**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 № \_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

#### Product.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace EFCoreApp.Models

{

public class Product

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

[Required]

[MaxLength(100)]

public string Name { get; set; } = string.Empty;

[MaxLength(500)]

public string? Description { get; set; }

[Required]

[Column(TypeName = "decimal(18,2)")]

public decimal Price { get; set; }

public int CategoryId { get; set; }

// Навигационное свойство

public virtual Category? Category { get; set; }

public DateTime CreatedDate { get; set; } = DateTime.UtcNow;

public DateTime UpdatedDate { get; set; } = DateTime.UtcNow;

public bool IsActive { get; set; } = true;

}

}

#### Category.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace EFCoreApp.Models

{

public class Category

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

[Required]

[MaxLength(50)]

public string Name { get; set; } = string.Empty;

[MaxLength(200)]

public string? Description { get; set; }

// Навигационное свойство для связи один-ко-многим

public virtual ICollection<Product> Products { get; set; } = new List<Product>();

public DateTime CreatedDate { get; set; } = DateTime.UtcNow;

}

}

#### Order.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace EFCoreApp.Models

{

public class Order

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

[Required]

[MaxLength(100)]

public string CustomerName { get; set; } = string.Empty;

[Required]

[EmailAddress]

public string CustomerEmail { get; set; } = string.Empty;

[Required]

[Column(TypeName = "decimal(18,2)")]

public decimal TotalAmount { get; set; }

public DateTime OrderDate { get; set; } = DateTime.UtcNow;

// Навигационное свойство для связи один-ко-многим

public virtual ICollection<OrderItem> OrderItems { get; set; } = new List<OrderItem>();

public OrderStatus Status { get; set; } = OrderStatus.Pending;

}

public enum OrderStatus

{

Pending = 0,

Confirmed = 1,

Shipped = 2,

Delivered = 3,

Cancelled = 4

}

}

#### OrderItem.cs

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace EFCoreApp.Models

{

public class OrderItem

{

[Key]

[DatabaseGenerated(DatabaseGeneratedOption.Identity)]

public int Id { get; set; }

public int OrderId { get; set; }

public int ProductId { get; set; }

[Required]

public int Quantity { get; set; }

[Required]

[Column(TypeName = "decimal(18,2)")]

public decimal UnitPrice { get; set; }

// Навигационные свойства

public virtual Order? Order { get; set; }

public virtual Product? Product { get; set; }

// Вычисляемое свойство (не сохраняется в БД)

[NotMapped]

public decimal TotalPrice => Quantity \* UnitPrice;

}

}

### 2. Контекст данных

#### AppDbContext.cs

csharp

using Microsoft.EntityFrameworkCore;

using EFCoreApp.Models;

namespace EFCoreApp.Data

{

public class AppDbContext : DbContext

{

// Конструктор для внедрения зависимостей

public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)

{

}

// DbSet для каждой модели

public DbSet<Product> Products { get; set; }

public DbSet<Category> Categories { get; set; }

public DbSet<Order> Orders { get; set; }

public DbSet<OrderItem> OrderItems { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

// Конфигурация модели Product

modelBuilder.Entity<Product>(entity =>

{

entity.HasKey(p => p.Id);

entity.Property(p => p.Name).IsRequired().HasMaxLength(100);

entity.Property(p => p.Description).HasMaxLength(500);

entity.Property(p => p.Price).HasColumnType("decimal(18,2)");

// Связь с Category

entity.HasOne(p => p.Category)

.WithMany(c => c.Products)

.HasForeignKey(p => p.CategoryId)

.OnDelete(DeleteBehavior.Restrict);

// Индекс для улучшения производительности

entity.HasIndex(p => p.CategoryId);

entity.HasIndex(p => p.IsActive);

});

// Конфигурация модели Category

modelBuilder.Entity<Category>(entity =>

{

entity.HasKey(c => c.Id);

entity.Property(c => c.Name).IsRequired().HasMaxLength(50);

entity.Property(c => c.Description).HasMaxLength(200);

// Уникальный индекс для имени категории

entity.HasIndex(c => c.Name).IsUnique();

});

// Конфигурация модели Order

modelBuilder.Entity<Order>(entity =>

{

entity.HasKey(o => o.Id);

entity.Property(o => o.CustomerName).IsRequired().HasMaxLength(100);

entity.Property(o => o.CustomerEmail).IsRequired();

entity.Property(o => o.TotalAmount).HasColumnType("decimal(18,2)");

