Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп'ютерних технологій, автоматики та метрології Спеціальність «Комп’ютерна Інженерія»

Кафедра СКС

Зображення, що містить текст, коло, Шрифт, емблема

Автоматично згенерований опис

**Звіт**

з лабораторної роботи №5

з дисципліни «Діагностика комп’ютерних засобів»

на тему «Тестування навантаження FastAPI-сервера та

аналіз продуктивності»

**Варіант №5**

Виконав:

Ст.гр KI-307

Ващишин І.І.

Прийняла:  
Клим Г. І.

Львів 2025

**Тема:** Тестування навантаження FastAPI-сервера та аналіз продуктивності.

**Мета:** Розробити простий сервер на FastAPI з трьома ендпоінтами та

провести тестування його продуктивності під навантаженням. Оцінити

продуктивність сервера, використовуючи інструменти тестування

навантаження, та зробити висновки на основі отриманих результатів.

**Теоретична частина**

Тестування навантаження FastAPI-сервера є ключовим етапом у

забезпеченні стабільності, масштабованості та ефективності веб-додатків і

API-сервісів. FastAPI відомий своєю високою продуктивністю завдяки

використанню асинхронної архітектури та підтримці Python asyncio, але без

ретельного тестування та аналізу продуктивності можуть виникнути

проблеми, пов’язані із перевантаженням сервера, некоректною роботою

асинхронних процесів або неефективним використанням ресурсів.

Тестування навантаження дозволяє визначити максимальну кількість

запитів, яку сервер може обробляти без деградації продуктивності. Це

допомагає виявити вузькі місця, такі як повільні ендпоінти, блокуючі операції,

перевантаження CPU, RAM або дискової підсистеми, що може негативно впливати на загальну продуктивність системи. Проведення навантажувальних

тестів із використанням таких інструментів, як Locust, K6 або Apache JMeter,

дозволяє змоделювати реальні сценарії використання сервера, наприклад

високе одночасне навантаження або тривале утримання з’єднань.

Аналіз продуктивності FastAPI-сервера також включає моніторинг часу

відповіді API, використання ресурсів та ефективності кешування. Наприклад,

якщо сервер починає відповідати із затримкою при збільшенні навантаження,

це може свідчити про проблеми з обмеженнями I/O, неефективним

управлінням підключеннями до бази даних або некоректним асинхронним

кодом. Оптимізація таких аспектів, як обробка запитів у фонових задачах,

використання кешу (Redis, Memcached) та балансування навантаження,

допомагає підвищити продуктивність сервера.

**Завдання:**

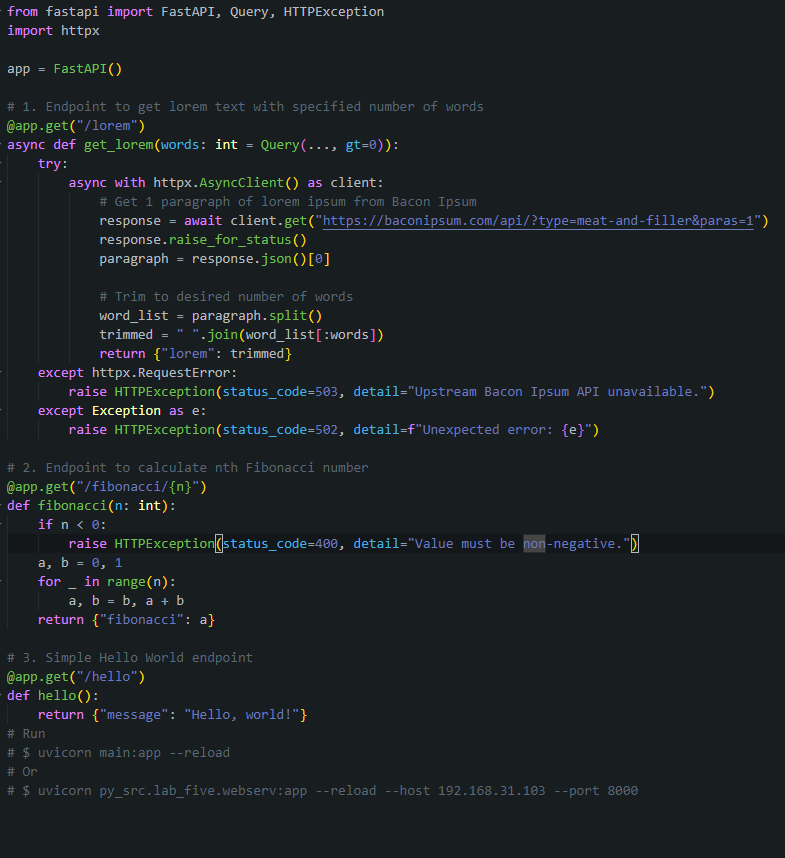
1. Встановити та налаштувати streamlit та pandas.

