere, opțional

În contextul aplicației descrise pentru gestionarea librăriei muzicale, există câteva **probleme sau funcționalități** care sunt acoperite sau sunt scopul principal al aplicației. Iată câteva dintre aceste probleme: **problema utilizatorilor**, aceasta este cea mai importantă funcționalitate, deoarece înregistrarea și autentificarea utilizatorilor sunt esențiale pentru a oferi acces personalizat și sigur la aplicație.

Problema melodiilor, aceasta implică înregistrarea, adăugarea și explorarea melodiilor sau calcularea ratingului mediu al melodiilor și asocierea lor cu genurile muzicale. **Problema Artiștilor și Albumelor**, această funcționalitate ajută utilizatorii să exploreze muzica în funcție de artiști și albume, contribuind la organizarea și descoperirea conținutului muzical.

Totodată câteva dintre cele mai importante relații dintre entități sunt:

- Utilizatorii se înregistrează în sistem, iar informațiile lor sunt stocate în tabela Utilizatori
- Fiecare utilizator are mai multe înregistrări în biblioteca utilizator care îi construiesc indirect propria lui bibliotecă, utilizatorul putând da o notă pentru fiecare melodie pe care o ascultă
- O înregistrare din biblioteca utilizator care vizează o anumită melodie corespunde unui singur utilizator
- Fiecare artist poate avea mai multe albume, iar fiecare melodie poate avea unul sau mai mulți artiști (colaborări muzicale)
- Fiecare melodie este legată de un album, iar un album poate avea una sau mai multe melodii
- Fiecare gen muzical poate să corespundă la mai multe melodii, iar o melodie poate avea și mai mult de un gen muzical, legătura între melodie și genul muzical putând fi stabilită în funcție de genul muzical al piesei
- Utilizatorul acordă un rating unei melodii pe care o ascultă care ulterior este văzută ca o înregistrare în bibliotecă utilizator
- Utilizatorii care acordă rating sunt legați prin intermediul notei pe care o acordă unei melodii indirect de tabela rating melodie, pe baza notelor date se face calculul ratingului mediu pentru fiecare melodie în parte
- Fiecare melodie îi corespunde unui rating unic din tabela RatingMelodie, spre exemplu melodia cu id-ul 1 va corespunde ratingului cu id-ul 1

De menționat sunt și problematicile pe care aplicația nu le tratează: funcționalități avansate de descoperire muzicală, playlist-uri și listele de redare personalizate sau comentarii și recenzii.

În contextul aplicației descrise pentru gestionarea librăriei muzicale vor fi prezentate mai jos **constrângerile pentru fiecare entitate** în parte:

1) **Entitatea Utilizator**:
    - **ID_utilizator**: primary key, not null
    - **nume**: conține doar litere mari sau mici, cratimă și punct, not null
    - **parola:** trebuie să aibă cel puțin 8 caractere
    - **email**: not null, unique și să respecte formatul de tipul: partea locală a adresei de email să conțină litere mici, cifre sau caracterele speciale precum “.”, “_”, ”%” sau “-”; “@” este necesar în orice adresă de email; după partea de domeniu a adresei de e-mail, care are aceleași caracteristici ca și partea locală; după se verifică prezența unui punct și după se definește domeniul superior (.com), unde se așteaptă litere mici de lungime între 2 și 4 caractere.
2) **Entitatea Artist**:
    - **ID_Artist**: primary key, not null
    - **nume**: unique, not null, lungimea numelui este mai mare sau egală cu 1 caractere și poate avea doar litere mari și mici, “.”, “!”, “?”, “$”, “&”, ”%”, ”-”, not null
3) **Entitatea Album**:
    - **ID_album**: primary key, not null
    - **nume**: not null, cu lungimea șirului de caractere mai mare decât 1 și poate avea doar litere mari și mici, “.”, “!”, “?”, “$”, “&”, ”%”, ”-”
4) **Entitatea Melodie**:
    - **ID_melodie**: primary key, not null
    - **nume**: not null, cu lungimea șirului de caractere mai mare decât 1 și poate avea doar litere mari și mici, “.”, “!”, “?”, “$”, “&”, ”%”, ”-”
5) **Entitatea BibliotecaUtilizator**:
    - **ID Bibliotecă Utilizator**: primary key, not null
    - **notă:** not null, în intervalul 0-5
6) **Entitatea GenMuzical**:
    - **ID_gen_muzical**: primary key, not null
    - **nume**: not null, unique, cu lungimea șirului de caractere mai mare decât 1 și să conțină doar litere mari și mici și cratimă, punct și cifre
7) **Entitatea RatingMelodie**:
    - **ID Rating**: primary key, not null
    - **rating**: not null, în intervalul 0-5

O **relație** reprezintă o asociere între entități. Relațiile sunt bidirecționale. Există trei tipuri de relații:

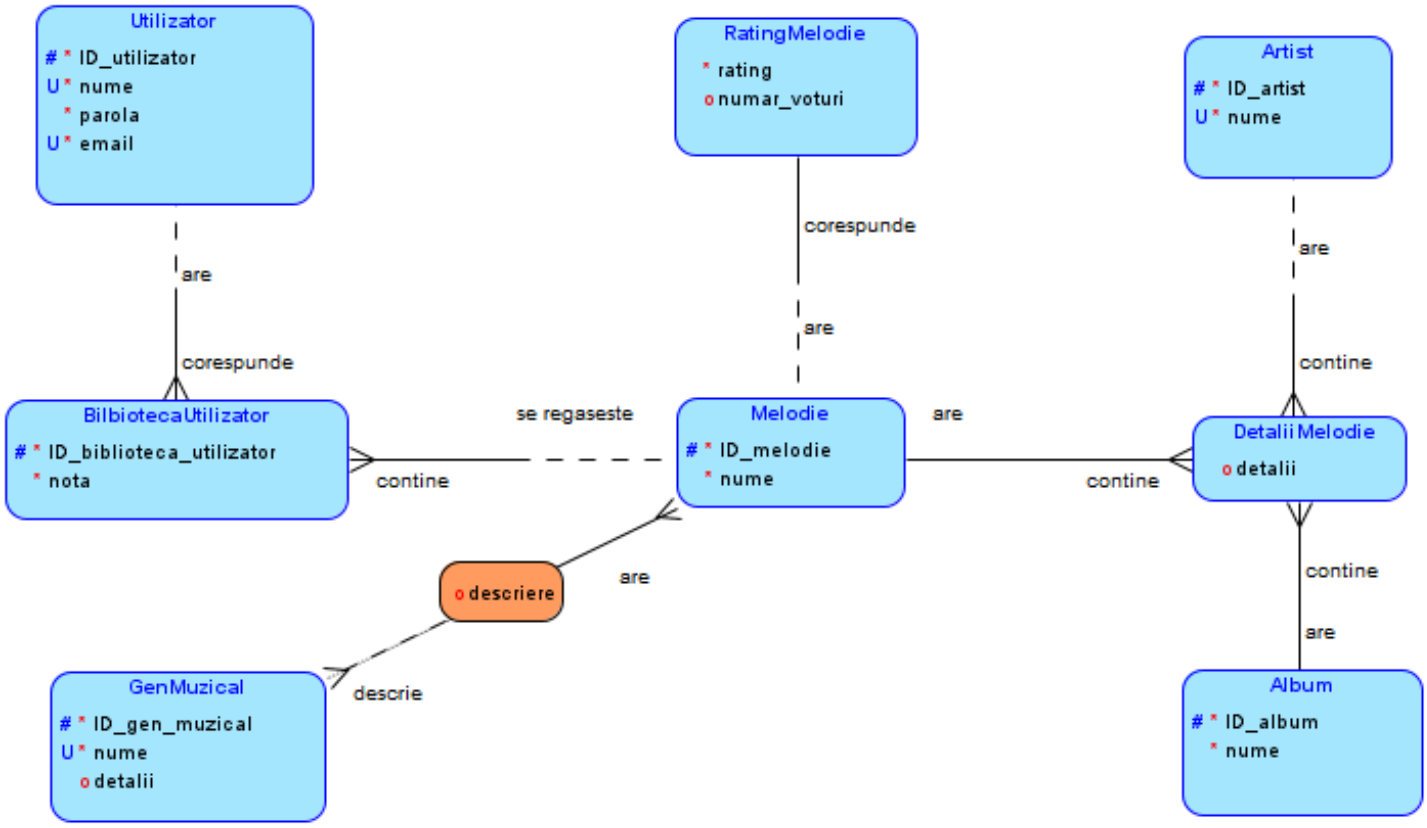
▪ **unu-la-mulți (one-to-many) – 1:M sau 1..***. Exemplu: Un utilizator poate avea mai multe înregistrări în biblioteca muzicală când ascultă o melodie, dar o înregistrare din bibliotecă vizează o anumită melodie care aparține unui singur utilizator.

