

Міністерство освіти і науки України
Львівський національний університет імені Івана Франка
Факультет електроніки та комп'ютерних технологій

ЛАБОРАТОРНА РОБОТА № 6
з курсу “Веб програмування на стороні сервера”
“Створення сервісу інвентаризації ”

Виконала:
Ст. ФЕІ-23
Гвоздевич Вікторія
Перевірив:
асистент Чмихало Олександр
Сергійович

Львів 2025

Мета роботи

- Робота з різними типами HTTP запитів
- Закріпити знання з використання Express.js для створення Веб-сервісів
- Використання Postman для тестування сервісу
- Створити документацію для веб сервісу з використанням Swagger

Хід роботи:

Підготовка:

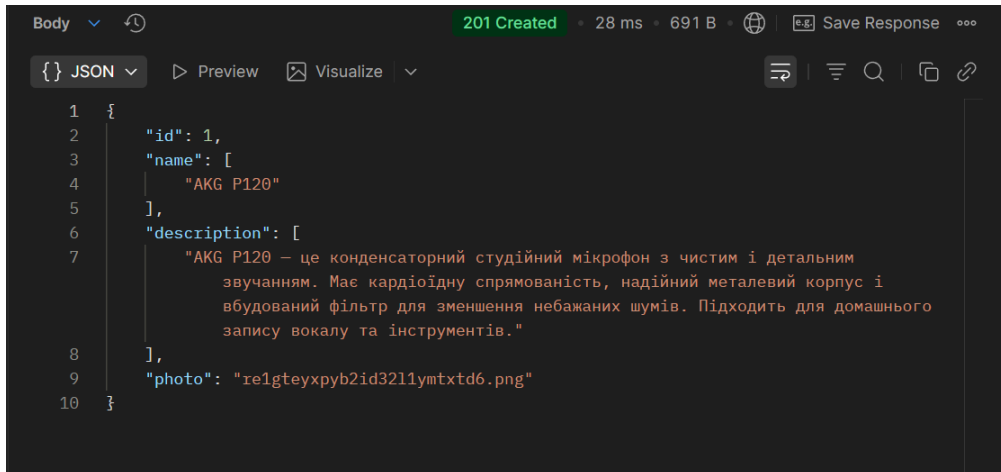
- Створено новий репозиторій backend-course-2025-6.
- Налаштовано ім'я користувача у форматі Ваше Ім'я bc2025-6 та електронну пошту для комітів за допомогою команди `git config`.
- Створено файл `package.json` у кореневій теці та додано головний файл програми; обидва файли закомічено.
- Локально встановлено пакет `Commander.js` для роботи з аргументами командного рядка. ○ Створено файл `.gitignore` та додано тека `node_modules` до виключень Git. ○ Підключено модуль `Commander.js` у програмі та реалізовано читання аргументів командного рядка.
- Локально встановлено пакет `superagent`.
- Додано пакет `nodemon` як залежність для розробки.
- Закомічено оновлений `package.json`, `package-lock.json` та головний файл програми.

Частина 1 - параметри командного рядка та Веб сервер

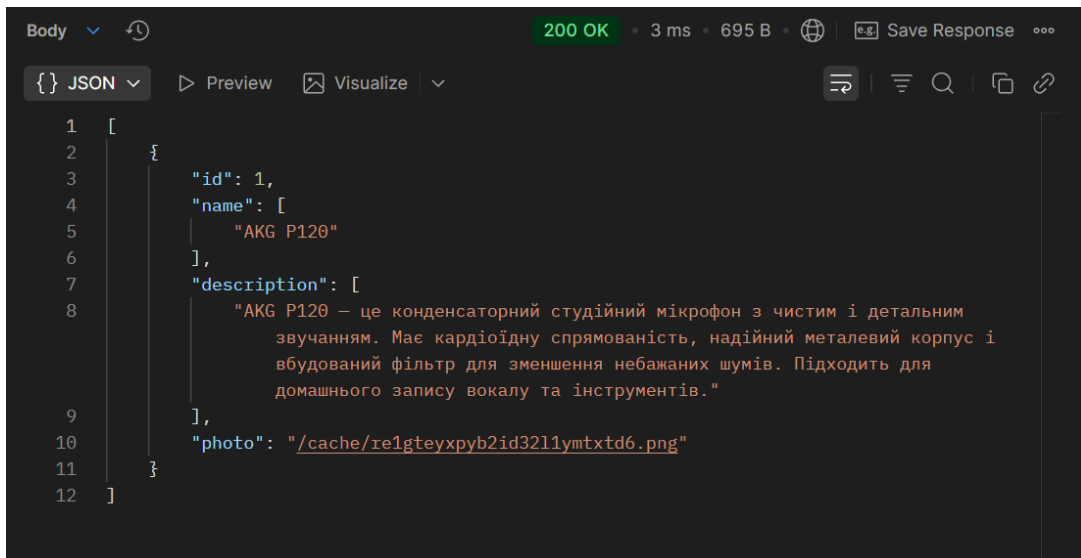
- Програма налаштована на приймання параметрів командного рядка за допомогою `Commander.js`: ○ `-h, --host` — обов'язковий параметр, що задає адресу сервера ○ `-p, --port` — обов'язковий параметр, що задає порт сервера ○ `-c, --cache` — обов'язковий параметр, що задає шлях до директорії для кешування файлів. Якщо вказано шлях до неіснуючої директорії, програма автоматично створює її під час запуску.
- Реалізовано перевірку: якщо будь-який обов'язковий параметр не передано — виводиться відповідна помилка.
- За допомогою модуля `http` запущено веб-сервер; значення параметрів `--host` та `--port` передаються у метод `http.Server.listen()`.
- Для автоматичного перезапуску сервера при зміні файлів сервер запускається через `nodemon`, встановлений як залежність лише для розробки.

