Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

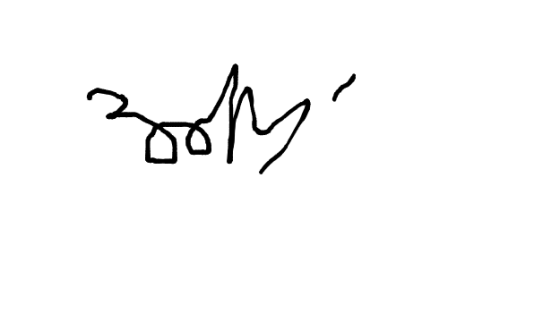
Кафедра системного аналізу та теорії прийняття рішень

Звіт

з лабораторної роботи № 3

Реалізація простого RESTful API

Виконав

студент групи К-23 Забровський В.Д.

Прийняв Махно М.Ф.

Київ – 2023

Варіант №6

**Мета:**

Створити простий веб-сервер, що надає RESTful API для керування певним ресурсом (за варіантом).

**Індивідуальне завдання:**1.Розробка моделі даних:

Мій варінт полягає у розробці наступної моделі даних “Музичні треки”:

з полями: id, song\_name, artist, album, release\_year.

2. Реалізація HTTP-методів:

Реалізовані наступні методи:

- GET для отримання списку ресурсів або одного ресурсу за ID.

- POST для створення нового ресурсу.

- PUT для повної зміни ресурсу.

- PATCH для часткової зміни ресурсу.

- DELETE для видалення ресурсу.

3. Розробка API:  
Для розробки програмного інтерфейсу було використано мову Python,зокрема фреймворк Flask(flask\_restful), який широко застосовується для розробки веб-застосунків.Також було використане розширення flask\_sqlalchemy для побудови бази даних для зберігання треків на базі SQLite.

4. Тестування API:

Для тестування API створено test.py файл, в якому передбачено тестові функції get\_track, post\_track, patch\_track, delete\_track, put\_track для тестування відповідних запитів за допомоги бібліотеки request

