**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 №12**

**Дисциплина:** BackEnd-разработка

**Тема:** Работа с базой данных и Entity Framework в 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:**

Первым шагом было создание нового проекта ASP.NET Core Web API. После этого в проект была добавлена модель данных Product, представляющая сущность товара. В ней определены три свойства: идентификатор, название и цена. Эта модель будет использоваться для взаимодействия с базой данных через Entity Framework.

Затем был создан класс AppDbContext, наследующий от DbContext, который представляет контекст базы данных. В этом классе определён DbSet<Product> — коллекция, представляющая таблицу товаров. В методе OnModelCreating добавлена инициализация базы начальными данными с помощью метода HasData, который автоматически заполняет таблицу при первом запуске приложения. Для упрощения тестирования использовалась встроенная база данных InMemory, которая не требует отдельного сервера и позволяет работать с данными напрямую в оперативной памяти.

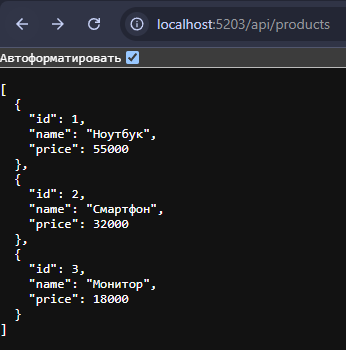
Далее была выполнена регистрация контекста данных в контейнере зависимостей через метод AddDbContext в файле Program.cs. Таким образом, Entity Framework и контекст базы данных подключаются к приложению с помощью встроенного механизма Dependency Injection, что обеспечивает гибкость и модульность кода.

Также в Program.cs был добавлен раздел конфигурации CORS. Политика “AllowAll” разрешает выполнение HTTP-запросов с любых источников, с любыми заголовками и методами. Это необходимо для взаимодействия серверного API с внешними клиентами — например, с фронтендом на JavaScript, работающим на другом домене.

Следующим шагом был создан контроллер ProductsController, в котором внедрён контекст базы данных через конструктор. Контроллер содержит метод GetAll, который выполняет выборку всех записей из таблицы Products и возвращает их в формате JSON в ответ на GET-запрос. Благодаря этому можно протестировать API, отправив запрос по адресу <http://localhost:5000/api/products> и получить список инициализированных данных.

Результатом выполнения лабораторной работы стало работающее ASP.NET Core приложение, использующее Entity Framework для хранения и получения данных, а также настроенное для безопасной работы с кросс-доменными запросами. Реализованы все основные этапы: создание модели данных, настройка контекста, регистрация зависимостей, инициализация базы и получение данных через контроллер.

**Скриншоты работы приложения:**

****

**Листинг:**

**Program.cs:**

|  |
| --- |
| using Lab12.Data;  using Microsoft.EntityFrameworkCore;  var builder = WebApplication.CreateBuilder(args);  // *Подключение контекста данных с использованием InMemory Database*  builder.Services.AddDbContext<AppDbContext>(options =>      options.UseInMemoryDatabase("ProductsDb"));  // *Разрешаем CORS-запросы (например, с фронтенда)*  builder.Services.AddCors(options =>  {      options.AddPolicy("AllowAll",          policy => policy.AllowAnyOrigin()                          .AllowAnyHeader()                          .AllowAnyMethod());  });  builder.Services.AddControllers();  var app = builder.Build();  app.UseCors("AllowAll");  app.MapControllers();  app.Run(); |

**Models/Product.cs:**

|  |
| --- |
| namespace Lab12.Models;  /// *Модель данных, описывающая товар*  public class Product  {      public int Id { get; set; }      public string Name { get; set; } = string.Empty;      public decimal Price { get; set; }  } |

**Data/AppDBContext.cs:**

|  |
| --- |
| using Lab12.Models;  using Microsoft.EntityFrameworkCore;  namespace Lab12.Data;  /// *Контекст базы данных для работы с Entity Framework*  public class AppDbContext : DbContext  {      public DbSet<Product> Products { get; set; }      public AppDbContext(DbContextOptions<AppDbContext> options)          : base(options)      {          // *Автоматическое создание базы при первом запуске*          Database.EnsureCreated();      }      protected override void OnModelCreating(ModelBuilder modelBuilder)      {          base.OnModelCreating(modelBuilder);          // *Инициализация базы начальными данными*          modelBuilder.Entity<Product>().HasData(              new Product { Id = 1, Name = "Ноутбук", Price = 55000 },              new Product { Id = 2, Name = "Смартфон", Price = 32000 },              new Product { Id = 3, Name = "Монитор", Price = 18000 }          );      }  } |

**Controllers/ProductsController.cs:**

|  |
| --- |
| using Lab12.Data;  using Microsoft.AspNetCore.Mvc;  using Microsoft.EntityFrameworkCore;  namespace Lab12.Controllers;  /// *Контроллер для работы с товарами*  [ApiController]  [Route("api/[controller]")]  public class ProductsController : ControllerBase  {      private readonly AppDbContext \_context;      public ProductsController(AppDbContext context)      {          \_context = context;      }      [HttpGet]      public async Task<IActionResult> GetAll()      {          var products = await \_context.Products.ToListAsync();          return Ok(products);      }  } |

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