



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

ЛАБОРАТОРНАЯ РАБОТА № _1_

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

1. Модели данных

2. Контекст данных

3. Репозитории

4. Контроллеры

ProductsController.cs

```
using Microsoft.AspNetCore.Mvc;

using EFCoreApp.Models;
using EFCoreApp.Repositories;
using EFCoreApp.DTOs;

namespace EFCoreApp.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class ProductsController : ControllerBase
    {
        private readonly IProductRepository _productRepository;

        private readonly ILogger<ProductsController> _logger;

        public ProductsController(IProductRepository productRepository,
            ILogger<ProductsController> logger)
        {
            _productRepository = productRepository;

            _logger = logger;
        }

        // GET: api/products
```

```
[HttpGet]

public async Task<ActionResult<IEnumerable<ProductDto>>>
GetProducts()

{
    _logger.LogInformation("Запрос на получение всех продуктов");

    try
    {
        var products = await
        _productRepository.GetAllProductsAsync();

        return Ok(products);
    }

    catch (Exception ex)
    {
        _logger.LogError(ex, "Ошибка при получении продуктов");
        return StatusCode(500, "Внутренняя ошибка сервера");
    }
}

// GET: api/products/5

[HttpGet("{id}")]

public async Task<ActionResult<ProductDetailDto>> GetProduct(int id)

{
    _logger.LogInformation("Запрос на получение продукта с ID:
{ProductId}", id);

    var product = await _productRepository.GetProductByIdAsync(id);

    if (product == null)
    {
        _logger.LogWarning("Продукт с ID {ProductId} не найден", id);
    }
}
```

```

        return NotFound();

    }

    return Ok(product);

}

// GET: api/products/category/1

[HttpGet("category/{categoryId}")]

public async Task<ActionResult<IEnumerable<ProductDto>>>
GetProductsByCategory(int categoryId)

{

    _logger.LogInformation("Запрос на получение продуктов категории:
{CategoryId}", categoryId);

    try

    {

        var products = await
        _productRepository.GetProductsByCategoryAsync(categoryId);

        return Ok(products);

    }

    catch (Exception ex)

    {

        _logger.LogError(ex, "Ошибка при получении продуктов
категории {CategoryId}", categoryId);

        return StatusCode(500, "Внутренняя ошибка сервера");

    }

}

// GET: api/products/search/query

[HttpGet("search/{query}")]

public async Task<ActionResult<IEnumerable<ProductDto>>>
SearchProducts(string query)

```

```

        {
            _logger.LogInformation("Поиск продуктов по запросу: {Query}",
query);

            if (string.IsNullOrEmpty(query))
            {
                return BadRequest("Поисковый запрос не может быть пустым");
            }

            try
            {
                var products = await
_productRepository.SearchProductsAsync(query);

                return Ok(products);
            }

            catch (Exception ex)
            {
                _logger.LogError(ex, "Ошибка при поиске продуктов по запросу:
{Query}", query);

                return StatusCode(500, "Внутренняя ошибка сервера");
            }
        }

        // POST: api/products

        [HttpPost]

        public async Task<ActionResult<ProductDetailDto>>
CreateProduct(Product product)

        {
            _logger.LogInformation("Создание нового продукта: {ProductName}",
product.Name);

            if (!ModelState.IsValid)

```

```

        {

            _logger.LogWarning("Некорректные данные продукта");

            return BadRequest(ModelState);

        }

        try

        {

            var createdProduct = await
            _productRepository.AddProductAsync(product);

            var productDto = await
            _productRepository.GetProductByIdAsync(createdProduct.Id);

            return CreatedAtAction(nameof(GetProduct), new { id =
            createdProduct.Id }, productDto);

        }

        catch (Exception ex)

        {

            _logger.LogError(ex, "Ошибка при создании продукта");

            return StatusCode(500, "Внутренняя ошибка сервера");

        }

    }

    // PUT: api/products/5

    [HttpPut("{id}")]

    public async Task<IActionResult> UpdateProduct(int id, Product
product)

    {

        id);

        _logger.LogInformation("Обновление продукта с ID: {ProductId}",

        id);

        if (id != product.Id)

        {

```

```

        _logger.LogWarning("Несоответствие ID в запросе: {RouteId} !=
{BodyId}", id, product.Id);

        return BadRequest();

    }

    if (!ModelState.IsValid)
    {
        _logger.LogWarning("Некорректные данные продукта для
обновления");
        return BadRequest(ModelState);
    }

    try
    {
        var updatedProduct = await
        _productRepository.UpdateProductAsync(product);

        if (updatedProduct == null)
        {
            _logger.LogWarning("Продукт с ID {ProductId} не найден
для обновления", id);
            return NotFound();
        }

        return NoContent();
    }

    catch (Exception ex)
    {
        _logger.LogError(ex, "Ошибка при обновлении продукта с ID:
{ProductId}", id);
        return StatusCode(500, "Внутренняя ошибка сервера");
    }
}

