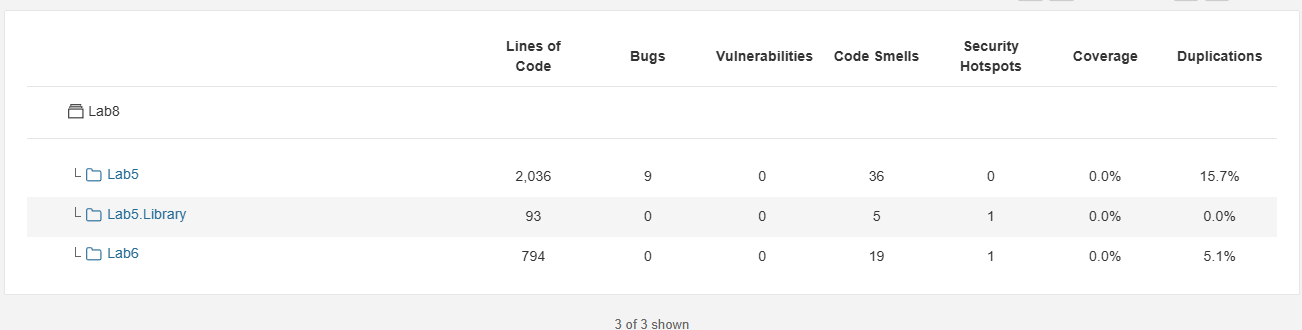
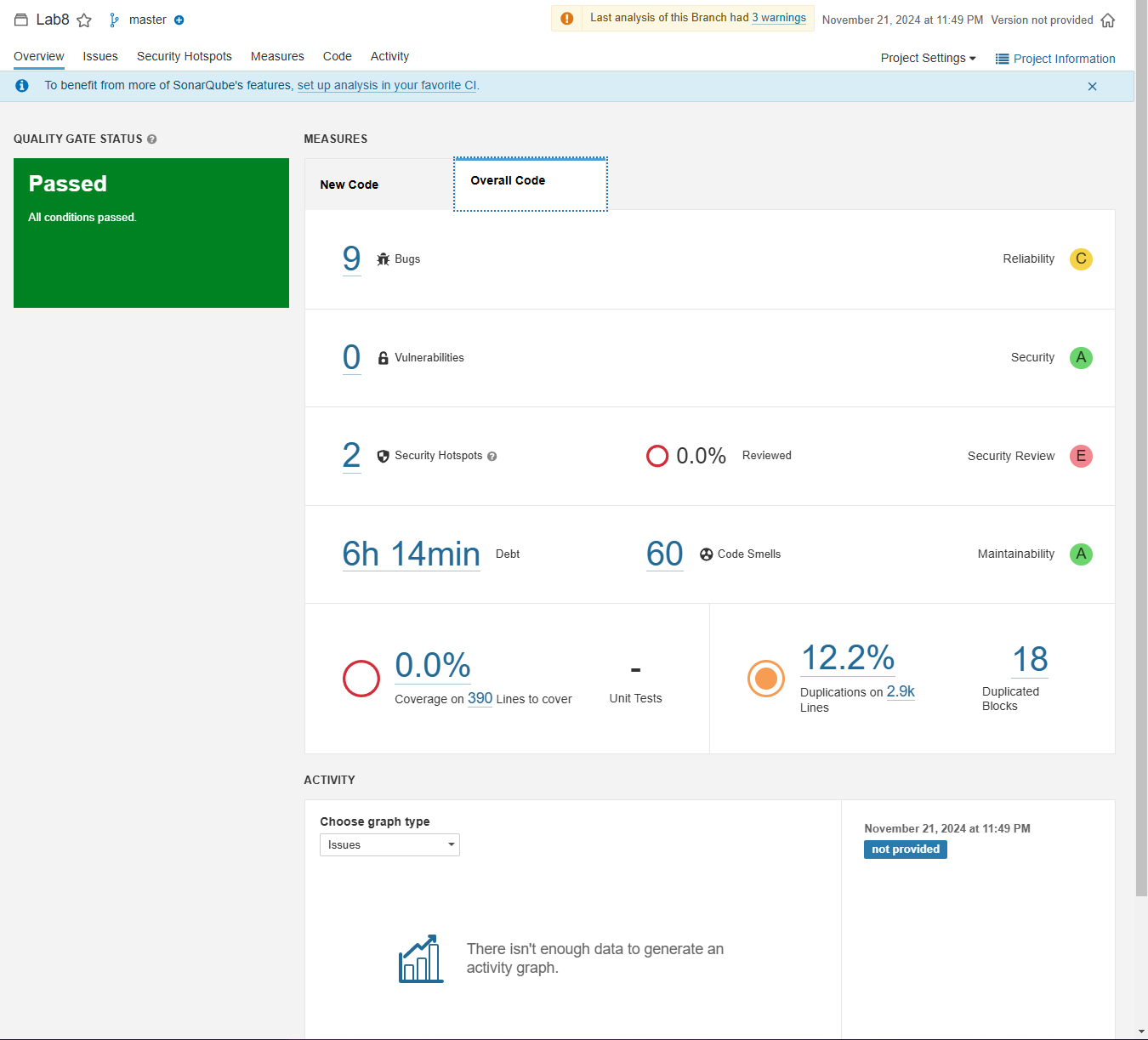
Шевченко Влада ІПЗ-34мс

