Universitatea Tehnica “Gheorghe Asachi” din Iasi

Facultatea de Automatica si Calculatoare

Domeniul: Calculatoare si tehnologia informatiei

MyAnimeList

Proiect la disciplina
Baze de Date

Grupa: 1309A

Echipa: Cojocaru Georgiana, Lupu Andra

Prof. Coordonator: Catalin Mironeanu

Iasi 2023

Descriere Proiect

Aplicatia noastra este creata cu scopul de a ajuta la tinerea evidentei si organizarea anime-urilor vizionate si manga-urilor citite de catre un utilizator.

Pentru un anime adaugat, utilizatorul incepe prin a introduce titlul. Acesta are optiunea de a alege dintre autorii/studiourile deja existente in baza de date sau de a introduce noi inregistrari. De asemenea, poate fi selectat genul anime-ului si progresul utilizatorului (watching, completed, on hold, dropped, plan to watch). Dupa introducerea numarului de episode vizualizate/totale, user-ul poate oferi o nota initiala de la 1 la 10. Asemanator se procedeaza si cu manga.

Baza noastra de date contine in total 6 tabele:

- Anime
- Anime_List
- Author
- Studio
- Manga
- Manga_List

Tehnologii folosite

Pentru partea de back-end vom folosi o baza de date SQL. Pentru crearea diagramelor am folosit aplicatia SQl Developer DataModeler de la Oracle.

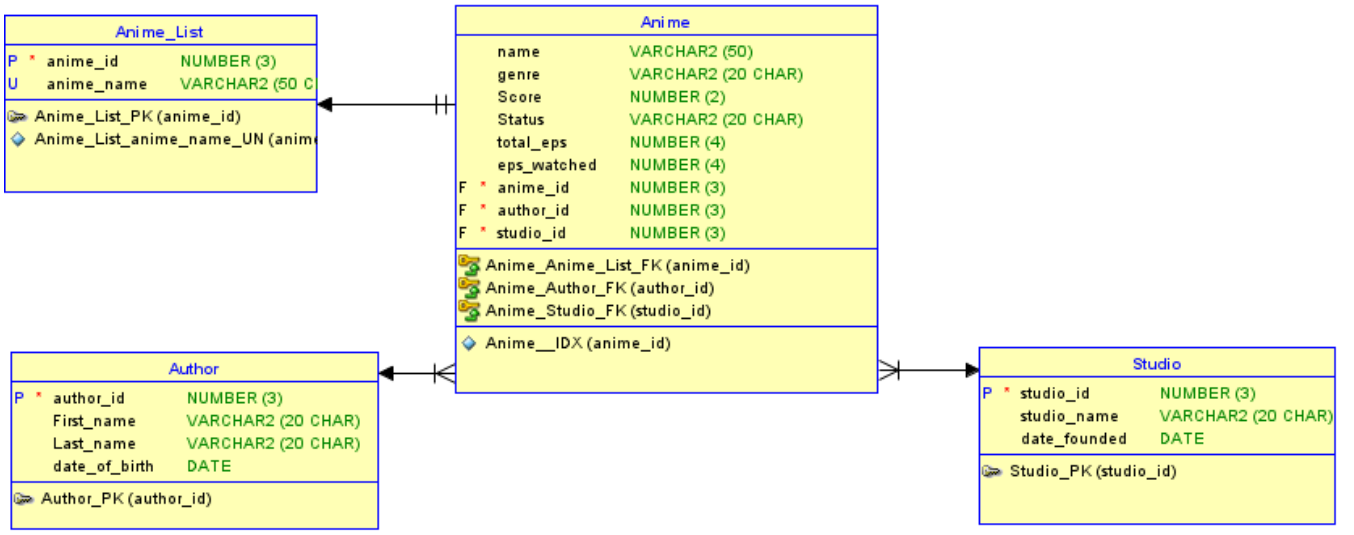
Pentru partea de front-end am folosit PyQt5 impreuna cu tool-ul Qt Designer pentru a usura task-ul crearii interfetei iar versiunea de python folosita este 3.9. Pentru manipularea bazei de date am folosit modulul Sqlite3.

