**Лабораторна робота**

з дисципліни **«Інформаційні системи»**

**Тема:** Прикладна інформаційна система *«Система моніторингу роботи СУБД»*

**Виконав:** студент групи ІПС-42  
Гоздецький Артур Романович

**Мета роботи**

**Розробити прикладну інформаційну систему для моніторингу роботи СУБД із використанням інструментів збору метрик та візуалізації даних (Prometheus, Grafana), а також реалізувати авторизацію користувачів і контейнеризацію сервісів.**

**Завдання**

1. **Розгорнути СУБД (MySQL / MongoDB / PostgreSQL) у Docker-контейнері.**
2. **Реалізувати генерацію актуалізації даних у БД за допомогою програмного засобу (Java Spring або Python).**
3. **Налаштувати збір даних у Prometheus.**
4. **Забезпечити збереження даних у Time Series Storage Prometheus.**
5. **Візуалізувати дані у Grafana.**
6. **Реалізувати механізм авторизації доступу до системи.**
7. **Виконати контейнеризацію всіх сервісів.**
8. **Використати RabbitMQ або Apache Kafka для обміну повідомленнями між сервісами**

**Виконання роботи**

**1. Розгортання СУБД**

**Було використано MySQL, запущений у Docker-контейнері.**

**2. Генерація даних**

**Розроблено Java Spring Application, що дозволяє створювати користувачів у БД через HTTP-запити.**

* **Для тестування запитів застосовувався Postman.**
* **Приклади API:**
  + **POST /auth – аутентифікація користувача**
  + **GET /users – отримати список користувачів**
  + **DELETE /users – видалити всіх користувачів**
  + **POST /users/generate/{n} – створити n користувачів із випадковими даними**

**3. Prometheus**

**Було налаштовано Prometheus для збору метрик зі Spring-застосунку.  
Фрагмент конфігурації prometheus.yml:**

**Изображение выглядит как текст, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.**

**Доступ до веб-інтерфейсу Prometheus:  
http://localhost:9090**

**4. Grafana**

* **UI доступний за посиланням: http://localhost:3000**
* **Вхід: admin / admin**
* **Налаштовано підключення до Prometheus як джерела даних.**
* **Створено Dashboard із власною метрикою users\_count (кількість користувачів у БД).**

**5. Авторизація та безпека**

* **Для Spring-застосунку реалізовано веб-фільтр, який перевіряє аутентифікацію користувачів.**
* **Авторизація виконується за запитом:  
  http://localhost:8080/auth**
* **Для Prometheus та Grafana використано Basic Auth:**

**Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.**

**6. Кастомні метрики**

**Розроблено власну метрику:**

* **users\_count – кількість користувачів у БД.**
* **Prometheus збирає ці дані, а Grafana візуалізує у вигляді графіка.**