// Индексы для улучшения производительности

entity.HasIndex(o => o.CustomerEmail);

entity.HasIndex(o => o.OrderDate);

entity.HasIndex(o => o.Status);

});

// Конфигурация модели OrderItem

modelBuilder.Entity<OrderItem>(entity =>

{

entity.HasKey(oi => oi.Id);

entity.Property(oi => oi.UnitPrice).HasColumnType("decimal(18,2)");

// Связь с Order

entity.HasOne(oi => oi.Order)

.WithMany(o => o.OrderItems)

.HasForeignKey(oi => oi.OrderId)

.OnDelete(DeleteBehavior.Cascade);

// Связь с Product

entity.HasOne(oi => oi.Product)

.WithMany()

.HasForeignKey(oi => oi.ProductId)

.OnDelete(DeleteBehavior.Restrict);

// Составной индекс

entity.HasIndex(oi => new { oi.OrderId, oi.ProductId });

});

// Заполнение начальными данными

SeedData(modelBuilder);

}

private void SeedData(ModelBuilder modelBuilder)

{

// Начальные категории

var categories = new List<Category>

{

new Category { Id = 1, Name = "Электроника", Description = "Электронные устройства и гаджеты" },

new Category { Id = 2, Name = "Книги", Description = "Художественная и образовательная литература" },

new Category { Id = 3, Name = "Одежда", Description = "Мужская и женская одежда" }

};

// Начальные продукты

var products = new List<Product>

{

new Product { Id = 1, Name = "Смартфон Samsung", Description = "Флагманский смартфон", Price = 59999.99m, CategoryId = 1 },

new Product { Id = 2, Name = "Ноутбук HP", Description = "Игровой ноутбук", Price = 89999.99m, CategoryId = 1 },

new Product { Id = 3, Name = "Научная фантастика", Description = "Сборник научно-фантастических рассказов", Price = 899.99m, CategoryId = 2 },

new Product { Id = 4, Name = "Джинсы", Description = "Мужские джинсы классического кроя", Price = 2999.99m, CategoryId = 3 },

new Product { Id = 5, Name = "Футболка", Description = "Хлопковая футболка", Price = 999.99m, CategoryId = 3 }

};

modelBuilder.Entity<Category>().HasData(categories);

modelBuilder.Entity<Product>().HasData(products);

}

public override async Task<int> SaveChangesAsync(CancellationToken cancellationToken = default)

{

// Автоматическое обновление дат

var entries = ChangeTracker.Entries()

.Where(e => e.Entity is Product &&

(e.State == EntityState.Added || e.State == EntityState.Modified));

foreach (var entityEntry in entries)

{

if (entityEntry.State == EntityState.Added)

{

((Product)entityEntry.Entity).CreatedDate = DateTime.UtcNow;

}

((Product)entityEntry.Entity).UpdatedDate = DateTime.UtcNow;

}

return await base.SaveChangesAsync(cancellationToken);

}

}

}

### 3. Репозитории

#### IProductRepository.cs

#### using EFCoreApp.Models;

#### using EFCoreApp.DTOs;

#### namespace EFCoreApp.Repositories

#### {

#### public interface IProductRepository

#### {

#### Task<IEnumerable<ProductDto>> GetAllProductsAsync();

#### Task<ProductDetailDto?> GetProductByIdAsync(int id);

#### Task<IEnumerable<ProductDto>> GetProductsByCategoryAsync(int categoryId);

#### Task<Product> AddProductAsync(Product product);

#### Task<Product?> UpdateProductAsync(Product product);

#### Task<bool> DeleteProductAsync(int id);

#### Task<IEnumerable<ProductDto>> SearchProductsAsync(string searchTerm);

#### }

#### }

#### ProductRepository.cs

using Microsoft.EntityFrameworkCore;

using EFCoreApp.Models;

using EFCoreApp.Data;

using EFCoreApp.DTOs;

using EFCoreApp.Extensions;

namespace EFCoreApp.Repositories

{

public class ProductRepository : IProductRepository

{

private readonly AppDbContext \_context;

private readonly ILogger<ProductRepository> \_logger;

public ProductRepository(AppDbContext context, ILogger<ProductRepository> logger)

{

\_context = context;

\_logger = logger;

}

public async Task<IEnumerable<ProductDto>> GetAllProductsAsync()

{

\_logger.LogInformation("Получение всех продуктов");

return await \_context.Products

.Include(p => p.Category)

.Where(p => p.IsActive)

.OrderBy(p => p.Name)

.Select(p => p.ToDto()) // Преобразование в DTO

.ToListAsync();

