Міністерство освіти та науки України

Харківський національний університет радіоелектроніки

Кафедра Інформатики

Звіт

з лабораторної роботи №3

з дисципліни «Теорія програмування»

Виконав: Перевірив:

Ст.гр. ІТІНФ-20-1 Сінельнікова. Т. Ф.

Самченко С. О.

Харків 2023

**Мета:** Ознайомитись з основними поняттями та принципом мікросервісної архітектури. Розглянути мікросервіси як незалежні компоненти, що виконують окремі функції всередині програми, та взаємодіють один з одним за допомогою мережевих дзвінків.

**Завдання:** Реалізувати мікросервіс обраною мовою програмування.

**Хід роботи:**

from flask import Flask, request, jsonify

from pymongo import MongoClient

from bson.objectid import ObjectId

app = Flask(\_\_name\_\_)

client = MongoClient('mongodb://localhost:27017/')

db = client['mydatabase']

@app.route('/users', methods=['GET'])

def get\_users():

    query = {}

    name = request.args.get('name')

    if name:

        query['name'] = name

    users = db.users.find(query)

    return jsonify(users)

@app.route('/users/<user\_id>', methods=['GET'])

def get\_user(user\_id):

    user = db.users.find\_one({'\_id': ObjectId(user\_id)})

    if not user:

        return jsonify({'message': 'User not found'}), 404

    return jsonify(user)

@app.route('/users', methods=['POST'])

def create\_user():

    data = request.get\_json()

    name = data.get('name')

    age = data.get('age')

    if not name or not age:

        return jsonify({'message': 'Name and age are required'}), 400

    user = {'name': name, 'age': age}

    result = db.users.insert\_one(user)

    return jsonify({'message': 'User created successfully', 'user\_id': str(result.inserted\_id)}), 201

@app.route('/users/<user\_id>', methods=['PUT'])

def update\_user(user\_id):

    data = request.get\_json()

    name = data.get('name')

    age = data.get('age')

    if not name or not age:

        return jsonify({'message': 'Name and age are required'}), 400

    result = db.users.update\_one({'\_id': ObjectId(user\_id)}, {'$set': {'name': name, 'age': age}})

    if result.modified\_count == 0:

        return jsonify({'message': 'User not found'}), 404

    return jsonify({'message': 'User updated successfully'})

@app.route('/users/<user\_id>', methods=['DELETE'])

def delete\_user(user\_id):

    result = db.users.delete\_one({'\_id': ObjectId(user\_id)})

    if result.deleted\_count == 0:

        return jsonify({'message': 'User not found'}), 404

    return jsonify({'message': 'User deleted successfully'})

if \_\_name\_\_ == '\_\_main\_\_':

    app.run()

У цьому прикладі ми використовуємо фреймворк Flask для створення маршрутів /users для отримання списку користувачів та створення нових користувачів.

Так само програма підтримує функціонал:

Отримання користувачів на ім'я: Ви можете вказати ім'я користувача у параметрах запиту для отримання лише тих користувачів, чиї імена відповідають зазначеному значенню.

Отримання користувача за ідентифікатором: існує ендпоінт /users/<user\_id>, який дозволяє отримати користувача за його унікальним ідентифікатором.

Оновлення користувача: існує ендпоінт /users/<user\_id> методом PUT, який дозволяє оновити інформацію про користувача щодо його ідентифікатора.

Видалення користувача: існує ендпоінт /users/<user\_id> методом DELETE, який дозволяє видалити користувача за його ідентифікатором.

Ми також підключаємося до бази даних MongoDB за допомогою pymongo. При зверненні до ендпоїнту /users методом GET, ми виймаємо всіх користувачів з колекції users у базі даних та повертаємо їх у форматі JSON.

При зверненні до ендпоінту /users методом POST ми отримуємо дані нового користувача із запиту, створюємо словник з полями name і age, а потім вставляємо його в колекцію users в базі даних.

**Висновок:** У рамках даної роботи відбулося ознайомлення з основними поняттями та принципом мікросервісної архітектури.