Universitatea Tehnică „Gheorghe Asachi” din Iași
Facultatea de Automatică și Calculatoare
Domeniul: Calculatoare și Tehnologia Informației

Aplicație simplă pe manageriere a unei baze de date You Tube

“YT database”

Proiect la disciplina

Baze de Date

Student: Chelarașu Elena - Denisa

Anul: 3

Grupa: 1308B

Coordonator: Cristian Buțincu

1. Introducere

Proiectul de față este o imitație a funcționării unei baze de date de tipul celei utilizate de către YouTube. Aceasta se dorește a fi o interfață pentru manipularea bazei de date prin intermediul unui API.

API-ul din proiectul curent permite utilizatorului în primă instanță să aleagă logarea, înregistrarea sau ieșirea din aplicație.

În cazul în caze utilizatorul alege “Sign Up”, acesta este trimis la o interfață pentru a-și introduce datele. După introducerea acestora, utilizatorul poate vedea și baza de date pentru verificare. Introducerea datelor se face cu o interogare INSERT.

Pentru a se loga, utilizatorul poate alege “Go back” din interfața “Sign Up” sau “Login” din interfața de bază.

Din butonul de Login se ajunge într-o interfață cu un meniu care permite manipularea bazei de date, acesta conținând opțiunile: Insert, Update, Delete și Quit. În plus, este valabilă și opțiunea de vizionare a orcărui tabel din baza de date.

În cazul alegerii oricărei modificări, se pot face acestea foarte intuitiv cu ajutorul unui mic meniu în partea stângă a widget-ului care va deschide diferite căsuțe pentru introducerea datelor, în funcție de opțiunea aleasă.

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

Baza de date folosită în această aplicație este SQLPlus. Partea de front-end a aplicației a fost realizată cu ajutrul librăriei tkinter din python. Am ales tkinter , fără ajutorul la pygubu, deoarece am putut să mă desfășor mai bine în acest mediu, fiindu-mi cunoscut dinainte (și neștiind Web Development).

Window-ul principal al aplicației este realizat în clasa PyApp() și acesta este cel care funcționează în mainloop(). Cu ajutorul funcțiilor se face și deschiderea celor două window-uri, Sign Up și Login.

Pentru criptarea parolei, s-au implementat metodele encrypt() și decrypt() în fișierul Crypt, dar nu a fost implementată criptarea parolei.

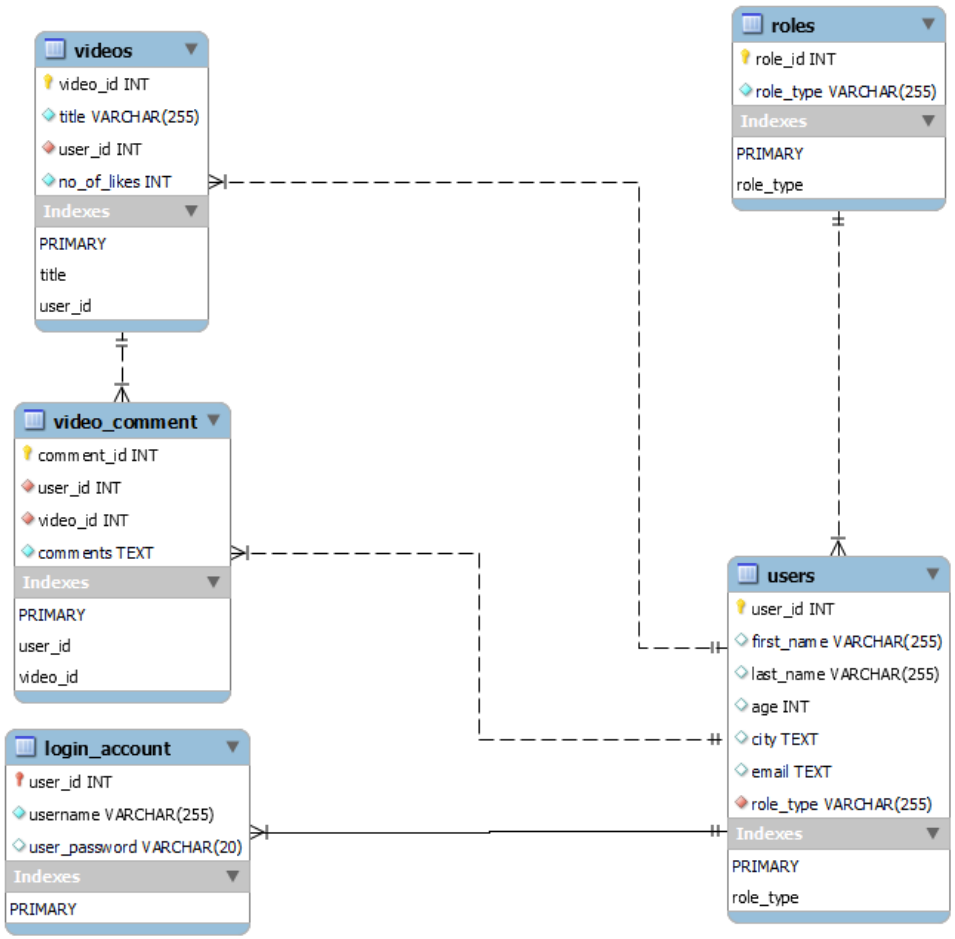
Scripturile pentru funcțiile generale de Insert, Delete și Update au fost realizate în deleteFile.py, modify.py și updateFile.py, cu menționarea că Update, deși implementat pentru tabela users, nu realizează update-ul tabelei.