**7. Контейнеризація**

**Було контейнеризовано всі компоненти:**

* **Prometheus**
* **Grafana**
* **Java Spring Application**
* **MySQL**

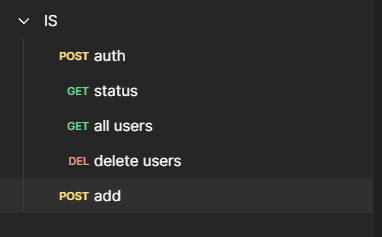
**Скріншоти лабораторної роботи**

**Вигляд контейнерів в Docker Desktop (можна побачити назви контейнерів та порти):**

**Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.**

**Колекція запитів постману:**

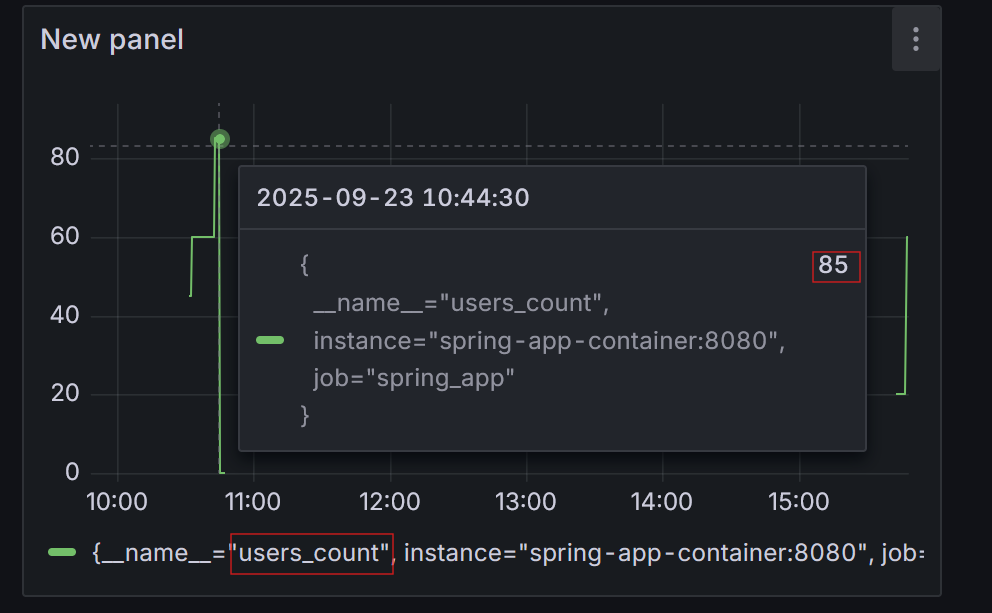
****

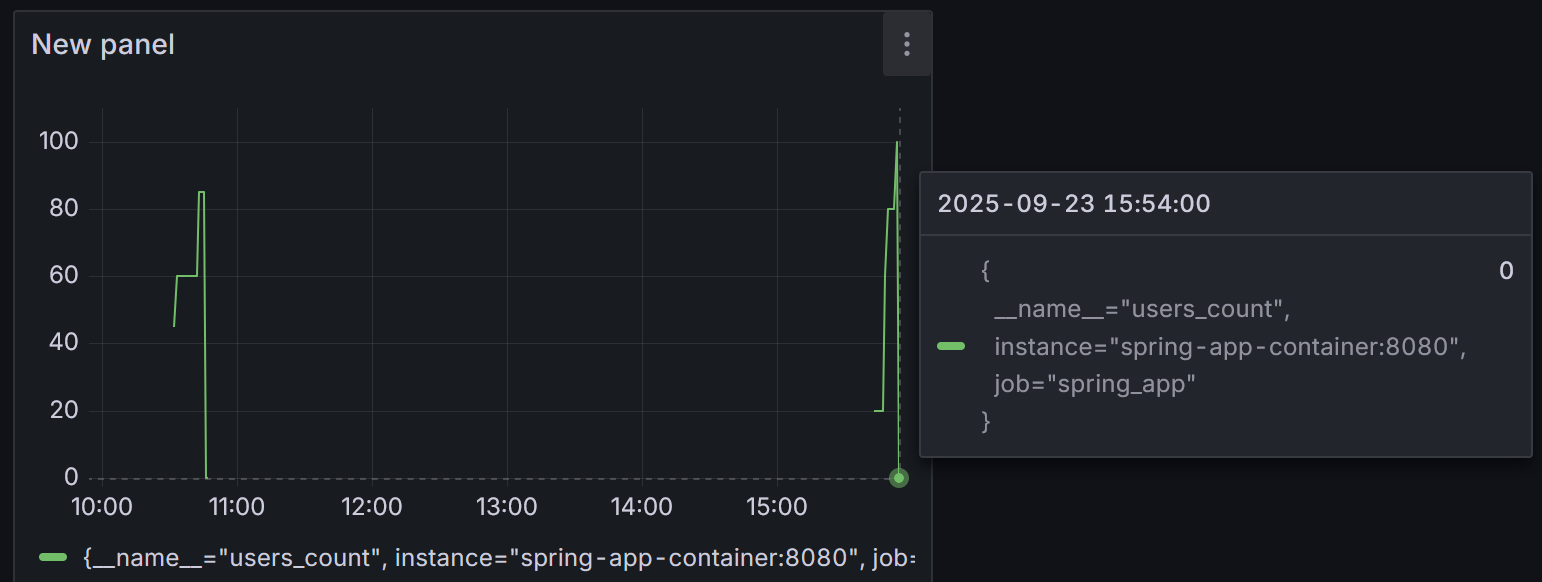
**Приклад створення юзерів. Можна побачити запит та створених юзерів.**

**Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.**

**Вигляд нашої кастомної метрики users\_count в Grafana:**

****

**А тепер давайте спробуємо додати якусь кількість юзерів та видалити їх всіх   
  
**

**Можемо побачити, що наш графік працює корекно, адже кількість юзерів підросла та знизилася до 0.**

**Також, можна помітити під графіком instance="spring-app-container:8080". Це значіть що Prometheus бере дані з нашого застосунку spring-app та Grafana відображає Prometheus.**

**Інформація для старту лабораторної роботи у Docker**

**Нижче наведені інструкції для налаштування та запуску необхідних сервісів у Docker: Prometheus, Grafana, MySQL та Spring Application.  
Також створюється спільна мережа для їх взаємодії.**

**1.** **Prometheus**

**cd "Шлях до Prometheus"**

**docker build -t my-prometheus .**

**docker run -d --name prometheus -p 9090:9090 my-prometheus**

**Prometheus буде доступний за адресою:**[**http://localhost:9090**](http://localhost:9090)

**2. Grafana**

**docker run -d -p 3000:3000 --name grafana grafana/grafana**

**Grafana буде доступна за адресою:**[**http://localhost:3000**](http://localhost:3000)

**3. MySQL**

**docker run --name mysql-container `**

**-e MYSQL\_ROOT\_PASSWORD=Artur.14022005 `**

**-e MYSQL\_DATABASE=isdb `**

**-d -p 3306:3306 mysql:8**

**Користувач: root**

**Пароль: Artur.14022005**

**База даних: isdb**

**4. Spring Application**

**cd "Шлях до Spring Application"**

**docker build -t spring-app .**

**docker run --name spring-app-container -p 8080:8080 -d spring-app**

**Spring-додаток буде доступний за адресою:**[**http://localhost:8080**](http://localhost:8080)

**5. Створення мережі та підключення контейнерів**

**docker network create spring-network**

**docker network connect spring-network mysql-container**

**docker network connect spring-network spring-app-container**

**docker network connect spring-network prometheus**

**docker network connect spring-network grafana**

**Результати роботи**

* **Реалізовано повний цикл збору, збереження та візуалізації метрик.**
* **Додано авторизацію користувачів та захист доступу до Prometheus і Grafana.**
* **Створено власну метрику users\_count, яка відображається у Grafana.**
* **Виконано контейнеризацію системи.**

**Висновки**

**У ході роботи було створено прикладну інформаційну систему моніторингу СУБД з використанням Prometheus, Grafana, Docker та Java Spring.  
Налаштування середовища дозволяє легко відтворити проєкт на іншому ПК.**