DOCUMENTATIE
INTERFATA CINEMA

Limbaje: Python, MySQL

GUI toolkit: CustomTkinter

Delcea Andrei-Iacob

434D

I. INTRODUCERE

Am avut de alcatuit o baza de date formata dintr-un table de filme, unul de locatii si unul care sa faca legatura intre ele. Acest tabel l-am luat ca fiind unul de rulari ale filmelor, avand filmul si locatia in care va fi rulat.

Am creat baza de date in MySQL Workbench, am creat niste atribute pentru fiecare tabel si am populat apoi fiecare tabel cu date relevante.

Dupa obtinerea bazei de date, am inceput sa creez clase pentru fiecare tabel in parte, pentru a facilita incarcarea in program a datelor din baza de date, modificarea lor, inserarea sau stergerea lor.

Pe urma am inceput sa creez aplicatia propriu zisa sub forma unei clase de obiecte. I-am creat un o functie de start care initiaza programul intr-un cadru de inceput, avand toate cele trei tabele afisate intr-un frame. De aici selectia oricarei tabele duce la functiile specifice pe care le-am dezvoltat, adica vizualizare, modificare, inserare si stergere. Functiile in sine afiseaza pe ecran tabelul respectiv, si eventual intrari necesare pentru a face prelucrarea dorita.

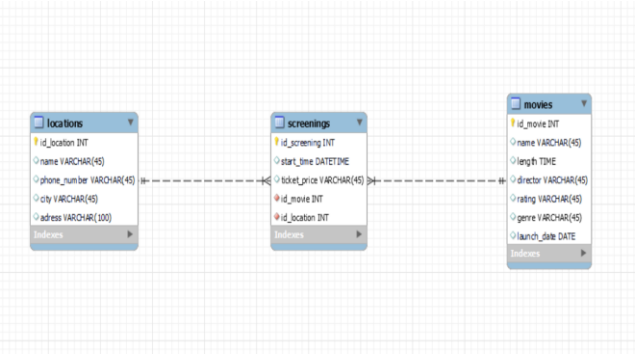
In final, am adaugat butoane de parcurgere inapoi a paginilor si am finisat elementul estetic.

II. TEHNOLOGII FOLOSITE

A. Python

Este un limbaj multi-paradigmă, concentrându-se asupra programării imperative, orientate pe obiecte și funcționale, ceea ce permite o flexibilitate mai mare în scrierea aplicațiilor. *Wikipedia / Python.* f*rom www.wikipedia.ro*

Am folosit Python datorita este un limbaj foarte versatil, are foarte multe biblioteci usor de folosit si are o complexitate mai mica in lucrul cu bazele de date.



B. MySQL

MySQL este un sistem de gestionare a bazelor de date relaționale open source care este utilizat în principal pentru aplicațiile online. MySQL poate crea și gestiona baze de date foarte utile (cum ar fi informații despre angajați, inventar și multe altele), la fel ca alte sisteme, cum ar fi popularul Microsoft Access. În timp ce Microsoft Access, MySQL și alte sisteme de gestionare a bazelor de date servesc scopuri similare (de a găzdui datele), utilizarea diferă foarte mult. *Nav Communications / Ce este MySQL?. from* [*www.nav.ro/blog/ce-este-mysql/*](www.nav.ro/blog/ce-este-mysql/)

Am folosit MySQL datorita preferintei mele legata de sistemele relationale de gestiune a datelor. De altfel, deoarece este un limbaj standard, rareori intervin modificari asupra lui in timp.

C. CustomTkinter

CustomTkinter este o prelungire a plugin-ului Tkinter, cu elemente suplimentare de UI.

Este un tool simplist, dar eficient pentru crearea de interfete in python. Are o gama variata de widget-uri cum ar fii tabelele, scrollbar-urile si butoanele. Pe langa asta are un aspect elegant si modern, ceea ce m-a atras in folosirea lui.

III. FUNCTIONALITATI

A. Incarcarea, modificarea, inserarea si stergerea din baza de date

Pentru aceste functii am creat clase de obiecte pentru fiecare tabela in parte, cu scopul de a prelua datele din baza de date, a le incarca local in aplicatia Python, si apoi a le reincarca in baza de date.

