Министерство науки и высшего образования РФ

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №1

по дисциплине «Программные технологии проектирования программного обеспечения вычислительных средств»

на тему «Создание простого Web-приложения»

Выполнили: студенты группы 22ВВП1

Беляев Д. И.

Демин М. С.

Приняли:

Патунин Д. В.

Деев М. В.

Пенза 2025

**Цель работы**

Создание простого Web-приложения

**Задание**

1. Установить Visual Studio 2022.

2. Установить .NET версии 6.0 или старше.

3. Создать проект по шаблону – web api (в зависимости от версии .net сгенерированный код может отличаться)

4. Запустить проект и продемонстрировать открывшуюся страницу в браузере

5. Инициализировать директорию с проектом для системы контроля версий и добавить ваш проект в созданный репозиторий (GitHub, GitLab).

6. Установить библиотеку для взаимодействия с базой данных.

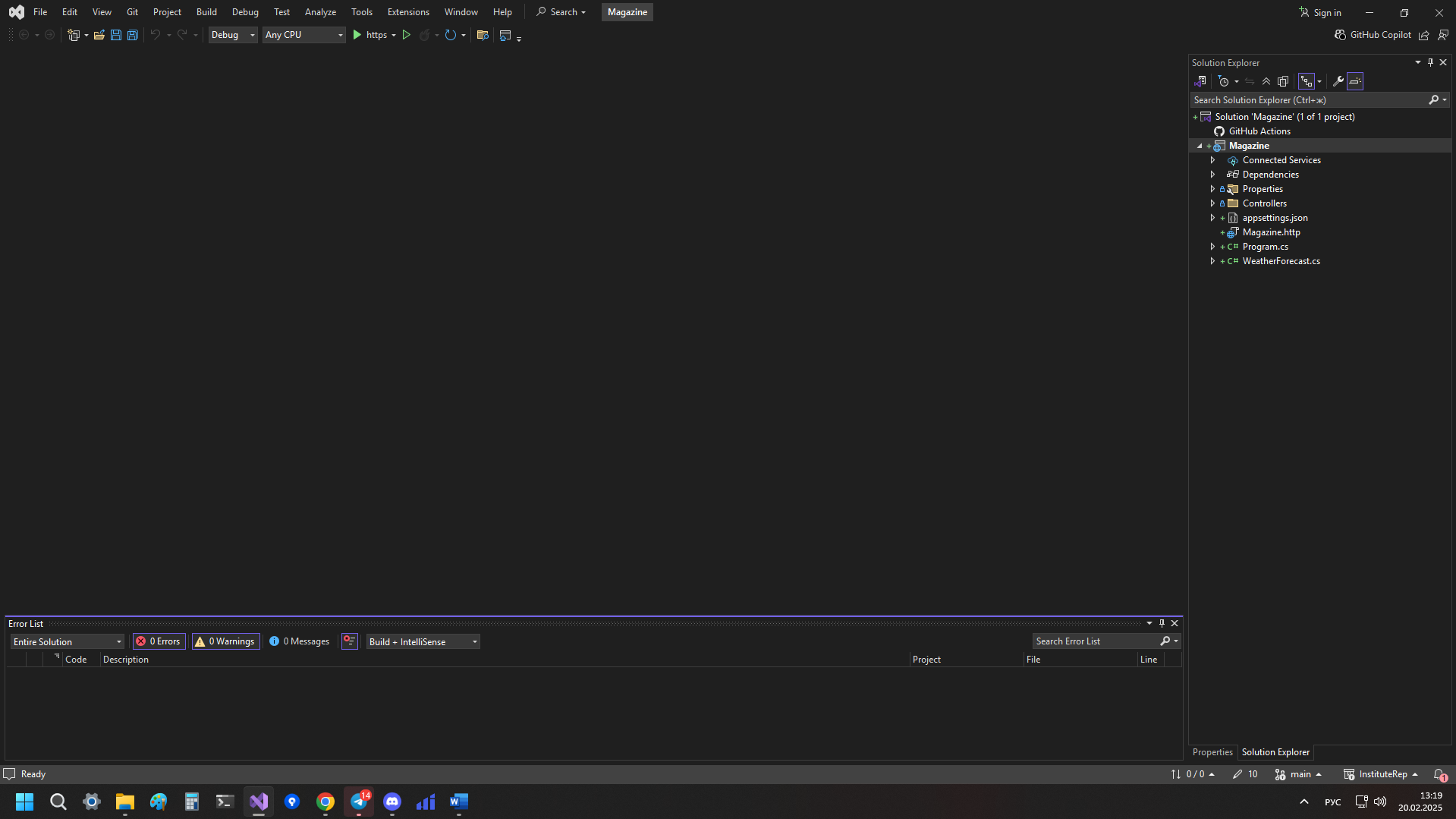
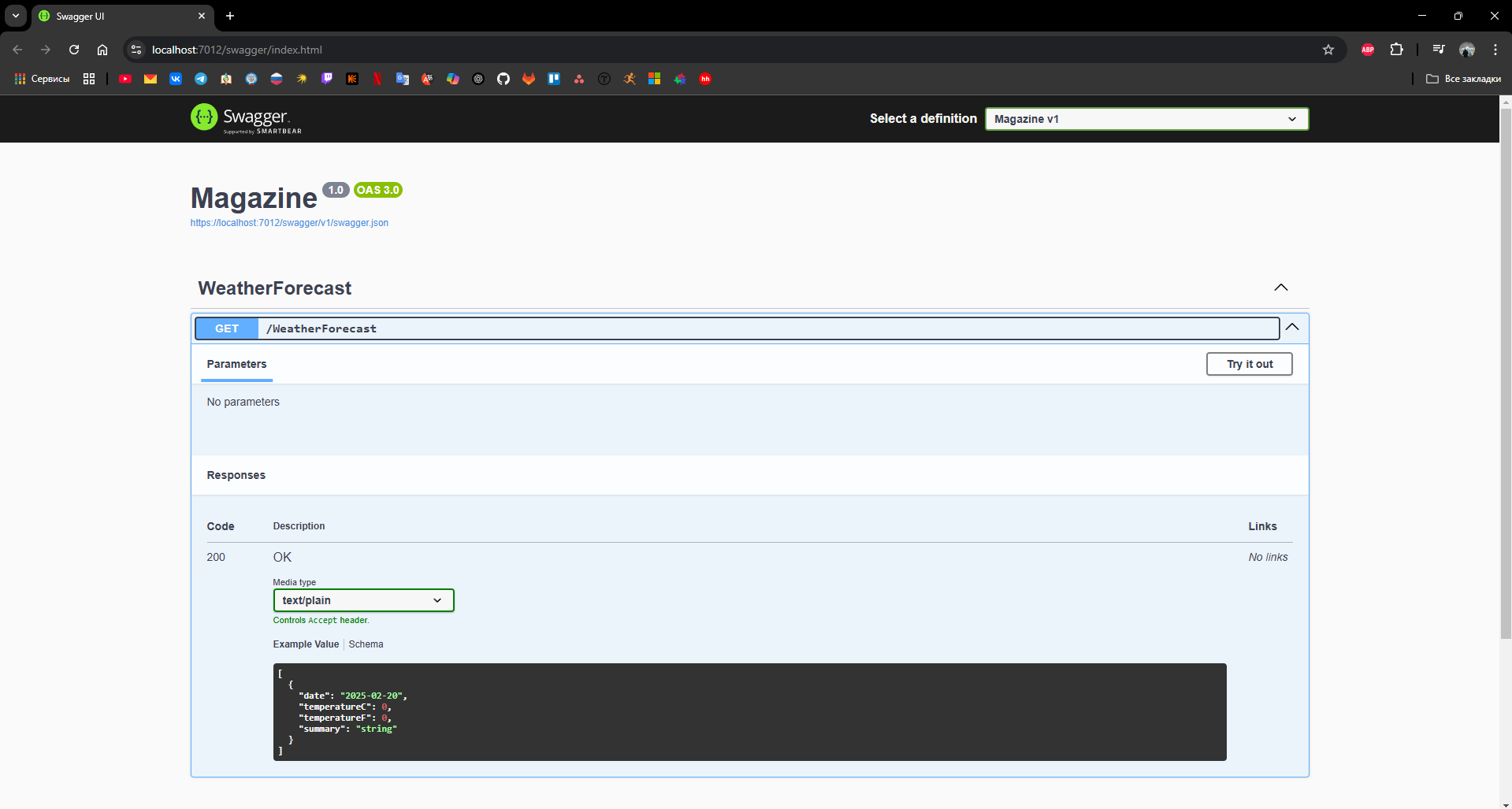
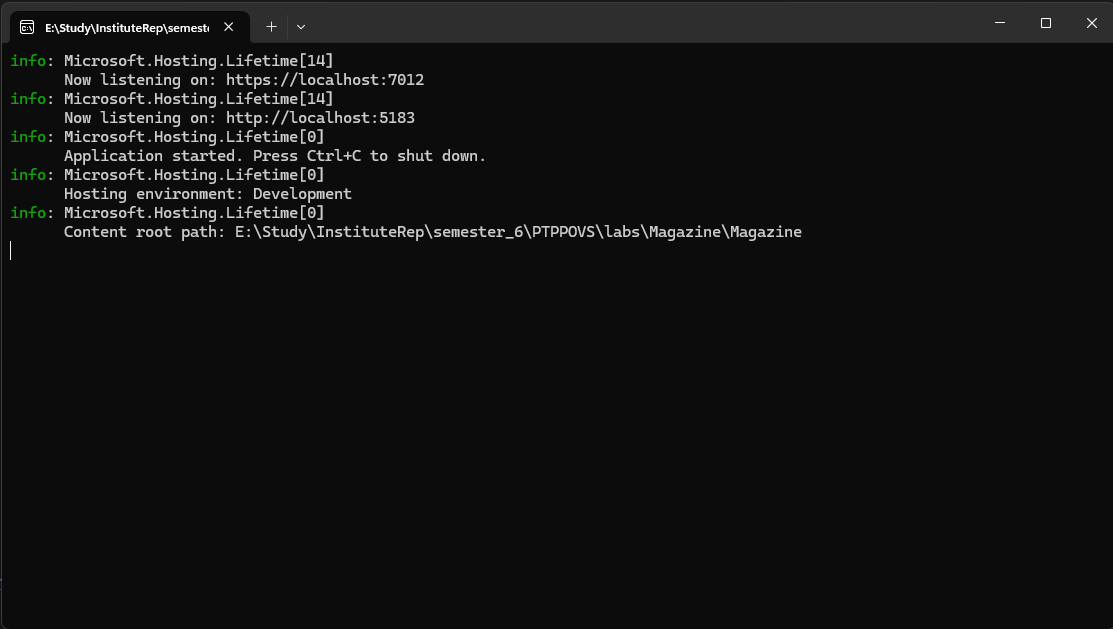
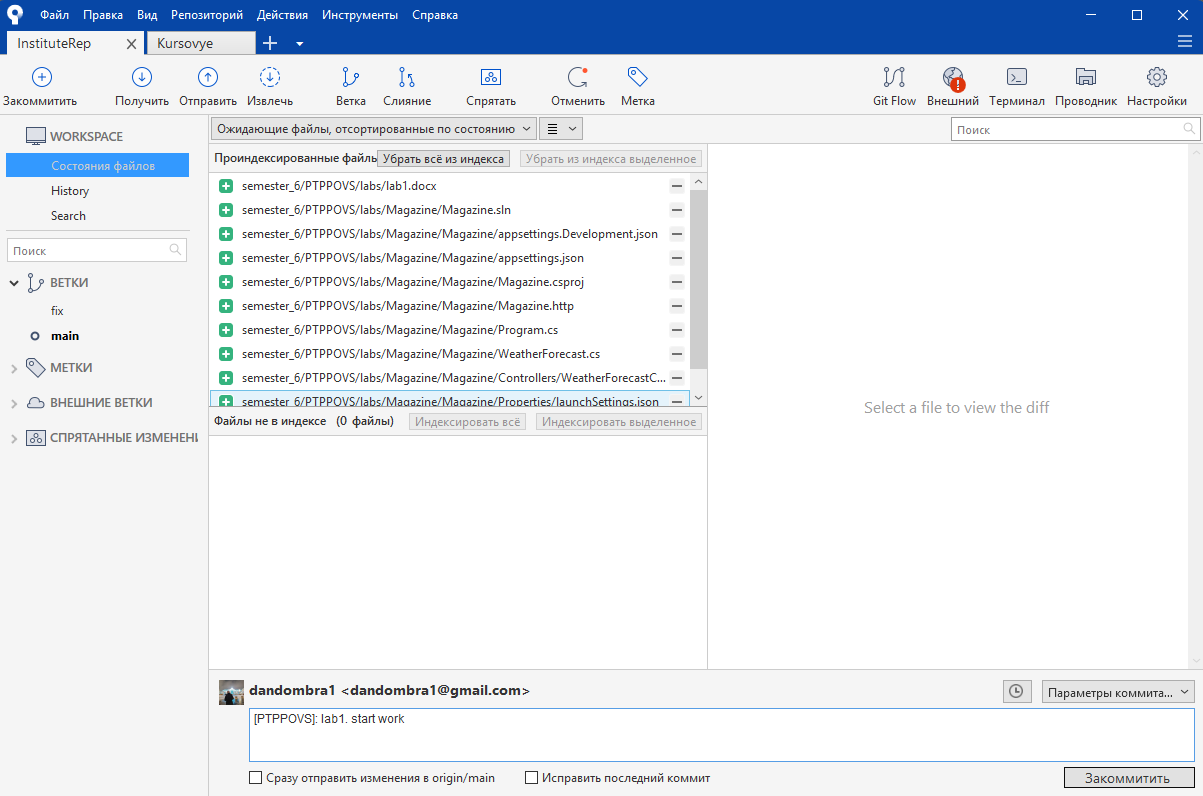
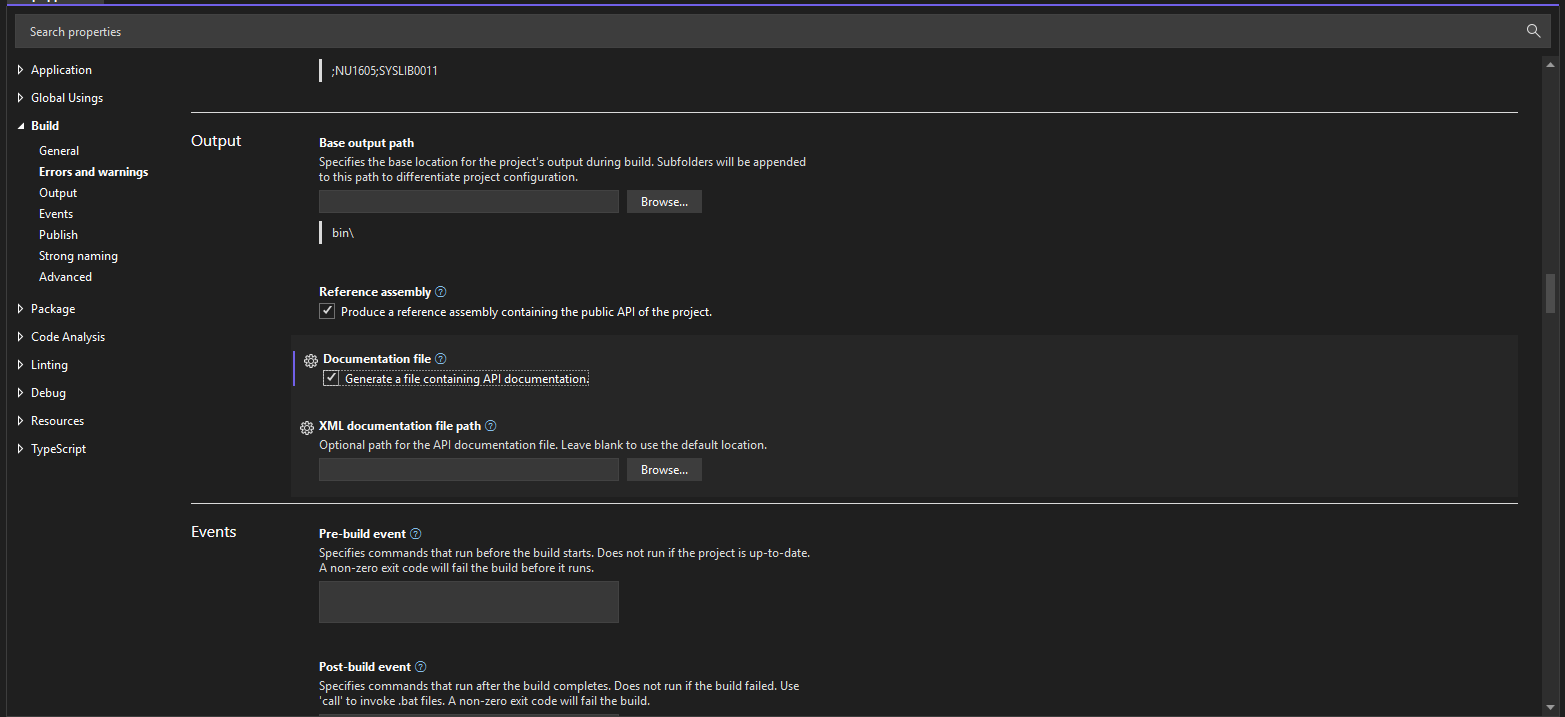
7. Реализовать класс Product, который инкапсулирует товар в интернет-магазине.