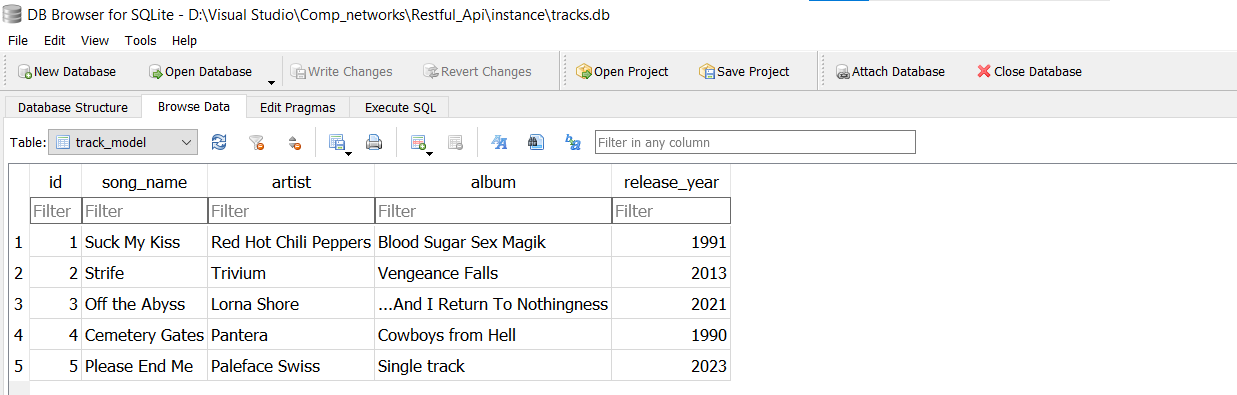
**Коди файлів**

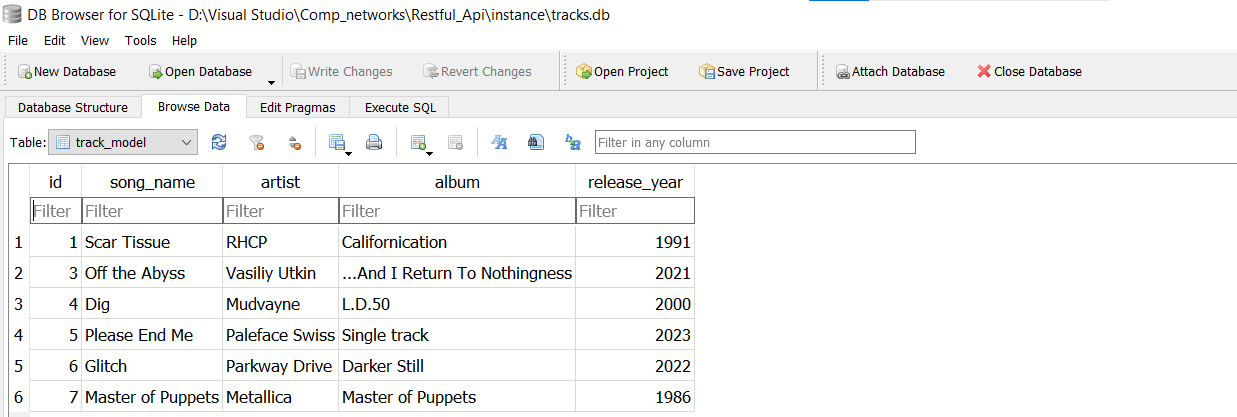
**main.py:**

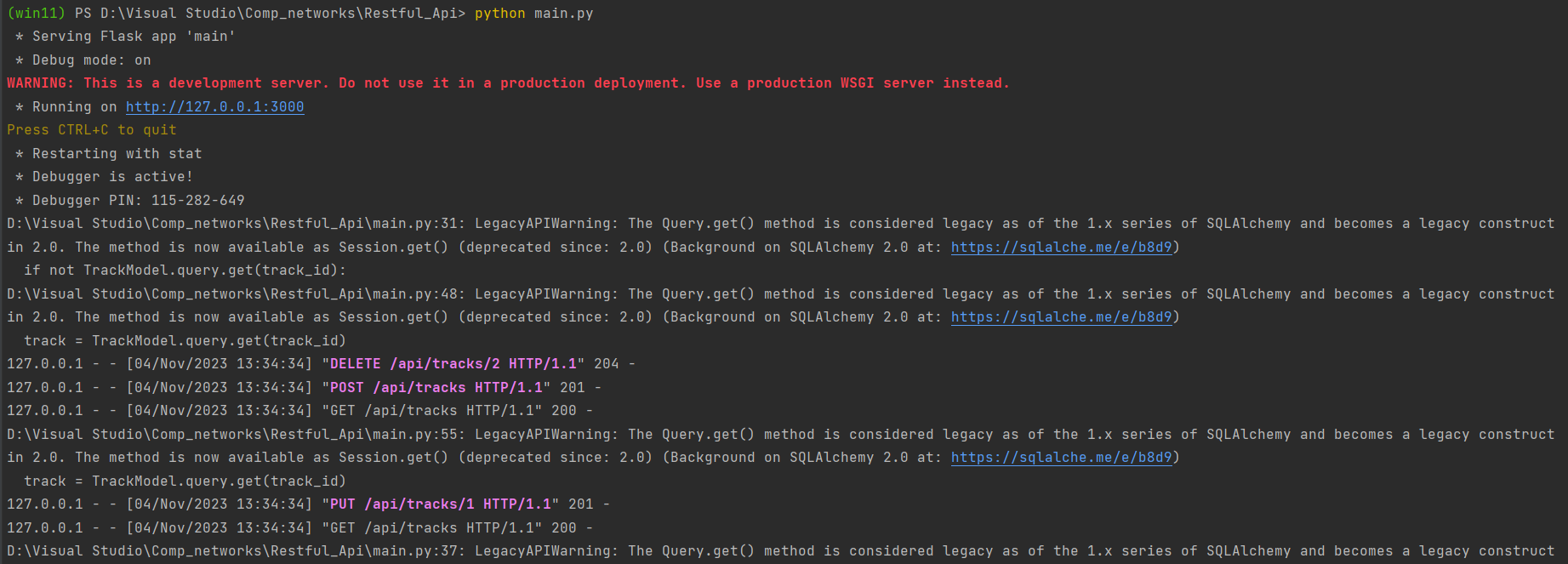
from flask import Flask  
from flask\_restful import Api, Resource, reqparse, abort  
from flask\_sqlalchemy import SQLAlchemy  
  