▪ **mulți-la-mulți (many-to-many) – M:N sau *..***. Exemplu: O melodie poate avea mai multe genuri muzicale și un gen muzical poate corespunde la mai multe melodii.

▪ **unu-la-unu (one-to-one) – 1:1 sau 1..1**. Exemplu: O melodie are un singur rating și un rating corespunde unei singuri melodii.

SCHEMA LOGICĂ

Soluționarea relației many-to-many: Pentru a soluționa relația many-to-many dintre Album-Melodie, Album-Artist și Melodie-Artist se folosește o entitate intermediară numită DetaliiMelodie care va avea foreign key pe toate cele trei entități (artist, album și melodie).



SCHEMA RELAȚIONALĂ

Normalizarea bazelor de date:

Reprezintă procesul de organizare (fără a pierde din informații) a atributelor și tabelelor dintr-o bază de date relațională, cu scopul de a minimiza redundanța datelor și implicit de a minimiza potențialele erori care pot apărea în manipularea datelor.

Prima formă normală (1NF)

O relație este în prima formă normală dacă satisface următoarele condiții:

- un atribut conține valori atomice din domeniul său (și nu grupuri de astfel de valori)
- nu conține grupuri care se repetă

A doua formă normală (2NF)

O relație este în a doua formă normală dacă satisface următoarele condiții:

- este în prima formă normală
- toate atributele non-cheie depind în totalitate de TOATE cheile candidat

Cu alte cuvinte, dacă un atribut non-cheie depinde doar parțial de o cheie candidat (depinde de un subset strict al acesteia), atunci relația nu se află în a doua formă normală. Dacă cheile candidat nu sunt de tipul cheie multiplă (sunt chei simple care conțin un singur atribut), o relație care este în prima formă normală este automat și în a doua formă normală.

A treia formă normală (3NF)

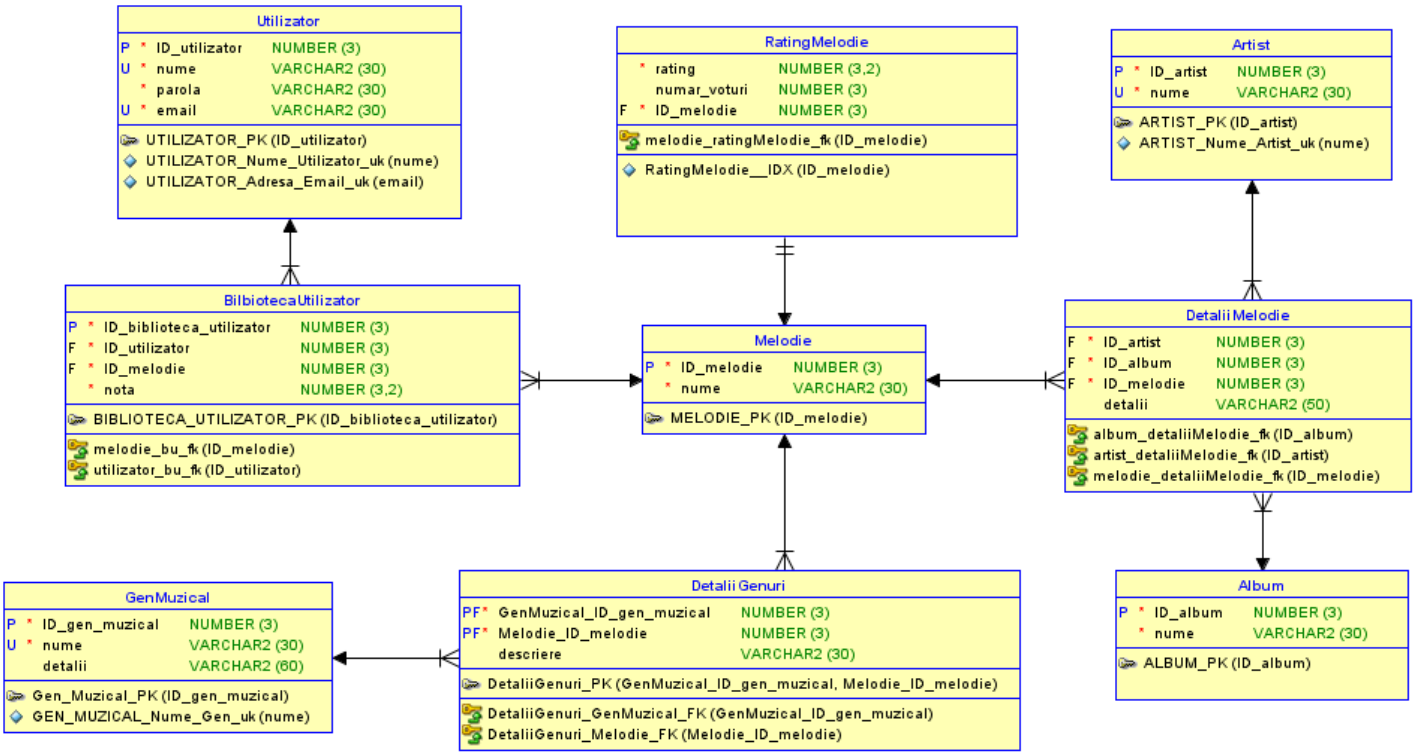
O relație este în a treia formă normală dacă satisface următoarele condiții:

- este în a doua formă normală
- toate atributele non-cheie sunt direct (non-tranzitiv) dependente de TOATE cheile candidat

Pentru prima formă normală, există tabele care au valori atomice în fiecare coloană și nu există grupuri sau array-uri repetitive. Pentru a doua formă normală, toate tabelele sunt deja în 1NF și nu există dependențe tranzitive, fiecare coloană care nu este cheie este complet funcțional dependentă de cheia primară. Pentru a treia formă, toate tabelele sunt deja în 2NF și nu există dependențe tranzitive, fiecare coloană care nu este cheie este non-tranzitivă dependentă de cheia primară.