3. Descrierea constrângerilor folosite

În tabelă avem relații

- one-to-many: un rol poate fi asignat mai multor utilizatori
- one-to-one: un utilizator poate avea un singur username

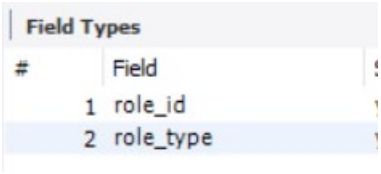
Constrângerile, atributele și cheile tabelelor sunt reprezentate în figura următoare



Constrângerile, atributele și cheile tabelelor

Aplicația conține cinci tabele, după cum urmează:

- **roles**



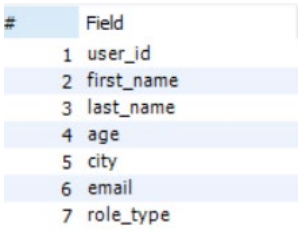
Field Types	
#	Field
1	role_id
2	role_type

Primary Key: role_id

Constrângere Unique: role_type (un rol are un nume unic)

Constrângere Not Null: role_id, role_type

- **users**



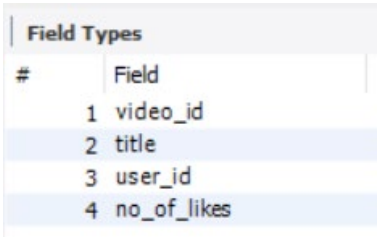
#	Field
1	user_id
2	first_name
3	last_name
4	age
5	city
6	email
7	role_type

Primary Key: user_id

Foreign Key: roles(role_type) (fiecare utilizator are un rol)

Constrângere Not Null: user_id, first_name, last_name, age, city, email, role_type

- **videos**



Field Types	
#	Field
1	video_id
2	title
3	user_id
4	no_of_likes

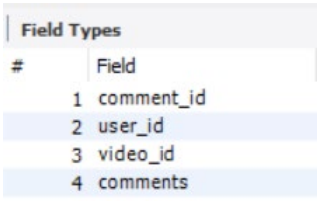
Primary Key: video_id

Foreign Key: users(user_id) (un video poate fi încărcat de către un utilizator)

Constrângere Unique: title (un video are un titlu unic)

Constrângere Not Null: video_id, title, user_id, no_of_likes

- **video_comment**



Primary Key: comment_id

Foreign Key: users(user_id) (un comentariu este scris de un utilizator)

Foreign Key: videos(video_id) (un comentariu este pus la un video)

Constrângere Not Null: comment_id, user_id, video_id, comments

- **login_account**

Primary Key: user_id

Constrângere Not Null: user_id, username, user_password

Constrângere Check: password_check (parola trebuie să aibă între 1 și 16 caractere)

4. Descrierea modalității de conectare la baza de date

Conectarea la baza de date a fost făcută prin intermediul librăriei sqlite3. Aceasta este o librărie care aparține limbajului C și care oferă o manipulare mai ușoară a unei baze de date. Nu necesită un proces server separat și accesează baza de date printr-o variantă nestandardizată a SQL. Aceasta este perfectă pentru proiectele de dimensiuni mici, precum acesta.

