****

Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №3

**Тенденції розвитку інформаційних систем та технологій**

*Централізовані системи логування. EFK.*

|  |  |  |
| --- | --- | --- |
| Виконав  студент групи ІТ-41ф |  | Перевірив: |
|  |  |  |
| Новиков Д. М. |  |  |
|  |  |  |
|  |  | ас. Цимбал С. І. |

Київ 2024

*Мета роботи:* ознайомлення із централізованими системами логування на прикладі EFK.

*Хід роботи:*

1. Запустити тестовий EFK стек

Оскільки я буду писати Web API на ASP.NET Core, провайдер Serilog взаємодіє з Elasticsearch напряму. Таким чином, зі стеку EFK (Elasticsearch, Fluentd, Kibana) мені не потрібен Fluentd. Його роль збирача та форматувача логів виконує спеціальний компонент Serilog - Sink. Підготуємо Docker-compose.yml для створення ELK:

services:

  elasticsearch:

    image: docker.elastic.co/elasticsearch/elasticsearch:7.17.0

    container\_name: elasticsearch

    environment:

      discovery.type: single-node

      xpack.monitoring.enabled: true

      xpack.watcher.enabled: false

    ports:

      - 9200:9200

      - 9300:9300

    volumes: # Stores elasticsearch data locally on the es\_data Docker volume

      - es\_data:/usr/share/elasticsearch/data

  kibana:

    image: docker.elastic.co/kibana/kibana:7.17.0

    container\_name: kibana

    environment:

      ELASTICSEARCH\_URL: http://elasticsearch:9200

    ports:

      - 5601:5601

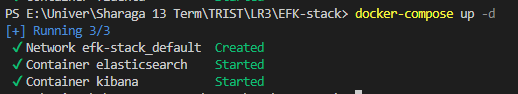
    depends\_on:

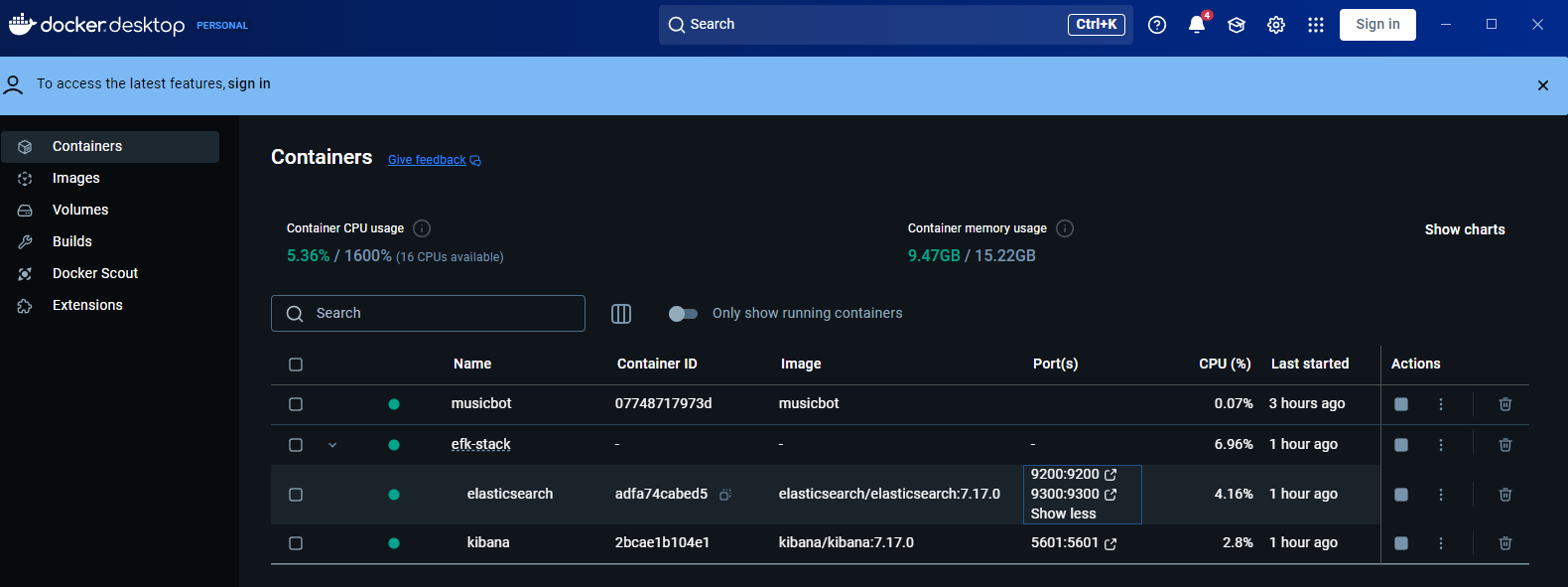
      - elasticsearch

volumes:

  es\_data:

Запустимо стек за допомогою команди docker-compose up:

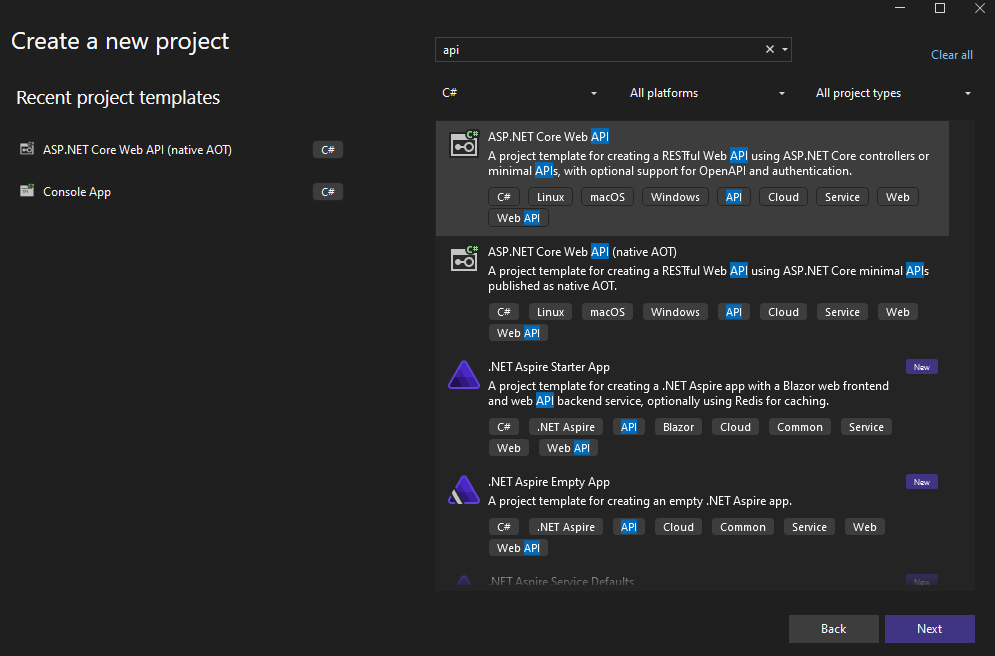


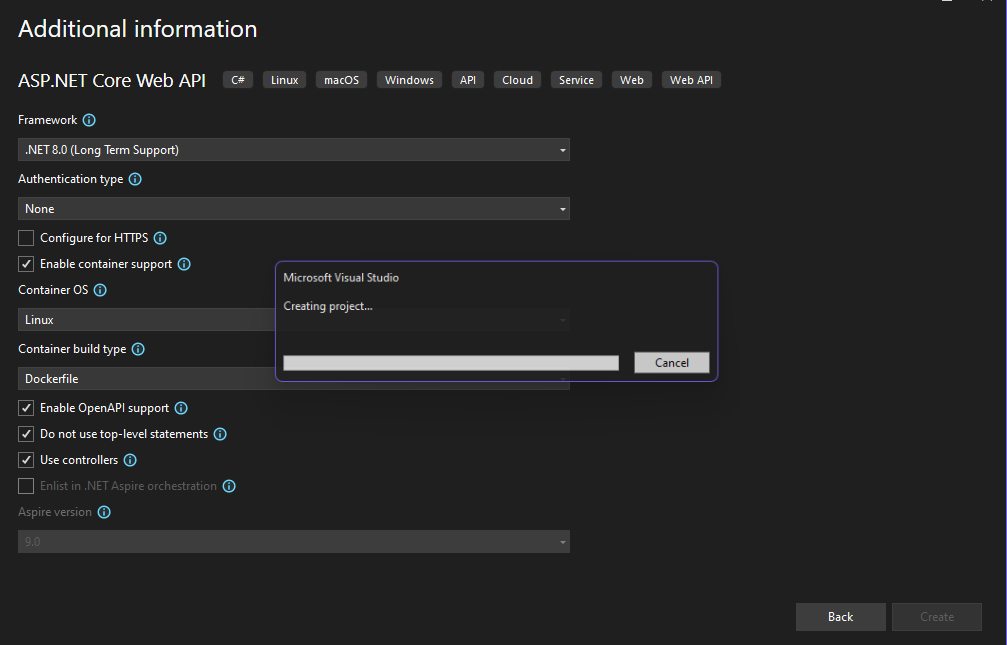


2. Розробити найпростіший застосунок, який буде записувати логи, щоб їх

можна було побачити , фільтрувати у Kibana.

Створимо Web API на ASP.NET Core, використовуючи стандартний шаблон:





1. Додамо необхідні залежності:

* Serilog;
* Serilog.AspNetCore;
* Serilog.Sinks.Console;
* Serilog.Sinks.Elasticsearch.

1. У файлі Program.cs налаштуємо конфігурацію для Serilog, а також додамо кілька логів:

public class Program

{

public static void Main(string[] args)

{

// Configure Serilog using appsettings.json.

Log.Logger = new LoggerConfiguration()

.ReadFrom.Configuration(new ConfigurationBuilder()

.AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)

.Build())

.CreateLogger();

try

{

Log.Information("Starting the application.");

var builder = WebApplication.CreateBuilder(args);

…

app.Run();

}

catch (Exception ex)

{

Log.Fatal(ex, "Application startup failed.");

}

finally

{

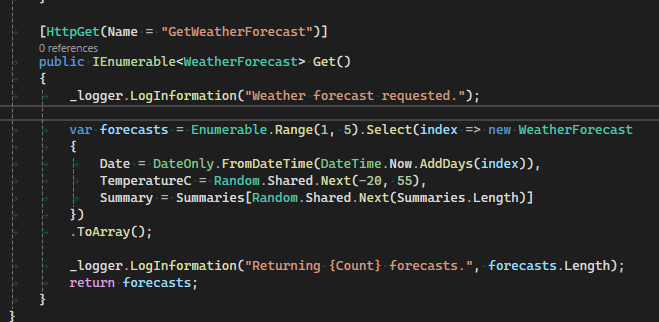
Log.CloseAndFlush();