```

```

        // DELETE: api/products/5

        [HttpDelete("{id}")]

        public async Task<IActionResult> DeleteProduct(int id)
        {
            _logger.LogInformation("Удаление продукта с ID: {ProductId}",
id);

            try
            {
                var result = await _productRepository.DeleteProductAsync(id);

                if (!result)
                {
                    _logger.LogWarning("Продукт с ID {ProductId} не найден
для удаления", id);

                    return NotFound();
                }

                return NoContent();
            }

            catch (Exception ex)
            {
                _logger.LogError(ex, "Ошибка при удалении продукта с ID:
{ProductId}", id);

                return StatusCode(500, "Внутренняя ошибка сервера");
            }
        }
    }
}

```

5. DTO классы

ProductDto.cs

```
namespace EFCoreApp.DTOs
{
    public class ProductDto
    {
        public int Id { get; set; }
        public string Name { get; set; } = string.Empty;
        public string? Description { get; set; }
        public decimal Price { get; set; }
        public int CategoryId { get; set; }
        public string? CategoryName { get; set; }
        public DateTime CreatedDate { get; set; }
        public DateTime UpdatedDate { get; set; }
        public bool IsActive { get; set; }
    }

    public class ProductDetailDto
    {
        public int Id { get; set; }
        public string Name { get; set; } = string.Empty;
        public string? Description { get; set; }
        public decimal Price { get; set; }
        public int CategoryId { get; set; }
        public CategoryDto? Category { get; set; }
        public DateTime CreatedDate { get; set; }
        public DateTime UpdatedDate { get; set; }
        public bool IsActive { get; set; }
    }
}
```

CategoryDto.cs

```
namespace EFCoreApp.DTOs
{
    public class CategoryDto
    {
        public int Id { get; set; }
        public string Name { get; set; } = string.Empty;
        public string? Description { get; set; }
        public DateTime CreatedDate { get; set; }
    }
}
```

6.Методы преобразования

ProductExtensions.cs

```
using EFCoreApp.Models;
using EFCoreApp.DTOs;

namespace EFCoreApp.Extensions
{
    public static class ProductExtensions
    {
        public static ProductDto ToDto(this Product product)
        {
            return new ProductDto
            {
                Id = product.Id,
                Name = product.Name,
                Description = product.Description,
                Price = product.Price,
                CategoryId = product.CategoryId,
                CategoryName = product.Category?.Name,
            }
        }
    }
}
```

```

        CreatedDate = product.CreatedDate,
        UpdatedDate = product.UpdatedDate,
        IsActive = product.IsActive
    };
}

public static ProductDetailDto ToDetailDto(this Product product)
{
    return new ProductDetailDto
    {
        Id = product.Id,
        Name = product.Name,
        Description = product.Description,
        Price = product.Price,
        CategoryId = product.CategoryId,
        Category = product.Category?.ToDto(),
        CreatedDate = product.CreatedDate,
        UpdatedDate = product.UpdatedDate,
        IsActive = product.IsActive
    };
}

}

public static class CategoryExtensions
{
    public static CategoryDto ToDto(this Category category)
    {
        return new CategoryDto
        {
            Id = category.Id,
            Name = category.Name,
            Description = category.Description,
            CreatedDate = category.CreatedDate
        };
    }
}
}

```

5. Настройка приложения

Program.cs

```

using Microsoft.EntityFrameworkCore;
using EFCoreApp.Data;
using EFCoreApp.Repositories;

var builder = WebApplication.CreateBuilder(args);

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

// Настройка Entity Framework Core с SQL Server
builder.Services.AddDbContext<AppDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnec
tion")));

// Регистрация репозитория
builder.Services.AddScoped<IProductRepository, ProductRepository>();

```

```

// Настройка CORS для кросс-доменных запросов
builder.Services.AddCors(options =>
{
    options.AddPolicy("AllowAll", policy =>
    {
        policy.AllowAnyOrigin() // Разрешить запросы с любого домена
        .AllowAnyMethod() // Разрешить любые HTTP методы (GET, POST,
PUT, DELETE и т.д.)
        .AllowAnyHeader(); // Разрешить любые заголовки
    });

    options.AddPolicy("AllowSpecificOrigin", policy =>
    {
        policy.WithOrigins("http://localhost:3000", "https://myapp.com")
        .WithMethods("GET", "POST", "PUT", "DELETE")
        .WithHeaders("Content-Type", "Authorization")
        .AllowCredentials();
    });
});

var app = builder.Build();

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

    // Инициализация базы данных с начальными данными в development
    using var scope = app.Services.CreateScope();
    var context = scope.ServiceProvider.GetRequiredService<AppDbContext>();
    context.Database.EnsureCreated(); // Создает БД если не существует
}

app.UseHttpsRedirection();

// Использование CORS политики
app.UseCors("AllowAll"); // Применяем политику CORS ко всем запросам

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

app.Run();