app = Flask(\_\_name\_\_)  
api = Api(app)  
  
  
app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///tracks.db'  
db = SQLAlchemy(app)  
parser = reqparse.RequestParser()  
parser.add\_argument("song\_name", type=str, required=False)  
parser.add\_argument("artist", type=str, required=False)  
parser.add\_argument("album", type=str, required=False)  
parser.add\_argument("release\_year", type=int, required=False)  
  
class TrackModel(db.Model):  
 id = db.Column(db.Integer, primary\_key=True)  
 song\_name = db.Column(db.String(255))  
 artist = db.Column(db.String(255))  
 album = db.Column(db.String(255))  
 release\_year = db.Column(db.Integer)  
  
 def \_\_init\_\_(self, song\_name, artist, album, release\_year):  
 self.song\_name = song\_name  
 self.artist = artist  
 self.album = album  
 self.release\_year = release\_year  
  
def abort\_if\_track\_doesnt\_exist(track\_id):  
 if not TrackModel.query.get(track\_id):  
 abort(404, message=f"Track {track\_id} doesn't exist")  
  
class Track(Resource):  
 def get(self, track\_id):  
 abort\_if\_track\_doesnt\_exist(track\_id)  
 track = TrackModel.query.get(track\_id)  
 return {  
 "id": track.id,  
 "song\_name": track.song\_name,  
 "artist": track.artist,  
 "album": track.album,  
 "release\_year": track.release\_year  
 }  
  
 def delete(self, track\_id):  
 abort\_if\_track\_doesnt\_exist(track\_id)  
 track = TrackModel.query.get(track\_id)  
 db.session.delete(track)  
 db.session.commit()  
 return '', 204  
  
 def put(self, track\_id):  
 args = parser.parse\_args()  
 track = TrackModel.query.get(track\_id)  
  
 if not track:  
 new\_track = TrackModel(  
 song\_name=args["song\_name"],  
 artist=args["artist"],  
 album=args["album"],  
 release\_year=args["release\_year"],  
 )  
 db.session.add(new\_track)  
 db.session.commit()  
 return {  
 "id": new\_track.id,  
 "song\_name": new\_track.song\_name,  
 "artist": new\_track.artist,  
 "album": new\_track.album,  
 "release\_year": new\_track.release\_year,  
 }  
 else:  
 track.song\_name = args["song\_name"]  
 track.artist = args["artist"]  
 track.album = args["album"]  
 track.release\_year = args["release\_year"]  
 db.session.commit()  
 return {  
 "id": track.id,  
 "song\_name": track.song\_name,  
 "artist": track.artist,  
 "album": track.album,  
 "release\_year": track.release\_year  
 }, 201  
  
 def patch(self, track\_id):  
 abort\_if\_track\_doesnt\_exist(track\_id)  
 args = parser.parse\_args()  
 track = TrackModel.query.get(track\_id)  
 track.song\_name = args.get("song\_name", track.song\_name)  
 track.artist = args.get("artist", track.artist)  
 track.album = args.get("album", track.album)  
 track.release\_year = args.get("release\_year", track.release\_year)  
 db.session.commit()  
 return {  
 "id": track.id,  
 "song\_name": track.song\_name,  
 "artist": track.artist,  
 "album": track.album,  
 "release\_year": track.release\_year  
 }, 200  
  
