


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

## Мета:

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

## Індивідуальне завдання:

### 1.Розробка моделі даних:

Мій варіант 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:

DB Browser for SQLite - D:\Visual Studio\Comp\_networks\Restful\_Api\instance\tracks.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

Table: track\_model Filter in any column

	id	song_name	artist	album	release_year
	Filter	Filter	Filter	Filter	Filter
1	1	Suck My Kiss	Red Hot Chili Peppers	Blood Sugar Sex Magik	1991
2	2	Strife	Trivium	Vengeance Falls	2013
3	3	Off the Abyss	Lorna Shore	...And I Return To Nothingness	2021
4	4	Cemetery Gates	Pantera	Cowboys from Hell	1990
5	5	Please End Me	Paleface Swiss	Single track	2023

DB Browser for SQLite - D:\Visual Studio\Comp\_networks\Restful\_Api\instance\tracks.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

Table: track\_model Filter in any column

	id	song_name	artist	album	release_year
	Filter	Filter	Filter	Filter	Filter
1	1	Scar Tissue	RHCP	Californication	1991
2	3	Off the Abyss	Vasiliy Utkin	...And I Return To Nothingness	2021
3	4	Dig	Mudvayne	L.D.50	2000
4	5	Please End Me	Paleface Swiss	Single track	2023
5	6	Glitch	Parkway Drive	Darker Still	2022
6	7	Master of Puppets	Metallica	Master of Puppets	1986



## Вивід main.py у терміналі:

```
(win11) PS D:\Visual Studio\Comp_networks\Restful_Api> python main.py
* Serving Flask app 'main'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:3000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 115-282-649
D:\Visual Studio\Comp_networks\Restful_Api\main.py:31: LegacyAPIWarning: The Query.get() method is considered legacy as of the 1.x series of SQLAlchemy and becomes a legacy construct in 2.0. The method is now available as Session.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
  if not TrackModel.query.get(track_id):
D:\Visual Studio\Comp_networks\Restful_Api\main.py:48: LegacyAPIWarning: The Query.get() method is considered legacy as of the 1.x series of SQLAlchemy and becomes a legacy construct in 2.0. The method is now available as Session.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
  track = TrackModel.query.get(track_id)
127.0.0.1 ~ - [04/Nov/2023 13:34:34] "DELETE /api/tracks/2 HTTP/1.1" 204 -
127.0.0.1 ~ - [04/Nov/2023 13:34:34] "POST /api/tracks HTTP/1.1" 201 -
127.0.0.1 ~ - [04/Nov/2023 13:34:34] "GET /api/tracks HTTP/1.1" 200 -
D:\Visual Studio\Comp_networks\Restful_Api\main.py:55: LegacyAPIWarning: The Query.get() method is considered legacy as of the 1.x series of SQLAlchemy and becomes a legacy construct in 2.0. The method is now available as Session.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
  track = TrackModel.query.get(track_id)
127.0.0.1 ~ - [04/Nov/2023 13:34:34] "PUT /api/tracks/1 HTTP/1.1" 201 -
127.0.0.1 ~ - [04/Nov/2023 13:34:34] "GET /api/tracks HTTP/1.1" 200 -
D:\Visual Studio\Comp_networks\Restful_Api\main.py:37: LegacyAPIWarning: The Query.get() method is considered legacy as of the 1.x series of SQLAlchemy and becomes a legacy construct in 2.0. The method is now available as Session.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
  track = TrackModel.query.get(track_id)
127.0.0.1 ~ - [04/Nov/2023 13:34:34] "GET /api/tracks/1 HTTP/1.1" 200 -
D:\Visual Studio\Comp_networks\Restful_Api\main.py:90: LegacyAPIWarning: The Query.get() method is considered legacy as of the 1.x series of SQLAlchemy and becomes a legacy construct in 2.0. The method is now available as Session.get() (deprecated since: 2.0) (Background on SQLAlchemy 2.0 at: https://sqlalche.me/e/b8d9)
  track = TrackModel.query.get(track_id)
127.0.0.1 ~ - [04/Nov/2023 13:34:34] "PATCH /api/tracks/1 HTTP/1.1" 200 -
127.0.0.1 ~ - [04/Nov/2023 13:34:34] "GET /api/tracks HTTP/1.1" 200 -
127.0.0.1 ~ - [04/Nov/2023 13:34:34] "PUT /api/tracks/4 HTTP/1.1" 201 -
127.0.0.1 ~ - [04/Nov/2023 13:34:34] "POST /api/tracks HTTP/1.1" 201 -
127.0.0.1 ~ - [04/Nov/2023 13:34:35] "GET /api/tracks/3 HTTP/1.1" 200 -
127.0.0.1 ~ - [04/Nov/2023 13:34:35] "PATCH /api/tracks/3 HTTP/1.1" 200 -
127.0.0.1 ~ - [04/Nov/2023 13:34:35] "GET /api/tracks/4 HTTP/1.1" 200 -
127.0.0.1 ~ - [04/Nov/2023 13:34:35] "PATCH /api/tracks/4 HTTP/1.1" 400 -
127.0.0.1 ~ - [04/Nov/2023 13:34:35] "GET /api/tracks HTTP/1.1" 200 -
127.0.0.1 ~ - [04/Nov/2023 13:34:35] "GET /api/tracks/2 HTTP/1.1" 404 -
127.0.0.1 ~ - [04/Nov/2023 13:34:35] "GET /api/tracks/~5 HTTP/1.1" 404 -
```