Conectarea la baza de date se face in crearea unui obiect de tipul Connection. In cazul in care nu exista baza de date la care se apelează constructorul, aceasta este creată automat.

După ce aceasta a fost creată, se instanțiază un obiect de tip Cursor() care ne va permite utilizarea funcției execute(). Cu ajutorul acestei funcții se pot executa interogări SQL și astfel putem manipula baza de date. De fiecare dată după ce am terminat lucrul cu elementul de tip Cursor(), trebuie făcut un commit() și închisă baza de date, ca în codul exemplu din imagine.

```
def video_comment_delete_submit():
    conn = sqlite3.connect('yt_database.db')

    c = conn.cursor()

    # insert into table
    c.execute("DELETE FROM video_comment WHERE oid=(:comment_id)",
              {
                  'comment_id': vcd_comment_id.get()
              })

    # clear the txt boxes
    vcd_comment_id.delete(0, END)

    conn.commit()

    conn.close()
```

Deschiderea, manipularea și închiderea bazei de date

Datele modificate și precedate de funcția commit() la final sunt pastrate și în următoarele deschideri ale aplicației. În cazul aplicației dezvoltate de mine, această proprietate nu este valabilă deoarece realizez o restartare a bazei de date la rularea programului. Acesta este inițializat cu cele 5 tabele și multiple înregistrări, exceptând tabela “roles” care conține doar cele două tipuri de roluri.

```
c.executescript("""
    INSERT INTO users VALUES (1,  'Paraschiva', 'Pristanda',    18, 'Iasi',        'paraschiva@yahoo.com'   , 'admin');
    INSERT INTO users VALUES (2,  'Paula',      'Horea',        34, 'Galati',      'daniel@yahoo.com'       , 'user');
    INSERT INTO users VALUES (3,  'Demetra',    'Daniel',       21, 'Bacau',       'paraschiva@yahoo.com'   , 'user');
    INSERT INTO users VALUES (4,  'Stefania',   'Petru',        9,  'Arad',        'petru@yahoo.com'        , 'user');
    INSERT INTO users VALUES (5,  'Rares',      'Veronica',     37, 'Olt',         'veronica@yahoo.com'     , 'user');
    INSERT INTO users VALUES (6,  'Sorin',      'Antalupei',    14, 'Cluj',        'antalupei@yahoo.com'    , 'user');
    INSERT INTO users VALUES (7,  'Ava',        'Bishop',       18, 'Philadelphia', 'bishop@yahoo.com'       , 'user');
    INSERT INTO users VALUES (8,  'River',      'Austin',       34, 'Chongqing',   'austin@yahoo.com'       , 'user');
    INSERT INTO users VALUES (9,  'Alex',       'Bailey',       21, 'Miami',       'bailey@yahoo.com'       , 'user');
    INSERT INTO users VALUES (10, 'Arianne',    'Coombes',      9,  'Tokyo',       'coombes@yahoo.com'      , 'user');
    INSERT INTO users VALUES (11, 'Maximus',    'Bean',         37, 'Gaborone',    'bean@yahoo.com'         , 'user');
    INSERT INTO users VALUES (12, 'Anish',      'Farrington',   14, 'Nashville',   'farrington@yahoo.com'   , 'user');
    INSERT INTO users VALUES (13, 'Rico',       'Beech',        18, 'Vienna',      'beech@yahoo.com'        , 'user');
    INSERT INTO users VALUES (14, 'Aysha',      'Orr',          34, 'Bratislava',  'orr@yahoo.com'          , 'user');
```