class TrackList(Resource):  
 def get(self):  
 tracks = TrackModel.query.all()  
 track\_data = [  
 {  
 "id": track.id,  
 "song\_name": track.song\_name,  
 "artist": track.artist,  
 "album": track.album,  
 "release\_year": track.release\_year,  
 }  
 for track in tracks  
 ]  
 return track\_data  
  
 def post(self):  
 args = parser.parse\_args()  
 new\_track = TrackModel(  
 song\_name=args["song\_name"],  
 artist=args["artist"],  
 album=args["album"],  
 release\_year=args["release\_year"],  
 )  
 db.session.add(new\_track)  
 db.session.commit()  
 return {  
 "id": new\_track.id,  
 "song\_name": new\_track.song\_name,  
 "artist": new\_track.artist,  
 "album": new\_track.album,  
 "release\_year": new\_track.release\_year,  
 }, 201  
  
api.add\_resource(TrackList, "/api/tracks")  
api.add\_resource(Track, "/api/tracks/<int:track\_id>")  
  
def initialize\_database\_with\_data():  
 data = {  
 1: {"song\_name": "Suck My Kiss", "artist": "Red Hot Chili Peppers", "album": "Blood Sugar Sex Magik", "release\_year": 1991},  
 2: {"song\_name": "Strife", "artist": "Trivium", "album": "Vengeance Falls", "release\_year": 2013},  
 3: {"song\_name": "Off the Abyss", "artist": "Lorna Shore", "album": "...And I Return To Nothingness", "release\_year": 2021},  
 4: {"song\_name": "Cemetery Gates", "artist": "Pantera", "album": "Cowboys from Hell", "release\_year": 1990},  
 5: {"song\_name": "Please End Me", "artist": "Paleface Swiss", "album": "Single track", "release\_year": 2023}  
 }  
  
 for track\_id, track\_info in data.items():  
 new\_track = TrackModel(  
 song\_name=track\_info["song\_name"],  
 artist=track\_info["artist"],  
 album=track\_info["album"],  
 release\_year=track\_info["release\_year"]  
 )  
 db.session.add(new\_track)  
  
 db.session.commit()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 with app.app\_context():  
 db.drop\_all()  
 db.create\_all()  
 initialize\_database\_with\_data()  
 app.run(debug=True, port=3000, host="127.0.0.1")

