**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**

## **Шаг 1: Анализ существующего кода и планирование**

### Исходное состояние приложения:

* 4 базовых маршрута: /, /about, /contact, /api/data
* Все маршруты возвращают HTML или JSON
* Отсутствуют параметризованные маршруты
* Нет обработки различных HTTP методов

### Планируемые улучшения:

1. Добавление параметризованных маршрутов
2. Реализация маршрутов с ограничениями
3. Добавление POST endpoints
4. Создание системы обработки ошибок
5. Улучшение навигации между страницами

## **Шаг 2: Добавление промежуточного ПО для логирования**

// Добавлено в начало конвейера middleware

app.Use(async (context, next) =>

{

Console.WriteLine($"Обработка запроса: {context.Request.Method} {context.Request.Path}");

await next();

});

**Цель**: Отслеживание всех входящих запросов для отладки маршрутизации.

## **Шаг 3: Модернизация главной страницы**

### Изменения в маршруте /:

* Добавлена навигация ко всем новым маршрутам
* Создан список доступных endpoints для удобства тестирования
* Добавлено описание функциональности маршрутизации

// Было: простой HTML с базовой навигацией

// Стало: расширенная страница с полным описанием API

## **Шаг 4: Добавление параметризованных маршрутов с ограничениями**

### Маршрут для товаров с ID:

app.MapGet("/products/{id:int:range(1,1000)}", (int id) =>

{

// Ограничение :int:range(1,1000) гарантирует, что:

// - id является целым числом

// - значение в диапазоне 1-1000

// - автоматическая валидация и возврат 404 при несоответствии

});

**Особенности реализации**:

* Использование встроенных ограничений маршрутов
* Автоматическая привязка параметров
* Возврат структурированного JSON

### Маршрут для категорий:

app.MapGet("/category/{categoryName}", (string categoryName) =>

{

// Обработка строковых параметров

// Валидация существующих категорий

// Возврат разных форматов ответа

});

## **Шаг 5: Реализация маршрутов с параметрами запроса**

### Маршрут поиска:

app.MapGet("/search", (string query) =>

{

// Параметр query извлекается из URL: /search?query=текст

// Проверка на пустой запрос

// Возврат релевантных результатов

});

**Особенности**:

* Необязательные параметры (можно вызвать /search без query)
* Обработка различных сценариев (с параметром/без)
* Структурированный JSON ответ

## **Шаг 6: Создание POST endpoint**

### Маршрут для создания товара:

app.MapPost("/api/products", async (HttpContext context) =>

{

// Асинхронное чтение тела запроса

// Десериализация JSON

// Валидация данных

// Имитация сохранения в БД

});

**Сложности и решения**:

* **Проблема**: Обработка потоков ввода
* **Решение**: Использование StreamReader с указанием кодировки
* **Проблема**: Обработка ошибок десериализации
* **Решение**: Try-catch блок для JsonException

## **Шаг 7: Добавление модели данных**

public class Product

{

public int Id { get; set; }

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

public decimal Price { get; set; }

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

public string? Description { get; set; }

public DateTime CreatedAt { get; set; }

}

**Цель**: Типизированная работа с данными в POST запросах.

## **Шаг 8: Реализация обработки несуществующих маршрутов**

app.MapFallback(() =>

{

// Вызывается для всех URL, не соответствующих другим маршрутам

// Возвращает пользовательскую страницу 404

// Сохраняет навигацию для удобства пользователя

});

**Преимущества**:

* Единая точка обработки ошибок
* Пользовательский интерфейс вместо стандартной страницы
* Сохранение навигации

## **Шаг 9: Тестирование всех маршрутов**

### Проверяемые сценарии:

1. **Валидные запросы**:
   * /products/123 - успешный ответ
   * /users/456 - профиль пользователя
   * /search?query=test - поиск с параметром
2. **Невалидные запросы**:
   * /products/abc - должен вернуть 404 (ограничение типа)
   * /products/0 - должен вернуть 404 (ограничение диапазона)
   * /nonexistent - должен вернуть fallback страницу
3. **POST запросы**:
   * Создание товара с валидным JSON
   * Обработка невалидного JSON
   * Проверка кодов ответа