2. Написати Python-скрипт, який виводить інформацію про систему на UI.

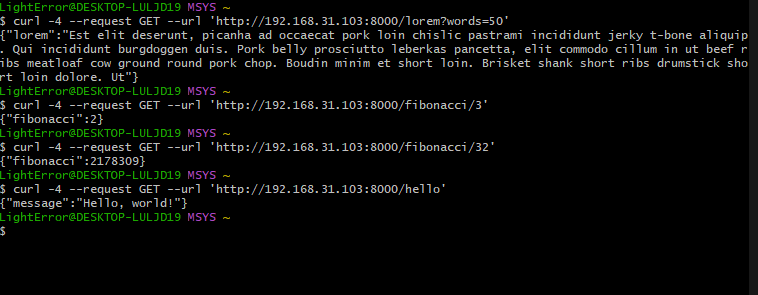
3. Проаналізувати результати, зробити висновки.

4. Оформити звіт.

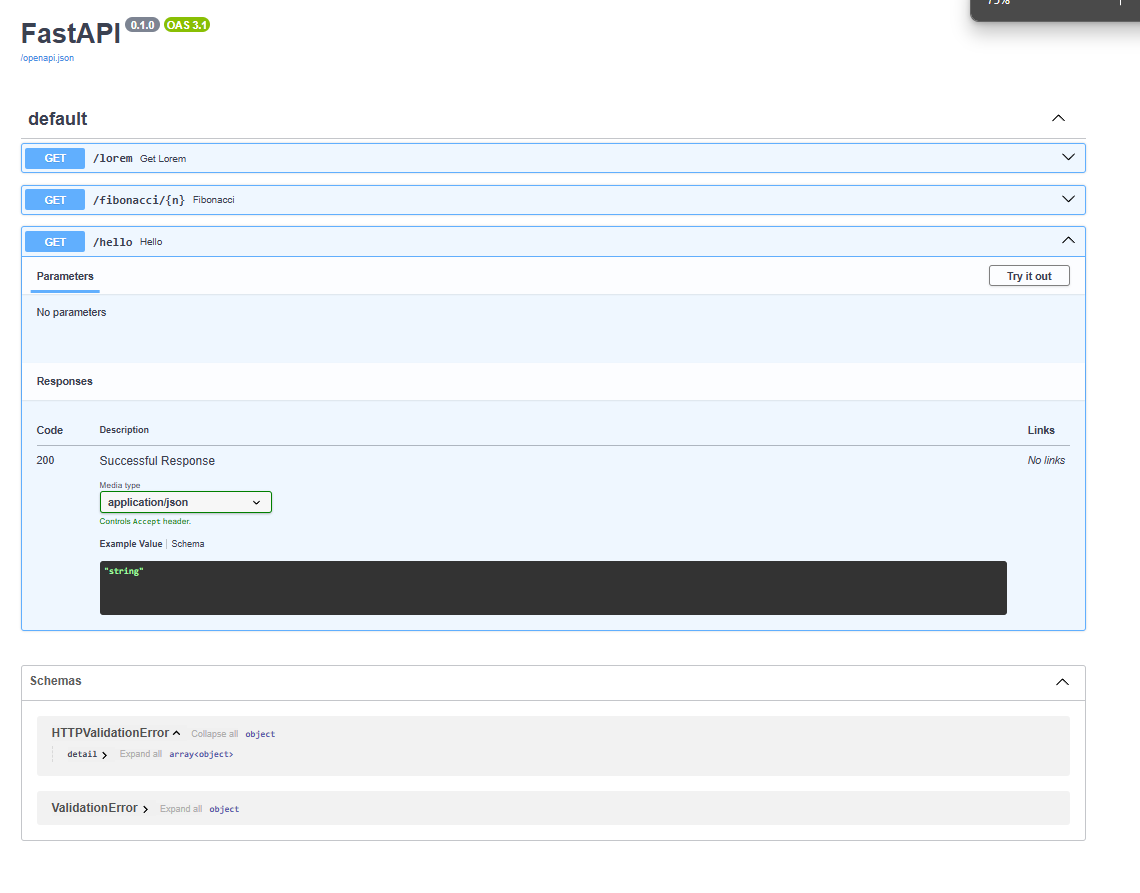
**Вихідний код**



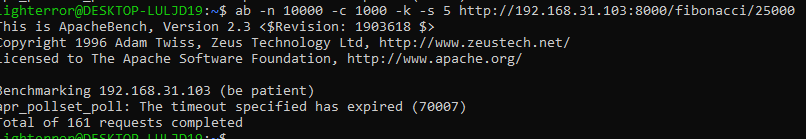
**Результат виконання:**



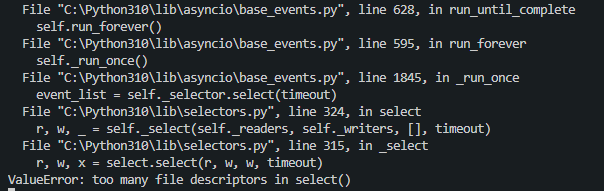
**Приклад Генрації Swagger (OpenApi 3.1) документації**

****

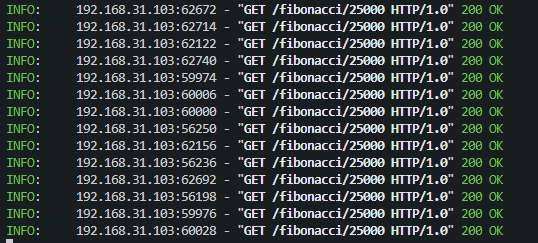
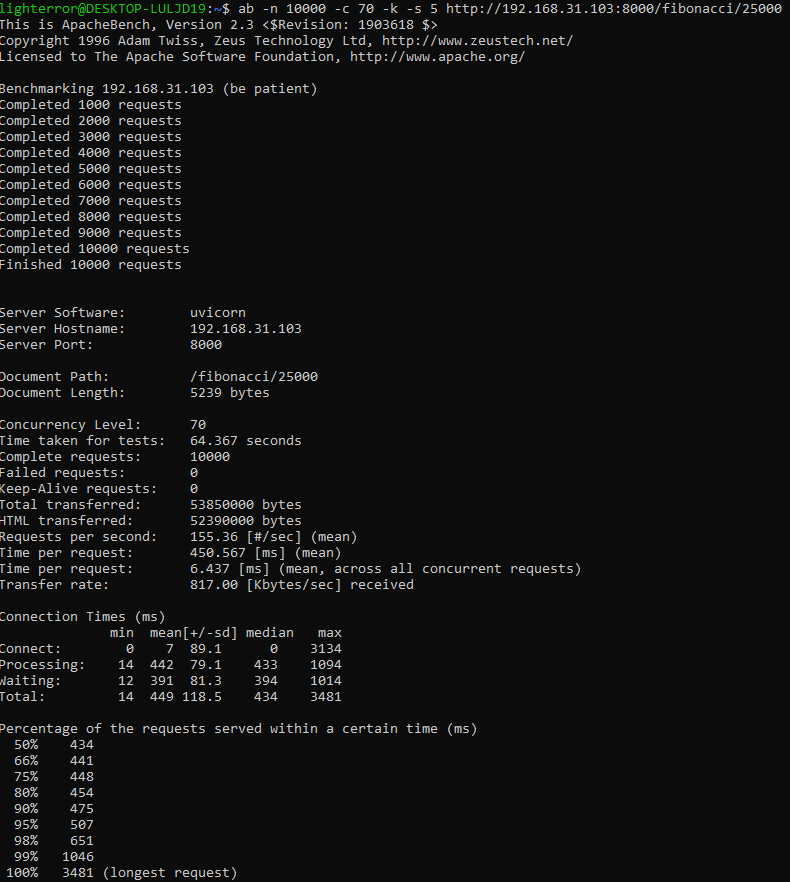
**Тести:**

**1. Apac****he Benchmark -с 1000 -n 1000. Виведення:**

Нажаль сервер не витримав (Причина це кількість відкритих сокетів, а не проблеми продуктивності Python)**:**

****

**2. Apache Benchmark -с 1000 -n 70. Виведення:**

**Тепер с сервером все ок:**

**Висновок:** На цій лабораторній роботі я розробив простий сервер на FastAPI з трьома ендпоінтами та провести тестування його продуктивності під навантаженням. Оцінити продуктивність сервера, використовуючи інструменти тестування навантаження, та зробити висновки на основі отриманих результатів