## Вивід test.py у терміналі:

```
(win11) PS D:\Visual Studio\Comp_networks\Restful_Api> python test.py
Delete_track
Track 2 not found
-----
Post_track
{'id': 8, 'song_name': 'Glitch', 'artist': 'Parkway Drive', 'album': 'Darker Still', 'release_year': 2022}
-----
Get_tracks
[{'id': 1, 'song_name': 'Scar Tissue', 'artist': 'RHCP', 'album': 'Californication', 'release_year': 1991}, {'id': 3, 'song_name': 'Off the Abyss', 'artist': 'Vasiliy Utkin', 'album': '...And I Return To Nothingness', 'release_year': 2021}, {'id': 4, 'song_name': 'Dig', 'artist': 'Mudvayne', 'album': 'L.D.50', 'release_year': 2000}, {'id': 5, 'song_name': 'Please End Me', 'artist': 'Paleface Swiss', 'album': 'Single track', 'release_year': 2023}, {'id': 6, 'song_name': 'Glitch', 'artist': 'Parkway Drive', 'album': 'Darker Still', 'release_year': 2022}, {'id': 7, 'song_name': 'Master of Puppets', 'artist': 'Metallica', 'album': 'Master of Puppets', 'release_year': 1986}, {'id': 8, 'song_name': 'Glitch', 'artist': 'Parkway Drive', 'album': 'Darker Still', 'release_year': 2022}]
-----
Put_track 1
{'id': 1, 'song_name': 'Sir Psycho Sexy', 'artist': 'Red Hot Chili Peppers', 'album': 'BSSM', 'release_year': 1991}
-----
```

```
Get_tracks
[{'id': 1, 'song_name': 'Sir Psycho Sexy', 'artist': 'Red Hot Chili Peppers', 'album': 'BSSM', 'release_year': 1991}, {'id': 3, 'song_name': 'Off the Abyss', 'artist': 'Vasiliy Utkin', 'album': '...And I Return To Nothingness', 'release_year': 2021}, {'id': 4, 'song_name': 'Dig', 'artist': 'Mudvayne', 'album': 'L.D.50', 'release_year': 2000}, {'id': 5, 'song_name': 'Please End Me', 'artist': 'Paleface Swiss', 'album': 'Single track', 'release_year': 2023}, {'id': 6, 'song_name': 'Glitch', 'artist': 'Parkway Drive', 'album': 'Darker Still', 'release_year': 2022}, {'id': 7, 'song_name': 'Master of Puppets', 'artist': 'Metallica', 'album': 'Master of Puppets', 'release_year': 1986}, {'id': 8, 'song_name': 'Glitch', 'artist': 'Parkway Drive', 'album': 'Darker Still', 'release_year': 2022}]
-----
Patch_track 1
Track 1 updated successfully
{'id': 1, 'song_name': 'Scar Tissue', 'artist': 'RHCP', 'album': 'Californication', 'release_year': 1991}
-----
Get_tracks