Relația many-to-many între tabelele GenMuzical și Melodie indică că o melodie poate aparține mai multor genuri muzicale și un gen muzical poat fi asociat mai multor melodii. În acest caz în modelul relațional apare tabela DetaliiGenuri pentru a gestiona această relație. Avem cheie străină genmuzical_id_gen_muzical către id_gen_muzical din tabela GenMuzical și avem cheie străină melodie_id_melodie către id_melodie din tabela Melodie. Această abordare permite ca fiecare

melodie să fie asociată cu mai multe genuri muzicale, iar fiecare gen muzical să fie asociat cu mai multe melodii. Prin intermediul tabelului DetaliiGenuri, se realizează conexiunea între aceste două entități, permițând astfel o relație many-to-many.



Folosirea **autoincrementului** permite generarea automată a valorilor cheilor primare din tabele. Prin această modalitate se permite gestionarea cheilor primare mult mai simplu, se evită introducerea de erori și nevoia de a atribui manual valori unice pentru fiecare înregistrare în scriptul de populare a bazei de date. Exemple de folosirea autoincrementului: pe id_utilizator, id_biblioteca_utilizator, id_gen_muzical, id_album, id_artist si id_melodie.

Folosirea **triggerului** pentru autoincrement, fiecare tabelă care utilizează o secvență pentru a implementa autoincrement are un trigger asociat. Triggerul asigură că înaintea fiecărei operații de inserare, secvența este apelată și valoarea generată este atribuită automat atributului din coloana corespunzătoare.

TEHNOLOGII FOLOSITE

Partea de back-end a fost realizată utilizând limbajul **Python**. Pentru **partea de front-end** s-a folosit pachetul **tkinter**, unul dintre cele mai utilizate pachete pentru implementarea interfețelor grafice în Python. Tkinter este o bibliotecă de interfață grafică (GUI) pentru limbajul de programare Python. Aceasta permite dezvoltatorilor să creeze interfețe grafice utilizator (UI) pentru aplicații desktop. Tkinter oferă un set de instrumente și widget-uri grafice, cum ar fi ferestre, butoane, etichete și câmpuri de text, care pot fi utilizate pentru a crea aplicații cu interfețe utilizator personalizate.

CONEXIUNEA LA BAZA DE DATE

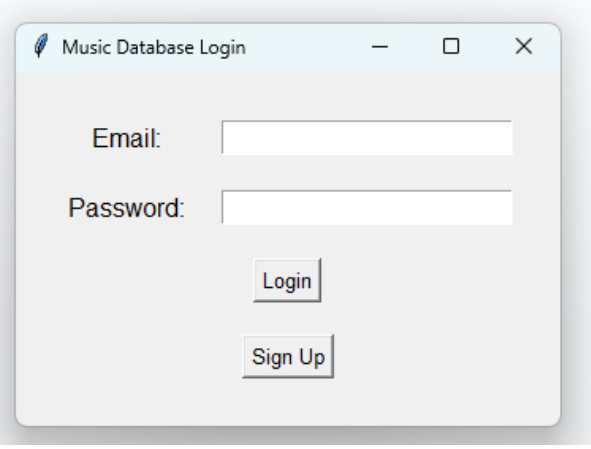
Pentru a realiza conexiunea la baza de date, am adoptat funcția **connect din modulul cx_Oracle**, avându-i ca parametri user-ul, parola și DSN-ul (data source name). Am creat un cursor prin intermediul metodei "cursor" a obiectului conexiune returnat de funcția "connect". Acest cursor este apoi utilizat pentru a executa comenzi SQL. Detaliile conexiunii la baza de date sunt stocate într-o variabilă globală în fișierul "database.py". Această variabilă poate fi importată în toate fișierele care necesită o conexiune la baza de date. La încheierea programului, am implementat închiderea conexiunii folosind metoda "connection.close()". Cx_Oracle este o bibliotecă Python pentru a lucra cu baze de date Oracle, permitând dezvoltatorilor să se conecteze la aceste baze de date, să execute interogări SQL și să manipuleze datele în ele folosind Python. Este utilă pentru dezvoltarea aplicațiilor care interacționează cu baze de date Oracle.

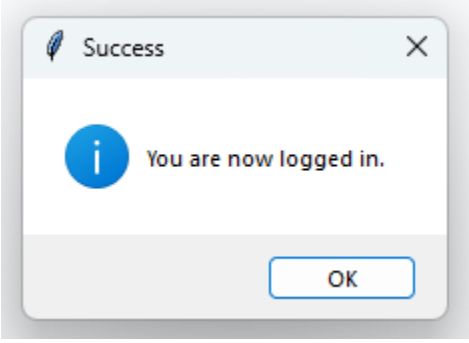
Exemplu de cod:

```
    import cx_Oracle
    cx_Oracle.init_oracle_client(lib_dir=r"C:\Programe_facultate\instantcli
ent_21_12")


    # Database connection parameters
    connStr = 'bd033/bd033@bd-dc.cs.tuiasi.ro:1539/orcl'


    # Function to connect to the Oracle database
    def create_connection():
        return cx_Oracle.connect(connStr)
```

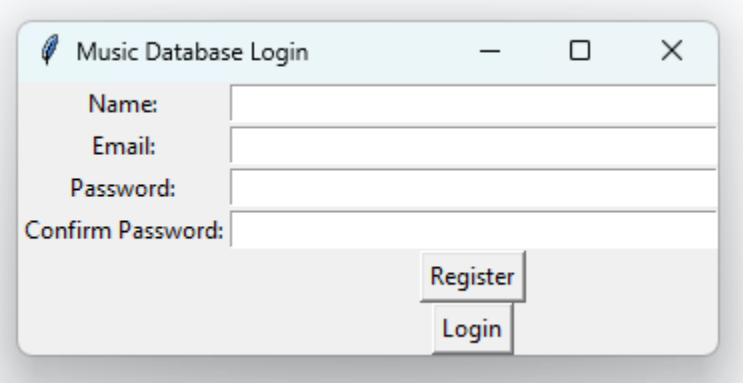
CAPTURI DIN INTERFAȚĂ GRAFICĂ

Fereastra de Login pentru user existent





Fereastra de Register pentru user nou



Fereastra pentru USER

- Prezintă o vizualizare a propriei biblioteci a userului care s-a logat în partea de sus în care acesta poate vedea ce melodii a adăugat și ce notă le-a dat

