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 № 3

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

з дисципліни

**«Технології інтеграції інформаційних ресурсів»**

Виконав: ст. гр. ІТІС-11

Стельмах В. Д.

Прийняв:

Щербак С.С.

Львів – 2025

**Лабораторна робота № 3**

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

**Мета роботи:** Ознайомитись із особливостями створення клієнтської частини системи інтеграції з використанням підходу Representational State Transfer (REST) для взаємодії з RESTful API.

**Порядок виконання роботи:**

1. Налаштувати середовище розробки Python та встановити необхідні бібліотеки (зокрема requests).
2. Розробити консольний клієнтський застосунок на Python, який буде взаємодіяти з RESTful API "Каталог товарів магазину електроніки" (розробленим у Лабораторній роботі №4, що працює за адресою http://127.0.0.1:8000).
3. Реалізувати функції для виконання CRUD-операцій (Create, Read, Update, Delete) над ресурсами товарів шляхом надсилання відповідних HTTP-запитів (POST, GET, PUT, DELETE) до API.
4. Забезпечити обробку відповідей від API, включаючи успішні результати (у форматі JSON) та можливі HTTP-помилки.
5. Реалізувати простий текстовий консольний інтерфейс (меню) для взаємодії користувача з клієнтом.
6. Продемонструвати роботу клієнтського застосунку шляхом взаємодії з API "Каталог товарів магазину електроніки".
7. Задокументувати результати роботи програми, включаючи ключові скріншоти (або текстові копії) консольного виводу, та надати посилання на репозиторій з кодом програми у звіті.

**Короткі теоретичні відомості**

REST (Representational State Transfer) – це архітектурний стиль для створення вебсервісів, який базується на наборі принципів та обмежень. Ключовими принципами REST є клієнт-серверна архітектура, відсутність стану (statelessness), можливість кешування відповідей, одноманітний інтерфейс, шарувата система та, опціонально, код на вимогу.

В REST архітектурі дані розглядаються як ресурси, кожен з яких має унікальний ідентифікатор (URI). Маніпуляції над ресурсами здійснюються за допомогою стандартних HTTP методів:

* GET: Отримання представлення ресурсу (або колекції ресурсів).
* POST: Створення нового ресурсу.
* PUT: Повне оновлення існуючого ресурсу.
* DELETE: Видалення ресурсу.
* PATCH: Часткове оновлення існуючого ресурсу.

Дані між клієнтом та сервером зазвичай передаються у форматі JSON (JavaScript Object Notation) – легкому текстовому форматі, зручному для читання та парсингу.

Для взаємодії з REST API з Python часто використовується бібліотека requests, яка спрощує надсилання HTTP-запитів та обробку відповідей. Вона дозволяє легко формувати запити з потрібними методами, заголовками, тілом (наприклад, JSON-даними) та отримувати відповіді, включаючи статус-коди, заголовки та тіло відповіді.

**Результати виконання роботи**

**Середовище розробки та налаштування**

Клієнтський застосунок було розроблено мовою програмування Python в операційній системі Windows з використанням WSL Ubuntu 22.04. Для здійснення HTTP-запитів до RESTful API було використано бібліотеку requests. Серверна частина (API "Каталог товарів магазину електроніки") була запущена локально за адресою http://127.0.0.1:8000.

**Демонстрація роботи клієнтського застосунку**

Клієнтський застосунок надає текстове меню для взаємодії з API. Нижче наведено ключові скріншоти (або текстові копії) консольного виводу, що демонструють основні операції.

**Загальний вигляд меню та створення нового товару:**

Представлено головне меню та процес додавання нового товару до каталогу з відповідним повідомленням від сервера.

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

Вміст, створений ШІ, може бути неправильним.

Рис. 1. Головне меню клієнта та приклад створення нового товару.

**Отримання списку товарів та перегляд конкретного товару:**

Демонструється отримання повного списку товарів з каталогу (після додавання кількох) та отримання інформації про один конкретний товар за його ID.

Зображення, що містить текст, знімок екрана

Вміст, створений ШІ, може бути неправильним.

Рис. 2. Перегляд списку товарів

**Оновлення та видалення товару:**

Показано процес оновлення даних існуючого товару (наприклад, зміна ціни) та подальше його видалення з каталогу.

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

Вміст, створений ШІ, може бути неправильним.

Рис. 3. Приклад оновлення товару

Зображення, що містить текст, знімок екрана

Вміст, створений ШІ, може бути неправильним.

Рис. 4. Приклад видалення товару

**Обробка помилкових ситуацій:**

Демонстрація реакції клієнта на спробу взаємодії з неіснуючим ресурсом (наприклад, запит товару за невірним ID) або передачу некоректних даних.

Зображення, що містить текст, знімок екрана

Вміст, створений ШІ, може бути неправильним.

Рис. 4. Приклад обробки помилки клієнтом.

**Аналіз протоколу роботи**

Клієнтський застосунок успішно взаємодіє з REST API "Каталог товарів магазину електроніки". Реалізовано всі CRUD-операції через текстове меню. Застосунок коректно надсилає HTTP-запити (GET, POST, PUT, DELETE) та обробляє JSON-відповіді від сервера, включаючи повідомлення про успішне виконання операцій та інформацію про помилки (наприклад, статус 404 "Not Found" або 422 "Unprocessable Entity"). Використання бібліотеки requests дозволило ефективно реалізувати HTTP-взаємодію.

**Посилання на вихідний код програми**

Повний вихідний код розробленого консольного клієнта на Python розміщено у репозиторії GitHub за наступним посиланням:

<https://github.com/VitaliiStelmakh/TIIR/blob/main/lab_3/lab_3.py>

**Висновки**

Під час виконання лабораторної роботи я ознайомився з архітектурним стилем REST та практично реалізував клієнтський застосунок для взаємодії з RESTful API. Було розроблено консольний клієнт на Python, що використовує бібліотеку requests для виконання HTTP-запитів та обробки JSON-даних. Ця робота дозволила закріпити розуміння принципів REST, HTTP методів та отримати навички створення програм, що інтегруються з вебсервісами.