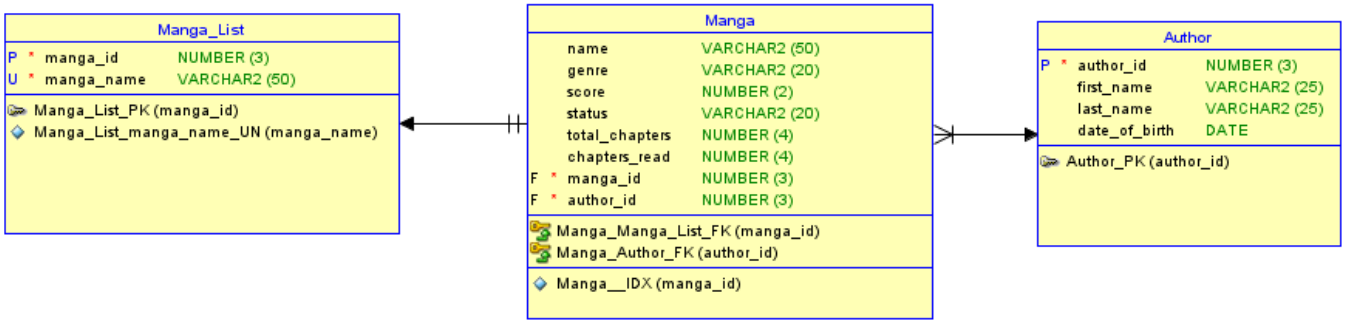
SQLite este o mică bibliotecă C care implementează un motor de baze de date SQL încapsulat si oferă posibilitatea de a-l introduce în diverse sisteme și necesită zero-configurare. SQLite este diferit de majoritatea altor motoare de baze de date SQL prin aceea că a fost proiectat pentru a fi simplu:

- Simplu de administrat
- Simplu de folosit
- Simplu de a fi incapsulat intr-un program mai mare
- Simplu de intretinut si setat

Iasi 2023

Structura si relatiile tabelelor





Descriere constrangeri si relatii

Am folosit constrangeri de integritate de tip unique si not null.Pe cea dintai am folosit-o in tabelele Anime/Manga_List pentru a ne asigura ca nu se va repata un anime/manga. Pe cea de a doua constrangere am folosit-o in 4 cazuri, in tabelele Anime/Manga/Author/Studio la total_eps/total_chapters/date_of_birth/date_founded deoarece constrangerea not null ne asigura faptul ca o coloana specificata nu poate primi valoarea null.

Constrangerile de integritate referentiala folosite sunt primary key si foreign key. Cea dintai constrangere combina alte doua, unique si not null. Am folosit aceasta constrangere in 4 tabele Anime_List/Manga_List/Studio/Author la coloana id-urilor din fiecare tabel mentionat pentru a asigura unicitate,pentru a nu avea valoarea null, dar si pentru a face legaturi cu alte tabele. Constrangerea foreign key am folosit-o deoarece asigura ca valorile unei coloane corespund unor valori PK din alte tabele, mai exact pentru a realiza legatura one-to-many.

In cazul nostru am folosit FK in doua tabele Anime/Manga la coloanele anime_id/studio_id/author_id/manga_id doarece acestea fac legatura cu PK-urile(coloanele id) din tabelele Anime_List/Manga_List/Studio/Author.

Relatia one-to-many se regaseste intre tabelele Author - Anime, Author-Manga, Studio-Anime. Aceasta relatie apare atunci cand o inregistrare dintr-un tabel se regaseste la una sau mai multe inregistrari din tabelul 2. In cazul nostru un autor poate scrie mai multe manga/anime-uri, un studio de animatie poate realiza mai multe anime-uri, ceea ce rezulta in faptul ca unele anime-uri vor avea acelasi autor/studio.

Relatia one-to-one se regaseste intre tabelele Anime_List - Anime, Manga_List - Manga. Aceaste relatie apare atunci cand o inregistrare dintr-un tabel se regaseste la o singura inregistrare din tabelul 2.In cazul nostru fiecare anime, din Anime_List, are un anime_id unic care se regaseste doar o singura data in tabelul Anime, acelazi caz pentru Manga_List - Manga.