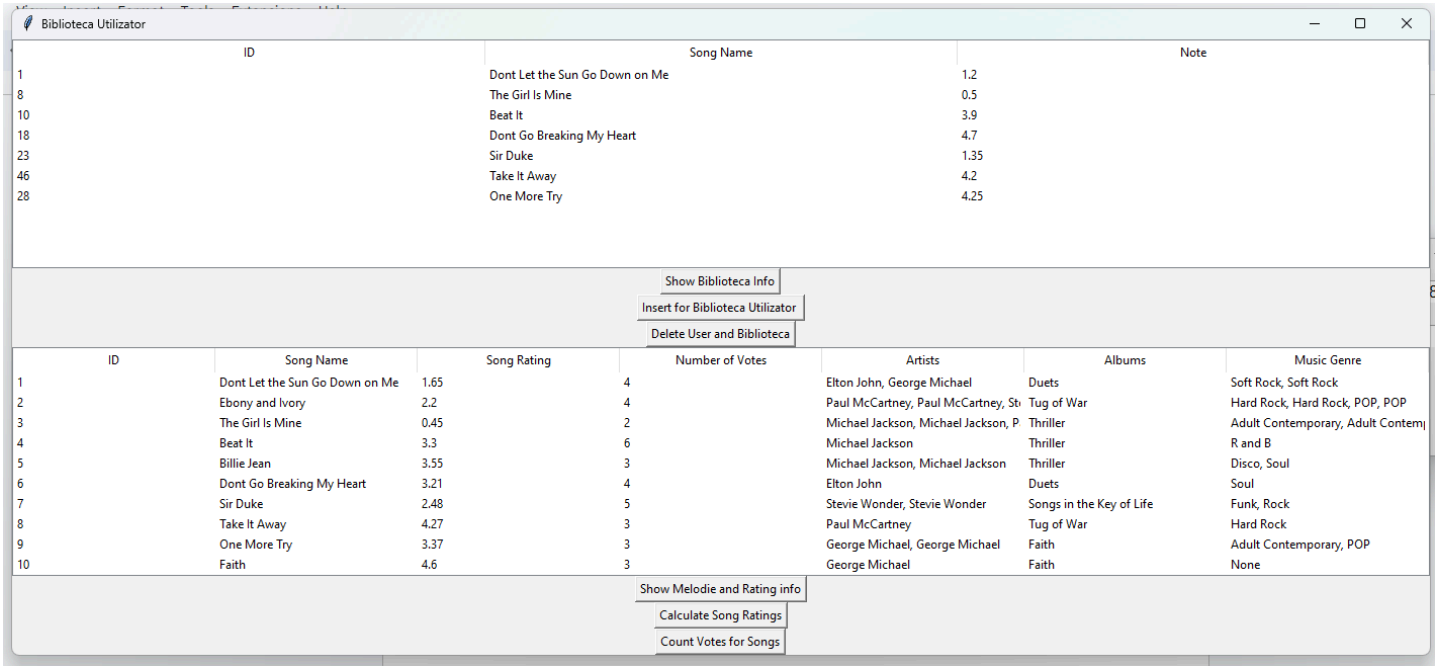
```
def open_biblioteca_utilizator_window(user_id):
    global tree_biblioteca, tree_melodie_rating

    # Create a new window
    new_window = tk.Toplevel()
    new_window.title("Biblioteca Utilizator")

    # Function to fetch user data
    def fetch_user_data_biblioteca():
        try:
            conn = create_connection()
            cursor = conn.cursor()
            # Adjust this query based on your actual table names and column names
            query = f"""
           SELECT bu.id_biblioteca_utilizator, m.nume, bu.nota
            FROM bilbiotecautilizator bu
            JOIN melodie m ON bu.id_melodie = m.id_melodie
            WHERE bu.id_utilizator = {user_id}
            """
            cursor.execute(query)
            rows = cursor.fetchall()

            # Clear existing data in the treeview
            for i in tree_biblioteca.get_children():
                tree_biblioteca.delete(i)

            # Inserting new data
            for row in rows:
                tree_biblioteca.insert("", tk.END, values=row)

            cursor.close()
            conn.close()

        except Exception as e:
            messagebox.showerror("Database Error", str(e))

    # Create a Treeview widget to show the data
    tree_biblioteca = ttk.Treeview(new_window, columns=('ID', 'Song Name',
'Note'), show='headings')
    tree_biblioteca.heading('ID', text='ID')
```

```
    tree_biblioteca.heading('Song Name', text='Song Name')
    tree_biblioteca.heading('Note', text='Note')
    tree_biblioteca.pack(fill='both', expand=True)

    # Add a button to fetch the data
    fetch_button = tk.Button(new_window, text="Show Biblioteca Info",
command=fetch_user_data_biblioteca)
    fetch_button.pack()

    # Initial call to fetch and display data
    fetch_user_data_biblioteca()
```

- În partea de jos fereastra utilizatorului prezintă o afișare amănunțită a melodiilor disponibile în întreaga bibliotecă muzicală, adică detalii precum numele melodiei, rating general calculat pe baza tuturor notelor date de utilizatori, numărul total de note date de toți utilizatorii (număr voturi), artistul sau artiștii corespunzători melodiei, albumul din care face piesa și genurile muzicale aferente

```
# Function to fetch user data
    def fetch_user_data_melodie_rating():
        try:
            conn = create_connection()
            cursor = conn.cursor()
            # Query to fetch user data including songs and their ratings

            # query = f"""
            # SELECT m.id_melodie, m.nume AS "Nume Melodie", r.rating AS
"Rating", r.numar_voturi AS "Numar Voturi"
            # FROM melodie m
            # LEFT JOIN ratingmelodie r ON m.id_melodie = r.id_melodie
            # conn.commit()

            # artists, albums and genres
            query = """
            SELECT m.id_melodie, m.nume AS "Song Name", rm.rating AS "Rating",
rm.numar_voturi AS "Number of Votes",
                LISTAGG(a.nume, ', ') WITHIN GROUP (ORDER BY a.nume) AS
"Artists",
                al.nume AS "Album",
                LISTAGG(g.nume, ', ') WITHIN GROUP (ORDER BY g.nume) AS "Genres"
            FROM melodie m
            LEFT JOIN ratingmelodie rm ON m.id_melodie = rm.id_melodie
            LEFT JOIN detaliimelodie dm ON m.id_melodie = dm.id_melodie
```

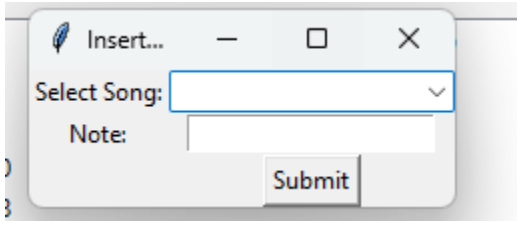
```
            LEFT JOIN artist a ON dm.id_artist = a.id_artist
            LEFT JOIN album al ON dm.id_album = al.id_album
            LEFT JOIN detaliigenuri dg ON m.id_melodie = dg.melodie_id_melodie
            LEFT JOIN genmuzical g ON dg.genmuzical_id_gen_muzical =
g.id_gen_muzical
            GROUP BY m.id_melodie, m.nume, rm.rating, rm.numar_voturi, al.nume
            ORDER BY m.id_melodie
            """
            cursor.execute(query)
            rows = cursor.fetchall()

            # Clear existing data in the treeview
            for i in tree_melodie_rating.get_children():
                tree_melodie_rating.delete(i)

            # Inserting new data
            for row in rows:
                tree_melodie_rating.insert("", tk.END, values=row)

            cursor.close()
            conn.close()

        except Exception as e:
            messagebox.showerror("Database Error", str(e))
```