## **Шаг 10: Документирование и рефакторинг**

### Улучшения кода:

* Добавление комментариев к каждому маршруту
* Единообразное форматирование JSON ответов
* Стандартизация HTML шаблонов
* Вынесение повторяющихся стилей

### Создание документации:

* Описание всех доступных endpoints
* Примеры использования
* Ограничения и требования

## **Детали реализации ключевых функций**

### 1. Обработка параметров маршрута:

// Автоматическая привязка и валидация

app.MapGet("/products/{id:int:min(1)}", (int id) => { ... });

// Ручная обработка строковых параметров

app.MapGet("/category/{categoryName}", (string categoryName) =>

{

// Приведение к нижнему регистру для унификации

var normalizedCategory = categoryName.ToLower();

});

### 2. Работа с различными типами ответов:

// HTML ответ

return Results.Text(htmlContent, "text/html");

// JSON ответ

return Results.Json(data, jsonOptions);

// Ошибки

return Results.NotFound("Сообщение об ошибке");

return Results.BadRequest("Неверный запрос");

### 3. Асинхронная обработка POST запросов:

app.MapPost("/api/products", async (HttpContext context) =>

{

// Чтение тела запроса

using var reader = new StreamReader(context.Request.Body, Encoding.UTF8);

var requestBody = await reader.ReadToEndAsync();

// Обработка и ответ

});

## **Проблемы и их решения**

### Проблема 1: Конфликт маршрутов

**Ситуация**: Маршрут /products и /products/{id} могут конфликтовать  
**Решение**: [ASP.NET](https://ASP.NET) Core правильно обрабатывает приоритеты - более конкретные маршруты имеют высший приоритет

### Проблема 2: Обработка невалидных данных

**Ситуация**: Пользователь передает нечисловой ID для /products/{id}  
**Решение**: Ограничения маршрутов автоматически возвращают 404