Modalitate de conectare

Pentru conectarea la baza de date, se apeleaza functia sqlite3.connect()

```
import sqlite3
con = sqlite3.connect("tutorial.db")
```

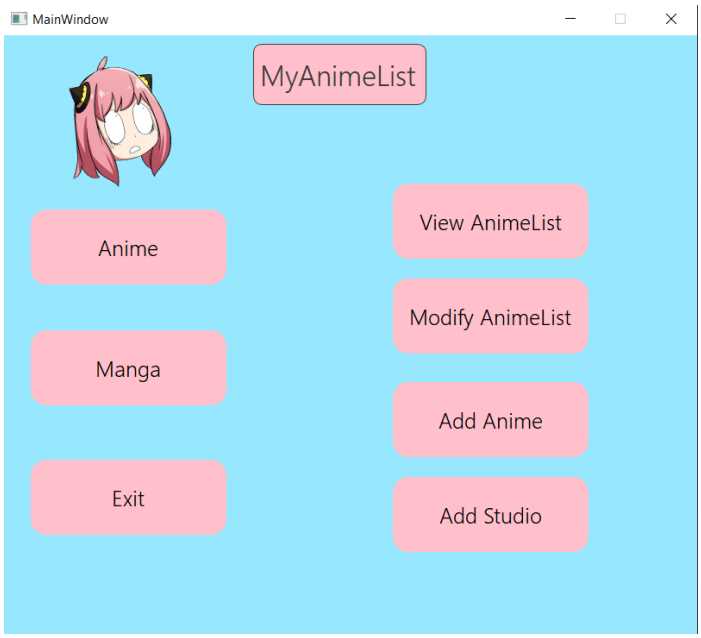
Pentru a putea executa instructiuni SQL si pentru a extrage infromatii vom avea nevoie de un database cursor. Acesta este obtinut prin apelarea functiei con.cursor()

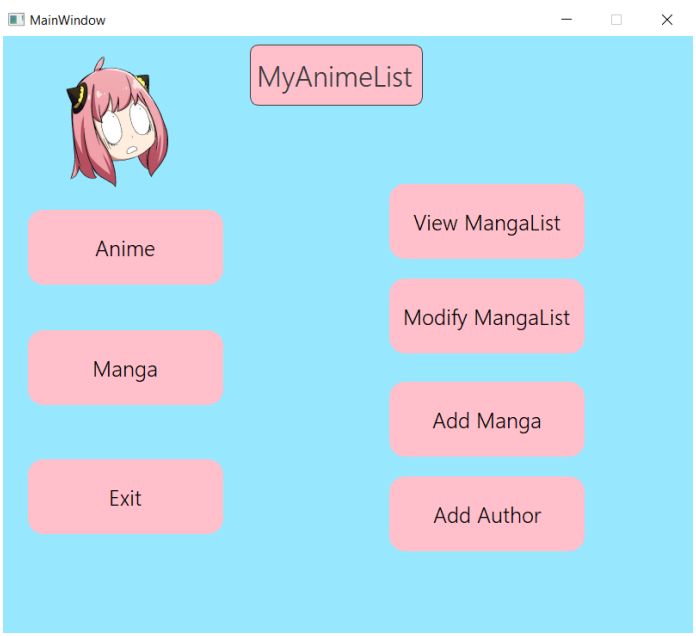
```
cur = con.cursor()
```

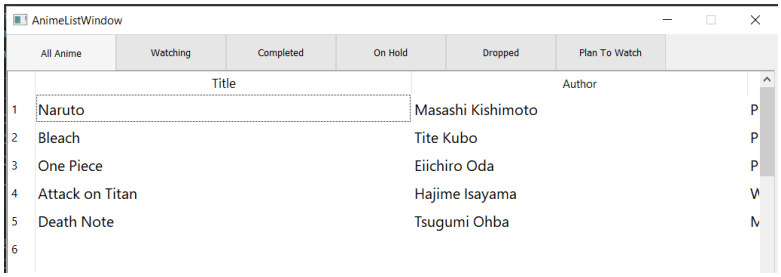
Odata indepliniti acesti pasi, instructiunile pot fi executate prin simplul apel al functiei con.execute()

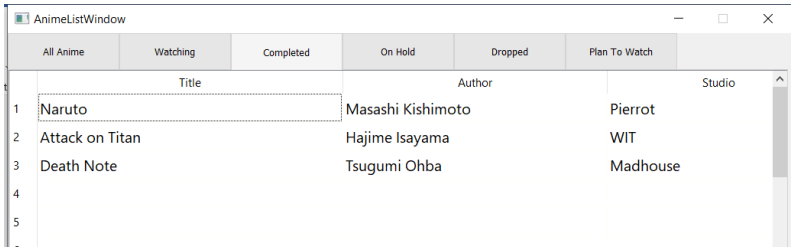
```
res = cur.execute("SELECT name FROM sqlite_master")
res.fetchone()
```