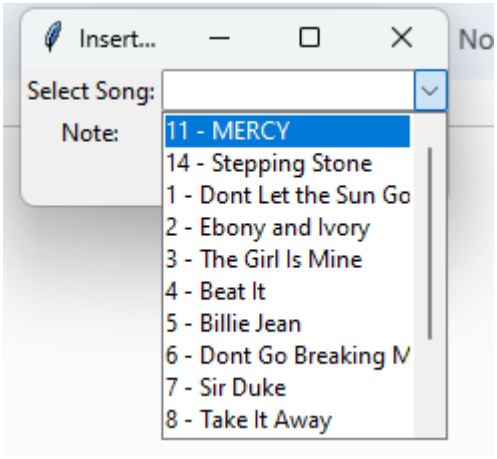


- Existența butonului pentru refresh pentru afișare bibliotecii utilizatorului

```
# Add a button to fetch the data
    fetch_button = tk.Button(new_window, text="Show Biblioteca Info",
command=fetch_user_data_biblioteca)
    fetch_button.pack()
```

- Existența unui buton de inset care deschide o altă fereastră de unde utilizatorul poate adăuga o nouă melodie și să îi ofere o notă. Dacă melodia există doar se actualizează nota melodiei deja existentă în biblioteca sa






```
def open_insert_data_window(user_id):
    insert_window = tk.Toplevel()
    insert_window.title("Insert Data")

    # Fetch songs from the database
    def fetch_songs():
        try:
            conn = create_connection()
            cursor = conn.cursor()
            cursor.execute("SELECT id_melodie, nume FROM melodie")  # Adjust
query as needed
            songs = cursor.fetchall()
            cursor.close()
            conn.close()
            return songs
        except Exception as e:
            messagebox.showerror("Database Error", str(e))
            return []

    # Dropdown for song selection
```

```
    songs = fetch_songs()
    song_var = tk.StringVar()
    song_combobox = ttk.Combobox(insert_window, textvariable=song_var,
values=[f"{song[0]} - {song[1]}" for song in songs])
    song_combobox.grid(row=0, column=1)
    tk.Label(insert_window, text="Select Song:").grid(row=0, column=0)

    # Entry for note
    tk.Label(insert_window, text="Note:").grid(row=1, column=0)
    note_entry = tk.Entry(insert_window)
    note_entry.grid(row=1, column=1)


    # Function to insert or update data in the database
    def insert_or_update_data():
        song_id = song_var.get().split(" - ")[0]
        note = note_entry.get()
        try:
            conn = create_connection()
            cursor = conn.cursor()

            # Check if the entry exists
            check_query = f"SELECT COUNT(*) FROM bilbiotecautilizator WHERE
id_utilizator = {user_id} AND id_melodie = {song_id}"
            cursor.execute(check_query)
            exists = cursor.fetchone()[0] > 0

            if exists:
                # Update the note if the entry exists
                update_query = f"UPDATE bilbiotecautilizator SET nota = '{note}'
WHERE id_utilizator = {user_id} AND id_melodie = {song_id}"
                cursor.execute(update_query)
                messagebox.showinfo("Success", "Note updated successfully")
            else:
                # Insert new data if the entry does not exist
                insert_query = f"INSERT INTO bilbiotecautilizator (id_utilizator,
id_melodie, nota) VALUES ({user_id}, {song_id}, '{note}')"
                cursor.execute(insert_query)
                messagebox.showinfo("Success", "Data inserted successfully")

            conn.commit()
            cursor.close()
            conn.close()
```

```
        except Exception as e:
            messagebox.showerror("Database Error", str(e))

        # Submit button
    submit_button = tk.Button(insert_window, text="Submit",
command=insert_or_update_data)
    submit_button.grid(row=3, column=1)
```

- Existența unui buton prin care utilizatorul poate alege dacă vrea sau nu să își șteargă contul

```
    # Button to delete the user and their biblioteca utilizator entries
    delete_button = tk.Button(new_window, text="Delete User and Biblioteca",
command=lambda: delete_user_and_biblioteca(user_id, new_window, root))
    delete_button.pack()
```

```
    def delete_user_and_biblioteca(user_id, biblioteca_window, main_window):
        # Confirmation dialog
        response = messagebox.askyesno("Confirm", "Are you sure you want to
delete this user and all their data?")
        if response:
            try:
                conn = create_connection()
                cursor = conn.cursor()

                # Delete user data from bilbiotecautilizator
                cursor.execute(f"DELETE FROM bilbiotecautilizator WHERE
id_utilizator = {user_id}")

                # Delete user from utilizator
                cursor.execute(f"DELETE FROM utilizator WHERE id_utilizator =
{user_id}")
                conn.commit()

                # Close the biblioteca window and return to login
                biblioteca_window.destroy()
                main_window.deiconify()
```

```
                messagebox.showinfo("Success", "User and data deleted
successfully.")

                cursor.close()
                conn.close()
            except Exception as e:
                messagebox.showerror("Database Error", str(e))
```

- Existența unui buton care calculează ratingul melodiilor pe baza notelor date de utilizator

```
calculate_rating_button = tk.Button(new_window, text="Calculate Song Ratings",
command=calculate_song_rating)
    calculate_rating_button.pack()
```

```
def calculate_song_rating():
        try:
            conn = create_connection()
            cursor = conn.cursor()

            # Calculate average user rating for each song
            cursor.execute("""
            UPDATE ratingmelodie rm
            SET rating = (
                SELECT COALESCE(AVG(bu.nota), 0)
                FROM bilbiotecautilizator bu
                WHERE bu.id_melodie = rm.id_melodie
            )
            """)

            conn.commit()
            cursor.close()
            conn.close()
            messagebox.showinfo("Success", "Song ratings updated
successfully.")
            fetch_user_data_melodie_rating()


        except Exception as e:
            messagebox.showerror("Database Error", str(e))
```

- Existența unui buton care calculează numărul de voturi pentru fiecare melodie

```
 count_votes_button = tk.Button(new_window, text="Count Votes for Songs",
command=count_votes_for_songs)
```

```
    count_votes_button.pack()
```

