Министерство науки и высшего образования Российской Федерации
ФГАОУ ВО «Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»
Институт новых материалов и технологий
Кафедра «Теплофизика и информатика в металлургии»

Использование Prometheus и Grafana
для эффективного мониторинга ASP.NET Core приложений

ОТЧЕТ
по практической работе № 5
по дисциплине «Основы методологии Development Operation»

Направление 09.03.02 «Информационные системы и технологии» (уровень бакалавриата)
Образовательная программа
09.03.02/33.02 «Информационные системы и технологии» (СУОС)

Студент
группы НМТ-413901 Я.В.Крашенинников

Преподаватель:
профессор, д.т.н. В.В.Лавров

Екатеринбург
2024

СОДЕРЖАНИЕ

Практическая работа

«Использование Prometheus и Grafana

для эффективного мониторинга ASP.NET Core приложений»

1.1 Цель работы

Познакомить студентов с технологией использования Prometheus и Grafana для эффективного мониторинга ASP.NET Core приложений.

1.2 Ход проведения работы

1.2.1 Разработка тестового приложения Visual Studio .NET Core

В ходе работы над проектом были внесены улучшения, включая добавление файла docker-compose.yml, а также небольшие модификации кода в файлах Program.cs и Dockerfile (обновлена версия .NET с 6.0 на 8.0).