### Проблема 3: Безопасность типов

**Ситуация**: Необходимость гарантировать тип параметров  
**Решение**: Использование строгой типизации и ограничений

## Итоговые изменения в архитектуре приложения

### До:

Приложение

├── Статические страницы

└── Простой API endpoint

### После:

Приложение

├── Статические страницы

├── Параметризованные маршруты

├── API с различными методами

├── Система валидации

## └── Обработка ошибок

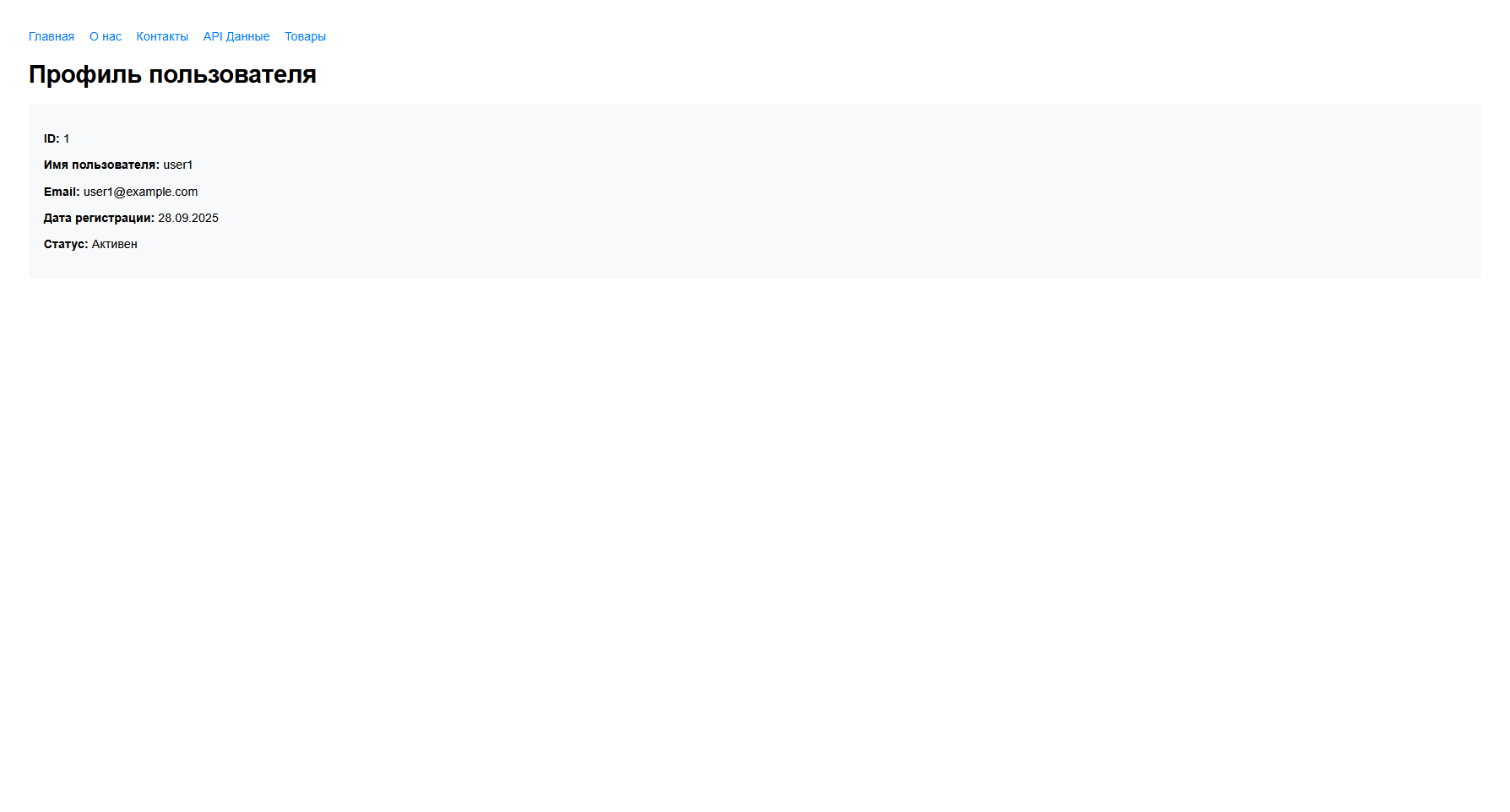
## **Скриншоты работающего приложения**

### Главная страница с навигацией

### Страница товаров

### API данные в JSON формате

### Профиль пользователя с параметром



## **Код приложения с комментариями**

using Microsoft.AspNetCore.Localization;

using Microsoft.AspNetCore.Mvc;

using System.Text;

using System.Text.Json;

var builder = WebApplication.CreateBuilder(args);

// Настройка сервисов приложения

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Настройка конвейера middleware

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

// Добавление статических файлов (wwwroot)

app.UseStaticFiles();

// Промежуточное ПО для логирования запросов

app.Use(async (context, next) =>

{

Console.WriteLine($"Обработка запроса: {context.Request.Method} {context.Request.Path}");

await next();

});

// Определение маршрутов приложения

// Главная страница

app.MapGet("/", () =>

{

return Results.Text(

"""

<!DOCTYPE html>

<html>

<head>

<meta charset=utf-8>

<title>Главная страница</title>

<style>

body { font-family: Arial, sans-serif; margin: 40px; }

nav { margin-bottom: 20px; }

a { margin-right: 15px; text-decoration: none; color: #007bff; }

a:hover { text-decoration: underline; }

.feature-list { margin: 20px 0; padding: 15px; background: #f8f9fa; border-radius: 5px; }