In aceste clase exista o functie pentru fiecare dintre aceste operatii, alcatuite din conectarea la baza de date, aplicarea unor query-uri si comiterea unei tranzactii, si inchiderea conexiunii cu baza de date.

```
class Location:

    def __init__(self, id_location=0, name="", phone_number="", city="",adress=""):
        self.id_location = id_location
        self.name = name
        self.phone_number = phone_number
        self.city = city
        self.adress = adress
```

```
def load_location(self, id_location):
    self.connection = mysql.connector.connect(host="localhost",user="root",passwd="a2mEfc@#uo%*cQ",d
    self.cursor = self.connection.cursor()
    self.cursor.execute("""
    SELECT * FROM locations
    WHERE id_location = {}
    """.format(id_location))

    results = self.cursor.fetchone()

    self.id_location = id_location
    self.name = results[1]
    self.phone_number = results[2]
    self.city = results[3]
    self.adress = results[4]


    self.connection.close()
```

B. Afisarea tabelelor

Sunt afisate trei butoane pe ecran, fiecare corespunzand unui tabel. La apasarea fiecarui buton vor fii afisate functionalitati legate de tabelul specific.

C. Selectarea functionalitatii

Sunt afisate patru butoane pe ecran in cadrul fiecarui tabel pentru vizualizarea, modificarea, inserarea sau stergerea din tabel.

D. Functionalitatea de viuzalizare

In functie de tabelul la care suntem, este construit pe ecran un tabel cu capetele de coloane specifice, dupa care sunt incarcate toate instantele din tabel in program pentru a popula tabelul.

E. Functionalitatea de modificare

Se face afisarea tabelului, dupa care este afisata o intrare pentru adaugarea unui id, si datelor corespunzatoare care se doresc a fi modificate. La apasarea butonului de modificare, se deschide o functie care verifica mai intai daca modificarile pot fi facute.

In cazul in care id-ul cautat nu a fost gasit, se afiseaza un mesaj de eroare. In cazul in care doar unele intrari au fost completate, doar acelea for afecta modificarea instantei din tabela. Daca nu a fost completat absolut nimic, solicitarea va fii ignorata.

La final se va folosi functia specifica de modificare a tabelei, si se va afisa un mesaj de confirmare.

F. Functionalitatea de inserare

Se face din nou afisarea tabelului, si se afiseaza alta intrare pe ecran, de data asta id-ul nu mai poate fi introdus la intrare. La apasarea butonului se verifica

```
def update_location(self):
    self.connection = mysql.connector.connect(host="localhost",user="root",passwd="a2mEfc@#uo%*cQ",database="cinema
    self.cursor = self.connection.cursor()
    self.cursor.execute("""
    UPDATE locations SET name = '{}',phone_number = '{}',city = '{}',adress = '{}' WHERE id_location = {}
    """.format(self.name,self.phone_number,self.city,self.adress,self.id_location))

    self.connection.commit()
    self.connection.close()
```

daca toate campurile atributelor au fost completate, si daca da, se va afisa un mesaj de confirmare si in baza de date se va incarca o noua instanta.

G. Functionalitatea de stergere

Se afiseaza tabelul, si o intrare pentru a completa id-ul instantei care se doreste a fi stearsa. Id-ul este cautat in baza de date, daca este gasit, instanta cu acel id va fii stearsa din tabel si se va afisa un mesaj de confirmare. Daca nu, va fii afisat un mesaj de eroare.

H. Functionalitatea de intoarcere la pagina precedenta

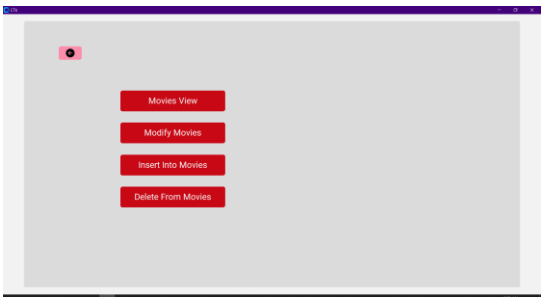
Pentru aceasta functionalitate am facut ca scena si fereastra curenta, dar si eventuale variabile necesare ale aplicatiei sa fie mereu transmisa intre paginile ei. Atunci cand butonul pentru pagina anterioara este apasat, se activeaza functia pentru pagina anterioara, avand toate infromatiile necesare pentru a continua de unde a ramas.