Лабораторна робота 8

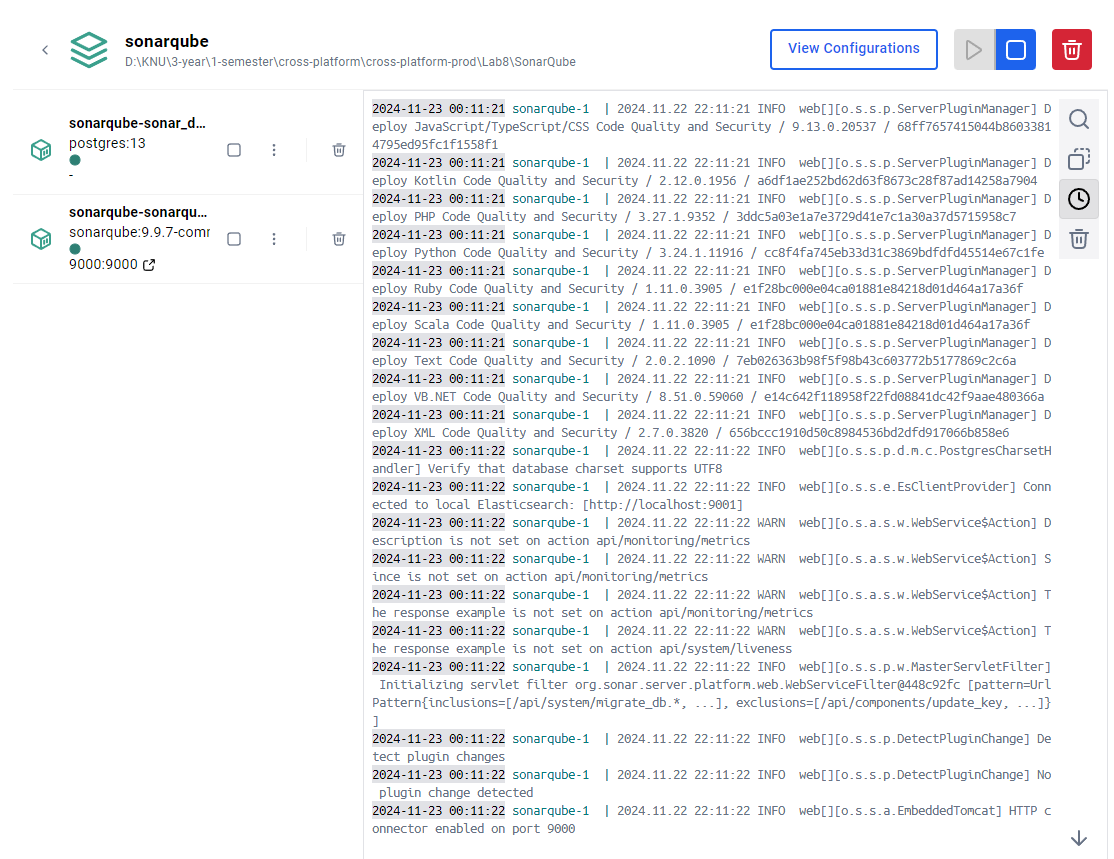
**Звіт**

**Завдання 1.**

Провести оцінку якості коду програмного забезпечення за допомогою SonarQube. Продемонструвати результати аналізу.



Для запуску було створено відповідний докер контейнер:



**Завдання 2:**

Інтегрувати OpenTelemetry в застосунки ПР5 та ПР6. Запустити програму з СУБД MsSQL та зібрати метрики (К-ть обєктів в поколіннях GC, розмір Heap, використані системні ресурси тощо. Все продемонструвати в Grafana, ElasticSearch), розподілений трейс, граф залежностей та логи (Zipkin, ElasticSearch) за допомогою.