</style>

</head>

<body>

<nav>

<a href="/">Главная</a>

<a href="/about">О нас</a>

<a href="/contact">Контакты</a>

<a href="/api/data">API Данные</a>

<a href="/products">Товары</a>

<a href="/users/1">Профиль пользователя</a>

</nav>

<h1>Добро пожаловать на главную страницу!</h1>

<p>Это минимальное веб-приложение на ASP.NET Core с использованием класса WebApplication.</p>

<div class="feature-list">

<h2>Доступные маршруты:</h2>

<ul>

<li><strong>GET /</strong> - Главная страница</li>

<li><strong>GET /about</strong> - Страница "О нас"</li>

<li><strong>GET /contact</strong> - Страница контактов</li>

<li><strong>GET /api/data</strong> - API данные (JSON)</li>

<li><strong>GET /products</strong> - Список товаров</li>

<li><strong>GET /products/{id}</strong> - Информация о товаре (ID: 1-1000)</li>

<li><strong>GET /users/{id}</strong> - Профиль пользователя (ID: int)</li>

<li><strong>GET /search?query=текст</strong> - Поиск товаров</li>

<li><strong>POST /api/products</strong> - Создание товара (JSON)</li>

<li><strong>GET /category/{categoryName}</strong> - Товары по категории</li>

</ul>

</div>

</body>

</html>

""", "text/html");

});

// Статические страницы

app.MapGet("/about", () =>

{

return Results.Text(

"""

<!DOCTYPE html>

<html>

<head>

<meta charset=utf-8>

<title>О нас</title>

<style>

body { font-family: Arial, sans-serif; margin: 40px; }

nav { margin-bottom: 20px; }

a { margin-right: 15px; text-decoration: none; color: #007bff; }

a:hover { text-decoration: underline; }

</style>

</head>

<body>

<nav>

<a href="/">Главная</a>

<a href="/about">О нас</a>

<a href="/contact">Контакты</a>

<a href="/api/data">API Данные</a>

<a href="/products">Товары</a>

</nav>

<h1>О нашей компании</h1>

<p>Мы создаем современные веб-приложения на ASP.NET Core.</p>

<ul>

<li>Профессиональная команда</li>

<li>Современные технологии</li>

<li>Качественные решения</li>

</ul>

</body>

</html>

""", "text/html");

});

app.MapGet("/contact", () =>

{

return Results.Text(

"""

<!DOCTYPE html>

<html>

<head>

<meta charset=utf-8>

<title>Контакты</title>

<style>

body { font-family: Arial, sans-serif; margin: 40px; }

nav { margin-bottom: 20px; }

a { margin-right: 15px; text-decoration: none; color: #007bff; }

a:hover { text-decoration: underline; }

.contact-info { background: #f8f9fa; padding: 20px; border-radius: 5px; }

</style>

</head>

<body>

<nav>

<a href="/">Главная</a>

<a href="/about">О нас</a>

<a href="/contact">Контакты</a>

<a href="/api/data">API Данные</a>

<a href="/products">Товары</a>

</nav>

<h1>Наши контакты</h1>

<div class="contact-info">

<p><strong>Email:</strong> info@example.com</p>

<p><strong>Телефон:</strong> +7 (123) 456-78-90</p>

<p><strong>Адрес:</strong> г. Москва, ул. Примерная, д. 123</p>

</div>

</body>

</html>

""", "text/html");

});

// API endpoints

app.MapGet("/api/data", () =>

{

var data = new

{

Message = "Добро пожаловать в наше API!",

Timestamp = DateTime.Now,

Version = "1.0",

Features = new[] { "Минимальный API", "Маршрутизация", "JSON ответы", "Параметры маршрутов" }

};

return Results.Json(data, new JsonSerializerOptions

{

WriteIndented = true,

PropertyNamingPolicy = JsonNamingPolicy.CamelCase

});

});

// Маршрут с параметром и ограничением (только числа от 1 до 1000)

app.MapGet("/products/{id:int:range(1,1000)}", (int id) =>

{

var product = new

{

Id = id,

Name = $"Товар {id}",

Price = (id \* 100) + 99.99m,

Category = "Электроника",

Description = $"Описание товара {id}",

InStock = true

};

return Results.Json(product, new JsonSerializerOptions

{

WriteIndented = true,

PropertyNamingPolicy = JsonNamingPolicy.CamelCase

});