Functionalitatea aplicatiei









Vizualizare AnimeList

pe Interfata

Selectare anime-uri din baza de date si pregatirea celulelor din tabelul interfetei

```
# ALL ANIME
cursor.execute("SELECT * from Anime ")
results = cursor.fetchall()
#print(results)
contor_linii = 0
contor_coloane = 0
for r in results:
    #print(contor_linii)
    titlu = self.ui.tableWidget.item(contor_linii, contor_coloane)
    autor = self.ui.tableWidget.item(contor_linii,contor_coloane+1)
    studio = self.ui.tableWidget.item(contor_linii,contor_coloane+2)
    genre = self.ui.tableWidget.item(contor_linii,contor_coloane+3)
    status = self.ui.tableWidget.item(contor_linii,contor_coloane+4)
    score = self.ui.tableWidget.item(contor_linii,contor_coloane+5)
    progres = self.ui.tableWidget.item(contor_linii,contor_coloane+6)
```

Cautarea autorului in functie de author_id

```
id_autor = str(r[7])
buff_autor = "Select first_name,last_name from Author where author_id = " + str(id_autor)
#print(buff_autor)
cursor.execute(buff_autor)
rez = cursor.fetchall()
```

Cautarea studioului in functie de studio_id

```
id_studio = str(r[8])
buff_studio = "Select studio_name from Studio where studio_id = " + id_studio
cursor.execute(buff_studio)
rez_studio = cursor.fetchall()
```

Afisare pe interfata si trecerea la urmatoarea inregistrare

```
    autor.setText(str(rez[0][0]) + " " + str(rez[0][1]))
    titlu.setText(str(r[0]))
    studio.setText(str(rez_studio[0][0]))
    genre.setText(str(r[1]))
    score.setText(str(r[2]))
    progres.setText(str(r[5]) + "/" + str(r[4]))
    status.setText(str(r[3]))

    contor_linii = contor_linii + 1
```

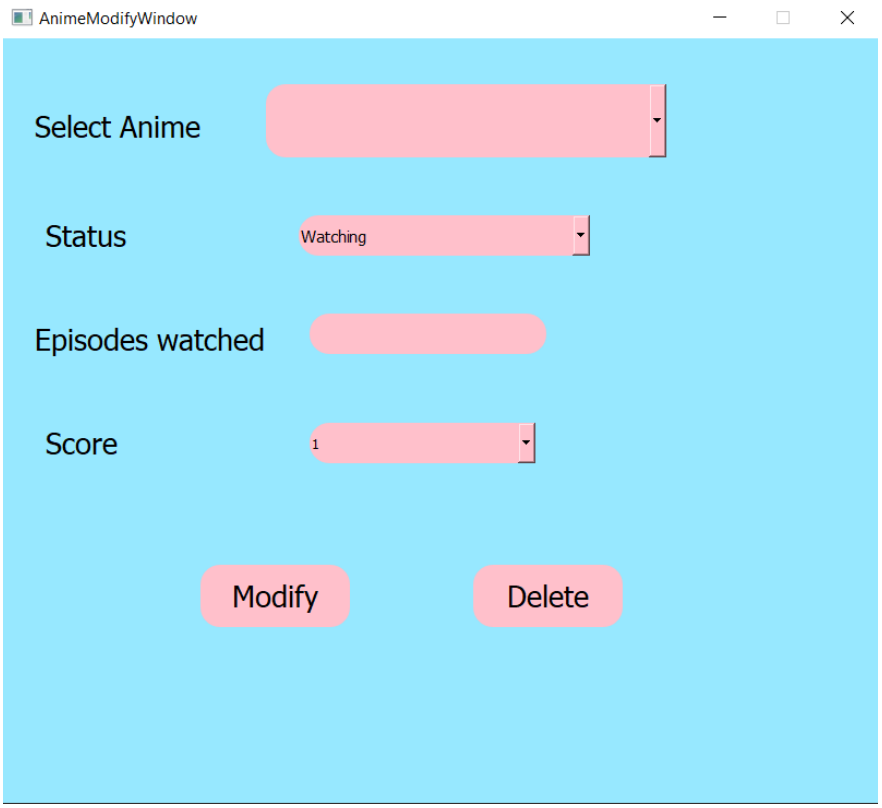
Stergere inregistrare in functie de numele dat de pe interfata