*Zipkin, Graphana, Prometheus*

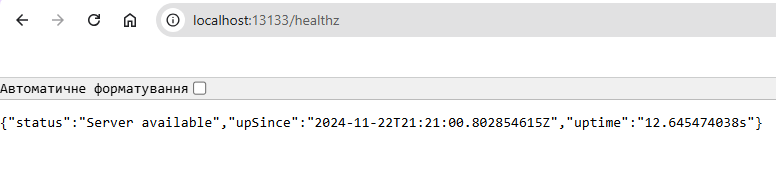
**Завдання 3:**

Додати додаткові можливості в систему трейсингу:

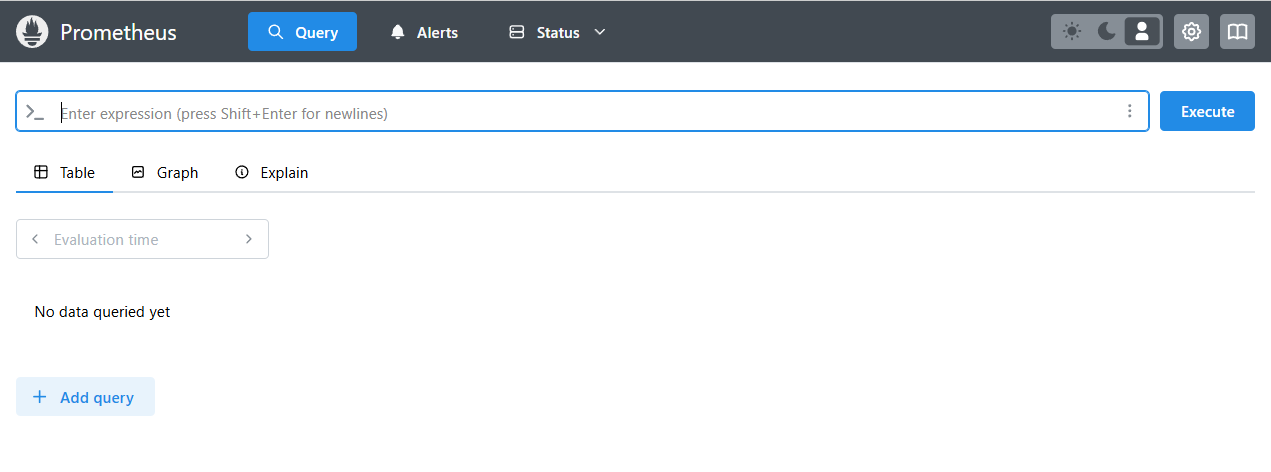
1. Додаткові поля.

2. Додатковий SPAN для зімітованого тривалого процесу.