});

// Маршрут для списка товаров

app.MapGet("/products", () =>

{

var products = new[]

{

new { Id = 1, Name = "Смартфон", Price = 29999.99m, Category = "Электроника" },

new { Id = 2, Name = "Ноутбук", Price = 59999.99m, Category = "Электроника" },

new { Id = 3, Name = "Книга", Price = 599.99m, Category = "Книги" },

new { Id = 4, Name = "Наушники", Price = 4999.99m, Category = "Электроника" }

};

return Results.Text(

$$"""

<!DOCTYPE html>

<html>

<head>

<meta charset=utf-8>

<title>Список товаров</title>

<style>

body { font-family: Arial, sans-serif; margin: 40px; }

nav { margin-bottom: 20px; }

a { margin-right: 15px; text-decoration: none; color: #007bff; }

a:hover { text-decoration: underline; }

.product { border: 1px solid #ddd; padding: 15px; margin: 10px 0; border-radius: 5px; }

.product h3 { margin-top: 0; }

</style>

</head>

<body>

<nav>

<a href="/">Главная</a>

<a href="/about">О нас</a>

<a href="/contact">Контакты</a>

<a href="/api/data">API Данные</a>

<a href="/products">Товары</a>

</nav>

<h1>Наши товары</h1>

{{string.Join("", products.Select(p => $$"""

<div class="product">

<h3>{{p.Name}}</h3>

<p><strong>Цена:</strong> {{p.Price}} руб.</p>

<p><strong>Категория:</strong> {{p.Category}}</p>

<a href="/products/{{p.Id}}">Подробнее</a>

</div>

"""))}}

<p>Попробуйте также:

<a href="/category/electronics">Товары электроники</a> |

<a href="/category/books">Книги</a>

</p>

</body>

</html>

""", "text/html");

});

// Маршрут с параметром категории (строковый параметр)

app.MapGet("/category/{categoryName}", (string categoryName) =>

{

var categories = new Dictionary<string, string>

{

{ "electronics", "Электроника" },

{ "books", "Книги" }

};

if (categories.TryGetValue(categoryName.ToLower(), out var displayName))

{

return Results.Text(

$$"""

<!DOCTYPE html>

<html>

<head>

<meta charset=utf-8>

<title>Категория: {{displayName}}</title>

<style>

body { font-family: Arial, sans-serif; margin: 40px; }

nav { margin-bottom: 20px; }

a { margin-right: 15px; text-decoration: none; color: #007bff; }

a:hover { text-decoration: underline; }

</style>

</head>

<body>

<nav>

<a href="/">Главная</a>

<a href="/about">О нас</a>

<a href="/contact">Контакты</a>

<a href="/api/data">API Данные</a>

<a href="/products">Товары</a>

</nav>

<h1>Категория: {{displayName}}</h1>

<p>Здесь будут товары категории "{{displayName}}"</p>

<a href="/products">Вернуться ко всем товарам</a>

</body>

</html>

""", "text/html");

}

else

{

return Results.NotFound($"Категория '{categoryName}' не найдена");

}

});

// Маршрут с параметром запроса (query parameter)

app.MapGet("/search", (string query) =>

{

if (string.IsNullOrEmpty(query))

{

return Results.BadRequest("Не указан параметр поиска");

}

var results = new

{

Query = query,

Results = new[]

{

new { Id = 1, Name = $"Результат 1 по запросу '{query}'", Relevance = 0.95 },

new { Id = 2, Name = $"Результат 2 по запросу '{query}'", Relevance = 0.87 },

new { Id = 3, Name = $"Результат 3 по запросу '{query}'", Relevance = 0.76 }

},

TotalCount = 3,

SearchTime = DateTime.Now

};

return Results.Json(results, new JsonSerializerOptions

{

WriteIndented = true,

PropertyNamingPolicy = JsonNamingPolicy.CamelCase

});