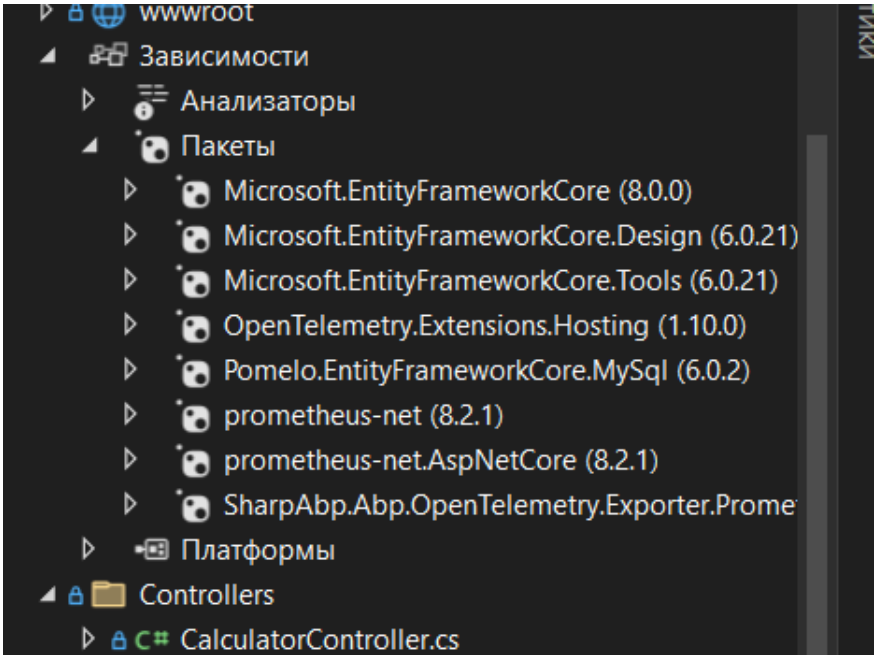

Рисунок 1 - Пакеты

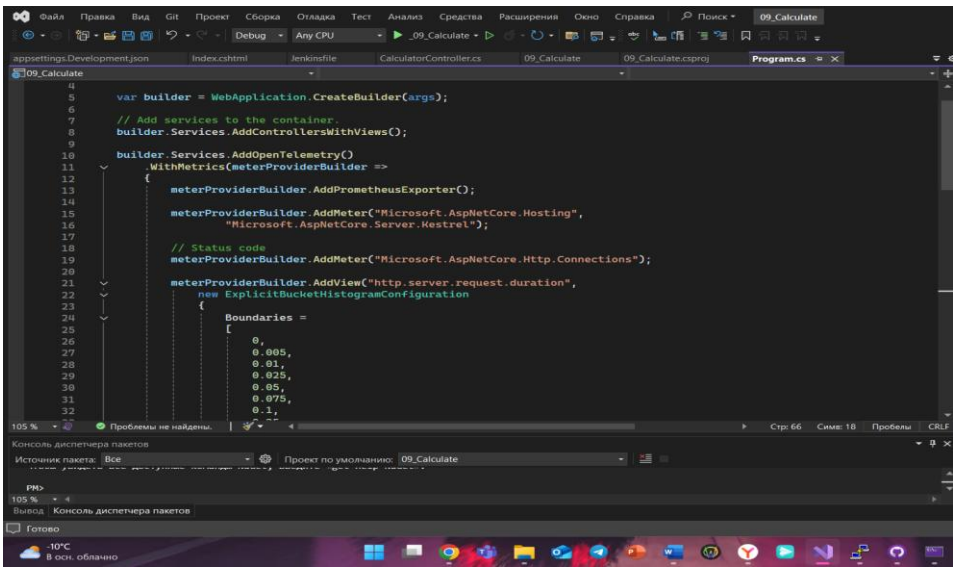

Рисунок 2 – Часть кода Program.cs

1.2.2 Создание файла docker-compose.yml для запуска простой конфигурации Prometeus и Grafana

В корневой папке проекта был создан файл docker-compose.yml. Этот файл содержит конфигурацию Docker Compose в формате YAML, описывающую процесс развёртывания веб-приложения в контейнере Docker. В файле указана версия синтаксиса Docker Compose — 3.7, обеспечивающая поддержку различных функций и параметров для настройки сервисов и контейнеров.

В разделе services описан сервис под названием web, который запускается в рамках данной конфигурации. Также установлено имя хоста app, используемое для взаимодействия между контейнерами внутри сети Docker.

Инструкция build: ./ указывает на необходимость сборки образа из текущей директории, где должен располагаться файл Dockerfile с инструкциями для сборки. В разделе ports настроено перенаправление портов: порт 5102 на хост-машина перенаправляется на порт 5009 внутри контейнера, обеспечивая доступ к приложению через порт 5109 хоста. (Рисунок 3)

Таким образом, файл docker-compose.yml определяет параметры и настройки для развёртывания и запуска веб-приложения в контейнере Docker. Полный листинг этого файла доступен в приложении Е.

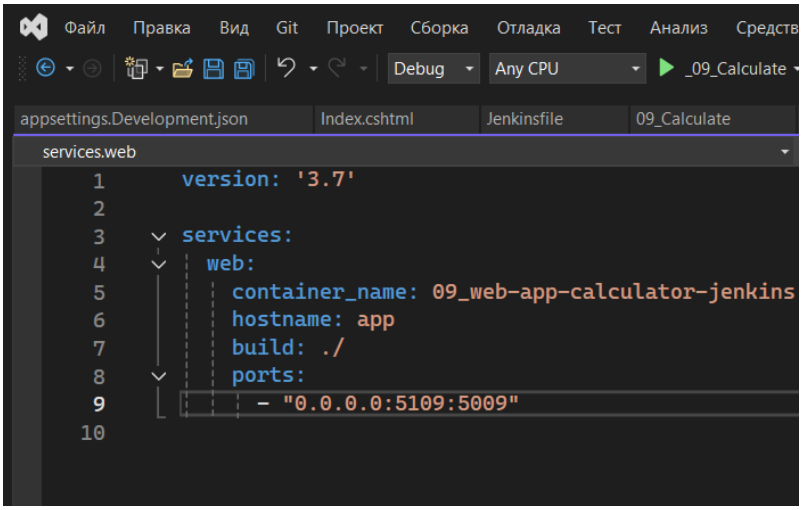

Рисунок 3 – Файл docker-compose

1.2.3 Настройка подключения Grafana к Prometeus и подготовка дашборда

После запуска контейнеров требуется настроить соединение между Grafana и Prometheus, а также настроить внешний вид дашборда. Инструкции по настройке соединения Grafana показаны на рисунке 4.