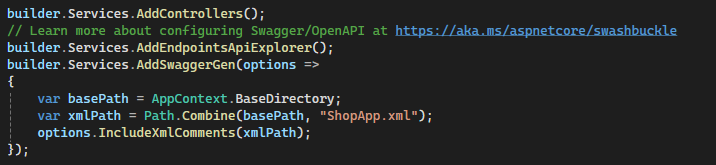
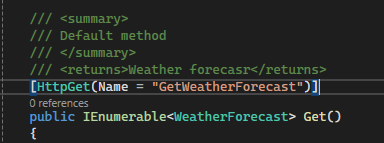
8. Разработать интерфейс IProductService, определяющий операции с товаром (добавление, удаление, изменение, поиск).

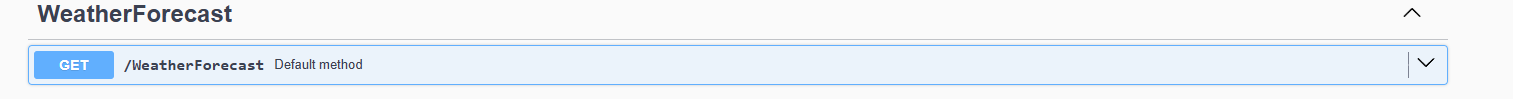
9. Разработать класс ProductService, реализующий интерфейс IProductService