});

// Маршрут для профиля пользователя с ограничением (только целые числа)

app.MapGet("/users/{id:int}", (int id) =>

{

var user = new

{

Id = id,

Username = $"user{id}",

Email = $"user{id}@example.com",

RegistrationDate = DateTime.Now.AddDays(-id \* 10),

IsActive = true

};

return Results.Text(

$$"""

<!DOCTYPE html>

<html>

<head>

<meta charset=utf-8>

<title>Профиль пользователя {{user.Username}}</title>

<style>

body { font-family: Arial, sans-serif; margin: 40px; }

nav { margin-bottom: 20px; }

a { margin-right: 15px; text-decoration: none; color: #007bff; }

a:hover { text-decoration: underline; }

.profile { background: #f8f9fa; padding: 20px; border-radius: 5px; }

</style>

</head>

<body>

<nav>

<a href="/">Главная</a>

<a href="/about">О нас</a>

<a href="/contact">Контакты</a>

<a href="/api/data">API Данные</a>

<a href="/products">Товары</a>

</nav>

<h1>Профиль пользователя</h1>

<div class="profile">

<p><strong>ID:</strong> {{user.Id}}</p>

<p><strong>Имя пользователя:</strong> {{user.Username}}</p>

<p><strong>Email:</strong> {{user.Email}}</p>

<p><strong>Дата регистрации:</strong> {{user.RegistrationDate:dd.MM.yyyy}}</p>

<p><strong>Статус:</strong> {{(user.IsActive ? "Активен" : "Неактивен")}}</p>

</div>

</body>

</html>

""", "text/html");

});

// POST маршрут для создания товара

app.MapPost("/api/products", async (HttpContext context) =>

{

try

{

// Чтение тела запроса

using var reader = new StreamReader(context.Request.Body, Encoding.UTF8);

var requestBody = await reader.ReadToEndAsync();

// Десериализация JSON

var product = JsonSerializer.Deserialize<Product>(requestBody, new JsonSerializerOptions

{

PropertyNameCaseInsensitive = true

});

if (product == null)

{

return Results.BadRequest("Неверный формат данных");

}

// Имитация сохранения в базу данных

product.Id = new Random().Next(1000, 10000);

product.CreatedAt = DateTime.Now;

return Results.Json(new

{

Message = "Товар успешно создан",

Product = product

}, new JsonSerializerOptions

{

WriteIndented = true,

PropertyNamingPolicy = JsonNamingPolicy.CamelCase

});

}

catch (JsonException)

{

return Results.BadRequest("Ошибка в формате JSON");

}

});

// Обработка несуществующих маршрутов

app.MapFallback(() =>

{

return Results.Text(

"""

<!DOCTYPE html>

<html>

<head>

<meta charset=utf-8>

<title>Страница не найдена</title>

<style>

body { font-family: Arial, sans-serif; margin: 40px; text-align: center; }

nav { margin-bottom: 20px; }

a { margin-right: 15px; text-decoration: none; color: #007bff; }

a:hover { text-decoration: underline; }

.error { color: #dc3545; }

</style>

</head>

<body>

<nav>

<a href="/">Главная</a>

<a href="/about">О нас</a>

<a href="/contact">Контакты</a>

<a href="/api/data">API Данные</a>

<a href="/products">Товары</a>

</nav>

<h1 class="error">404 - Страница не найдена</h1>

<p>Запрошенная страница не существует.</p>

<p><a href="/">Вернуться на главную страницу</a></p>

</body>

</html>

""", "text/html");

});

// Запуск приложения

app.Run();

// Модель данных для товара должна быть объявлена После любых исполняемых инструкций

public class Product

{

public int Id { get; set; }

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

public decimal Price { get; set; }

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

public string? Description { get; set; }

public DateTime CreatedAt { get; set; }

}