**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

***Факультет Информационных технологий***

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА №18**

**Дисциплина:** BackEnd-разработка

**Тема:** Мониторинг веб-приложения на основе ASP.NET Core

**Выполнил: студент группы 231-332**

**Чесноков Александр Евгеньевич**

(Фамилия И.О. обучающегося)

**Дата, подпись** 01.10.2025 

(Дата)(Подпись)

**Проверил:** \_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_*

(Фамилия И.О., степень, звание) (Оценка)

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *\_\_\_ \_\_\_\_\_\_\_\_*

(Дата)(Подпись)

**Москва**

**2025**

**Цель:**

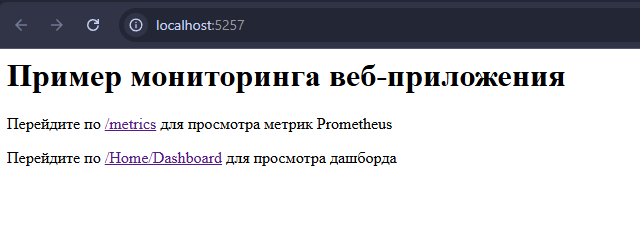
Овладеть навыками настройки мониторинга веб-приложений на платформе ASP.NET Core для наблюдения и анализа работы приложения в реальном времени.

**Ход работы:**

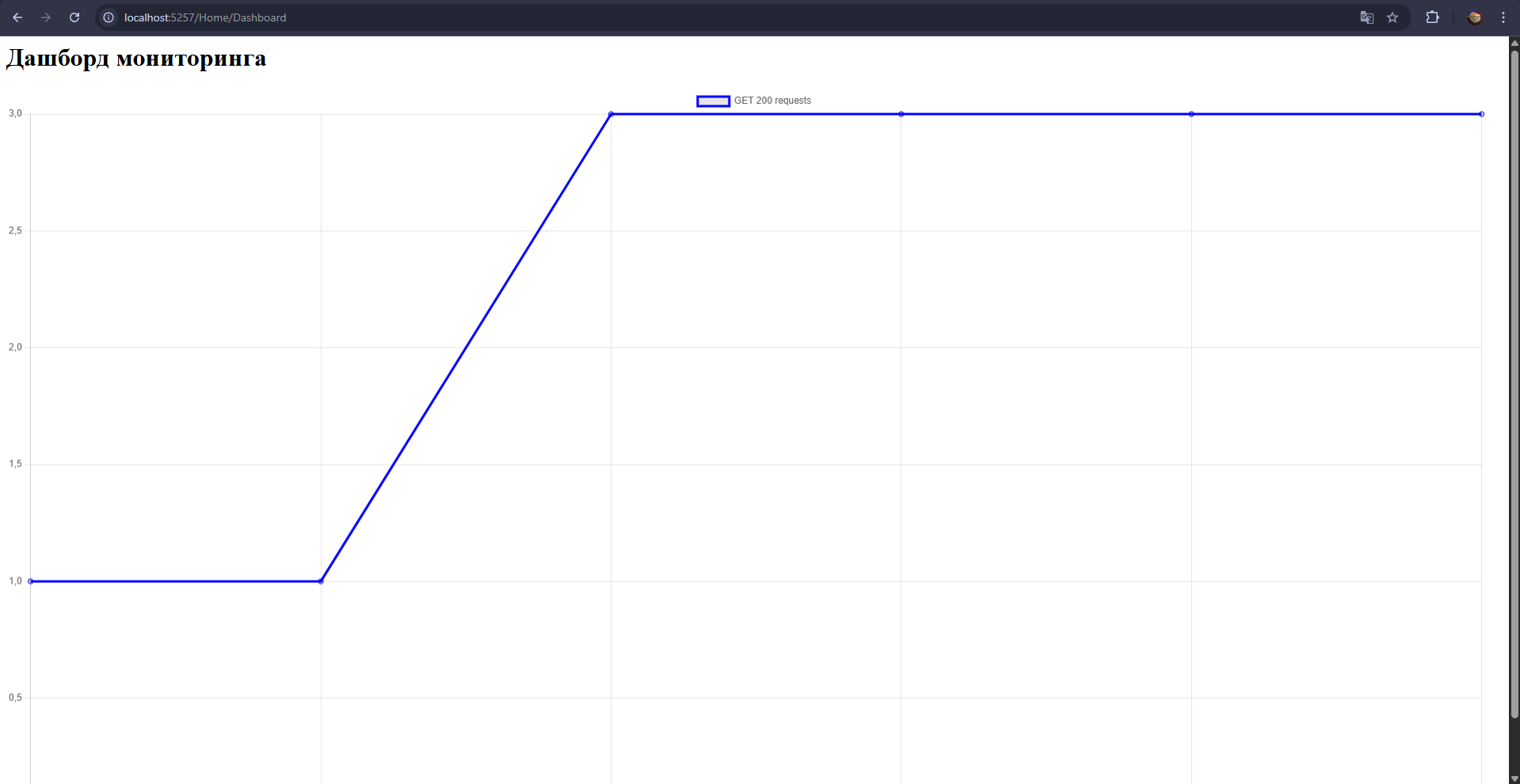
В ходе выполнения практического занятия №18 был настроен мониторинг веб-приложения на платформе ASP.NET Core с использованием встроенных инструментов и Prometheus. На первом этапе проект был подготовлен для работы с метриками: подключена библиотека Prometheus и настроена регистрация стандартных HTTP-метрик, что позволило собирать информацию о количестве запросов к серверу, их статусах и времени обработки. Далее был реализован контроллер с действиями для отображения основной страницы приложения, генерации тестовых ошибок и предоставления интерфейса метрик в формате Prometheus, что обеспечило возможность их дальнейшего анализа.