IV. Mai multe imagini cu functionalitatile interfetei

```
frame = customtkinter.CTkFrame(master=root)
frame.pack(pady=20, padx=60, fill="both", expand=True)

button1 = customtkinter.CTkButton(master=frame, width=400, height=100, fg_color="#FF00CC",
                                  hover_color="#FFC1CC", font=("Roboto", 33), text="Movies",
                                  command=lambda: self.features(frame, root, "Movies"))
button1.pack(pady=12, padx=10)
button1.place(relx=0.5, rely=0.20, anchor='center')

button2 = customtkinter.CTkButton(master=frame, width=400, height=100, fg_color="#77C3EC",
                                  hover_color="#44CCEF", font=("Roboto", 33), text="Locations",
                                  command=lambda: self.features(frame, root, "Locations"))
button2.pack(pady=12, padx=10)
button2.place(relx=0.5, rely=0.45, anchor='center')

button3 = customtkinter.CTkButton(master=frame, width=400, height=100, fg_color="#66FF00",
                                  hover_color="#80FC38", font=("Roboto", 33), text="Screenings",
                                  command=lambda: self.features(frame, root, "Screenings"))
button3.pack(pady=12, padx=10)
button3.place(relx=0.5, rely=0.70, anchor='center')
```

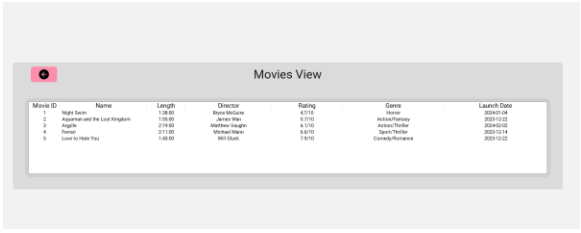
A. Afisarea celor 3 tabele

```
def features(self, frame, root, app):
    frame.destroy()
    customtkinter.set_default_color_theme("green")


    frame = customtkinter.CTkFrame(master=root)
    frame.pack(pady=20, padx=60, fill="both", expand=True)

    button1 = customtkinter.CTkButton(master=frame, width=300, height=60, fg_color="#C85815", hover_color="#8F9DE0A", font=("Roboto", 25),
                                      text=app + " View",
                                      command=lambda: self.viewtabel(frame, root, app))

    button1.pack(pady=12, padx=10)
    button1.place(relx=0.3, rely=0.3, anchor='center')

    button2 = customtkinter.CTkButton(master=frame, width=300, height=60, fg_color="#C85815", hover_color="#8F9DE0A", font=("Roboto", 25),
                                      text="Modify " + app,
                                      command=lambda: self.modifytabel(frame, root, app))

    button2.pack(pady=12, padx=10)
    button2.place(relx=0.3, rely=0.42, anchor='center')

    button3 = customtkinter.CTkButton(master=frame, width=300, height=60, fg_color="#C85815", hover_color="#8F9DE0A", font=("Roboto", 25),
                                      text="Insert into " + app,
                                      command=lambda: self.insertintotabel(frame, root, app))
```



B. Functia care afiseaza functionalitatile

```
                    show='headings')

        table.column("#1", anchor="c", minwidth=100, width=100)
        table.column("#2", anchor="w", minwidth=270, width=270)
        table.column("#3", anchor="c", minwidth=120, width=120)
        table.column("#4", anchor="c", minwidth=270, width=270)
        table.column("#5", anchor="c", minwidth=220, width=220)
        table.column("#6", anchor="c", minwidth=320, width=320)
        table.column("#7", anchor="c", minwidth=320, width=320)

        table.heading('id_movie', text='Movie ID')
        table.heading('name', text='Name')
        table.heading('length', text='Length')
        table.heading('director', text='Director')
        table.heading('rating', text='Rating')
        table.heading('genre', text='Genre')
        table.heading('launch_date', text='Launch Date')

        self.connection = mysql.connector.connect(host="localhost", user="root", passwd="a2eEfc@#us%+cQ",
                                                  database="cinema")
        self.cursor = self.connection.cursor()
        self.cursor.execute("""
                                SELECT id_movie FROM movies
                                """)
        results = [row[0] for row in self.cursor.fetchall()]
        self.connection.close()
        vector = []

        for x in results:
            d = Movie()
            d.load_movie(x)
            vector.append((d.id_movie, d.name, d.length, d.director, d.rating, d.genre, d.launch_date))

        for x in vector:
            table.insert('', tk.END, values=x)

        table.grid(row=1, column=0, padx=10, pady=10)
        table.bind('<Motion>', 'break')
```