```

6. Конфигурация

Скриншоты работы приложения

Curl

```
curl -X 'GET' \
  'https://localhost:7277/api/Products' \
  -H 'accept: text/plain'
```



Request URL

```
https://localhost:7277/api/Products
```

Server response

Code

Details

200

Response body

```
[
  {
    "id": 4,
    "name": "Джинсы",
    "description": "Мужские джинсы классического кроя",
    "price": 2999.99,
    "categoryId": 3,
    "categoryName": "Одежда",
    "createdDate": "2025-10-07T21:13:35.6502835",
    "updatedAt": "2025-10-07T21:13:35.6502835",
    "isActive": true
  },
  {
    "id": 3,
    "name": "Научная фантастика",
    "description": "Сборник научно-фантастических рассказов",
    "price": 899.99,
    "categoryId": 2,
    "categoryName": "Книги",
    "createdDate": "2025-10-07T21:13:35.6502832",
    "updatedAt": "2025-10-07T21:13:35.6502833",
    "isActive": true
  },
  {
    "id": 2,
    "name": "Ноутбук HP",
    "description": "Игровой ноутбук",
    "price": 89999.99,
```



Download

Response headers

```
content-type: application/json; charset=utf-8
date: Tue, 07 Oct 2025 21:30:49 GMT
server: Kestrel
x-firefox-spdy: h2
```

Responses

GET

/api/Products/{id}

^

Parameters

Cancel

Name	Description
id * required integer(\$int32) (path)	<input type="text" value="1"/>

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
'https://localhost:7277/api/Products/1' \
-H 'accept: text/plain'
```

Request URL

```
https://localhost:7277/api/Products/1
```

Server response

Code

Details

200

Response body

```
{
  "id": 1,
  "name": "Смартфон Samsung",
  "description": "Флагманский смартфон",
  "price": 59999.99,
  "categoryId": 1,
  "category": {
    "id": 1,
    "name": "Электроника",
    "description": "Электронные устройства и гаджеты",
    "createdAt": "2025-10-07T21:13:35.6499482"
  },
  "createdAt": "2025-10-07T21:13:35.6500659",
  "updatedAt": "2025-10-07T21:13:35.650066",
  "isActive": true
}
```

Download

Response headers

```
content-type: application/json; charset=utf-8
date: Tue, 07 Oct 2025 21:34:26 GMT
server: Kestrel
x-firefox-spy: h2
```

Тестирование API endpoints

GET /api/products

text

```
[
  {
    "id": 4,
    "name": "Джинсы",
    "description": "Мужские джинсы классического кроя",
    "price": 2999.99,
    "categoryId": 3,
    "categoryName": "Одежда",
```

```

      "createdAt": "2025-10-07T21:13:35.6502835",
      "updatedAt": "2025-10-07T21:13:35.6502835",
      "isActive": true
    },
    {
      "id": 3,
      "name": "Научная фантастика",
      "description": "Сборник научно-фантастических рассказов",
      "price": 899.99,
      "categoryId": 2,
      "categoryName": "Книги",
      "createdAt": "2025-10-07T21:13:35.6502832",
      "updatedAt": "2025-10-07T21:13:35.6502833",
      "isActive": true
    },
    {
      "id": 2,
      "name": "Ноутбук HP",
      "description": "Игровой ноутбук",
      "price": 89999.99,
      "categoryId": 1,
      "categoryName": "Электроника",
      "createdAt": "2025-10-07T21:13:35.6502828",
      "updatedAt": "2025-10-07T21:13:35.6502829",
      "isActive": true
    },
    {
      "id": 1,
      "name": "Смартфон Samsung",
      "description": "Флагманский смартфон",
      "price": 59999.99,
      "categoryId": 1,
      "categoryName": "Электроника",
      "createdAt": "2025-10-07T21:13:35.6500659",
      "updatedAt": "2025-10-07T21:13:35.650066",
      "isActive": true
    },
    {
      "id": 5,
      "name": "Футболка",
      "description": "Хлопковая футболка",
      "price": 999.99,
      "categoryId": 3,
      "categoryName": "Одежда",
      "createdAt": "2025-10-07T21:13:35.6502836",
      "updatedAt": "2025-10-07T21:13:35.6502836",
      "isActive": true
    }
  ]
}

```

GET /api/products/category/1

text

```

{
  "id": 1,
  "name": "Смартфон Samsung",
  "description": "Флагманский смартфон",
  "price": 59999.99,
  "categoryId": 1,
  "category": {
    "id": 1,
    "name": "Электроника",
    "description": "Электронные устройства и гаджеты",
    "createdAt": "2025-10-07T21:13:35.6499482"
  },
}

```

```
"createdDate": "2025-10-07T21:13:35.6500659",  
"updatedAt": "2025-10-07T21:13:35.650066",  
"isActive": true  
}
```