МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет “Львівська політехніка”

Інститут комп’ютерних наук та інформаційних технологій

Кафедра інформаційних систем та мереж



**Звіт**

до лабораторної роботи № 4

**«Побудова системи інтеграції даних за допомогою архітектурного стилю RESTful (Серверна частина)»**

з дисципліни

**«Технології інтеграції інформаційних ресурсів»**

Виконав: ст. гр. ІТІС-11

Стельмах В. Д.

Прийняв:

Щербак С.С.

Львів – 2025

**Лабораторна робота № 4**

**Тема:** Побудова системи інтеграції даних за допомогою архітектурного стилю RESTful (Серверна частина)

**Мета роботи:** Ознайомитись із особливостями створення серверної частини системи інтеграції даних з використанням підходу Representational State Transfer (REST) та фреймворку FastAPI.

**Порядок виконання роботи:**

1. Налаштувати середовище розробки Python та встановити необхідні бібліотеки (FastAPI, Uvicorn, Pydantic).
2. Розробити RESTful API "Каталог товарів магазину електроніки" на Python з використанням фреймворку FastAPI.
3. Визначити моделі даних (Pydantic) для товарів (створення, оновлення, відображення).
4. Реалізувати ендпоінти для виконання CRUD-операцій (Create, Read, Update, Delete) над ресурсами товарів, використовуючи відповідні HTTP методи.
5. Забезпечити валідацію вхідних даних за допомогою Pydantic.
6. Використати in-memory сховище для зберігання даних товарів на час роботи сервера.
7. Продемонструвати загальну структуру API за допомогою автоматично генерованої документації FastAPI (Swagger UI).
8. Продемонструвати роботу ендпоінтів API шляхом взаємодії з консольним клієнтом, розробленим у Лабораторній роботі №3.

**Короткі теоретичні відомості**

REST (Representational State Transfer) – це архітектурний стиль, що визначає набір обмежень для створення вебсервісів. RESTful API використовують стандартні HTTP методи (GET, POST, PUT, DELETE тощо) для взаємодії з ресурсами, які ідентифікуються за допомогою URI. Дані зазвичай передаються у форматі JSON або XML.

FastAPI – це сучасний, швидкий (високопродуктивний) вебфреймворк для створення API на Python 3.7+ на основі стандартних підказок типів Python. Ключові особливості FastAPI:

* Швидкість виконання: Завдяки Starlette (для вебчастини) та Pydantic (для даних).
* Швидкість розробки: Значно скорочує час написання коду.
* Менше помилок: Завдяки підказкам типів та валідації.
* Інтуїтивність: Простий у використанні та вивченні.
* Автоматична документація: Генерує інтерактивну документацію API (Swagger UI та ReDoc) "з коробки".

\* Валідація даних: Pydantic використовується для визначення моделей даних та автоматичної валідації запитів та відповідей.

Pydantic – це бібліотека для валідації даних та управління налаштуваннями за допомогою підказок типів Python. FastAPI використовує моделі Pydantic для визначення структури запитів та відповідей, що забезпечує автоматичну валідацію, серіалізацію та документацію.

Uvicorn – це ASGI (Asynchronous Server Gateway Interface) сервер, який використовується для запуску FastAPI додатків.

**Результати виконання роботи**

**Демонстрація роботи REST API**

API "Каталог товарів магазину електроніки" було запущено локально. Загальна структура API та можливості тестування були переглянуті через автоматично генеровану інтерактивну документацію Swagger UI. Подальша демонстрація функціональності ендпоінтів проводилася шляхом взаємодії з консольним клієнтом, розробленим у Лабораторній роботі №3.

Загальний вигляд документації API (Swagger UI) та приклад створення нового товару (POST /products):

Представлено головну сторінку документації API в Swagger UI, де видно всі доступні ендпоінти та їхні описи.

Зображення, що містить текст, знімок екрана, програмне забезпечення, Комп’ютерна піктограма

Вміст, створений ШІ, може бути неправильним.

Рис. 1. Загальний вигляд документації API в Swagger UI та приклад POST-запиту для створення товару.

Демонструється взаємодія консольного клієнта (з ЛР3) з API для отримання повного списку товарів та інформації про один конкретний товар за його ID. Консольний вивід показує запити клієнта та відповідні JSON-відповіді, отримані від сервера.

Зображення, що містить текст, знімок екрана

Вміст, створений ШІ, може бути неправильним.

Рис. 2. Демонстрація роботи GET-запитів через консольний клієнт: отримання списку товарів та конкретного товару.

Оновлення (PUT /products/{product\_id}) та видалення товару (DELETE /products/{product\_id}) через клієнт:

Показано процес оновлення даних існуючого товару (наприклад, зміна кількості на складі) через PUT-запит, ініційований консольним клієнтом, та подальше його видалення з каталогу через DELETE-запит. Консольний вивід відображає відповіді сервера (200 OK для PUT, 204 No Content для DELETE).

Зображення, що містить текст, знімок екрана, програмне забезпечення, Мультимедійне програмне забезпечення

Вміст, створений ШІ, може бути неправильним.

Рис. 3. Демонстрація роботи PUT запитів через консольний клієнт: оновлення та видалення товару.

Зображення, що містить текст, знімок екрана

Вміст, створений ШІ, може бути неправильним.

Рис. 4. Демонстрація роботи DELETE запитів через консольний клієнт: оновлення та видалення товару.

Обробка помилкових ситуацій сервером (демонстрація через клієнт):

Демонстрація реакції API на некоректні запити, ініційовані консольним клієнтом: спроба взаємодії з неіснуючим ресурсом (очікувана відповідь сервера 404 Not Found) Консольний вивід клієнта показує ці відповіді від сервера.

Зображення, що містить текст, знімок екрана

Вміст, створений ШІ, може бути неправильним.

Рис. 4. Демонстрація обробки помилок сервером: відповіді на некоректні запити, показані через консольний клієнт.

**Аналіз функціональності API**

Розроблене REST API "Каталог товарів магазину електроніки" успішно реалізує всі заплановані CRUD-операції над ресурсами товарів. Фреймворк FastAPI та бібліотека Pydantic забезпечили швидку розробку, автоматичну валідацію вхідних даних та генерацію інтерактивної документації (Swagger UI). Ендпоінти коректно обробляють запити, що надходять від консольного клієнта, повертають дані у форматі JSON та відповідні HTTP статус-коди, включаючи обробку помилкових ситуацій. In-memory сховище даних є достатнім для демонстрації функціональності в рамках лабораторної роботи.

**Посилання на вихідний код програми**

Повний вихідний код розробленого REST API на Python з використанням FastAPI розміщено у репозиторії GitHub за наступним посиланням:

<https://github.com/VitaliiStelmakh/TIIR/blob/main/lab_4/lab_4.py>

**Висновки**

Під час виконання лабораторної роботи я ознайомився з принципами побудови серверної частини RESTful API та практично реалізував такий API за допомогою фреймворку FastAPI на Python. Було розроблено ендпоінти для CRUD-операцій, визначено моделі даних з Pydantic для валідації, та протестовано роботу API через взаємодію з розробленим раніше клієнтом. Ця робота дозволила закріпити розуміння архітектурного стилю REST, HTTP методів, формату JSON та отримати навички створення ефективних, документованих та валідованих вебсервісів.