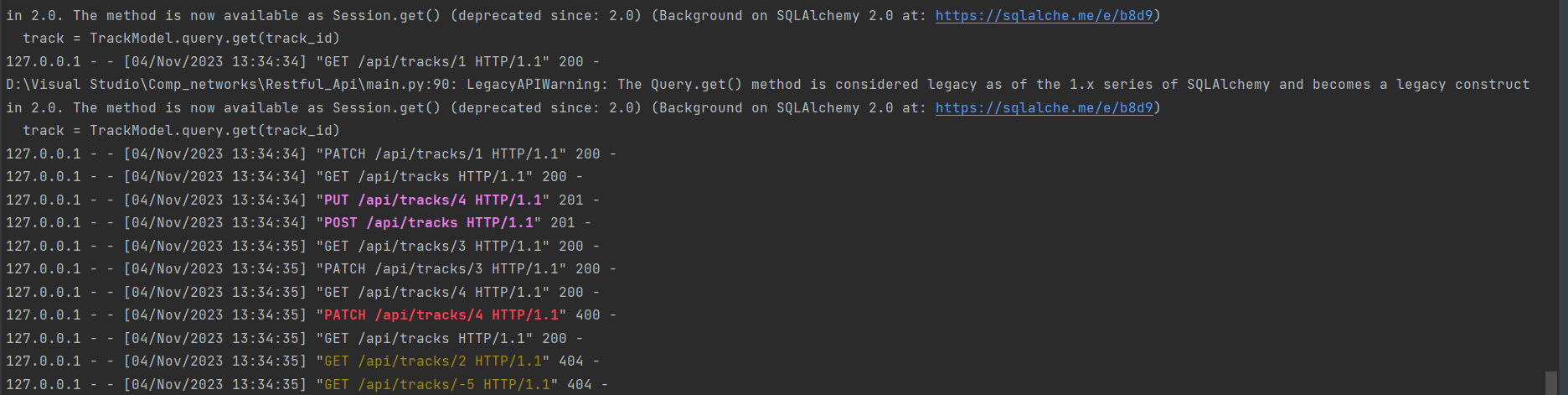
**test.py**

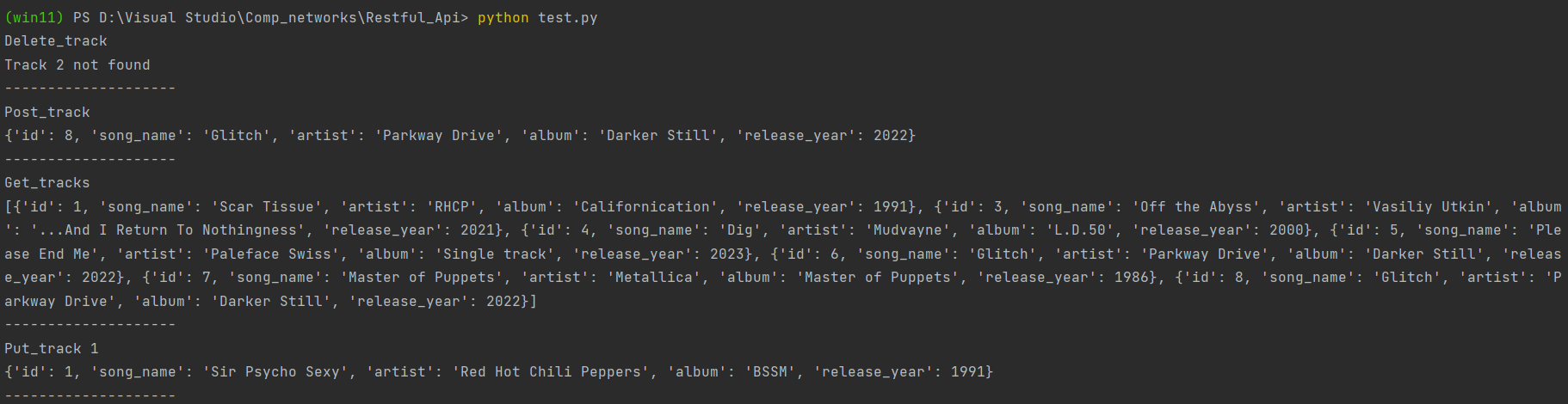
import requests  
import json  
  
BASE\_URL = "http://127.0.0.1:3000/api/tracks"  
  
def get\_track(track\_id=None):  
 try:  
 if track\_id is None:  
 url = f"{BASE\_URL}"  
 res = requests.get(url)  
 res.raise\_for\_status()  
 print('Get\_tracks')  
 print(res.json())  
 print('-' \* 20)  
 else:  
 url = f"{BASE\_URL}/{track\_id}"  
 res = requests.get(url)  
 if res.status\_code == 404:  
 print(f'Get\_track {track\_id}')  
 print(f"Track {track\_id} not found")  
 print('-' \* 20)  
 else:  
 res.raise\_for\_status()  
 print('Get\_track')  
 print(res.json())  
 print('-' \* 20)  
 except requests.exceptions.RequestException as e:  
 print(f"An error occurred while making the request: {e}")  
  