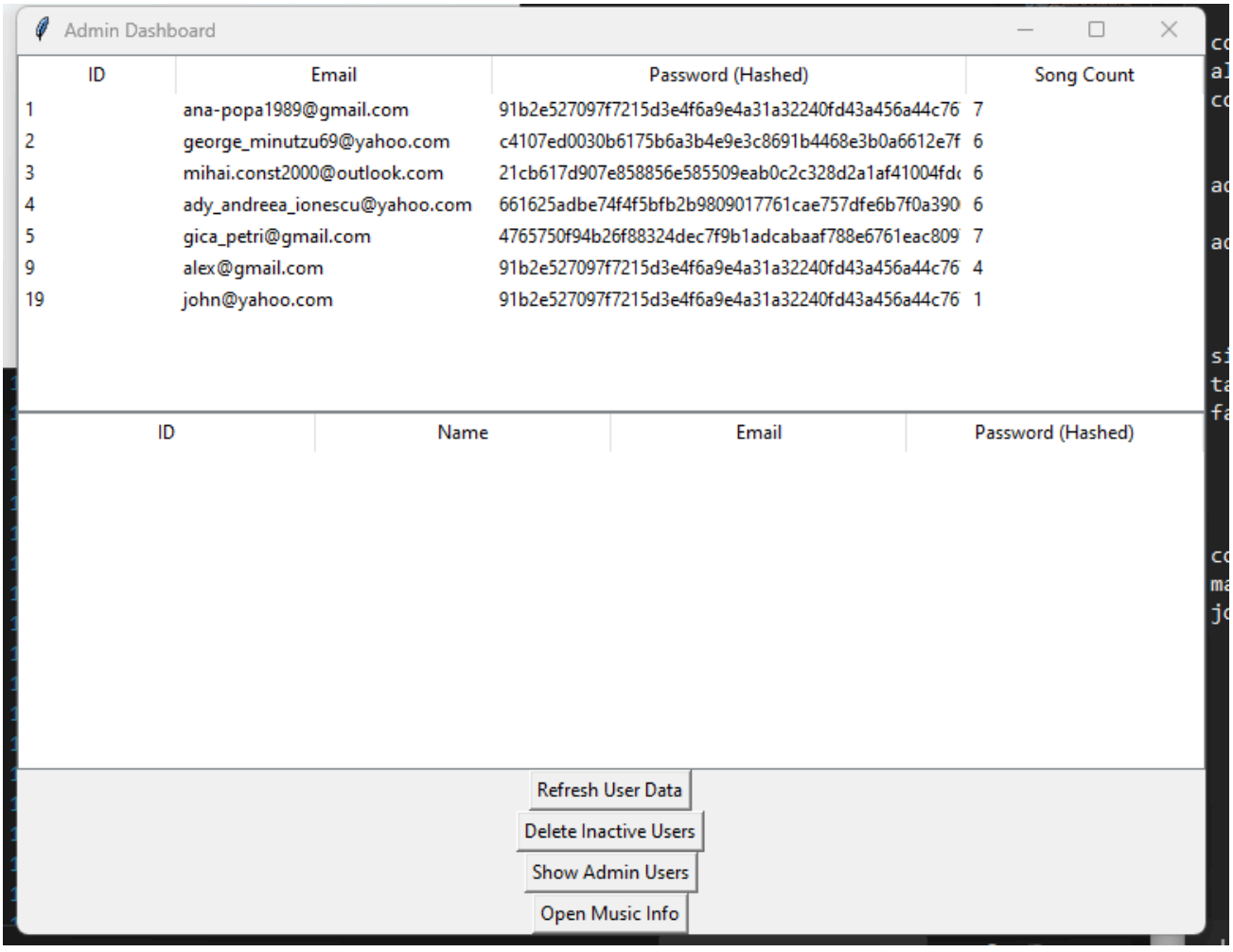
```
def count_votes_for_songs():
        try:
            conn = create_connection()
            cursor = conn.cursor()

            # Calculate the number of votes for each song
            cursor.execute("""
            UPDATE ratingmelodie rm
            SET numar_voturi = (
            SELECT COUNT(bu.id_biblioteca_utilizator)
            FROM bilbiotecautilizator bu
            WHERE rm.id_melodie = bu.id_melodie)
            """)
            conn.commit()
            cursor.close()
            conn.close()
            messagebox.showinfo("Success", "Vote counts updated
successfully.")
            # fetch_user_data_melodie_rating()

        except Exception as e:
            messagebox.showerror("Database Error", str(e))
```

