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:

_____

Выполнил(а): студент(ка) группы __231-336__

_____Канищев И.М_____
(Фамилия И.О.)

Дата, подпись _____ _____
(Дата) (Подпись)

Проверил: _______
_____
(Фамилия И.О., степень, звание) **(Оценка)**

Дата, подпись _____ _____
(Дата) (Подпись)

Замечания:

Москва

2025

Шаг 1: Создание проекта

Создан новый проект ASP.NET Core Web API с использованием .NET 6.0. Выбран шаблон "ASP.NET Core Web API" с включенной поддержкой контроллеров.

Шаг 2: Определение моделей данных

Созданы модели данных для представления сущностей в системе.

Шаг 3: Реализация контроллеров

Реализованы контроллеры с методами для обработки CRUD операций (Create, Read, Update, Delete).

Шаг 4: Настройка зависимостей

Настроены необходимые сервисы в контейнере зависимостей.

Шаг 5: Тестирование API

Протестированы все endpoints с помощью Postman.

Листинги кода с комментариями

Модель данных (Product.cs)

```
namespace WebAPIApp.Models
{
    /// <summary>
    /// Модель данных для товара
    /// </summary>
    public class Product
    {
        /// <summary>
        /// Уникальный идентификатор товара
        /// </summary>
        public int Id { get; set; }

        /// <summary>
        /// Название товара
        /// </summary>
        public string Name { get; set; } = string.Empty;

        /// <summary>
        /// Цена товара
        /// </summary>
        public decimal Price { get; set; }

        /// <summary>
        /// Категория товара
        /// </summary>
        public string Category { get; set; } = string.Empty;

        /// <summary>
        /// Описание товара
        /// </summary>
        public string Description { get; set; } = string.Empty;
```

```
        /// <summary>
        /// Дата создания записи
        /// </summary>
        public DateTime CreatedDate { get; set; }
    }
}
```

Репозиторий для работы с данными (IProductRepository.cs и ProductRepository.cs)

IProductRepository.cs

```
using WebAPIApp.Models;

namespace WebAPIApp.Repositories
{
    /// <summary>
    /// Интерфейс репозитория для работы с товарами
    /// Определяет контракт для операций CRUD (Create, Read, Update, Delete)
    /// </summary>
    public interface IProductRepository
    {
        /// <summary>
        /// Получить все товары
        /// </summary>
        /// <returns>Коллекция всех товаров</returns>
        IEnumerable<Product> GetAll();

        /// <summary>
        /// Получить товар по идентификатору
        /// </summary>
        /// <param name="id">Уникальный идентификатор товара</param>
        /// <returns>Найденный товар или null</returns>
        Product GetById(int id);

        /// <summary>
        /// Добавить новый товар
        /// </summary>
        /// <param name="product">Объект товара для добавления</param>
        void Add(Product product);

        /// <summary>
        /// Обновить существующий товар
        /// </summary>
        /// <param name="product">Объект товара с обновленными
данными</param>
        void Update(Product product);

        /// <summary>
        /// Удалить товар по идентификатору
        /// </summary>
        /// <param name="id">Идентификатор товара для удаления</param>
        void Delete(int id);

        /// <summary>
        /// Проверить существование товара по идентификатору
        /// </summary>
        /// <param name="id">Идентификатор товара</param>
        /// <returns>True если товар существует, иначе False</returns>
        bool Exists(int id);
    }
}
```