На следующем этапе была создана визуализация данных в виде дашборда. Для этого был добавлен метод Dashboard, который возвращает HTML-страницу с графиком, построенным с использованием библиотеки Chart.js. Скрипт периодически опрашивает маршрут /metrics и отображает динамически обновляющийся график количества запросов к серверу. Это позволило наблюдать за изменением нагрузки на веб-приложение в реальном времени и проверять работу мониторинга при возникновении различных событий, включая ошибки и обновления страниц.

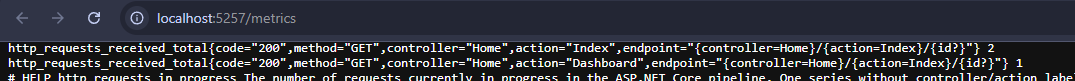
**Скриншоты работы приложения:**

****

Пример графика посещения главной страницы

****

Данные, возвращаемы Prometheus (количество посещений и ошибок):

****

**Листинг:**

**Program.cs:**

|  |
| --- |
| using Prometheus;  var builder = WebApplication.CreateBuilder(args);  builder.Services.AddControllersWithViews();  var app = builder.Build();  app.UseHttpMetrics();  app.UseStaticFiles();  app.MapControllerRoute(      name: "default",      pattern: "{controller=Home}/{action=Index}/{id?}"  );  app.MapMetrics("/metrics");  app.Run(); |

**Controllers/HomeController.cs:**

|  |
| --- |
| using Microsoft.AspNetCore.Mvc;  using System.Net.Http;  using System.Text.RegularExpressions;  namespace Lab18.Controllers  {      public class HomeController : Controller      {          public IActionResult Index()          {              return Content(@"                  <html>                      <head>                          <title>Мониторинг ASP.NET Core</title>                      </head>                      <body>                          <h1>Пример мониторинга веб-приложения</h1>                          <p>Перейдите по <a href='/metrics'>/metrics</a> для просмотра метрик Prometheus</p>                          <p>Перейдите по <a href='/Home/Dashboard'>/Home/Dashboard</a> для просмотра дашборда</p>                      </body>                  </html>", "text/html; charset=utf-8");          }          public IActionResult TriggerError()          {              throw new Exception("Тестовая ошибка для мониторинга");          }          public IActionResult Dashboard()          {              return Content(@"                  <html>                      <head>                          <title>Дашборд мониторинга</title>                      </head>                      <body>                          <h1>Дашборд мониторинга</h1>                          <canvas id='requestsChart'></canvas>                          <script src='https://cdn.jsdelivr.net/npm/chart.js'></script>                          <script>                          async function updateChart(chart) {                              const response = await fetch('/metrics');                              const text = await response.text();                                // Найдём строку с нужной метрикой                              const match = text.match(/http\_requests\_received\_total\{code=\""200\"",method=\""GET\"",controller=\""Home\"",action=\""Index\"",endpoint=\""\{controller=Home\}\/\{action=Index\}\/\{id\?\}\""\}\s+(\d+)/);                              const value = match ? parseInt(match[1]) : 0;                              // Добавим новое значение                              const now = new Date().toLocaleTimeString();                              chart.data.labels.push(now);                              chart.data.datasets[0].data.push(value);                              // Оставим последние 20 точек                              if (chart.data.labels.length > 20) {                                  chart.data.labels.shift();                                  chart.data.datasets[0].data.shift();                              }                              chart.update();                          }                          // Создаём график                          const ctx = document.getElementById('requestsChart').getContext('2d');                          const requestsChart = new Chart(ctx, {                              type: 'line',                              data: {                                  labels: [],                                  datasets: [{                                      label: 'GET 200 requests',                                      data: [],                                      borderColor: 'blue',                                      fill: false                                  }]                              },                              options: {                                  responsive: true,                                  scales: {                                      x: { display: true },                                      y: { beginAtZero: true }                                  }                              }                          });                          // Обновляем график каждые 5 секунд                          setInterval(() => updateChart(requestsChart), 5000);                          </script>                      </body>                  </html>              ", "text/html; charset=utf-8");          }      }  } |

**Ссылка на репозиторий:** <https://github.com/avoryyy/backend_mospoly>