}

public async Task<ProductDetailDto?> GetProductByIdAsync(int id)

{

\_logger.LogInformation("Получение продукта по ID: {ProductId}", id);

var product = await \_context.Products

.Include(p => p.Category)

.FirstOrDefaultAsync(p => p.Id == id && p.IsActive);

return product?.ToDetailDto(); // Преобразование в Detail DTO

}

public async Task<IEnumerable<ProductDto>> GetProductsByCategoryAsync(int categoryId)

{

\_logger.LogInformation("Получение продуктов по категории: {CategoryId}", categoryId);

return await \_context.Products

.Include(p => p.Category)

.Where(p => p.CategoryId == categoryId && p.IsActive)

.OrderBy(p => p.Name)

.Select(p => p.ToDto()) // Преобразование в DTO

.ToListAsync();

}

public async Task<IEnumerable<ProductDto>> SearchProductsAsync(string searchTerm)

{

\_logger.LogInformation("Поиск продуктов по запросу: {SearchTerm}", searchTerm);

return await \_context.Products

.Include(p => p.Category)

.Where(p => p.IsActive &&

(p.Name.Contains(searchTerm) ||

p.Description != null && p.Description.Contains(searchTerm)))

.OrderBy(p => p.Name)

.Select(p => p.ToDto()) // Преобразование в DTO

.ToListAsync();

}

// Остальные методы остаются без изменений

public async Task<Product> AddProductAsync(Product product)

{

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

\_context.Products.Add(product);

await \_context.SaveChangesAsync();

return product;

}

public async Task<Product?> UpdateProductAsync(Product product)

{

\_logger.LogInformation("Обновление продукта: {ProductId}", product.Id);

var existingProduct = await \_context.Products.FindAsync(product.Id);

if (existingProduct == null)

return null;

existingProduct.Name = product.Name;

existingProduct.Description = product.Description;

existingProduct.Price = product.Price;

existingProduct.CategoryId = product.CategoryId;

existingProduct.UpdatedDate = DateTime.UtcNow;

await \_context.SaveChangesAsync();

return existingProduct;

}

public async Task<bool> DeleteProductAsync(int id)

{

\_logger.LogInformation("Удаление продукта: {ProductId}", id);

var product = await \_context.Products.FindAsync(id);

if (product == null)

return false;

product.IsActive = false;

product.UpdatedDate = DateTime.UtcNow;

await \_context.SaveChangesAsync();

return true;

}

}

}

### 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.IsNullOrWhiteSpace(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)

### {

### \_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("DefaultConnection")));

// Регистрация репозиториев

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

#### appsettings.json

{

"ConnectionStrings": {

"DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=EFCoreAppDb;Trusted\_Connection=true;MultipleActiveResultSets=true"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning",

"Microsoft.EntityFrameworkCore.Database.Command": "Warning"