Scriptul de inserare inițială în tabela users

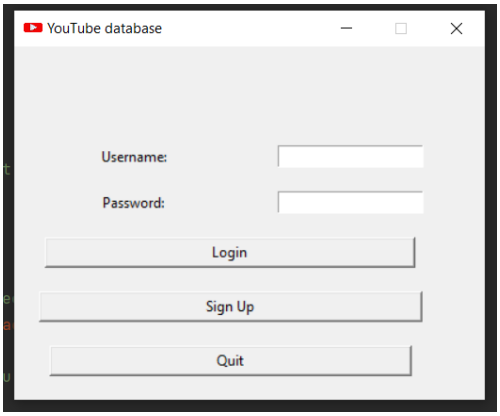
În proiectul de față, eu fac trei operații mari pentru conectarea la baza de date: execut trei scripturi. Pentru aceste scripturi am creat funcții diferite care se află în fișierul “table_management.py”. După conectarea primului window la baza de date, realizez o operațiunie de “DROP” pentru toate tabelele realizate anterior. După aceasta, se execută scriptul de creare a tabelelor și cel e inserare a datelor în tabelele create.

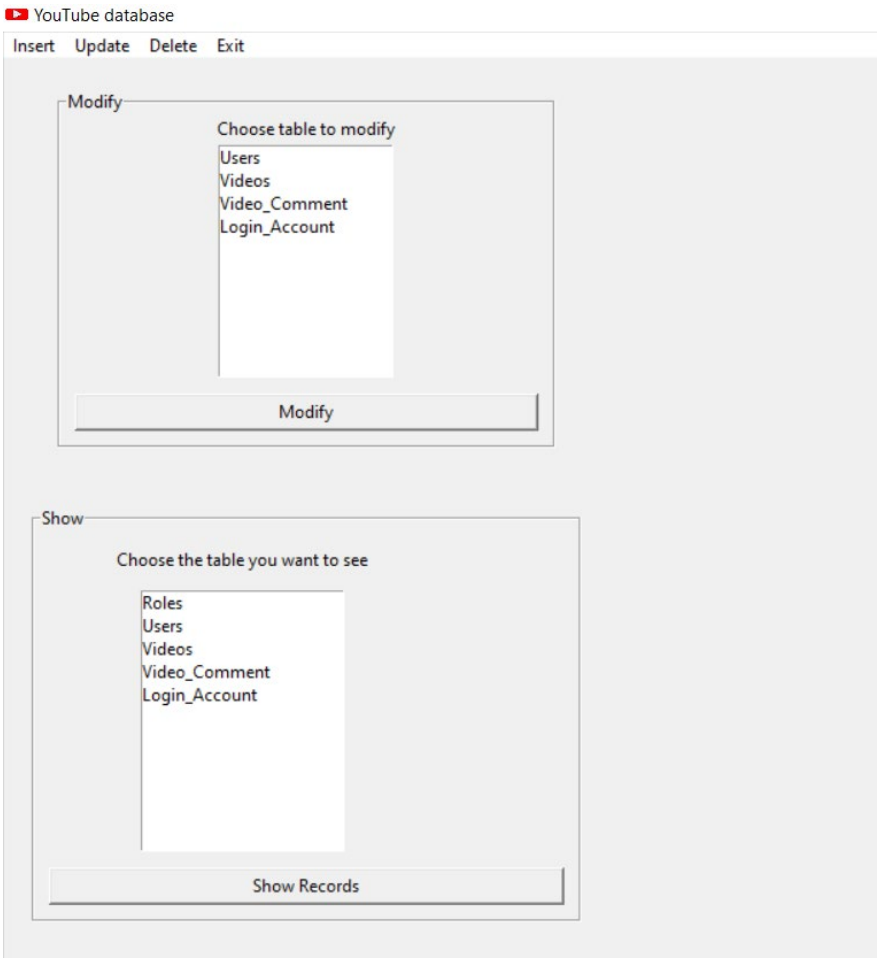
```python
def initGUI(self):
    # connect to db
    conn = sqlite3.connect('yt_database.db')

    # create cursor instance
    c = conn.cursor()

    tm.dropTables()

    # create the tables
    tm.createTables()

    # insert into tables
    tm.insertIntoTables()
```