ProductRepository.cs

```
using WebAPIApp.Models;

namespace WebAPIApp.Repositories
{
    /// <summary>
    /// Реализация репозитория товаров с хранением в памяти
    /// В реальном приложении здесь будет работа с базой данных
    /// </summary>
    public class ProductRepository : IProductRepository
    {
        // Коллекция для хранения товаров в памяти
        private readonly List<Product> _products;

        // Счетчик для генерации уникальных идентификаторов
        private int _nextId = 1;

        /// <summary>
        /// Конструктор репозитория
        /// Инициализирует коллекцию начальными данными
        /// </summary>
        public ProductRepository()
        {
            _products = new List<Product>
            {
                new Product {
                    Id = _nextId++,
                    Name = "Ноутбук",
                    Price = 50000,
                    Category = "Электроника",
                    Description = "Мощный ноутбук для работы и игр",
                    CreatedDate = DateTime.Now
                },
                new Product {
                    Id = _nextId++,
                    Name = "Смартфон",
                    Price = 25000,
                    Category = "Электроника",
                    Description = "Современный смартфон с отличной камерой",
                    CreatedDate = DateTime.Now
                },
                new Product {
                    Id = _nextId++,
                    Name = "Книга по программированию",
                    Price = 1500,
                    Category = "Книги",
                    Description = "Учебное пособие по C# и .NET",
                    CreatedDate = DateTime.Now
                }
            };
        }

        /// <summary>
        /// Получить все товары
        /// </summary>
        public IEnumerable<Product> GetAll()
        {
            // Возвращаем копию коллекции для избежания изменений извне
            return _products.ToList();
        }

        /// <summary>
```

```
        /// Получить товар по идентификатору
        /// </summary>
        public Product GetById(int id)
        {
            // Используем FirstOrDefault для безопасного поиска
            return _products.FirstOrDefault(p => p.Id == id);
        }

        /// <summary>
        /// Добавить новый товар
        /// </summary>
        public void Add(Product product)
        {
            // Проверка входного параметра
            if (product == null)
                throw new ArgumentNullException(nameof(product));

            // Генерируем новый ID и устанавливаем дату создания
            product.Id = _nextId++;
            product.CreatedDate = DateTime.Now;

            // Добавляем товар в коллекцию
            _products.Add(product);
        }

        /// <summary>
        /// Обновить существующий товар
        /// </summary>
        public void Update(Product product)
        {
            if (product == null)
                throw new ArgumentNullException(nameof(product));

            // Находим существующий товар
            var existingProduct = GetById(product.Id);
            if (existingProduct != null)
            {
                // Обновляем все свойства, кроме ID и даты создания
                existingProduct.Name = product.Name;
                existingProduct.Price = product.Price;
                existingProduct.Category = product.Category;
                existingProduct.Description = product.Description;
            }
        }

        /// <summary>
        /// Удалить товар по идентификатору
        /// </summary>
        public void Delete(int id)
        {
            var product = GetById(id);
            if (product != null)
            {
                _products.Remove(product);
            }
        }

        /// <summary>
        /// Проверить существование товара по идентификатору
        /// </summary>
        public bool Exists(int id)
        {
            return _products.Any(p => p.Id == id);
        }
```

```
    }
}
```

Контроллер продуктов (ProductsController.cs)

```
using Microsoft.AspNetCore.Mvc;
using WebAPIApp.Models;
using WebAPIApp.Repositories;

namespace WebAPIApp.Controllers
{
    /// <summary>
    /// Контроллер для управления товарами
    /// </summary>
    [ApiController]
    [Route("api/[controller]")]
    public class ProductsController : ControllerBase
    {
        private readonly IProductRepository _productRepository;

        /// <summary>
        /// Конструктор с внедрением зависимости репозитория
        /// </summary>
        public ProductsController(IProductRepository productRepository)
        {
            _productRepository = productRepository;
        }

        /// <summary>
        /// GET: api/products
        /// Получение всех товаров
        /// </summary>
        [HttpGet]
        public ActionResult<IEnumerable<Product>> GetProducts()
        {
            var products = _productRepository.GetAll();
            return Ok(products);
        }

        /// <summary>
        /// GET: api/products/{id}
        /// Получение товара по ID
        /// </summary>
        [HttpGet("{id}")]
        public ActionResult<Product> GetProduct(int id)
        {
            var product = _productRepository.GetById(id);

            if (product == null)
            {
                return NotFound($"Товар с ID {id} не найден");
            }

            return Ok(product);
        }

        /// <summary>
        /// POST: api/products
        /// Создание нового товара
        /// </summary>
        [HttpPost]
        public ActionResult<Product> CreateProduct(Product product)
        {
```

```
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            _productRepository.Add(product);

            // Возвращаем созданный товар с кодом 201
            return CreatedAtAction(nameof(GetProduct), new { id = product.Id
}, product);
        }

        /// <summary>
        /// PUT: api/products/{id}
        /// Обновление существующего товара
        /// </summary>
        [HttpPut("{id}")]
        public IActionResult UpdateProduct(int id, Product product)
        {
            if (id != product.Id)
            {
                return BadRequest("ID в пути не совпадает с ID в теле
запроса");
            }

            if (!_productRepository.Exists(id))
            {
                return NotFound($"Товар с ID {id} не найден");
            }

            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            _productRepository.Update(product);

            return NoContent(); // 204 No Content
        }

        /// <summary>
        /// DELETE: api/products/{id}
        /// Удаление товара
        /// </summary>
        [HttpDelete("{id}")]
        public IActionResult DeleteProduct(int id)
        {
            if (!_productRepository.Exists(id))
            {
                return NotFound($"Товар с ID {id} не найден");
            }

            _productRepository.Delete(id);

            return NoContent(); // 204 No Content
        }
    }
}
```