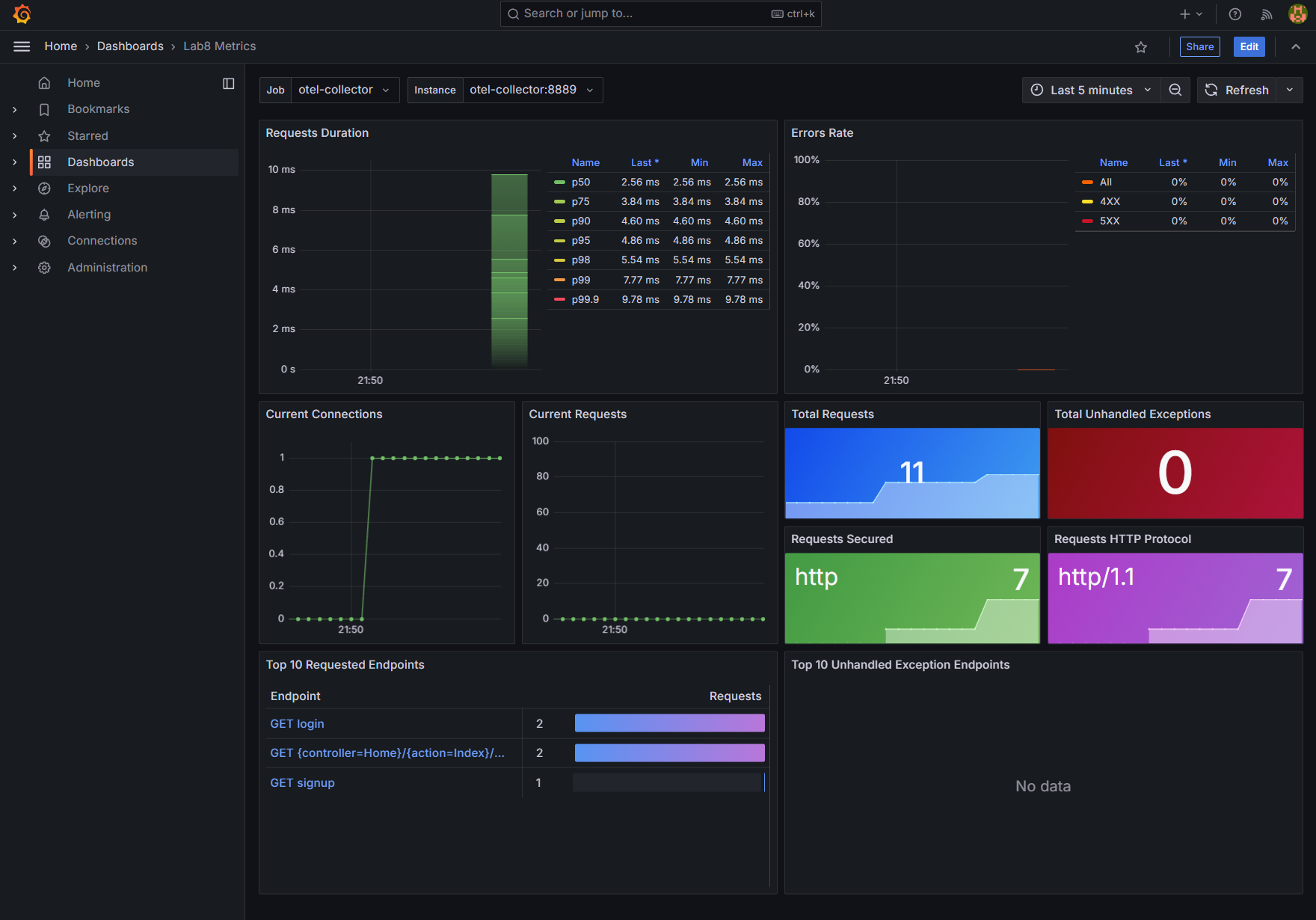
Видно, що сервер працює*:*



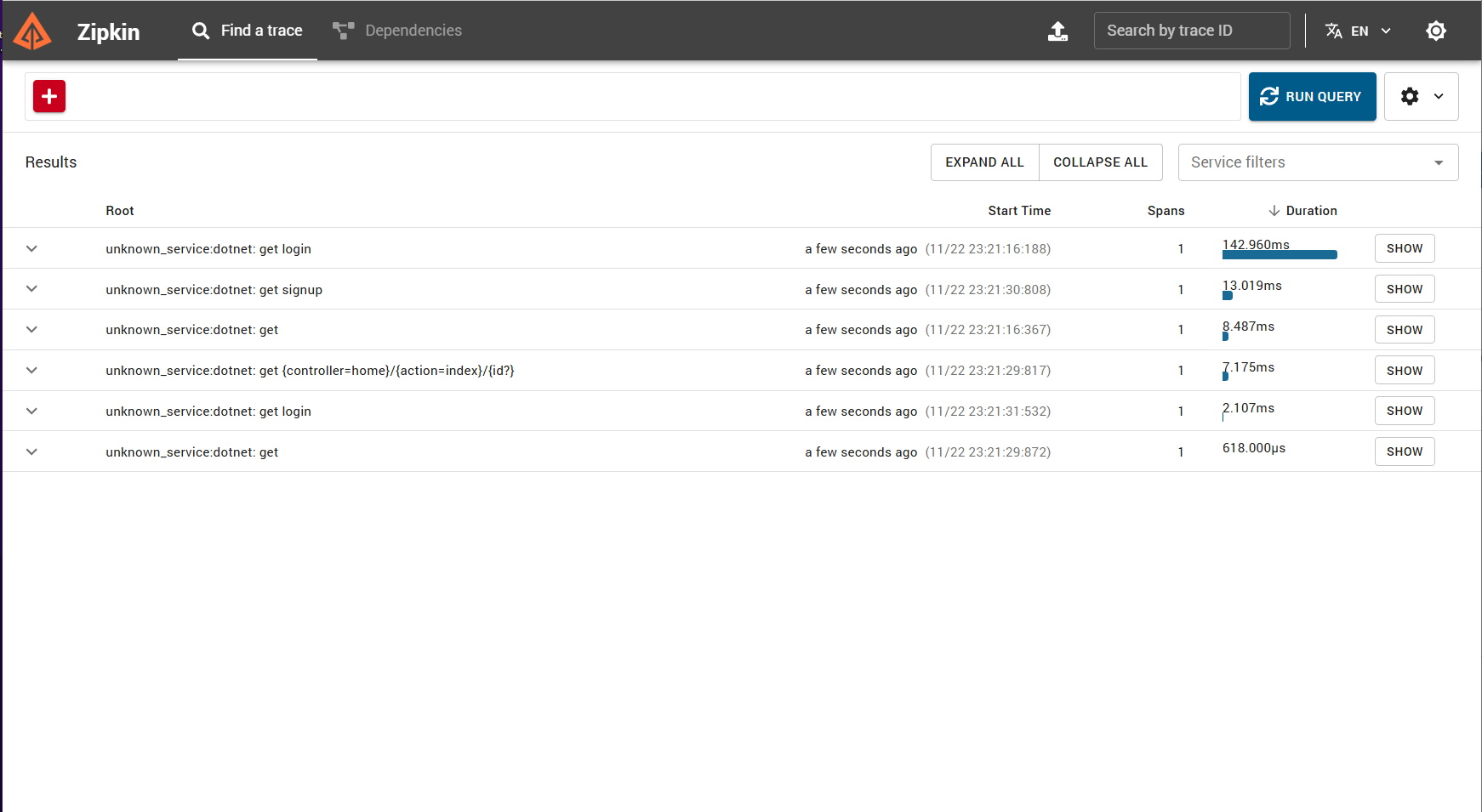
Працюючий *Prometheus:*



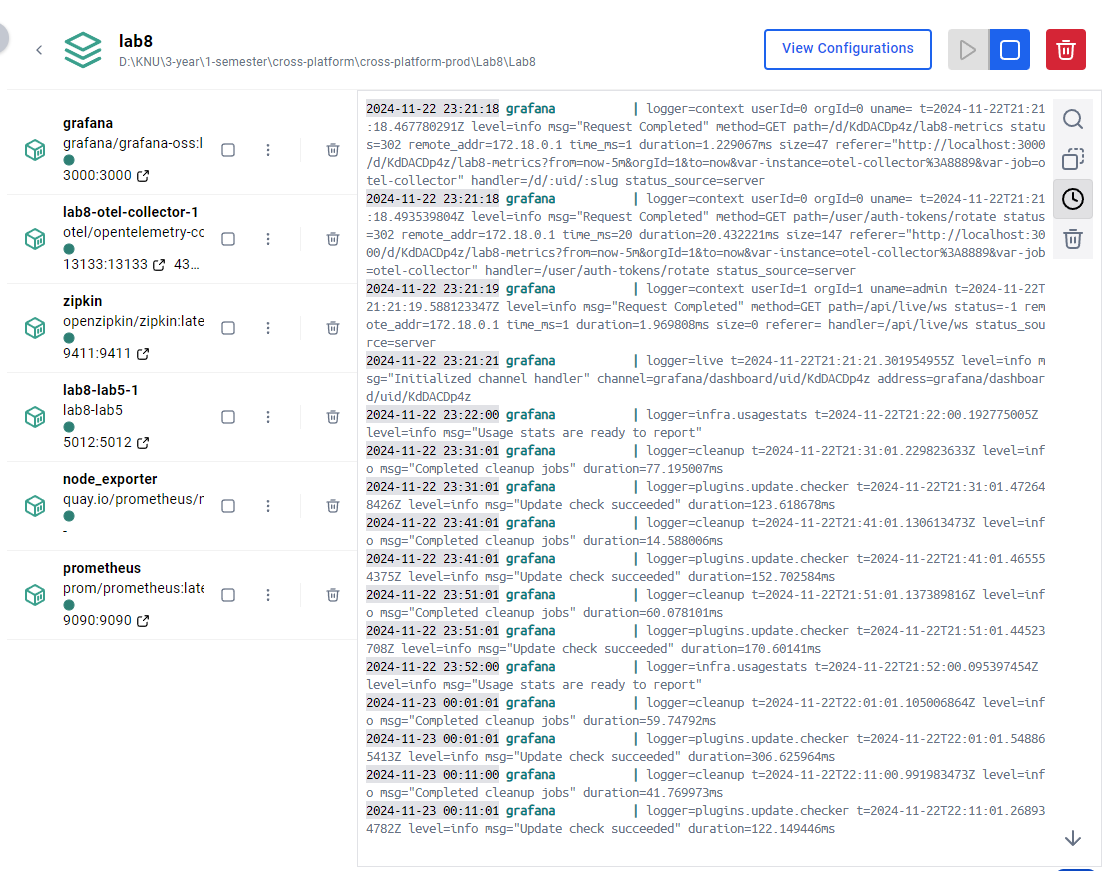
Отримані результати у *Graphana:*



Отримані результати у *Zipkin:*



Для запуску було створено відповідний докер контейнер:



**Завдання 4:**

Додати в СУБД достатьню кількість записів (Seed) X > 10000 в будь-яку з центральних таблиць будь-яким найпростішим способом (наприклад скориставшись згенерованим кліентом створити консольний застосунок для швидкого наповнення). Провести навантажувальне тестування Asp.Net MVC застосунку (відключити Okta на час тестування) за допомогою jMeter+Influx+Grafana. Оцінити поведінку программи для 1,5,20,50,100,300 одночасних користувачів.