Частина 2 - Реалізація WebAPI сервісу

Реалізовано POST /register Обробляється multipart form data за допомогою formidable
Приймає поля inventory_name description photo Якщо ім'я не передано сервер повертає
400 Bad Request У разі успіху повертається 201 Created



Реалізовано GET /inventory Повертає список усіх інвентаризованих речей у форматі
JSON Кожен елемент містить ID опис інформацію та посилання на фото



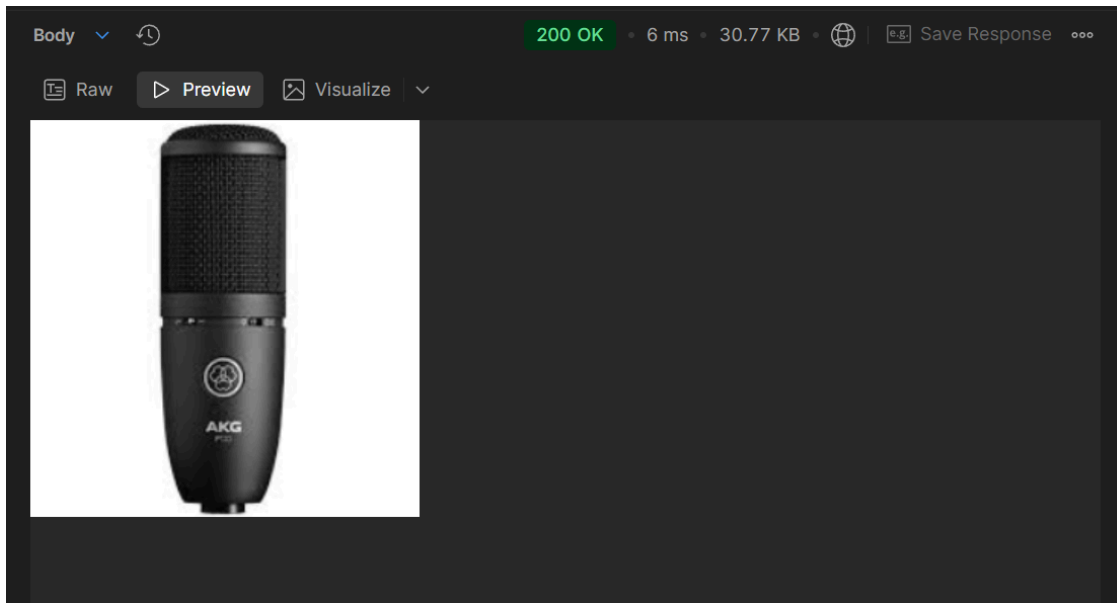
Реалізовано GET /inventory/ Повертає інформацію про конкретну річ Якщо річ не
знайдена повертається 404 Not Found

```
Body 200 OK • 3 ms • 693 B Save Response
{} JSON Preview Visualize
1 {
2   "id": 1,
3   "name": [
4     "AKG P120"
5   ],
6   "description": [
7     "AKG P120 — це конденсаторний студійний мікрофон з чистим і детальним
8     звучанням. Має кардіоїдну спрямованість, надійний металевий корпус і
9     вбудований фільтр для зменшення небажаних шумів. Підходить для домашнього
10    запису вокалу та інструментів."
11  ],
12  "photo": "/cache/re1gteuxpyb2id3211ymtxtd6.png"
13 }
```