Программный класс (Program.cs)

```
using WebAPIApp.Repositories;
```

```
var builder = WebApplication.CreateBuilder(args);

// Добавление сервисов в контейнер
builder.Services.AddControllers();

// Регистрация репозитория как Singleton
builder.Services.AddSingleton<IProductRepository, ProductRepository>();

// Настройка Swagger для документирования API
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Настройка конвейера HTTP запросов
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();
app.UseAuthorization();
app.MapControllers();

app.Run();
```

Тестирование API с помощью Postman

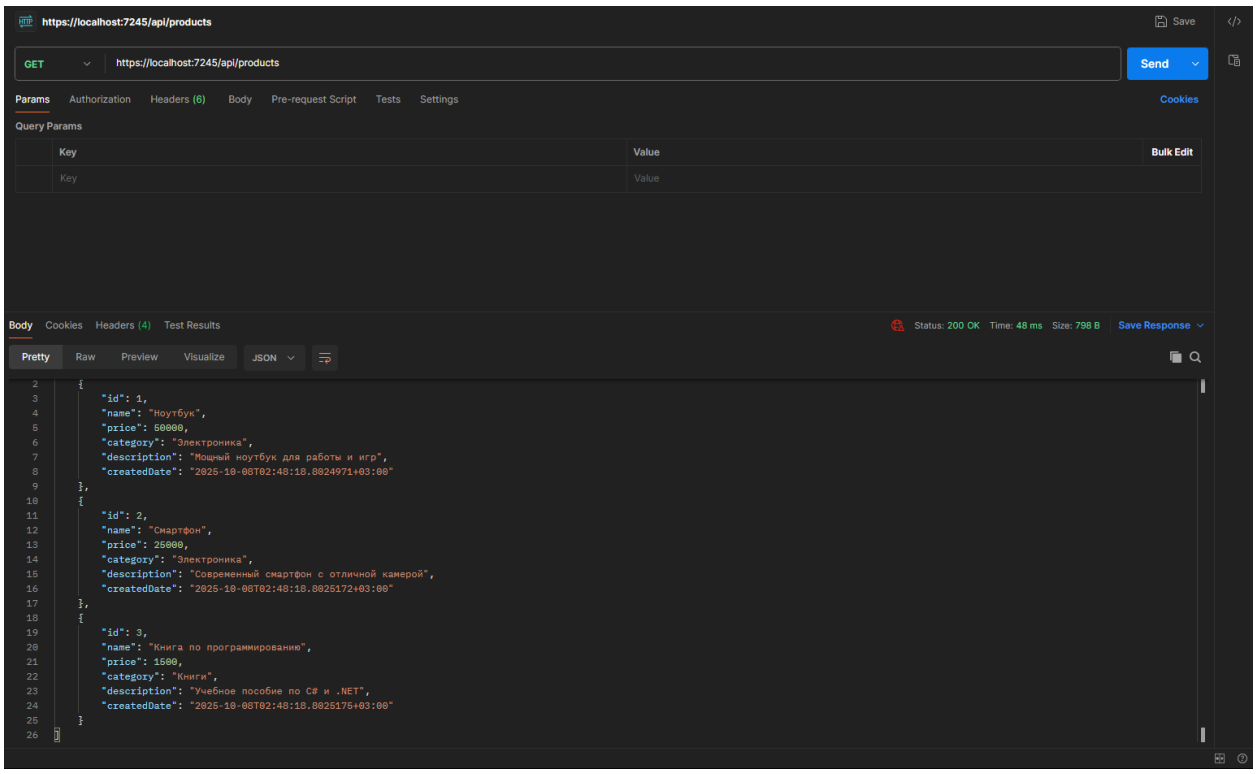
GET /api/products - Получение всех товаров