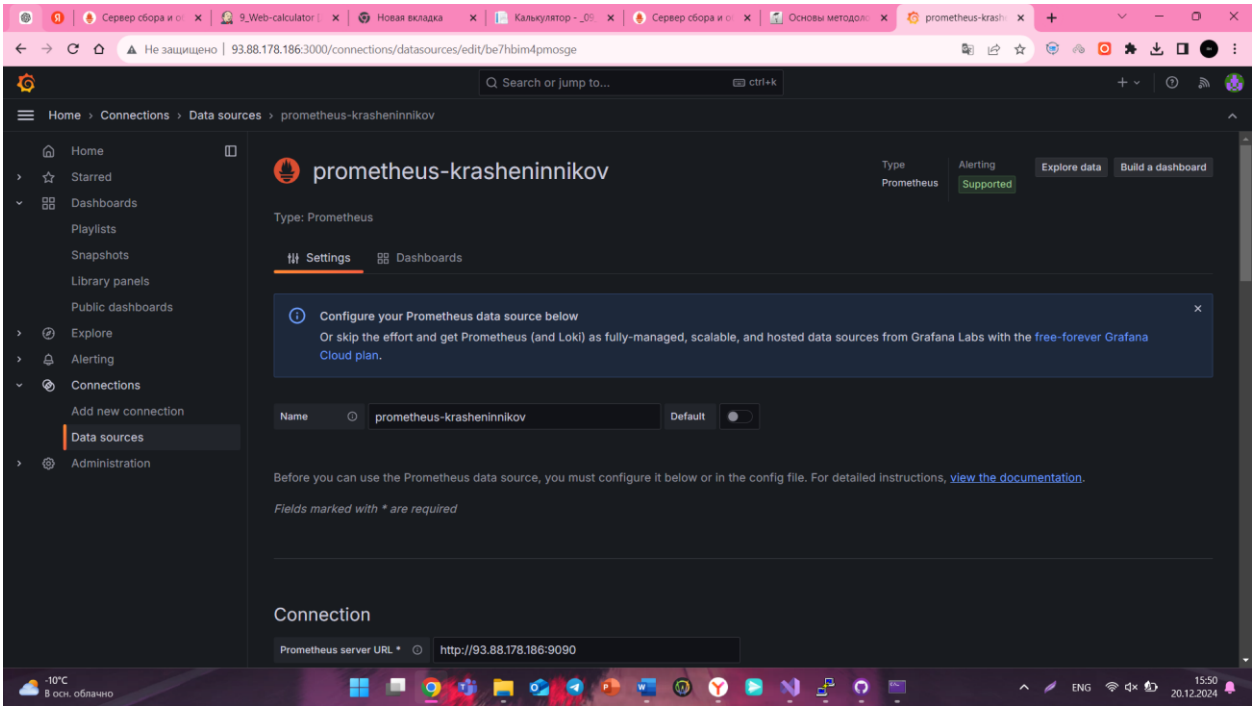

Рисунок 4 - Окно веб-браузера с демонстрацией окна настройки подключения Grafana

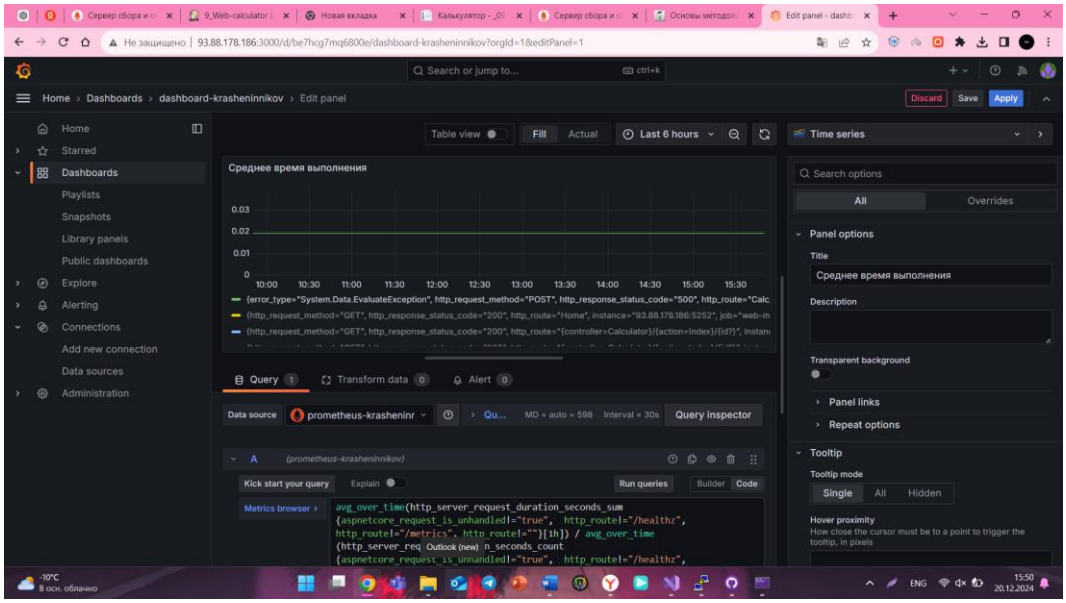

Рисунок 5 - Окно веб-браузера с демонстрацией настройки панели дашборда «Среднее время выполнения»