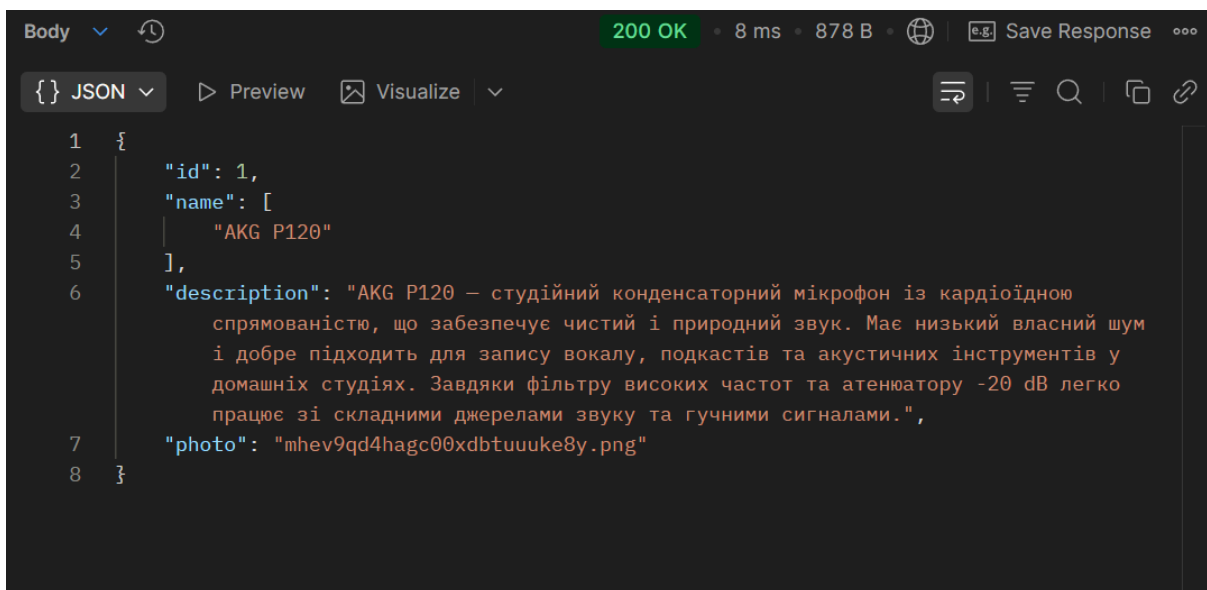
Реалізовано PUT /inventory/ Приймає JSON з новим ім'ям або описом Оновлює відповідні дані Якщо річ не існує повертається 404 Not Found

```
Body 200 OK • 3 ms • 878 B Save Response
{} JSON Preview Visualize
1 {
2   "id": 1,
3   "name": [
4     "AKG P120"
5   ],
6   "description": "AKG P120 — студійний конденсаторний мікрофон із кардіоїдною
7   спрямованістю, що забезпечує чистий і природний звук. Має низький власний шум і
8   добре підходить для запису вокалу, подкастів та акустичних інструментів у
9   домашніх студіях. Завдяки фільтру високих частот та атенюатору -20 dB легко
10  працює зі складними джерелами звуку та гучними сигналами.",
11  "photo": "/cache/re1gteuxpyb2id3211ymtxtd6.png"
12 }
```

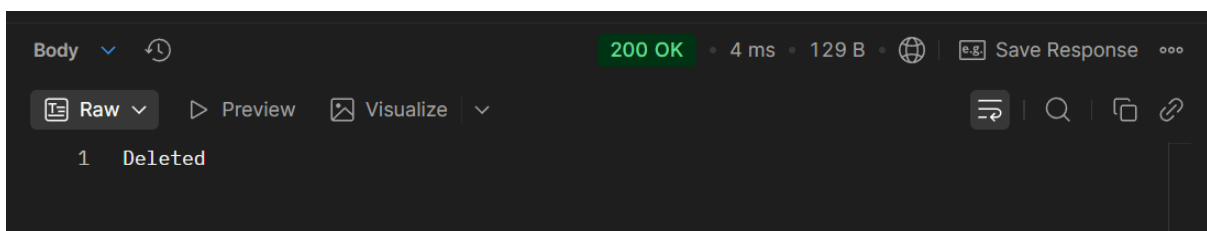
Реалізовано GET /inventory//photo Повертає зображення речі Додається заголовок Content-Type image/jpeg Якщо фото або річ не існує повертається 404 Not Found