[{'id': 1, 'song_name': 'Scar Tissue', 'artist': 'RHCP', 'album': 'Californication', 'release_year': 1991}, {'id': 3, 'song_name': 'Off the Abyss', 'artist': 'Vasiliy Utkin', 'album': '...And I Return To Nothingness', 'release_year': 2021}, {'id': 4, 'song_name': 'Dig', 'artist': 'Mudvayne', 'album': 'L.D.50', 'release_year': 2000}, {'id': 5, 'song_name': 'Please End Me', 'artist': 'Paleface Swiss', 'album': 'Single track', 'release_year': 2023}, {'id': 6, 'song_name': 'Glitch', 'artist': 'Parkway Drive', 'album': 'Darker Still', 'release_year': 2022}, {'id': 7, 'song_name': 'Master of Puppets', 'artist': 'Metallica', 'album': 'Master of Puppets', 'release_year': 1986}, {'id': 8, 'song_name': 'Glitch', 'artist': 'Parkway Drive', 'album': 'Darker Still', 'release_year': 2022}]
-----
Put_track 4
```

```
Post_track 4
{'id': 4, 'song_name': 'Dig', 'artist': 'Mudvayne', 'album': 'L.D.50', 'release_year': 2000}
-----
Post_track
{'id': 9, 'song_name': 'Master of Puppets', 'artist': 'Metallica', 'album': 'Master of Puppets', 'release_year': 1986}
-----
Patch_track 3
Track 3 updated successfully
{'id': 3, 'song_name': 'Off the Abyss', 'artist': 'Vasilii Utkin', 'album': '...And I Return To Nothingness', 'release_year': 2021}
-----
Patch_track 4
{'release_year': "invalid literal for int() with base 10: 'dsdjsoiddjdid'"}
{'message': {'release_year': "invalid literal for int() with base 10: 'dsdjsoiddjdid'"}}
Get_tracks
[{'id': 1, 'song_name': 'Scar Tissue', 'artist': 'RHCP', 'album': 'Californication', 'release_year': 1991}, {'id': 3, 'song_name': 'Off the Abyss', 'artist': 'Vasilii Utkin', 'album': '...And I Return To Nothingness', 'release_year': 2021}, {'id': 4, 'song_name': 'Dig', 'artist': 'Mudvayne', 'album': 'L.D.50', 'release_year': 2000}, {'id': 5, 'song_name': 'Please End Me', 'artist': 'Paleface Swiss', 'album': 'Single track', 'release_year': 2023}, {'id': 6, 'song_name': 'Glitch', 'artist': 'Parkway Drive', 'album': 'Darker Still', 'release_year': 2022}, {'id': 7, 'song_name': 'Master of Puppets', 'artist': 'Metallica', 'album': 'Master of Puppets', 'release_year': 1986}, {'id': 8, 'song_name': 'Glitch', 'artist': 'Parkway Drive', 'album': 'Darker Still', 'release_year': 2022}, {'id': 9, 'song_name': 'Master of Puppets', 'artist': 'Metallica', 'album': 'Master of Puppets', 'release_year': 1986}]
-----
Get_track 2
Track 2 not found
-----
Get_track -5
Track -5 not found
-----
```