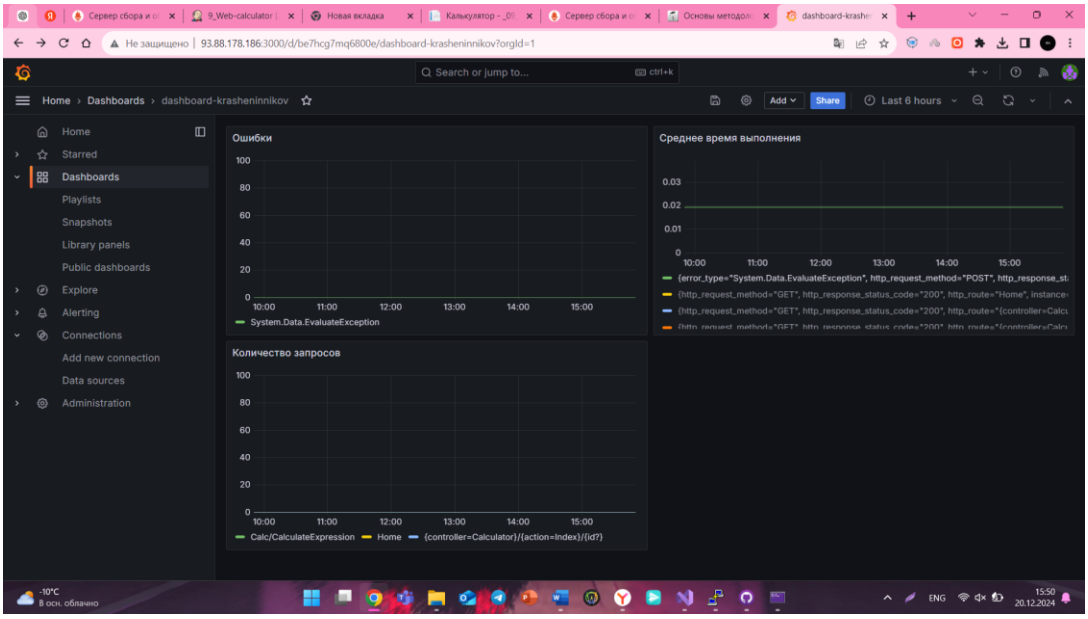

Рисунок 6 – Ошибки и количество запросов

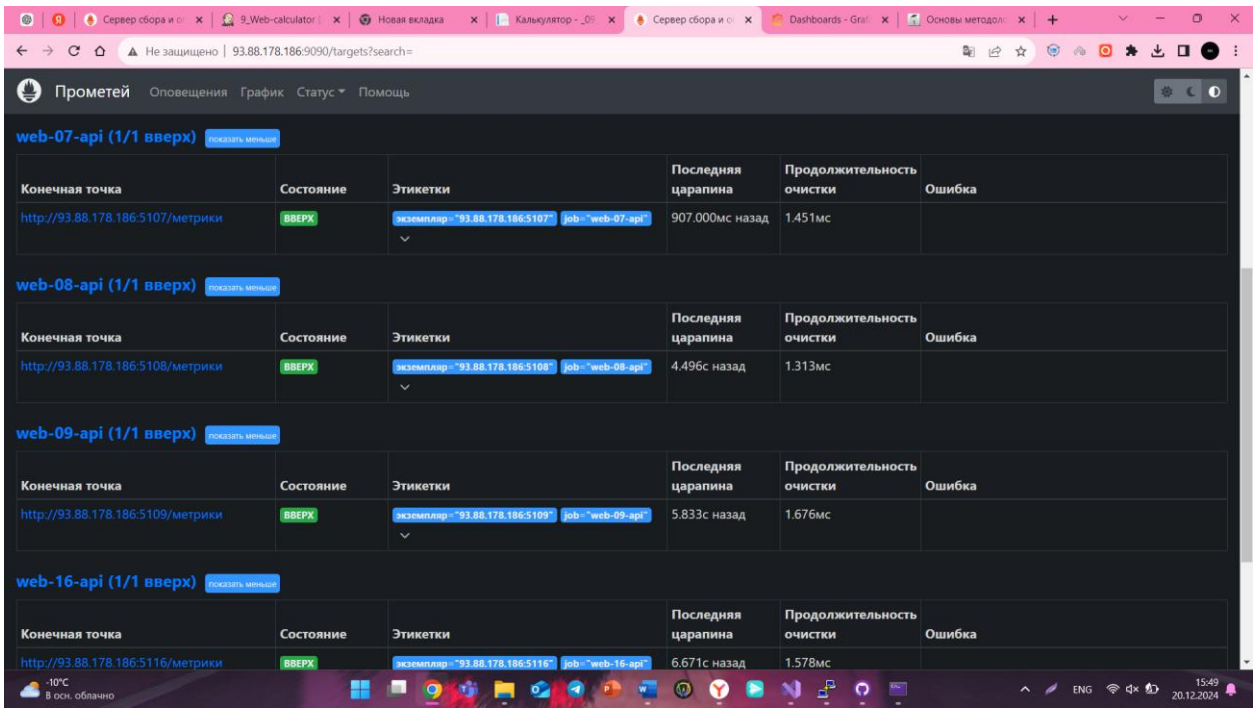

Рисунок 7 – Мой проект на сервере

1.3 Выводы

Была подробно изучена методика использования Prometheus и Grafana для эффективного мониторинга приложений ASP.NET Core в ходе выполнения данной практической работы. В ходе работы было улучшено тестовое приложение "Калькулятор", внесены обновления в код файла docker-compose.yml, и настроено соединение Grafana с Prometheus.

Приложение А Листинг программного кода приложения «Калькулятор» (контроллер)

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;


namespace _09_Calculate.Controllers
{
   public enum Operation { Add, Subtract, Multiply, Divide }

   public class CalculatorController : Controller
   {
      [HttpGet]
      public IActionResult Index()
      {
         return View();
      }

      [HttpPost]
      [ValidateAntiForgeryToken]
      public IActionResult Calculate(double num1, double num2, Operation operation)
      {
         double result = 0;

            switch (operation)
            {
               case Operation.Add:
                  result = num1 + num2;
```

```
                    break;
                case Operation.Subtract:
                    result = num1 - num2;
                    break;
                case Operation.Multiply:
                    result = num1 * num2;
                    break;
                case Operation.Divide:
                    result = num1 / num2;
                    break;
            }
            ViewBag.Result = result;
            return View("Index");
        }

    }
}
```

Приложение Б Листинг программного кода приложения «Калькулятор» (appsettings.json)

```
{

  "Kestrel": {
    "Endpoints": {
      "Http": {
        "Url": "http://0.0.0.0:5009"
      }
    }
  },

  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