Запрос:

```
GET https://localhost:7000/api/products
```

Ответ:

```
[
    {
        "id": 1,
        "name": "Ноутбук",
        "price": 50000,
        "category": "Электроника",
        "description": "Мощный ноутбук для работы",
        "createdDate": "2024-01-15T10:30:00"
    },
    {
        "id": 2,
        "name": "Смартфон",
        "price": 25000,
        "category": "Электроника",
        "description": "Современный смартфон",
        "createdDate": "2024-01-15T10:30:00"
    }
]
```



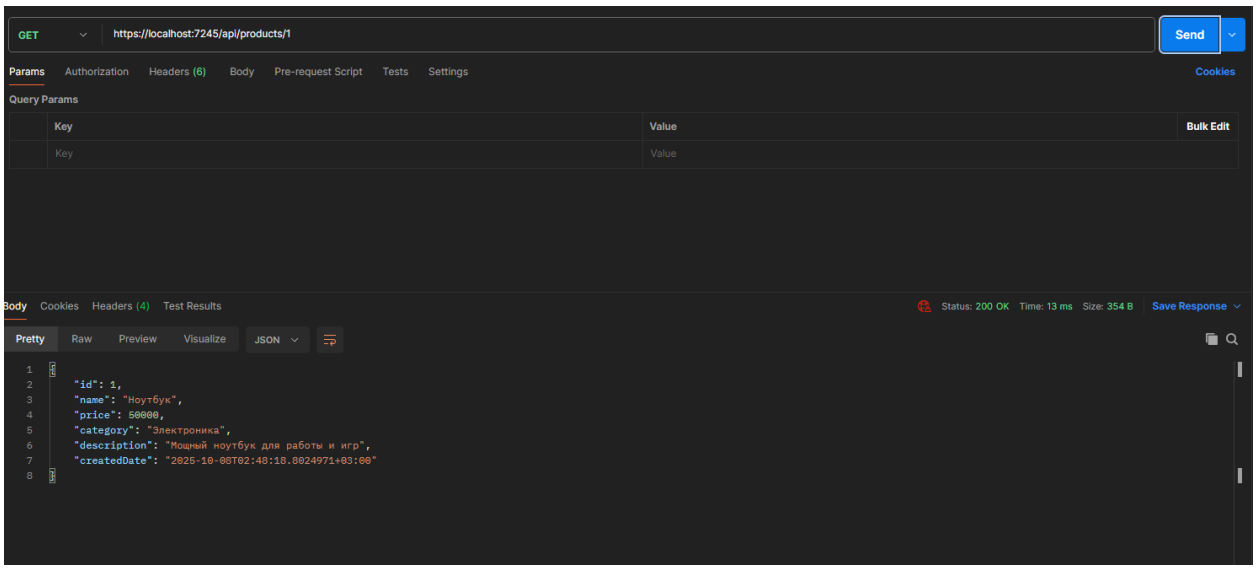
GET /api/products/1 - Получение товара по ID

Запрос:

text
```
GET https://localhost:7000/api/products/1
```

Ответ:

json
```
{
    "id": 1,
    "name": "Ноутбук",
    "price": 50000,
    "category": "Электроника",
    "description": "Мощный ноутбук для работы",
    "createdDate": "2024-01-15T10:30:00"
}
```



POST /api/products - Создание нового товара

Запрос:

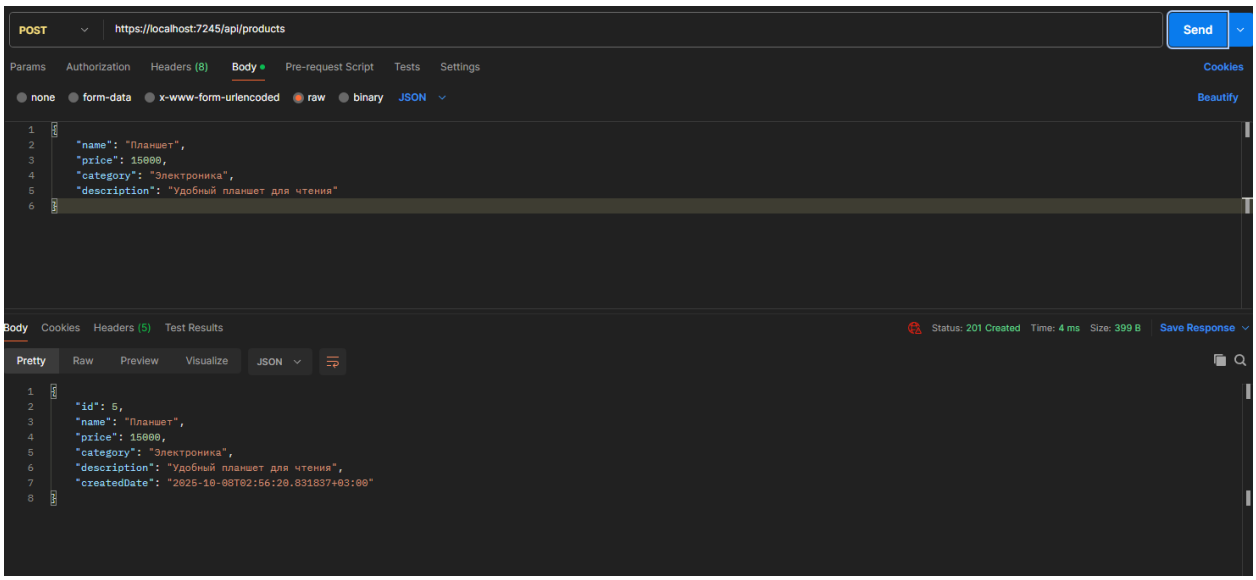
```
POST https://localhost:7000/api/products
Content-Type: application/json

{
    "name": "Планшет",
    "price": 15000,
    "category": "Электроника",
    "description": "Удобный планшет для чтения"
}
```

Ответ (201 Created):

json

```
{
    "id": 4,
    "name": "Планшет",
    "price": 15000,
    "category": "Электроника",
    "description": "Удобный планшет для чтения",
    "createdDate": "2024-01-15T11:45:00"
}
```



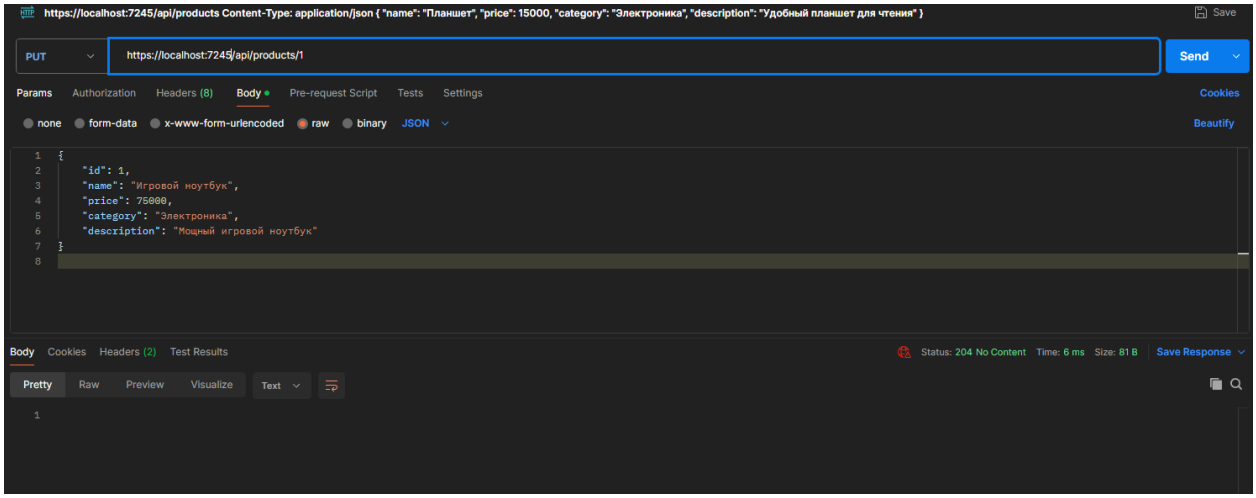
PUT /api/products/1 - Обновление товара

Запрос:

text
```
PUT https://localhost:7000/api/products/1
Content-Type: application/json

{
    "id": 1,
    "name": "Игровой ноутбук",
    "price": 75000,
    "category": "Электроника",
    "description": "Мощный игровой ноутбук"
}
```

Ответ: 204 No Content



DELETE /api/products/2 - Удаление товара

Запрос:

text
```
DELETE https://localhost:7000/api/products/2
```

Ответ: 204 No Content