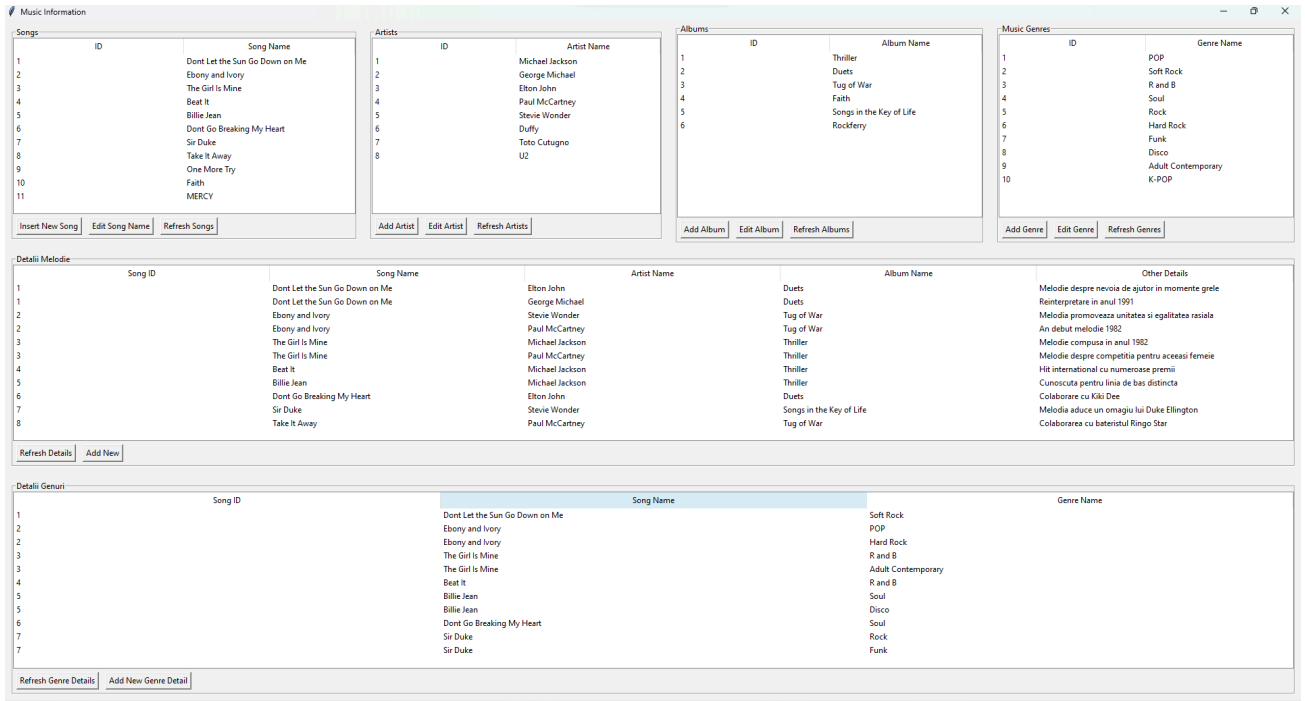
Fereastra pentru ADMIN

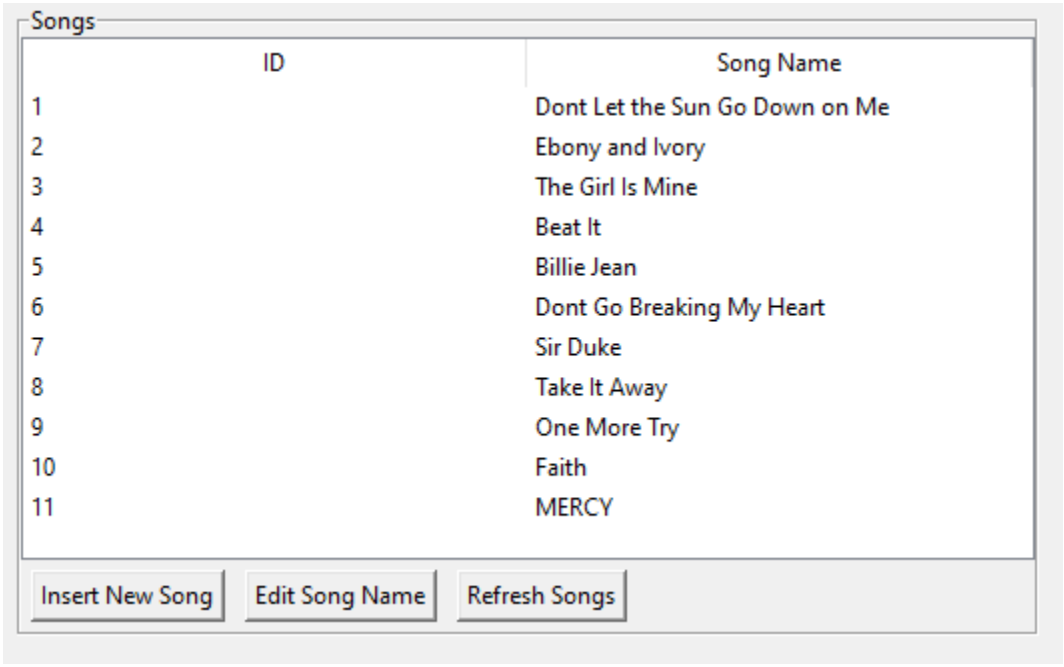
Fereastra de admin prezintă în partea de sus o vizualizare a tuturor userilor existenți în aplicație și în partea de jos a tuturor adminilor. Există un pentru afișare și refresh al userilor, unul pentru afișarea adminilor. Parolele în cazul ferestrei adminului sunt afișate criptat, folosind modulul **hashlib**. Modulul hashlib este o bibliotecă standard în Python care oferă funcții pentru calcularea valorilor hash criptografice. Hashurile criptografice sunt valori unice și fixe generate din datele de intrare, ceea ce le face utile pentru verificarea integrității datelor sau pentru stocarea securizată a parolelor. Unul dintre algoritmii de hashing disponibile în modulul hashlib este SHA-256. SHA-256 (Secure Hash Algorithm 256-bit) este un algoritm de hashing criptografic care produce un hash de 256 de biți (32 de caractere hexazecimale) din datele de intrare. Acest algoritm este considerat destul de sigur și este folosit în numeroase aplicații pentru a asigura integritatea datelor și securitatea acestora. De asemena, există un buton care șterge toți utilizatorii care nu au nicio melodie adăugată în biblioteca lor și mai avem și un buton care deschide fereastra Music Info.

Fereastra de Music Info prezintă posibilitatea vizualizării oricărei tabele (album, artist, melodie, genuri muzicale, detalii genuri și detalii melodie). Prezintă butoane de edit pentru tabelele melodie, artist, album și gen muzical în cazul în care se introduce greșit o înregistrare. De altfel toate tabelele prezintă buton de refresh al vizualizarii datelor.

```
      import hashlib
      . . . . . .
      for user in users:
                        hashed_password =
hashlib.sha256(user[2].encode()).hexdigest()  # Hashing the password
                        tree_users.insert("", tk.END, values=(user[0], user[1],
hashed_password, user[3]))
```




Music Information
Songs
ID
Song Name
1 Dont Let the Sun Go Down on Me
2 Ebony and Ivory
3 The Girl Is Mine
4 Beat It
5 Billie Jean
6 Dont Go Breaking My Heart
7 Sir Duke
8 Take It Away
9 One More Try
10 Faith
11 MERCY
Insert New Song
Edit Song Name
Refresh Songs
Artists
ID
Artist Name
1 Michael Jackson
2 George Michael
3 Elton John
4 Paul McCartney
5 Stevie Wonder
6 Duffy
7 Toto Cutugno
8 U2
Add Artist
Edit Artist
Refresh Artists
Albums
ID
Album Name
1 Thriller
2 Duets
3 Tug of War
4 Faith
5 Songs in the Key of Life
6 Rockferry
Add Album
Edit Album
Refresh Albums
Music Genres
ID
Genre Name
1 POP
2 Soft Rock
3 R and B
4 Soul
5 Rock
6 Hard Rock
7 Funk
8 Disco
9 Adult Contemporary
10 K-POP
Add Genre
Edit Genre
Refresh Genres
Detalii Melodie
Song ID
Song Name
Artist Name
Album Name
Other Details
1 Dont Let the Sun Go Down on Me Elton John Duets Melodie despre nevoia de ajutor in momente grele
1 Dont Let the Sun Go Down on Me George Michael Duets Reinterpretare in anul 1991
2 Ebony and Ivory Stevie Wonder Tug of War Melodia promoveaza unitatea si egalitatea rasiala
2 Ebony and Ivory Paul McCartney Tug of War An debut melodie 1982
3 The Girl Is Mine Michael Jackson Thriller Melodie compusa in anul 1982
3 The Girl Is Mine Paul McCartney Thriller Melodie despre competitia pentru aceeasi femeie
4 Beat It Michael Jackson Thriller Hit international cu numeroase premii
5 Billie Jean Michael Jackson Thriller Cunoscuta pentru linia de bas distincta
6 Dont Go Breaking My Heart Elton John Duets Colaborare cu Kiki Dee
7 Sir Duke Stevie Wonder Songs in the Key of Life Melodia aduce un omagiu lui Duke Ellington
8 Take It Away Paul McCartney Tug of War Colaborarea cu bateristul Ringo Star
Refresh Details
Add New
Detalii Genuri
Song ID
Song Name
Genre Name
1 Dont Let the Sun Go Down on Me Soft Rock
2 Ebony and Ivory POP
2 Ebony and Ivory Hard Rock
3 The Girl Is Mine R and B
3 The Girl Is Mine Adult Contemporary
4 Beat It R and B
5 Billie Jean Soul
5 Billie Jean Disco
6 Dont Go Breaking My Heart Soul
7 Sir Duke Rock
7 Sir Duke Funk
Refresh Genre Details
Add New Genre Detail


Songs
ID
Song Name
1 Dont Let the Sun Go Down on Me
2 Ebony and Ivory
3 The Girl Is Mine
4 Beat It
5 Billie Jean
6 Dont Go Breaking My Heart
7 Sir Duke
8 Take It Away
9 One More Try
10 Faith
11 MERCY
Insert New Song
Edit Song Name
Refresh Songs

Artists

ID	Artist Name
1	Michael Jackson
2	George Michael
3	Elton John
4	Paul McCartney
5	Stevie Wonder
6	Duffy
7	Toto Cutugno
8	U2

Add Artist | Edit Artist | Refresh Artists

Albums

ID	Album Name
1	Thriller
2	Duets
3	Tug of War
4	Faith
5	Songs in the Key of Life
6	Rockferry

Add Album | Edit Album | Refresh Albums

Music Genres

ID	Genre Name
1	POP
2	Soft Rock
3	R and B
4	Soul
5	Rock
6	Hard Rock
7	Funk
8	Disco
9	Adult Contemporary
10	K-POP

Add Genre | Edit Genre | Refresh Genres

Detalii Melodie

Song ID	Song Name	Artist Name	Album Name	Other Details
1	Dont Let the Sun Go Down on Me	Elton John	Duets	Melodie despre nevoia de ajutor in momente grele
1	Dont Let the Sun Go Down on Me	George Michael	Duets	Reinterpretare in anul 1991
2	Ebony and Ivory	Stevie Wonder	Tug of War	Melodia promoveaza unitatea si egalitatea rasiala
2	Ebony and Ivory	Paul McCartney	Tug of War	An debut melodie 1982
3	The Girl Is Mine	Michael Jackson	Thriller	Melodie compusa in anul 1982
3	The Girl Is Mine	Paul McCartney	Thriller	Melodie despre competitia pentru aceeasi femeie
4	Beat It	Michael Jackson	Thriller	Hit international cu numeroase premii
5	Billie Jean	Michael Jackson	Thriller	Cunoscuta pentru linia de bas distincta
6	Dont Go Breaking My Heart	Elton John	Duets	Colaborare cu Kiki Dee
7	Sir Duke	Stevie Wonder	Songs in the Key of Life	Melodia aduce un omagiu lui Duke Ellington
8	Take It Away	Paul McCartney	Tug of War	Colaborarea cu bateristul Ringo Star

Refresh Details | Add New

Detalii Genuri

Song ID	Song Name	Genre Name
1	Dont Let the Sun Go Down on Me	Soft Rock
2	Ebony and Ivory	POP
2	Ebony and Ivory	Hard Rock
3	The Girl Is Mine	R and B
3	The Girl Is Mine	Adult Contemporary
4	Beat It	R and B
5	Billie Jean	Soul
5	Billie Jean	Disco
6	Dont Go Breaking My Heart	Soul
7	Sir Duke	Rock
7	Sir Duke	Funk

Refresh Genre Details | Add New Genre Detail

Ins...