def post\_track(song\_name, artist, album, release\_year):  
 try:  
 data = {  
 "song\_name": song\_name,  
 "artist": artist,  
 "album": album,  
 "release\_year": release\_year  
 }  
 headers = {  
 "Content-Type": "application/json"  
 }  
 res = requests.post(BASE\_URL, data=json.dumps(data), headers=headers)  
 res.raise\_for\_status()  
 print('Post\_track')  
 print(res.json())  
 print('-' \* 20)  
 except requests.exceptions.RequestException as e:  
 print('Post\_track')  
 print(f"An error occurred while making the request: {e}")  
 print('-' \* 20)  
def put\_track(track\_id, song\_name=None, artist=None, album=None, release\_year=None):  
 try:  
 url = f"{BASE\_URL}/{track\_id}"  
 data = {}  
 if song\_name:  
 data["song\_name"] = song\_name  
 if artist:  
 data["artist"] = artist  
 if album:  
 data["album"] = album  
 if release\_year:  
 data["release\_year"] = release\_year  
 headers = {  
 "Content-Type": "application/json"  
 }  
 res = requests.put(url, data=json.dumps(data), headers=headers)  
 if res.status\_code == 404:  
 print(f"Track {track\_id} not found")  
 elif res.status\_code == 400:  
 print(res.json()["message"])  
 print('-' \* 20)  
 else:  
 res.raise\_for\_status()  
 print(f'Put\_track {track\_id}')  
 print(res.json())  
 print('-' \* 20)  
 except requests.exceptions.RequestException as e:  
 print(f'Put\_track {track\_id}')  
 print(f"An error occurred while making the request: {e}")  
 print('-' \* 20)  
def patch\_track(track\_id, \*\*kwargs):  
 try:  
 url = f"{BASE\_URL}/{track\_id}"  
 headers = {  
 "Content-Type": "application/json"  
 }  
 res = requests.get(url)  
 if res.status\_code == 404:  
 print(f"Track {track\_id} not found")  
 return  
 track = res.json()  
 for key, value in kwargs.items():  
 if value is not None:  
 track[key] = value  
 res = requests.patch(url, data=json.dumps(track), headers=headers)  
 if res.status\_code == 400:  
 print(f'Patch\_track {track\_id}')  
 print(res.json()["message"])  
 print(res.json())  
 elif res.status\_code == 200:  
 res.raise\_for\_status()  
 print(f'Patch\_track {track\_id}')  
 print(f"Track {track\_id} updated successfully")  
 print(res.json())  
 print('-' \* 20)  
 else:  
 print(f"Failed to update track {track\_id}")  
 except requests.exceptions.RequestException as e:  
 print("Patch\_track")  
 print(f"An error occurred while making the request: {e}")  
 print('-' \* 20)  
  
def delete\_track(track\_id):  
 try:  
 url = f"{BASE\_URL}/{track\_id}"  
 res = requests.delete(url)  
 if res.status\_code == 404:  
 print('Delete\_track')  
 print(f"Track {track\_id} not found")  
 print('-' \* 20)  
 else:  
 res.raise\_for\_status()  
 print(f'Delete\_track {track\_id}')  
 print(f"Track {track\_id} deleted")  
 print('-' \* 20)  
 except requests.exceptions.RequestException as e:  
 print('Delete\_track')  
 print(f"An error occurred while making the request: {e}")  
 print('-' \* 20)  
  
  
delete\_track(2)  
post\_track("Glitch", 'Parkway Drive', 'Darker Still', 2022)  
get\_track()  
put\_track(1, 'Sir Psycho Sexy', 'Red Hot Chili Peppers', 'BSSM', 1991)  
get\_track()  
patch\_track(1, song\_name="Scar Tissue", artist="RHCP", album='Californication')  
get\_track()  
put\_track(4, 'Dig', 'Mudvayne', 'L.D.50', 2000)  
post\_track("Master of Puppets", 'Metallica', 'Master of Puppets', 1986)  
patch\_track(3, artist='Vasiliy Utkin')  
patch\_track(4, album='Vulgar Display of Power', song\_name='Walk', release\_year='dsdjsoiddjdid')  
get\_track()  
get\_track(2)  
get\_track(-5)

**Стан бази даних tracks.db після відповідних запитів:  
  
**



**Вивід main.py у терміналі:**



**Вивід test.py у терміналі:**