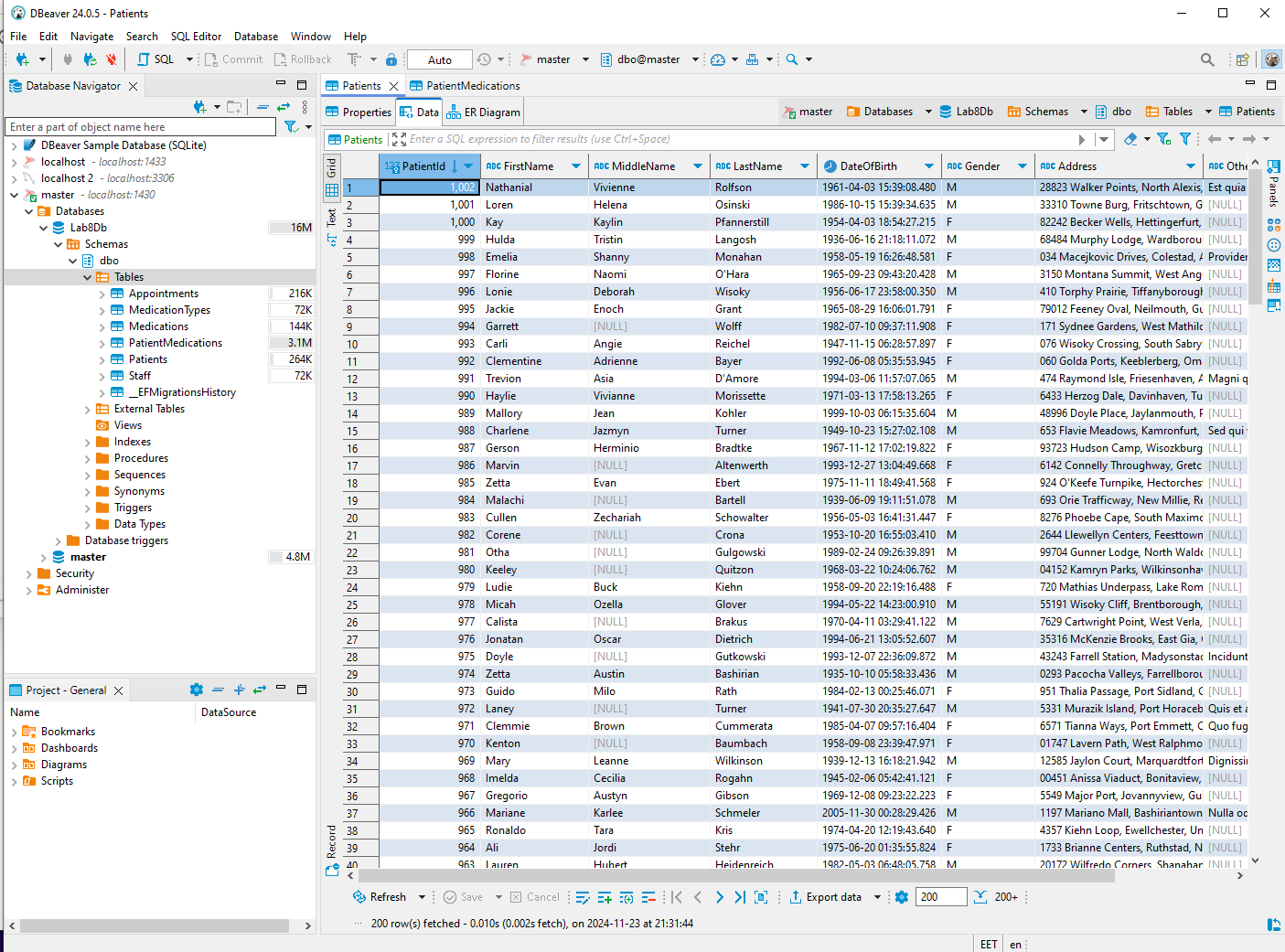
1. Знайти:

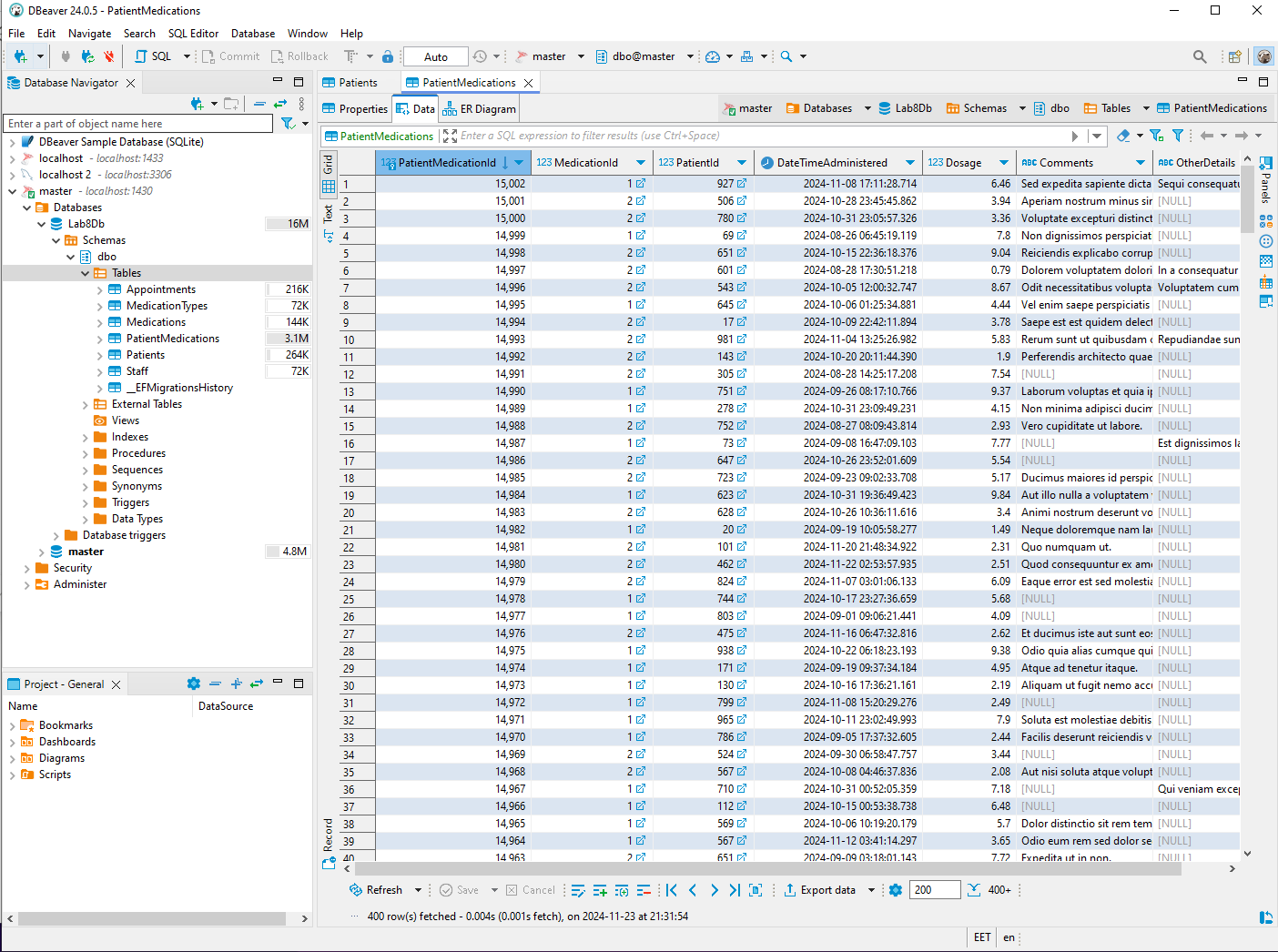
1. Error rate для кожного

2. Ф-ю залежності між к-тю користувачів та часом на відповідь. Дослідити тип ф-ЇЇ – лінійна, квадратична, логарифмічна тощо.

2. За допогою бекенду трасування Zipkin + Graphana + Prometheus знайти та продемонструвати найповільнішу частину застосунків. Запропонувати метод покращення (оптимізуємо що? як? а це можливо прискорити взагалі? ).