Song Name:

Save

Edit Son...

Song ID:

New Song Name:

Update

Ad...

Artist Name:

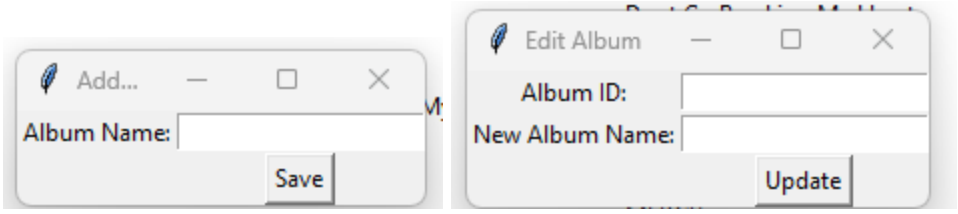
Save

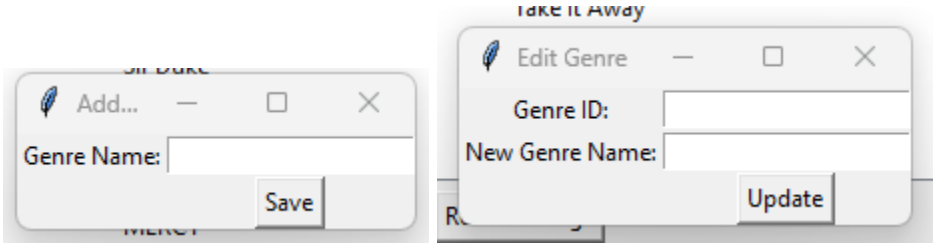
Edit Arti...

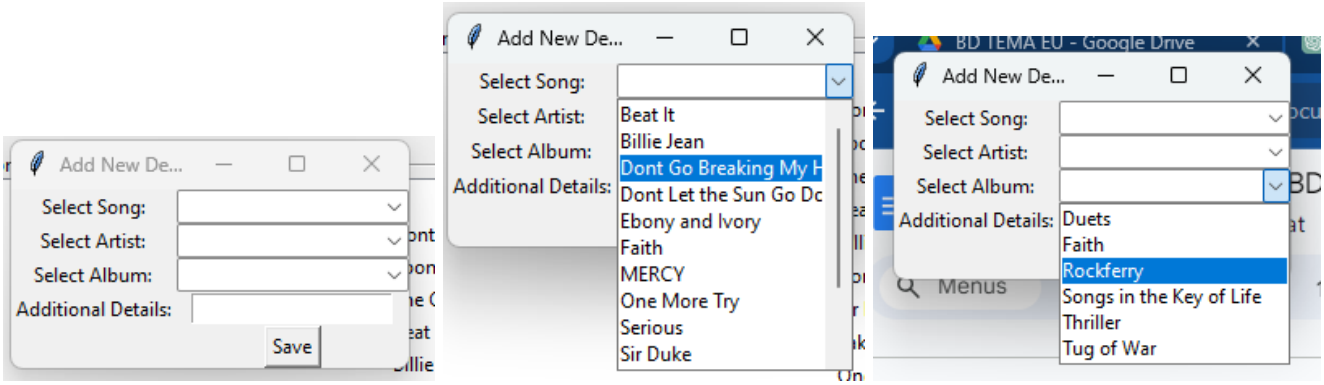
Artist ID:

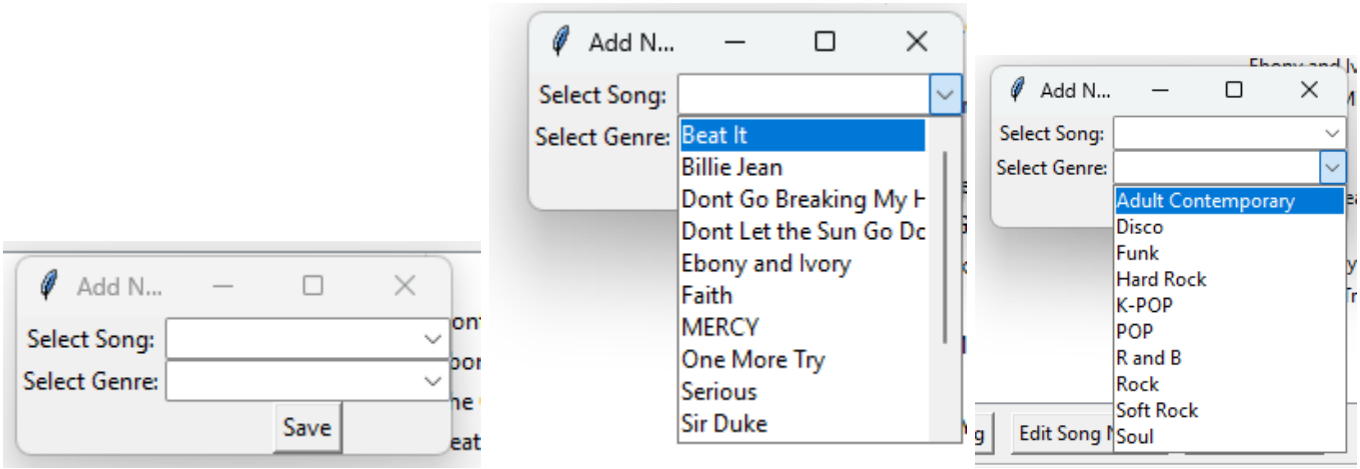
New Artist Name:

Update


Add...
Album Name:
Save
Edit Album
Album ID:
New Album Name:
Update


Add...
Genre Name:
Save
Edit Genre
Genre ID:
New Genre Name:
Update


Add New De...
Select Song:
Select Artist:
Select Album:
Additional Details:
Save
Beat It
Billie Jean
Dont Go Breaking My H
Dont Let the Sun Go Dc
Ebony and Ivory
Faith
MERCY
One More Try
Serious
Sir Duke
Duets
Faith
Rockferry
Songs in the Key of Life
Thriller
Tug of War


Add N...
Select Song:
Select Genre:
Save
Beat It
Billie Jean
Dont Go Breaking My H
Dont Let the Sun Go Dc
Ebony and Ivory
Faith
MERCY
One More Try
Serious
Sir Duke
Adult Contemporary
Disco
Funk
Hard Rock
K-POP
POP
R and B
Rock
Soft Rock
Soul