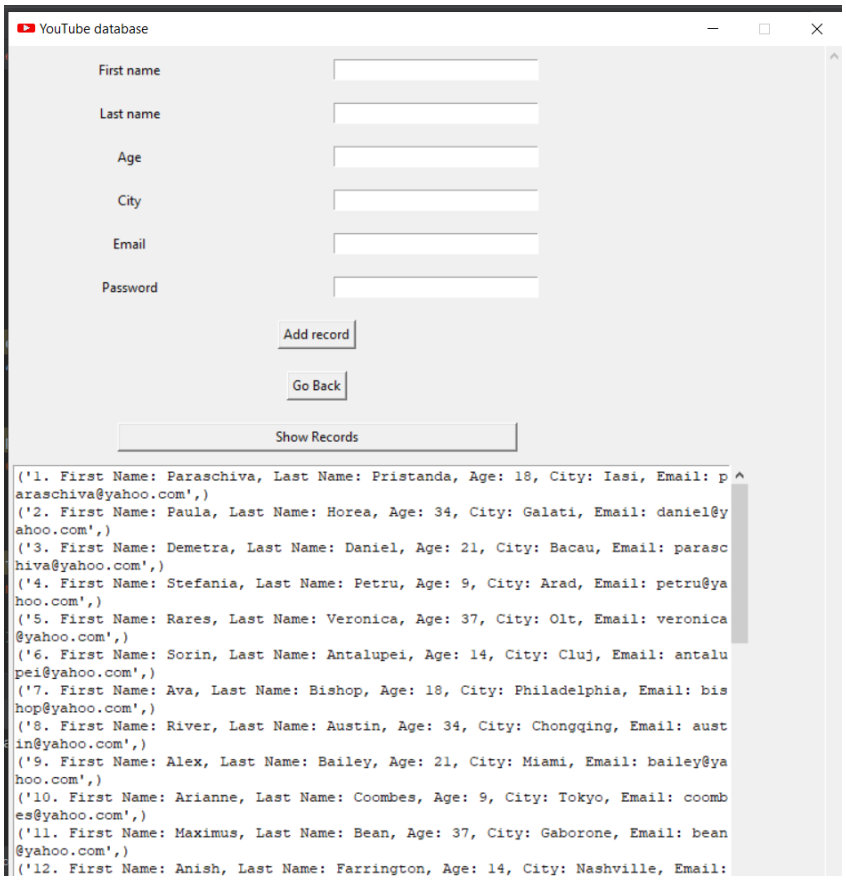
Rularea celor trei scripturi

5. Capturi de ecran concludente

Interfața grafică a proiectului







Exemple de cod

```python
# for the menu form
def openWinDb():
    # connect to db
    conn = sqlite3.connect('yt_database.db')

    # create cursor instance
    c = conn.cursor()

    global bottom_frame, up_frame

    top1 = Toplevel()
    top1.title('YouTube database')
    top1.iconbitmap('Images/ytDatabaseIcon.ico')

    width = 1200
    height = 700

    screen_width = top1.winfo_screenwidth()
    screen_height = top1.winfo_screenheight()

    top1.geometry(f'{screen_width}x{screen_height}')
```

```python
def __init__(self):
    super().__init__()

    self.resizable(0, 0)
    self.geometry("400x300")
    self.title('YouTube database')
    self.iconbitmap('Images/ytDatabaseIcon.ico')
    self.eval('tk::PlaceWindow . center')

    self.width = 400
    self.height = 300

    scr_width = self.winfo_screenwidth()
    scr_height = self.winfo_screenheight()

    x = (scr_width / 2) - (self.width / 2)
    y = (scr_height / 2) - (self.height / 2)

    self.geometry(f'{self.width}x{self.height}+{int(x)}+{int(y)}')

    self.initGUI()
```

```python
def login_submit():
    conn = sqlite3.connect('yt_database.db')

    c = conn.cursor()

    # insert into table
    c.execute("INSERT INTO login_account(username, user_password) "
              "VALUES (:username, :user_password)",
              {
                  'username': ls_username.get(),
                  'user_password': ls_user_password.get()
              })

    # clear the txt boxes
    ls_username.delete(0, END)
    ls_user_password.delete(0, END)

    conn.commit()

    conn.close()
```