(se cauta anime-idul corespunzator)

```
def delete(self):
    mn=self.comboBox_2.currentText()
    con=sl.connect('TEMA.db')
    cursor=con.cursor()
    cursor.execute('Select anime_id FROM Anime_List WHERE anime_name=?',(mn,))
    results=cursor.fetchall()
    mid=results[0][0]
    print(mid)
    cursor.execute('DELETE FROM Anime_List WHERE anime_id=?',(mid,))
    cursor.execute('DELETE FROM Anime WHERE anime_id=?', (mid,))
    con.commit()
```

Iasi 2023

Actualizare




```
def modify(self):
    mn = self.comboBox_2.currentText()
    con = sl.connect('TEMA.db')
    cursor = con.cursor()
    cursor.execute('Select anime_id FROM Anime_List WHERE anime_name=?', (mn,))
    results = cursor.fetchall()
    mid = results[0][0]
    print(mid)
    status = self.comboBox_5.currentText()
    score = self.comboBox.currentText()
    chr = self.lineEdit_5.text()
    print(status)
    cursor.execute('UPDATE Anime SET status=? WHERE anime_id=?', (status, mid,))
    con.commit()
    cursor.execute('UPDATE Anime SET score=? WHERE anime_id=?', (score, mid,))
    con.commit()
    cursor.execute('UPDATE Anime SET eps_watched=? WHERE anime_id=?', (chr, mid,))
    con.commit()
```

Inserare


```
def addAnime(self):
    con = sl.connect('TEMA.DB')
    cursor = con.cursor()
    cursor.execute("SELECT anime_id  FROM Anime_List")
    results = cursor.fetchall()
    max = results[0][0]
    print(max)
    for r in results:
        if (max < r[0]):
            max = r[0]
    print(max)
    max = max + 1       # GENERARE ANIME_ID

    sql = 'INSERT INTO Anime_List (anime_id,anime_name) values( ?, ?)' # INSERARE IN TABELA ANIME_LIST
    data = [
        (max, self.titleEdit.text())
    ]
    with con:
        con.executemany(sql, data)
```

```
ln = self.authorComboBox.currentText()
print(ln)
cursor2 = con.cursor()
cursor2.execute("SELECT author_id  FROM Author WHERE last_name=?", (ln,))
aid = cursor2.fetchall()
aidd = aid[0][0]
print(aidd)

sn=self.studioComboBox.currentText()
cursor3 = con.cursor()
cursor3.execute("SELECT studio_id  FROM Studio WHERE studio_name=?", (sn,))
print(sn)
sid = cursor3.fetchall()
sidd = sid[0][0]
print(sidd)

 # EXTRAGERE AUTHOR_ID IN FUNCTIE DE NUMELE DAT
 ln = self.authorComboBox.currentText()
 print(ln)
 cursor2 = con.cursor()
 cursor2.execute("SELECT author_id  FROM Author WHERE last_name=?", (ln,))
 aid = cursor2.fetchall()
 aidd = aid[0][0]
 print(aidd)

 # EXTRAGERE STUDIO_ID IN FUNCTIE DE NUMELE DAT
 sn=self.studioComboBox.currentText()
 cursor3 = con.cursor()
 cursor3.execute("SELECT studio_id  FROM Studio WHERE studio_name=?", (sn,))
 print(sn)
 sid = cursor3.fetchall()
 sidd = sid[0][0]
 print(sidd)
```

```
sql2 = 'INSERT INTO Anime (name,genre,score,status,total_eps,eps_watched,anime_id,author_id,studio_id) values(?,?,?,?,?,?,?,?,?)'
data2 = [
    (self.titleEdit.text(), self.genreComboBox.currentText(), self.scoreComboBox.currentText(),
     self.statusComboBox.currentText()
     , self.totalNrEpEdit.text(), self.lineEdit_5.text(), max, aidd,sidd)
]
with con:
    con.executemany(sql2, data2)
```

Impartire task-uri

Cojocaru:

- realizare interfata
- citire din baza de date si afisarea pe interfata

Lupu:

- Realizare baza de date
- Inserare in baza de date de pe interfata