**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 №11**

**Дисциплина:** BackEnd-разработка

**Тема:** Создание веб-API 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:**

Освоить создание веб-API приложения на платформе ASP.NET Core для обработки HTTP-запросов и предоставления данных через API.

**Ход работы:**

На первом этапе был создан новый проект типа **ASP.NET Core Web API**. В файле Program.cs настроена базовая инфраструктура приложения: добавлены сервисы для контроллеров и подключен Swagger для удобного тестирования API через браузер. Swagger автоматически генерирует интерактивную страницу документации по адресу /swagger, где можно выполнять запросы без использования сторонних инструментов вроде Postman.

Далее была создана модель данных Product, содержащая три свойства: идентификатор Id, название Name и цену Price. Эта модель используется для представления объектов, с которыми работает API.

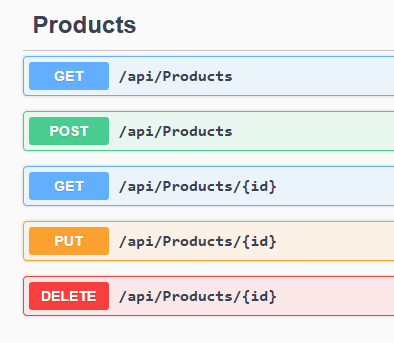
Контроллер ProductsController реализует полный набор CRUD-операций:

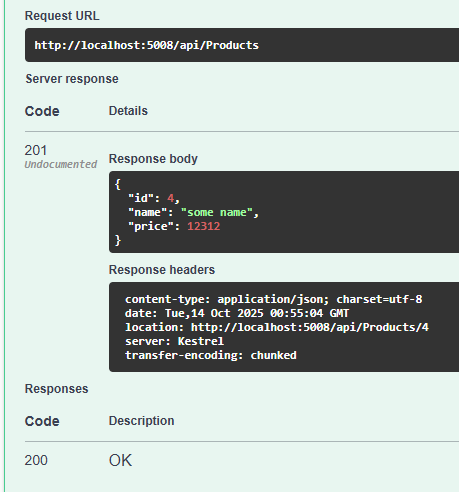
* **GET /api/products** — возвращает список всех продуктов.
* **GET /api/products/{id}** — возвращает продукт по его идентификатору, если он существует.
* **POST /api/products** — добавляет новый продукт, присваивая ему уникальный идентификатор.
* **PUT /api/products/{id}** — обновляет данные существующего продукта.
* **DELETE /api/products/{id}** — удаляет продукт по идентификатору.

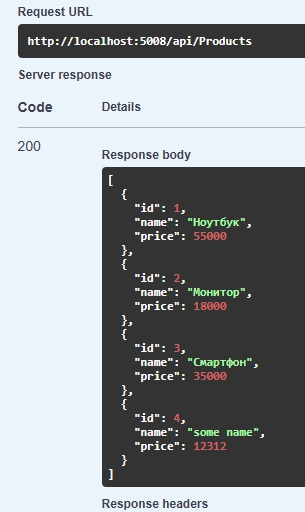
Все данные хранятся в статическом списке Products, что позволяет имитировать работу с базой данных без использования внешних источников.

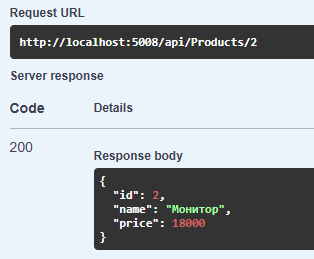
После запуска приложения Swagger автоматически отображает доступные маршруты и позволяет тестировать их прямо из браузера. В процессе тестирования было подтверждено, что все запросы обрабатываются корректно, возвращая ожидаемые ответы с нужными кодами состояния: 200 OK, 201 Created, 204 No Content, 404 Not Found.

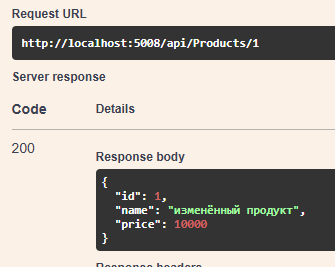
**Скриншоты работы приложения:**

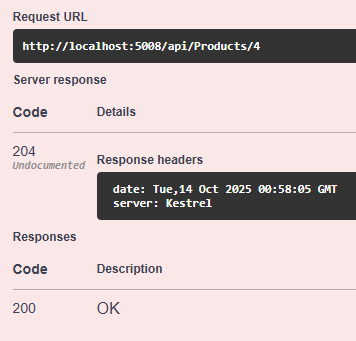












**Листинг:**

**Program.cs:**

|  |
| --- |
| var builder = WebApplication.CreateBuilder(args);  // *Добавляем контроллеры и Swagger*  builder.Services.AddControllers();  builder.Services.AddEndpointsApiExplorer();  builder.Services.AddSwaggerGen();  var app = builder.Build();  // *Подключаем Swagger*  if (app.Environment.IsDevelopment())  {      app.UseSwagger();      app.UseSwaggerUI();  }  // *Маршрутизация*  app.MapControllers();  app.Run(); |

**Controllers/ProductsController.cs:**

|  |
| --- |
| using Microsoft.AspNetCore.Mvc;  using Lab11.Models;  namespace Lab11.Controllers;  /// *Контроллер для CRUD операций с продуктами*  [ApiController]  [Route("api/[controller]")]  public class ProductsController : ControllerBase  {      private static readonly List<Product> Products = new()      {          new Product { Id = 1, Name = "Ноутбук", Price = 55000 },          new Product { Id = 2, Name = "Монитор", Price = 18000 },          new Product { Id = 3, Name = "Смартфон", Price = 35000 }      };      //*GET — получение всех продуктов*      [HttpGet]      public IActionResult GetAll()      {          return Ok(Products);      }      //*GET — получение продукта по Id*      [HttpGet("{id:int}")]      public IActionResult GetById(int id)      {          var product = Products.FirstOrDefault(p => p.Id == id);          if (product == null)              return NotFound(new { message = $"Продукт с Id = {id} не найден." });          return Ok(product);      }      //*POST — добавление нового продукта*      [HttpPost]      public IActionResult Create(Product newProduct)      {          newProduct.Id = Products.Any() ? Products.Max(p => p.Id) + 1 : 1;          Products.Add(newProduct);          return CreatedAtAction(nameof(GetById), new { id = newProduct.Id }, newProduct);      }      //*PUT — обновление существующего продукта*      [HttpPut("{id:int}")]      public IActionResult Update(int id, Product updatedProduct)      {          var product = Products.FirstOrDefault(p => p.Id == id);          if (product == null)              return NotFound(new { message = $"Продукт с Id = {id} не найден." });          product.Name = updatedProduct.Name;          product.Price = updatedProduct.Price;          return Ok(product);      }      //*DELETE — удаление продукта*      [HttpDelete("{id:int}")]      public IActionResult Delete(int id)      {          var product = Products.FirstOrDefault(p => p.Id == id);          if (product == null)              return NotFound(new { message = $"Продукт с Id = {id} не найден." });          Products.Remove(product);          return NoContent();      }  } |

**Models/Product.cs:**

|  |
| --- |
| namespace Lab11.Models;  /// *Модель данных для демонстрации CRUD операций через API*  public class Product  {      public int Id { get; set; }      public string Name { get; set; } = string.Empty;      public decimal Price { get; set; }  } |

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