}

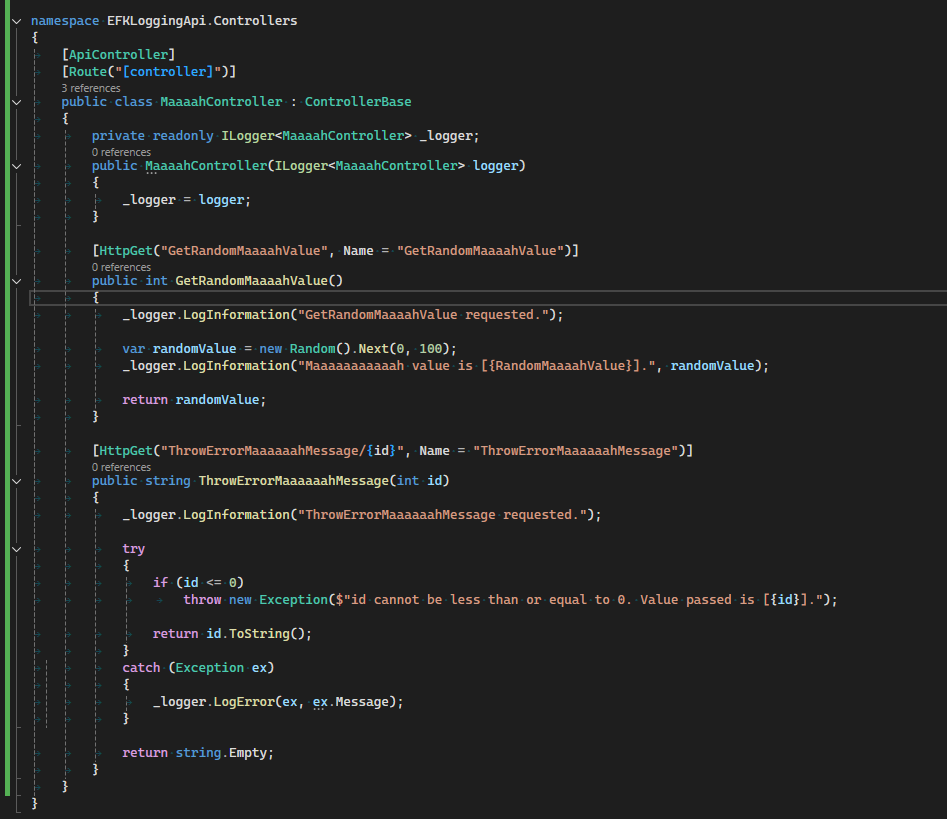
}

}

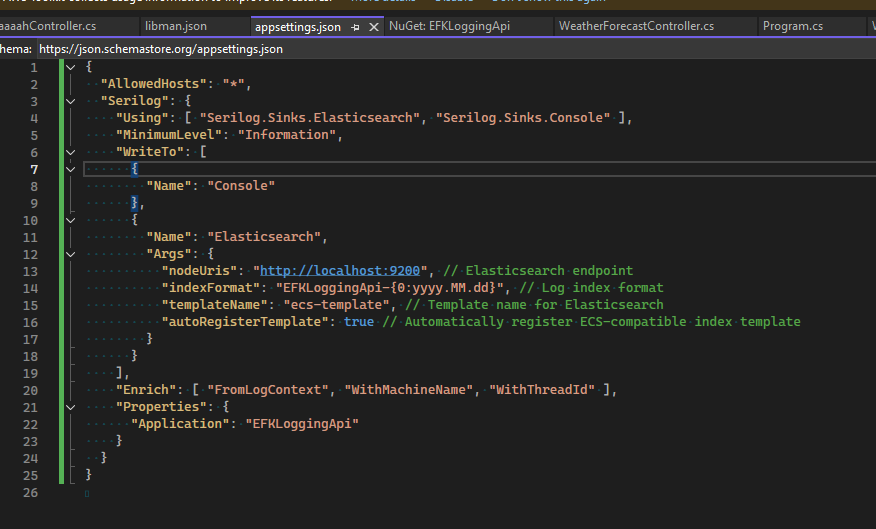
1. Модифікуємо існуючий контролер WeatherForecastController, додавши логування на початку виклику методу та перед поверненням результату:



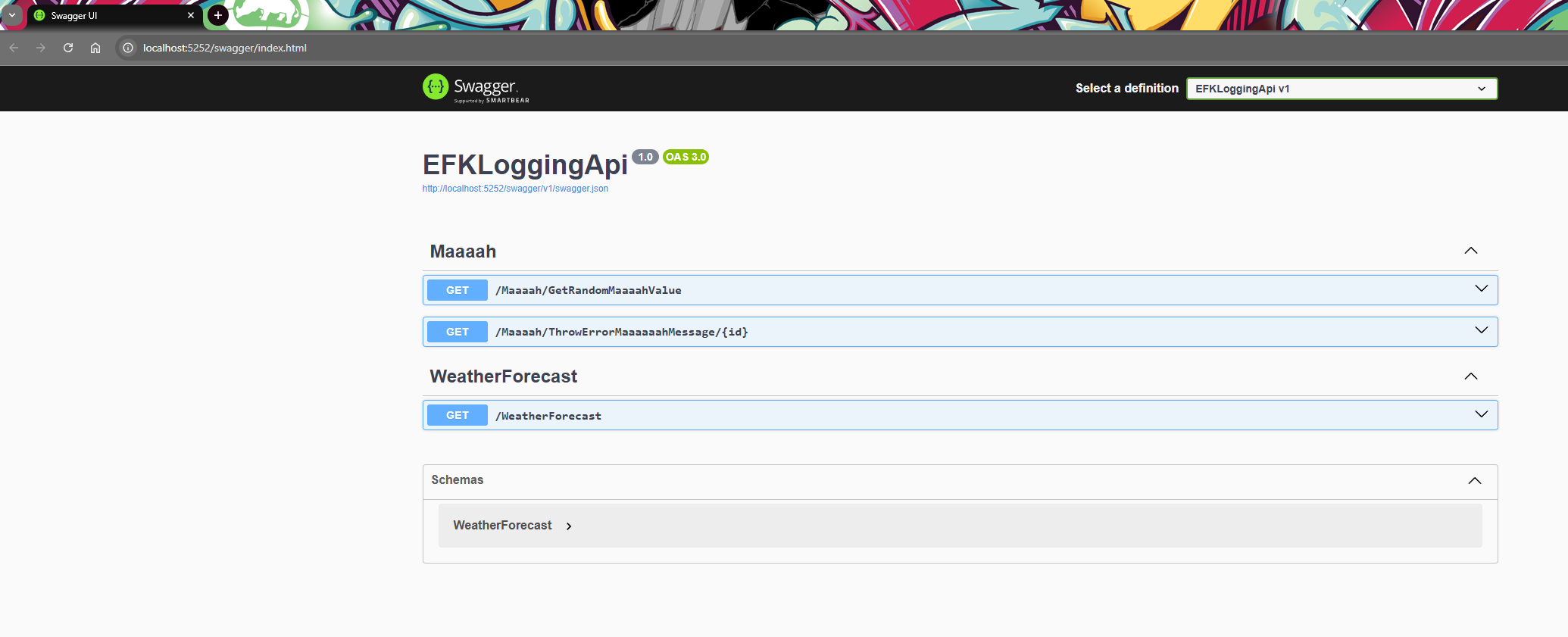
1. Створимо додатковий контролер для тестування:

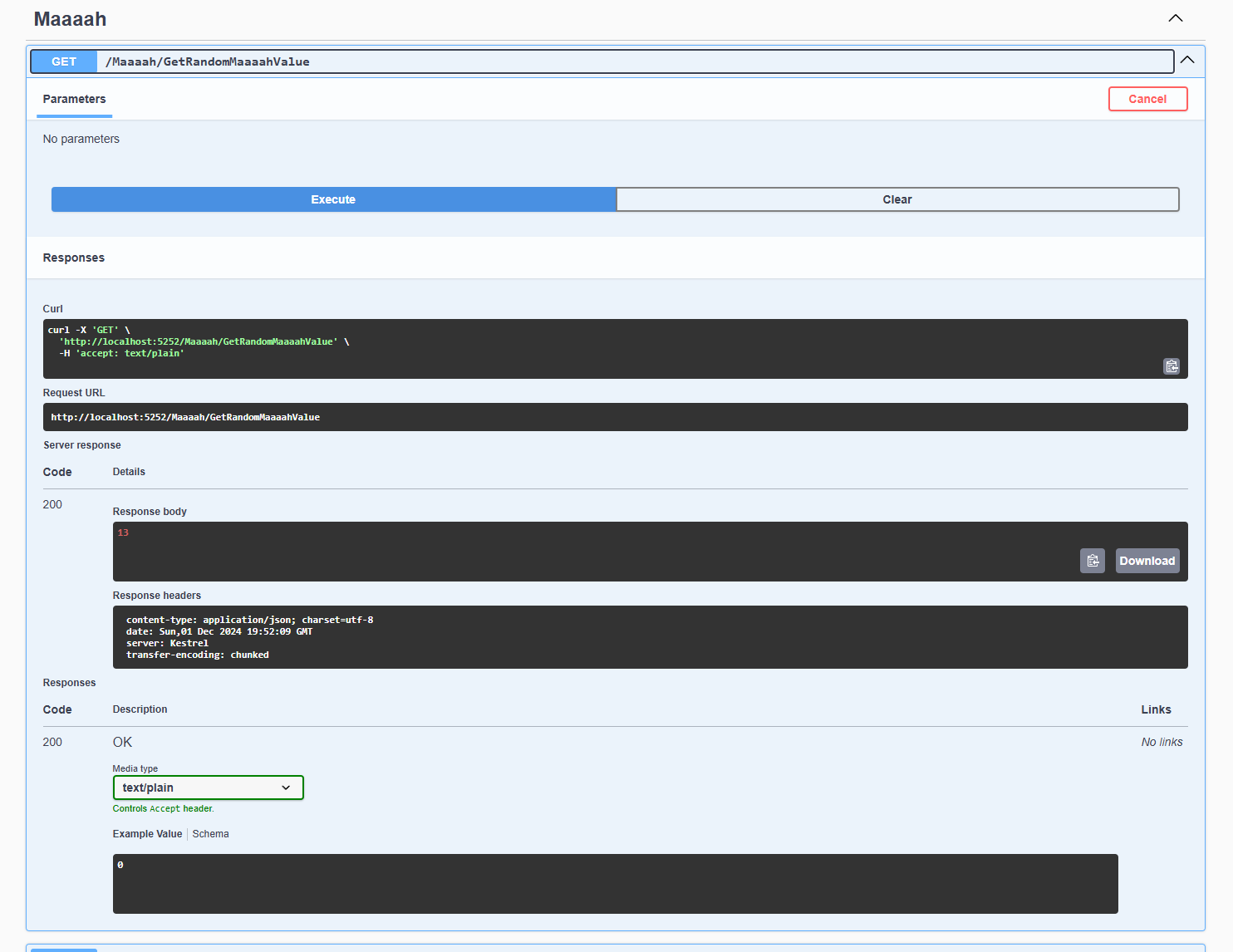


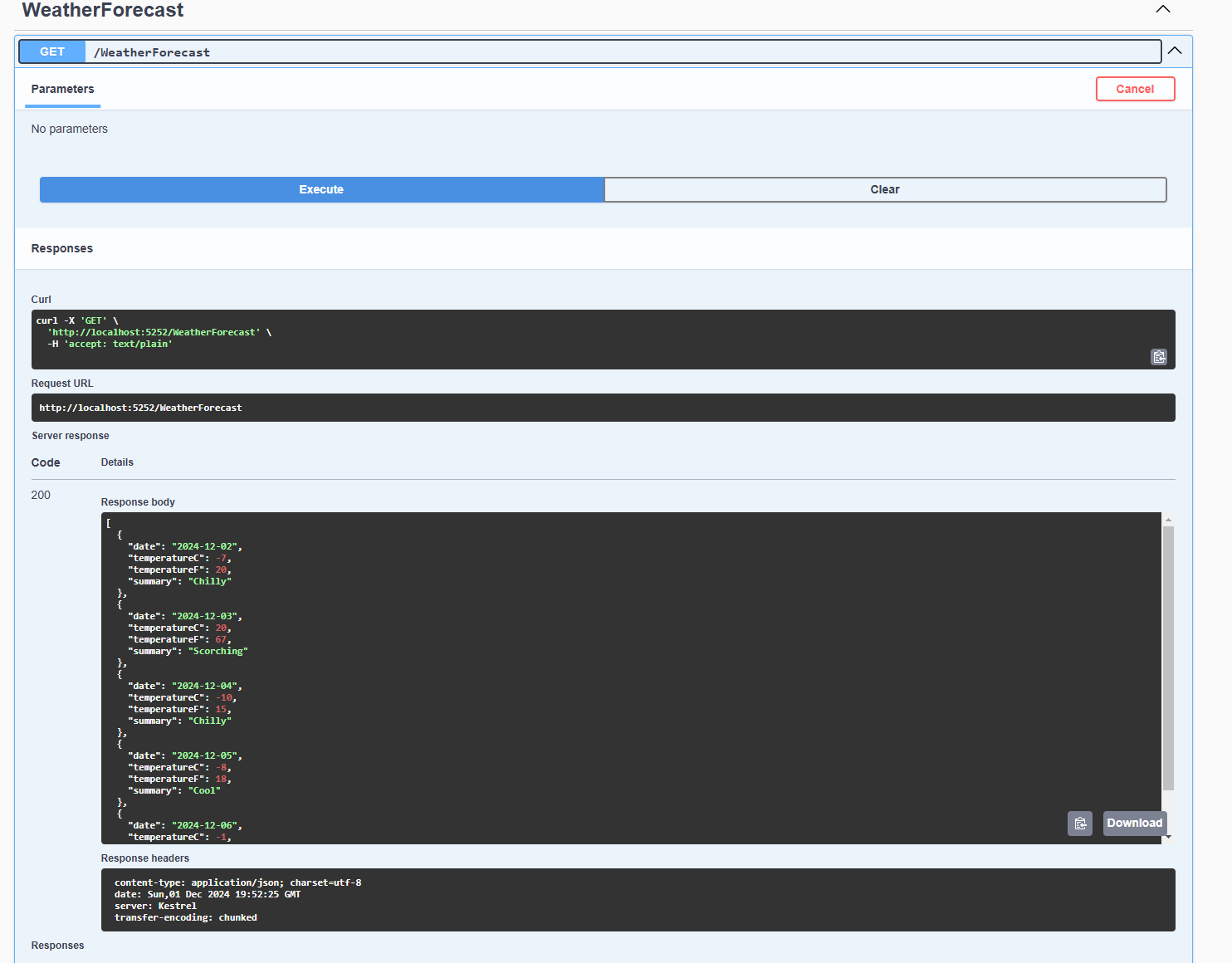
1. Модифікуємо конфігураційний файл appsettings.json, щоб налаштувати Serilog. Вкажемо формат індексу, адресу (endpoint) Elasticsearch і додаткові поля:

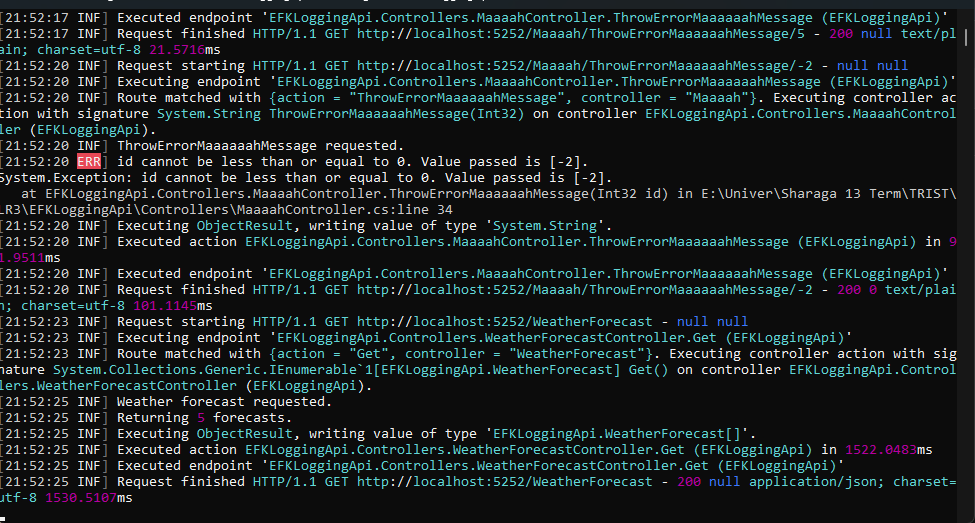


Запустимо API та перевіримо його працездатність:

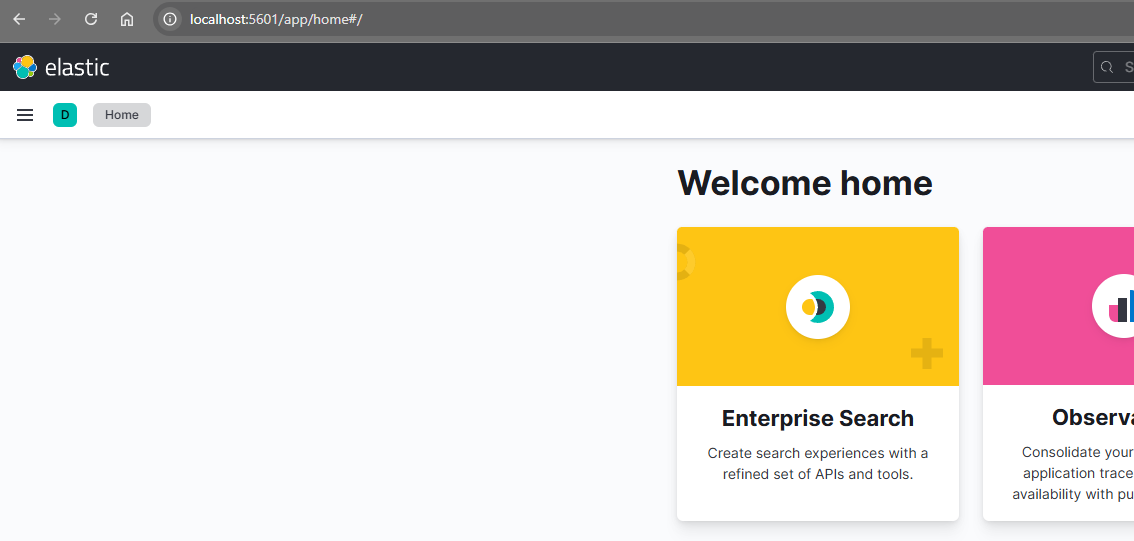


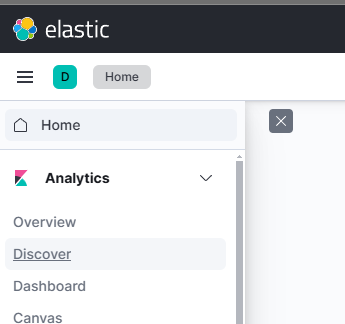
****

****

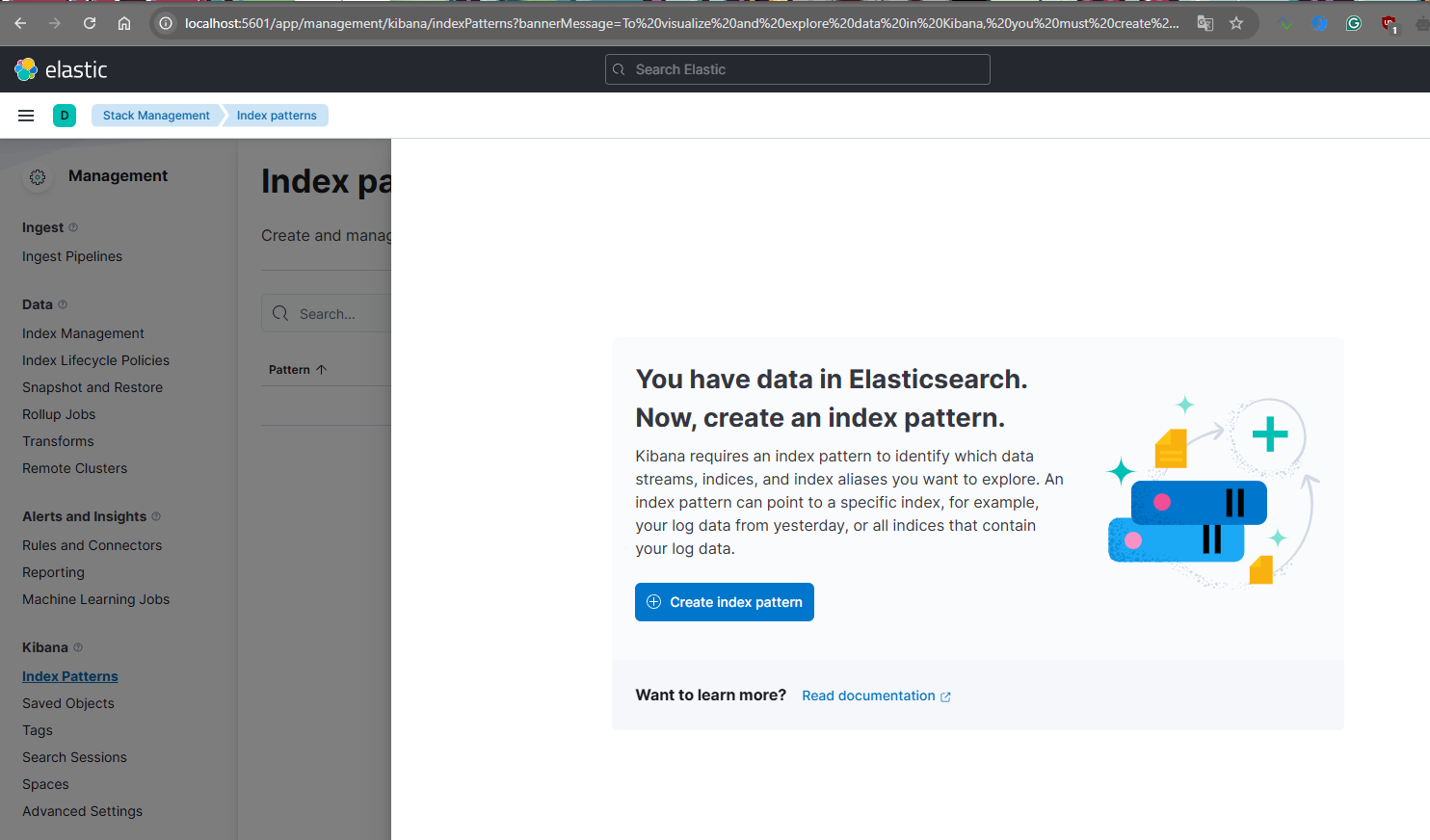
****

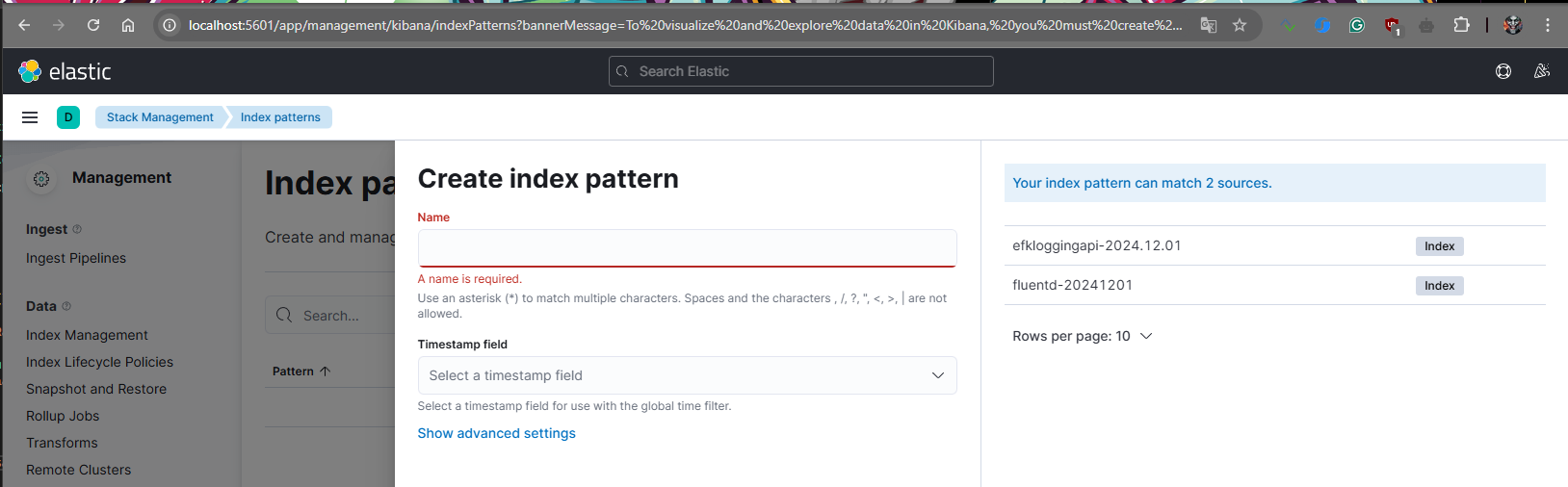
Далі перейдемо до Kibana для перевірки логів:

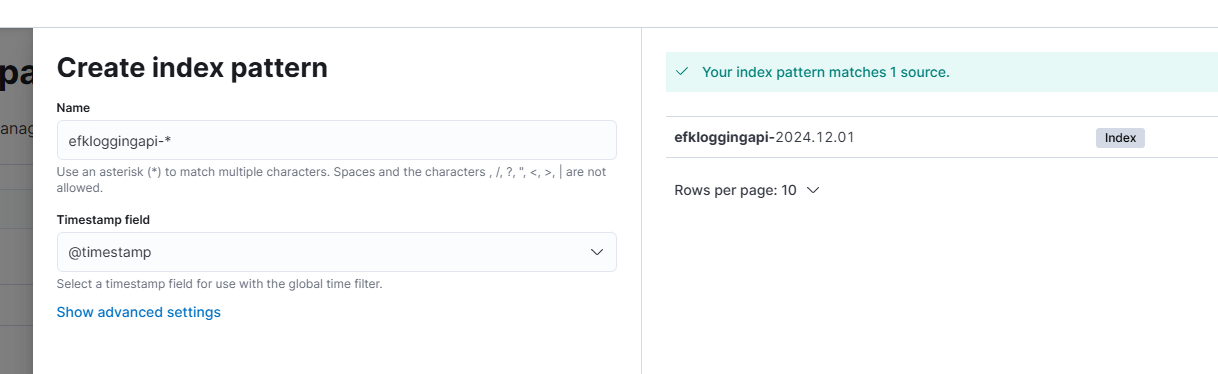


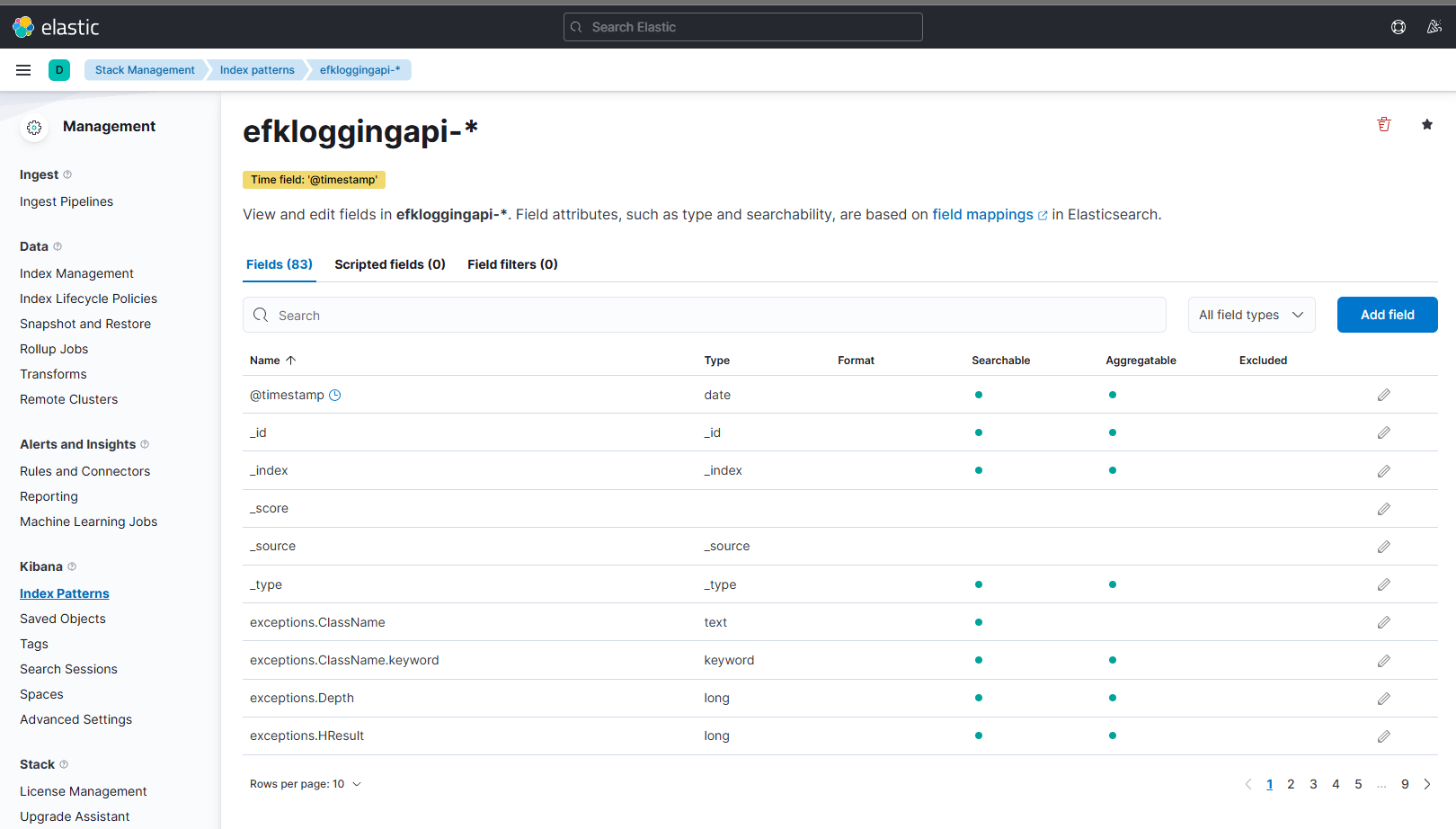


1. Створимо шаблон індексу для даних з нашого API:

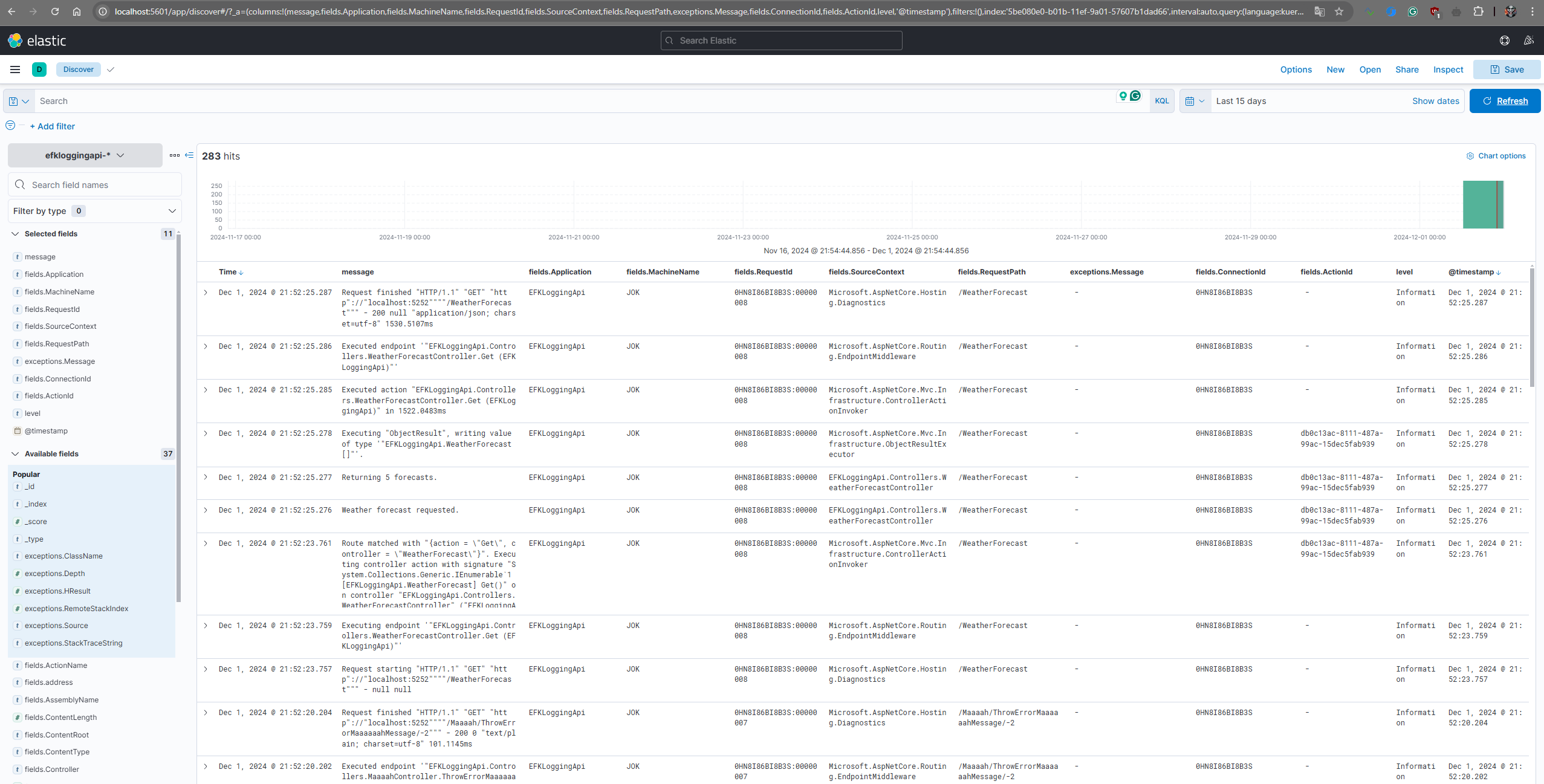




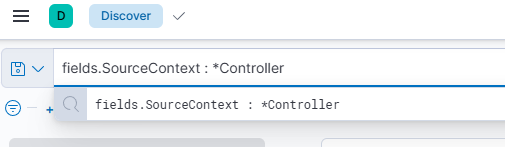


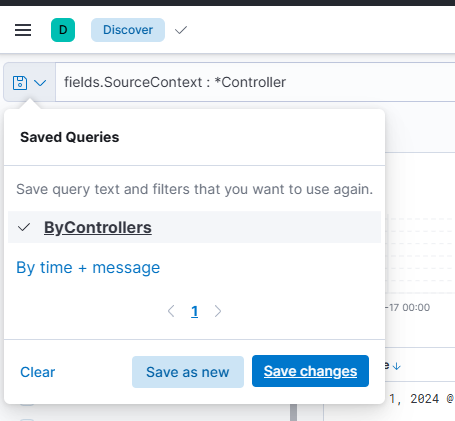


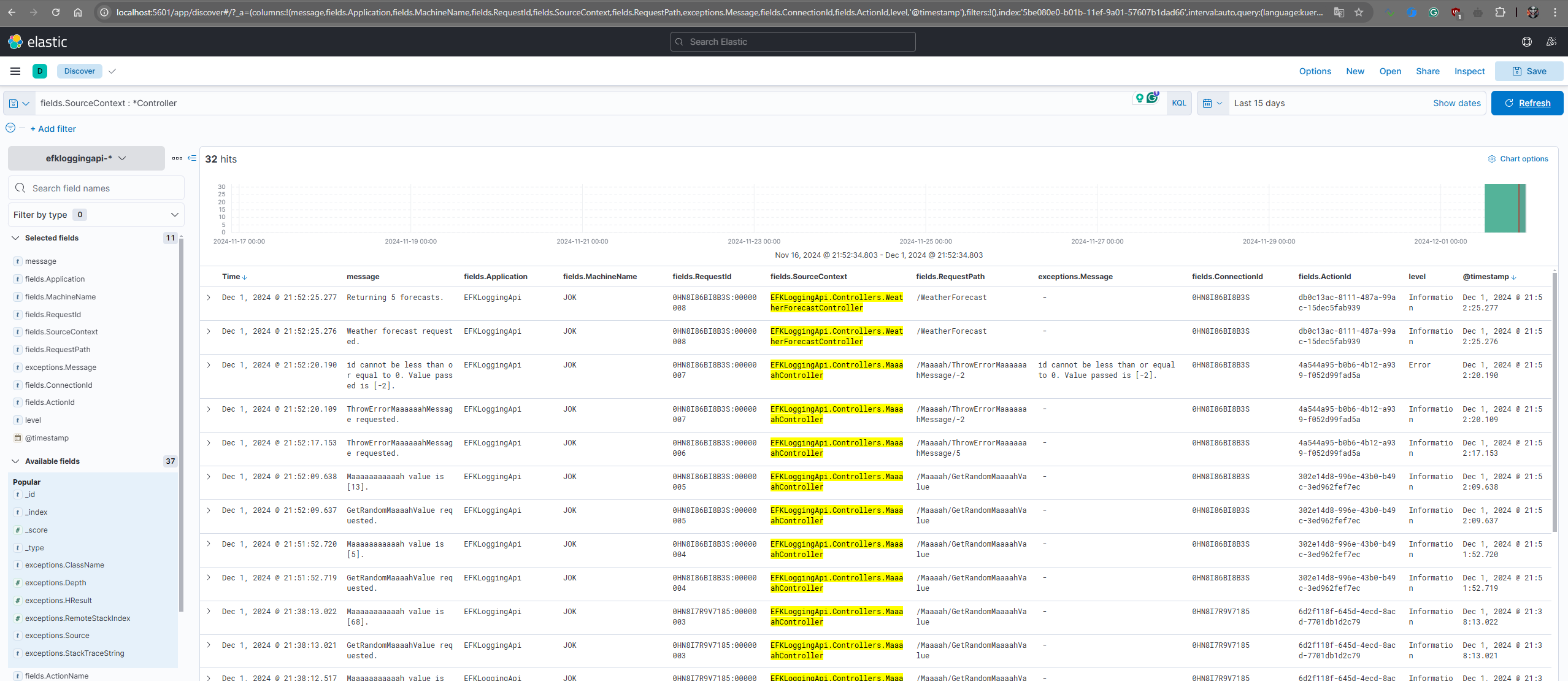
1. Перейдемо до розділу Discovery і перевіримо наші логи:



1. Додамо фільтр за SourceContext, щоб відобразити лише ті логи, які були згенеровані методами контролерів:







*Висновки:* в результаті виконання цієї лабораторної роботи було ознайомлено з базовими концепціями технологій централізованих систем логування на прикладі EFK. Однак через вибір мови програмування Fluentd не використовувався.

На основі отриманих знань було реалізовано практичну частину, яка полягала у створенні застосунку, що передає логи до централізованої системи для їх збереження, подальшої обробки та аналізу.

Вихідний код застосунку можна знайти за наступним посиланням на [GitHub](https://github.com/JokerFunny/TRIST/tree/main/LR3).