**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 №14**

**Дисциплина:** BackEnd-разработка

**Тема:** Работа с кросс-доменными запросами веб-приложение 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:**

Ознакомиться с работой с кросс-доменными запросами (CORS) в веб-приложениях на платформе ASP.NET Core для обеспечения безопасности и разрешения запросов с других источников.

**Ход работы:**

Современные браузеры по умолчанию блокируют запросы с других доменов, отличных от источника веб-приложения. Это ограничение предназначено для повышения безопасности, чтобы предотвратить несанкционированный доступ к данным. Однако в некоторых случаях требуется разрешить такие запросы — например, при работе с фронтендом, размещённым на другом сервере (например, React-приложение на localhost:3000 обращается к API на localhost:5000).

В рамках лабораторной работы был создан новый проект **ASP.NET Core Web API**. Основной файл Program.cs был расширен добавлением настройки CORS через встроенный сервис AddCors. В приложении определено две политики:

* **AllowSpecificOrigin** — разрешает запросы только с определённых доменов (http://localhost:3000 и https://example.com), что является рекомендуемым подходом для реальных систем.
* **AllowAll** — разрешает доступ с любых источников, что удобно для тестирования, но не должно использоваться в продакшене.

В контроллере DataController реализованы два метода для демонстрации работы CORS:

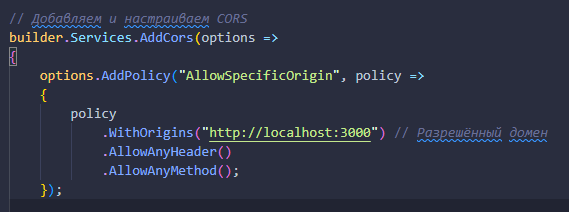
* GET /api/data/info — возвращает JSON с сообщением и временем обработки.
* POST /api/data/submit — принимает JSON-объект и возвращает подтверждение приёма данных.

После запуска приложения и открытия Swagger-интерфейса (/swagger) можно протестировать эти методы. Также можно проверить работу CORS, отправляя запросы с других доменов (например, из фронтенда, размещённого на localhost:3000). В зависимости от политики, браузер либо разрешит, либо отклонит запрос.

В процессе тестирования было подтверждено, что при использовании политики **AllowSpecificOrigin** запросы, отправленные с разрешённых доменов, обрабатываются успешно, тогда как попытка обращения с других источников приводит к ошибке CORS policy error.

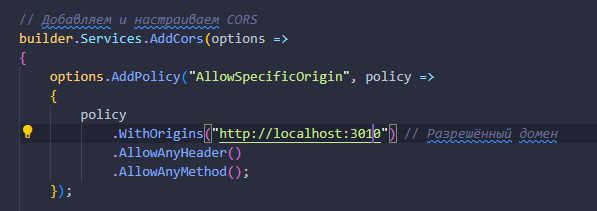
**Скриншоты работы приложения:**

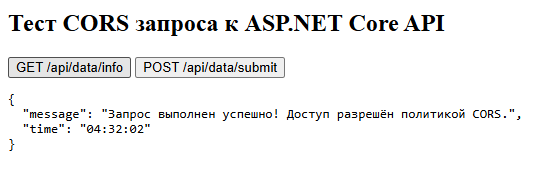
Верный порт в настройках CORS:





Нерный порт в настройках CORS:







**Листинг:**

**Program.cs:**

|  |
| --- |
| var builder = WebApplication.CreateBuilder(args);  // *Добавляем контроллеры*  builder.Services.AddControllers();  // *Добавляем и настраиваем CORS*  builder.Services.AddCors(options =>  {      options.AddPolicy("AllowSpecificOrigin", policy =>      {          policy              .WithOrigins("http://localhost:3000") // *Разрешённый домен*              .AllowAnyHeader()              .AllowAnyMethod();      });      // *Также можно разрешить все источники (для теста)*      options.AddPolicy("AllowAll", policy =>      {          policy.AllowAnyOrigin()                .AllowAnyHeader()                .AllowAnyMethod();      });  });  // *Добавляем Swagger*  builder.Services.AddEndpointsApiExplorer();  builder.Services.AddSwaggerGen();  var app = builder.Build();  // *Подключаем Swagger*  if (app.Environment.IsDevelopment())  {      app.UseSwagger();      app.UseSwaggerUI();  }  // *Подключаем CORS (можно выбрать нужную политику)*  app.UseCors("AllowSpecificOrigin");  // *Подключаем маршрутизацию*  app.MapControllers();  app.Run(); |

**Controllers/DataController.cs:**

|  |
| --- |
| using Microsoft.AspNetCore.Mvc;  namespace Lab14.Controllers;  /// *Контроллер для демонстрации работы CORS*  [ApiController]  [Route("api/[controller]")]  public class DataController : ControllerBase  {      // *Пример GET-запроса, который может вызываться с другого домена*      [HttpGet("info")]      public IActionResult GetInfo()      {          return Ok(new          {              message = "Запрос выполнен успешно! Доступ разрешён политикой CORS.",              time = DateTime.Now.ToString("HH:mm:ss")          });      }      // *Пример POST-запроса для проверки CORS при отправке данных*      [HttpPost("submit")]      public IActionResult Submit([FromBody] dynamic payload)      {          return Ok(new          {              status = "Получено",              data = payload,              time = DateTime.Now.ToString("HH:mm:ss")          });      }  } |

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