Створені 1000 записів у таблицю Patient та 15000 у таблицю PatientMedication



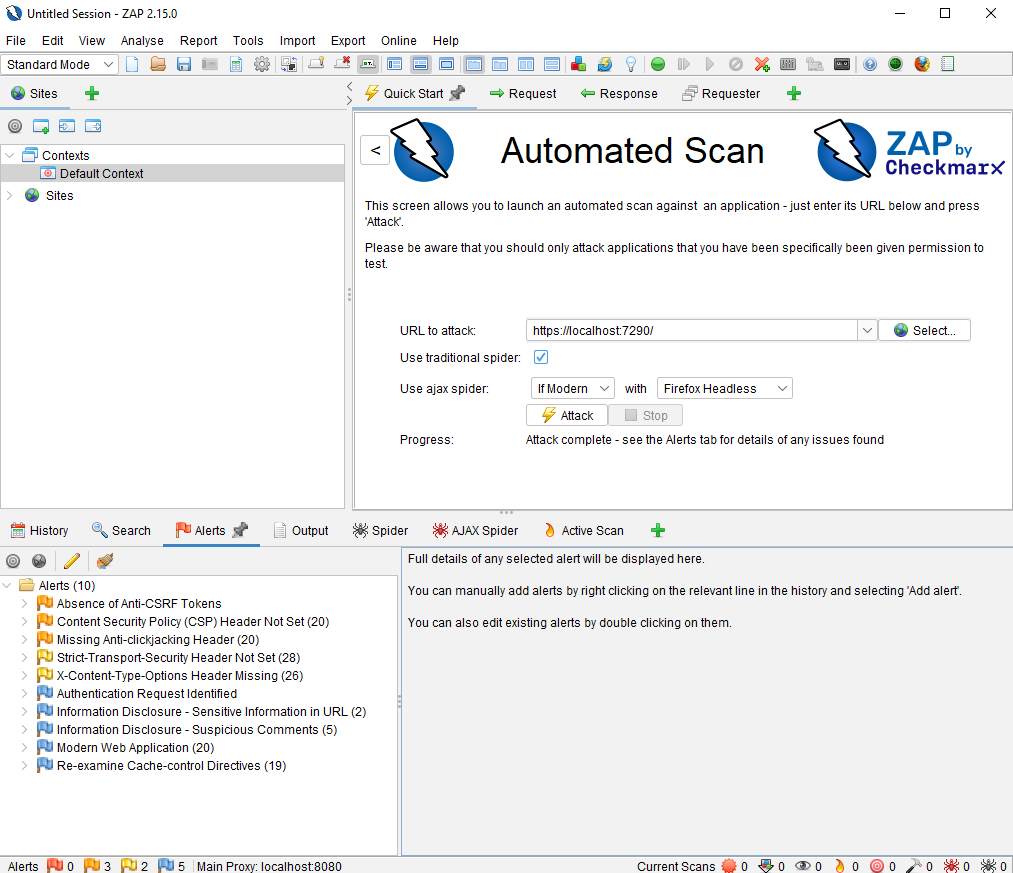


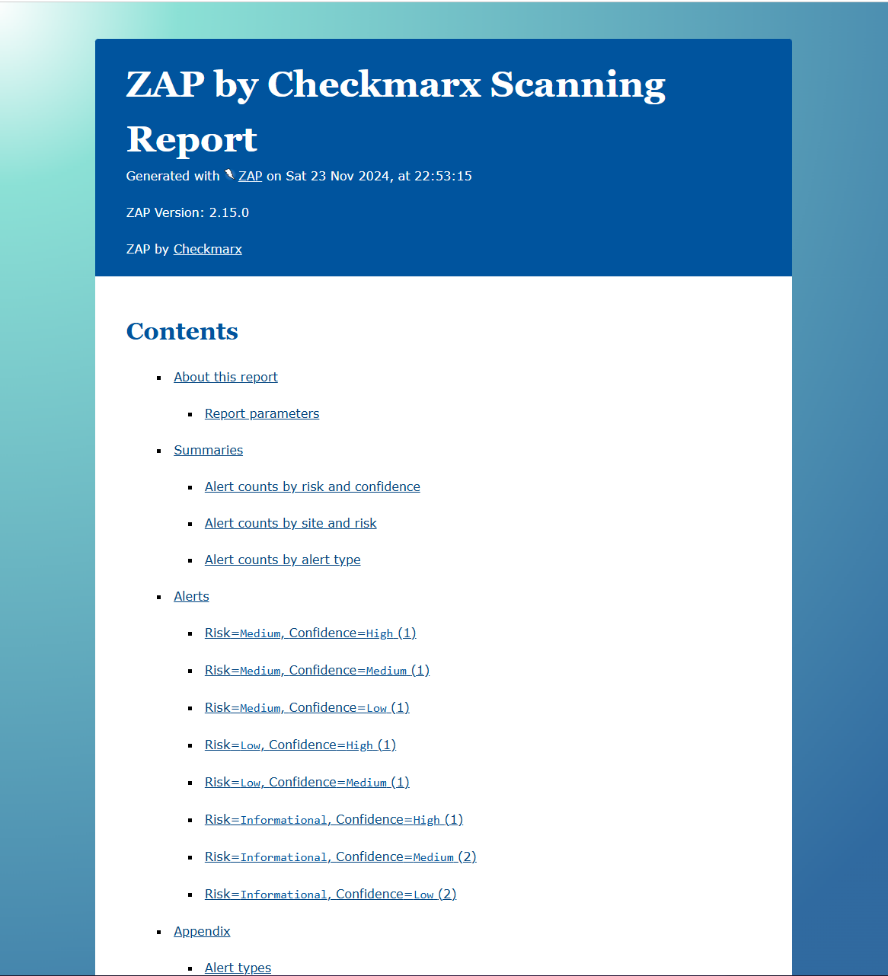
Для подальшого покращення продуктивності Asp.Net MVC застосунку можна також налаштувати:

* кешування даних на рівні контролера або додатку для зменшення кількості запитів до бази даних;
* оптимізацію SQL-запитів через використання індексів та підготовлених запитів;
* застосувати асинхронні методи у контролерах для покращення масштабованості;
* впровадити пагінацію для API, що повертає великі набори даних, а також системи моніторингу і профілювання для своєчасного виявлення проблем у продуктивності.

**Завдання 5:**

Провести тестування застосунку на безпечність використовуючи OWASP ZAP. Продемонструвати звіт.





Повний звіт у репозиторії, у папці Lab8/OWASP\_ZAP: <https://github.com/vladashvch/cross-platform-labs/Lab8/OWASP_ZAP>