C. Afisarea si popularea tabelului cu date

```
    table2.column("#1", anchor="c", minwidth=100, width=100)
    table2.column("#2", anchor="w", minwidth=100, width=100)
    table2.column("#3", anchor="c", minwidth=100, width=100)
    table2.column("#4", anchor="c", minwidth=100, width=100)
    table2.column("#5", anchor="c", minwidth=100, width=100)
    table2.column("#6", anchor="c", minwidth=100, width=100)
    table2.column("#7", anchor="c", minwidth=100, width=150)

    table2.heading('id_movie', text='Movie ID')
    table2.heading('name', text='Name')
    table2.heading('length', text='Length')
    table2.heading('director', text='Director')
    table2.heading('rating', text='Rating')
    table2.heading('genre', text='Genre')
    table2.heading('launch_date', text='Launch Date')
    table2.grid(row=3, column=0, padx=10, pady=40)
    table2.bind('<Motion>', 'break')

    entry1 = customtkinter.CTkEntry(master=frame, width=80, placeholder_text="Movie ID")
    entry1.grid(row=4, column=0, padx=35, pady=10, sticky='w')
    entry2 = customtkinter.CTkEntry(master=frame, width=80, placeholder_text="Name")
    entry2.grid(row=4, column=0, padx=110, pady=10, sticky='w')
    entry3 = customtkinter.CTkEntry(master=frame, width=80, placeholder_text="Length")
    entry3.grid(row=4, column=0, padx=190, pady=10, sticky='w')
    entry4 = customtkinter.CTkEntry(master=frame, width=80, placeholder_text="Director")
    entry4.grid(row=4, column=0, padx=270, pady=10, sticky='w')
    entry5 = customtkinter.CTkEntry(master=frame, width=80, placeholder_text="Rating")
    entry5.grid(row=4, column=0, padx=350, pady=10, sticky='w')
    entry6 = customtkinter.CTkEntry(master=frame, width=80, placeholder_text="Genre")
    entry6.grid(row=4, column=0, padx=430, pady=10, sticky='w')
    entry7 = customtkinter.CTkEntry(master=frame, width=120, placeholder_text="YYYY-MM-DD")
    entry7.grid(row=4, column=0, padx=510, pady=10, sticky='w')

    entry_button = customtkinter.CTkButton(master=frame, width=100, height=26, text="Modify",
                                           fg_color="#FC6E4C", hover_color="#FC4C0E",
                                           command=lambda: self.checkmodify(frame, root, app, results, entry1.get(), entry2.get(), entry3.get(), en
    entry_button.grid(row=5, column=0, padx=45, pady=15, sticky="w")
```

D. Modificarea tabelului, colectarea intrarilor

```
def checkmodify(self, frame, root, app, results, e1, e2, e3, e4, e5, e6, e7):
    if e1 != '':
        k = 0
        for x in results:
            if int(e1) == x:
                d = Movie()
                d.load_movie(x)
                if e2 != '':
                    d.name = e2
                if e3 != '':
                    d.length = e3
                if e4 != '':
                    d.director = e4
                if e5 != '':
                    d.rating = e5
                if e6 != '':
                    d.genre = e6
                if e7 != '':
                    d.launch_date = e7
                d.update_movie()
                k = 1
                label1 = customtkinter.CTkLabel(master=frame, width=300, text="Row modified",
                                                text_color='green', font=("Roboto", 16))
                label1.grid(row=5, column=0, padx=45, pady=10)
                break

        if k == 0:
            label1 = customtkinter.CTkLabel(master=frame, width=300, text="There are no row with the selected ID",
                                            text_color='red', font=("Roboto", 16))
            label1.grid(row=5, column=0, padx=45, pady=10)
```

E. Prelucrarea tabelului in functie de intrari

```
def checkinsert(self, frame, root, app, results, e2, e3, e4, e5, e6, e7):
    if e2 != '' and e3 != '' and e4 != '' and e5 != '' and e6 != '' and e7 != '':
        d = Movie()
        d.name = e2
        d.length = e3
        d.director = e4
        d.rating = e5
        d.genre = e6
        d.launch_date = e7
        d.insert_movie()
        label1 = customtkinter.CTkLabel(master=frame, width=300, text="Row inserted",
                                        text_color='green', font=("Roboto", 16))
        label1.grid(row=5, column=0, padx=45, pady=10)
    else:
        label1 = customtkinter.CTkLabel(master=frame, width=300,
                                        text="You have to complet the blank spaces",
                                        text_color='red', font=("Roboto", 16))
        label1.grid(row=5, column=0, padx=45, pady=10)
```

F. Inserarea in tabel

V. Bibliografie

DESCRIERE PYTHON *from* *https://ro.wikipedia.org/wiki/Python*

DESCRIERE MySQL *from* *https://nav.ro*

INFORMATII INSTRUCTIUNI PYTHON *from* *https://www.scaler.com/topics*

INFORMATII INSTRUCTIUNI CTKINTER *from github.com/TomSchimansky/CustomTkinter*

PALETE DE CULORI *from* *https://creativebooster.net/blogs/colors*

DEPANARE ERORI *from* *https://stackoverflow.com/*