Приложение В Листинг программного кода приложения «Калькулятор» (appsettings.Development.json)

```
{
  "DetailedErrors": true,
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  }
}
```

Приложение Г Листинг программного кода приложения «Калькулятор» (Program.cs)

```
using Microsoft.EntityFrameworkCore;
using OpenTelemetry.Metrics;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();

builder.Services.AddOpenTelemetry()
   .WithMetrics(meterProviderBuilder =>
   {
      meterProviderBuilder.AddPrometheusExporter();

      meterProviderBuilder.AddMeter("Microsoft.AspNetCore.Hosting",
          "Microsoft.AspNetCore.Server.Kestrel");

      // Status code

meterProviderBuilder.AddMeter("Microsoft.AspNetCore.Http.Connections");

      meterProviderBuilder.AddView("http.server.request.duration",
        new ExplicitBucketHistogramConfiguration
        {
          Boundaries =
          [
            0,
```

```
                0.005,
                0.01,
                0.025,
                0.05,
                0.075,
                0.1,
                0.25,
                0.5,
                0.75,
                1,
                2.5,
                5,
                7.5,
                10
            ]
        });
    });



builder.Services.AddRazorPages();

var app = builder.Build();


// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
```

```
        // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
        app.UseHsts();
    }

    app.UseHttpsRedirection();
    app.UseStaticFiles();

    app.MapPrometheusScrapingEndpoint();

    app.UseRouting();

    app.UseAuthorization();

    app.MapControllerRoute(
        name: "default",
        pattern: "{controller=Calculator}/{action=Index}/{id?}");

    app.Run();
```

Приложение Д Листинг файла docker-compose.yml

```
version: '3.7'

services:
  web:
    container_name: 09_web-app-calculator-jenkins
    hostname: app
    build: ./
    ports:
      - "0.0.0.0:5109:5009"
```