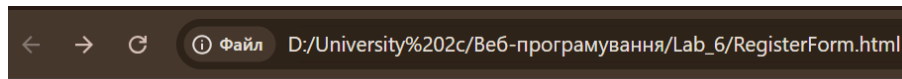
Реалізовано PUT /inventory//photo Приймає multipart form data з фото Оновлює файл фото у кеші Якщо річ не існує повертається 404 Not Found



Реалізовано DELETE /inventory/ Видаляє річ зі списку Якщо ID неправильний повертається 404 Not Found



Реалізовано GET /RegisterForm.html та GET /SearchForm.html HTML форма реєстрації пристрою HTML форма пошуку за ID з прапорцем has_photo

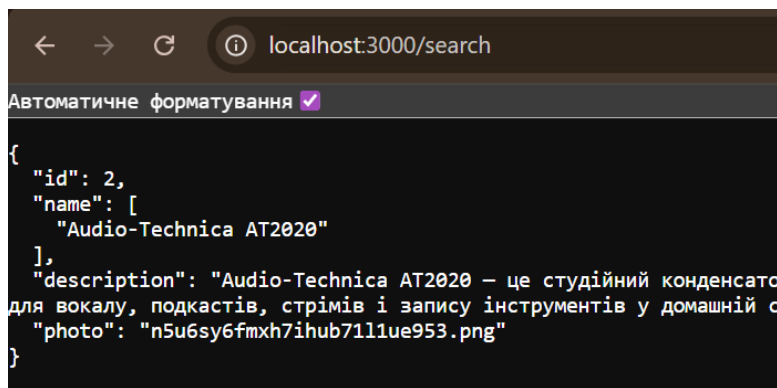


Register New Inventory Item

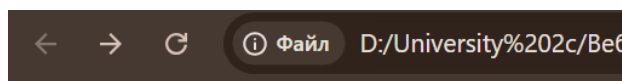
Name (required):

Description:

Photo:

 Знімок ек...142648.png

Реалізовано POST /search Приймає дані у форматі x www form urlencoded Обробляє id та has_photo Повертає знайдений елемент або 404 Not Found Якщо прапорець увімкнений додає посилання на фото до опису



Search Inventory Item

Item ID:

☒ Add photo link to description

```
localhost:3000/search
Автоматичне форматування
{
  "id": 2,
  "name": [
    "Audio-Technica AT2020"
  ],
  "description": "Audio-Technica AT2020 – це студійний конденсатор для вокалу, подкастів, стрімів і запису інструментів у домашній студії",
  "photo": "n5u6sy6fmxh7ihub711ue953.png"
}
```

Додано обробку неправильних методів Для будь якого іншого методу повертається 405 Method Not Allowed

Частина 3 - тестування та документування сервісу

Додано Swagger документацію (/docs) Створено ендпоінт /swagger.json Створено інтерфейс Swagger UI на /docs

Inventory Service 1.0.0 OAS 3.0

[/swagger.json](#)

API Documentation for Inventory Service (Lab 6)

default

| | | | |
|--------|-----------------------|-------------------|---|
| POST | /register | Register new item | ⌵ |
| GET | /inventory | Get all items | ⌵ |
| GET | /inventory/{id} | Get item by ID | ⌵ |
| PUT | /inventory/{id} | Update item info | ⌵ |
| DELETE | /inventory/{id} | Delete item | ⌵ |
| GET | /inventory/{id}/photo | Get item photo | ⌵ |
| PUT | /inventory/{id}/photo | Update item photo | ⌵ |
| POST | /search | Search item | ⌵ |

Docker підтримка Створено Dockerfile Перевірено запуск сервера всередині контейнера

Після реалізації кожного ендпоінта було виконано тестування та зроблено окремий коміт

Висновок: У ході виконання лабораторної роботи №6 було створено веб-сервіс інвентаризації з підтримкою різних типів HTTP-запитів, HTML-форм, пошуку, роботи з файлами та документації Swagger. Сервіс було повністю протестовано за допомогою Postman та запущено у Docker.

Під час виконання виникло дві основні проблеми. Перша — помилковий порядок методів у сервері, через що деякі маршрути перекривали інші (наприклад, `/inventory/:id` перехоплював запити до `/inventory/:id/photo`). Після впорядкування маршрутів сервер почав працювати коректно. Друга — Docker не запускався через помилку імпорту, оскільки Node.js у контейнері не розпізнавав деякі модулі. Проблема було вирішено шляхом додавання параметра `"type": "module"` у `package.json`, після чого контейнер успішно запустився.

Лабораторна робота виконана повністю, усі вимоги реалізовано.