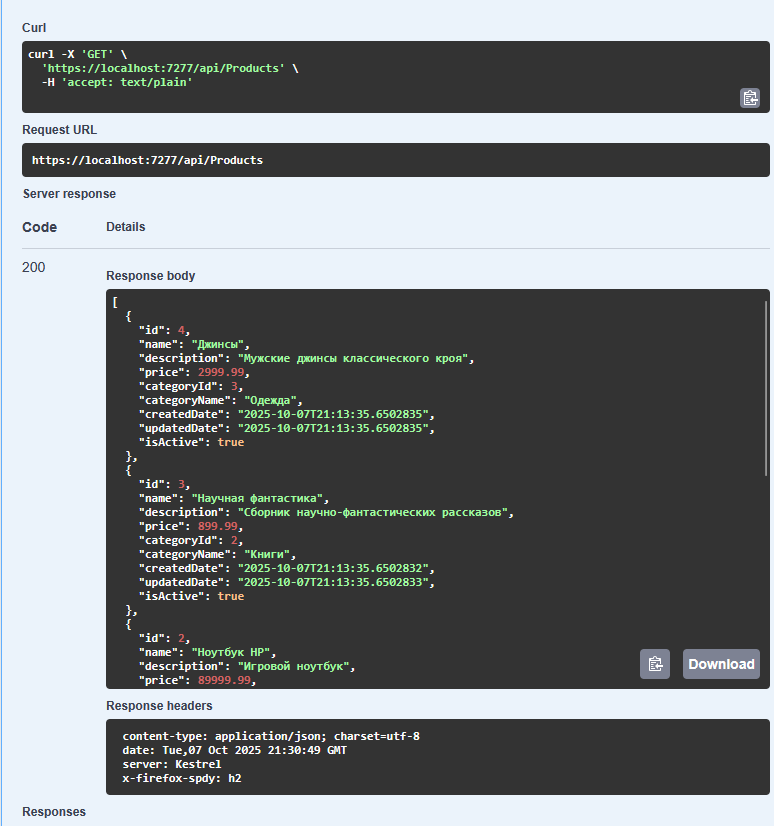
}

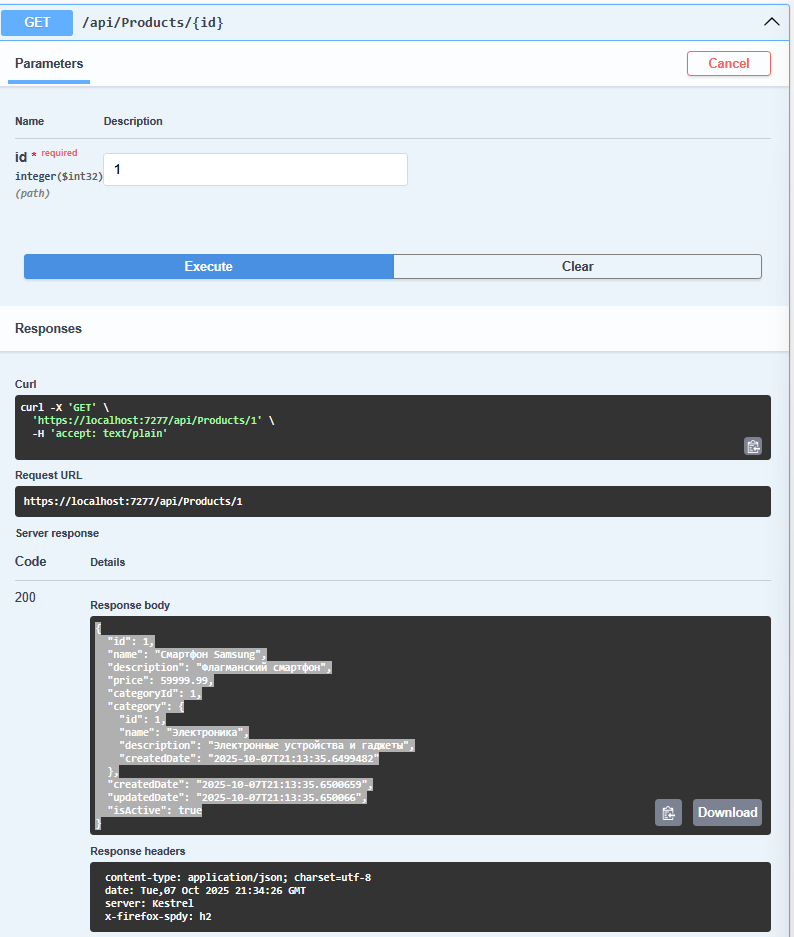
},

"AllowedHosts": "\*"

}

## Скриншоты работы приложения





### Тестирование API endpoints

#### GET /api/products

text

[

{

"id": 4,

"name": "Джинсы",

"description": "Мужские джинсы классического кроя",

"price": 2999.99,

"categoryId": 3,

"categoryName": "Одежда",

"createdDate": "2025-10-07T21:13:35.6502835",

"updatedDate": "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",

"updatedDate": "2025-10-07T21:13:35.6502833",

"isActive": true

},

{

"id": 2,

"name": "Ноутбук HP",

"description": "Игровой ноутбук",

"price": 89999.99,

"categoryId": 1,

"categoryName": "Электроника",

"createdDate": "2025-10-07T21:13:35.6502828",

"updatedDate": "2025-10-07T21:13:35.6502829",

"isActive": true

},

{

"id": 1,

"name": "Смартфон Samsung",

"description": "Флагманский смартфон",

"price": 59999.99,

"categoryId": 1,

"categoryName": "Электроника",

"createdDate": "2025-10-07T21:13:35.6500659",

"updatedDate": "2025-10-07T21:13:35.650066",

"isActive": true

},

{

"id": 5,

"name": "Футболка",

"description": "Хлопковая футболка",

"price": 999.99,

"categoryId": 3,

"categoryName": "Одежда",

"createdDate": "2025-10-07T21:13:35.6502836",

"updatedDate": "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",

"createdDate": "2025-10-07T21:13:35.6499482"

},

"createdDate": "2025-10-07T21:13:35.6500659",

"updatedDate": "2025-10-07T21:13:35.650066",

"isActive": true

}