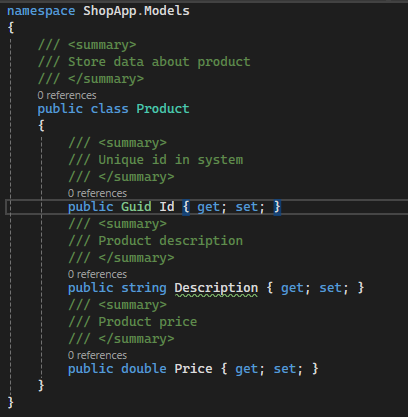
10. Создать контроллер ProductController, принимающий пользовательские запросы и возвращающий результат выполнения операции в формате json

**Ход работы**

1. Создали проект по шаблону и запустили его  
     
     
   
2. Сделали коммит с изменениями  
   
3. Включили генерацию документации  
   



1. Реализовали класс Product

Реализовали интерфейс и сервис продуктов  
IProductService  
namespace ShopApp.Services

{

/// <summary>

/// Abstract product service

/// </summary>

/// <typeparam name="T">Type of Product</typeparam>

public interface IProductService<T>

where T : class

{

/// <summary>

/// Add product to collection

/// </summary>

/// <param name="product">Product</param>

/// <returns>True if add was successful, else = false</returns>

bool Add(T product);

/// <summary>

/// Remove product from collection

/// </summary>

/// <param name="product">Product</param>

/// <returns>Product if remove was successful, else = null</returns>

T? Remove(T product);

/// <summary>

/// Edit product in collection

/// </summary>

/// <param name="product">Product</param>

/// <returns>Product if edit was successful, else = null</returns>

T? Edit(T product);

/// <summary>

/// Search product in collection

/// </summary>

/// <param name="product">Product</param>

/// <returns>Product if search was successful, else = null</returns>

T? Search(Guid product);

}

}

ProductService  
using ShopApp.Models;

namespace ShopApp.Services

{

/// <summary>

/// 'Product' realization of ProductService

/// </summary>

public class ProductService : IProductService<Product>

{

private readonly Dictionary<Guid, Product> \_products;

/// <summary>

/// Base constructor. randomly generate products collection

/// </summary>

public ProductService()

{

\_products = new Dictionary<Guid, Product>();

for (int i = 0; i < 5; i++)

{

Product tempProduct = new Product()

{

Id = Guid.NewGuid(),

Description = $"Random description of product {i}",

Price = Random.Shared.NextDouble()

};

\_products.Add(tempProduct.Id, tempProduct);

}

}

/// <summary>

/// Base constructor. Fill products from parametres

/// </summary>

/// <param name="initialProducts">Initial products</param>

public ProductService(List<Product> initialProducts)

{

\_products = new Dictionary<Guid, Product>();

foreach (var product in initialProducts)

{

\_products.Add(product.Id, product);

}

}

/// <summary>

/// Add product to collection

/// </summary>

/// <param name="product">Product</param>

/// <returns>True if add was successful, else = false</returns>

public bool Add(Product product)

{

if (\_products.ContainsKey(product.Id))

{

return false;

}

\_products.Add(product.Id, product);

return true;

}

/// <summary>

/// Edit product in collection

/// </summary>

/// <param name="product">Product</param>

/// <returns>Product if edit was successful, else = null</returns>

public Product? Edit(Product product)

{

if (!\_products.ContainsKey(product.Id))

{

return null;

}

\_products[product.Id] = (Product)product.Clone();

return \_products[product.Id];

}

/// <summary>

/// Remove product from collection

/// </summary>

/// <param name="product">Product</param>

/// <returns>Product if remove was successful, else = null</returns>

public Product? Remove(Product product)

{

if (!\_products.TryGetValue(product.Id, out Product? removedProduct))

{

return null;

}

\_products.Remove(product.Id);

return removedProduct;

}

/// <summary>

/// Search product in collection

/// </summary>

/// <param name="itemGuid">Product</param>

/// <returns>Product if search was successful, else = null</returns>

public Product? Search(Guid itemGuid)

{

if (!\_products.TryGetValue(itemGuid, out Product? foundProduct))

{

return null;

}

return foundProduct;

}

}

}

1. Реализовали контроллер

using Microsoft.AspNetCore.Mvc;

using ShopApp.Models;

using ShopApp.Services;

namespace ShopApp.Controllers

{

/// <summary>

/// Provides endpoints for interacting with products

/// </summary>

[ApiController]

[Route("api/[controller]")]

public class ProductsController : ControllerBase

{

private readonly ILogger<ProductsController> \_logger;

private readonly IProductService<Product> \_productService;

/// <summary>

/// Base constructor

/// </summary>

/// <param name="logger">ProductsController logger</param>

public ProductsController(ILogger<ProductsController> logger)

{

\_logger = logger;

#if DEBUG

\_productService = new ProductService();

#endif

}

/// <summary>

/// Create new product

/// </summary>

/// <param name="description">Product description</param>

/// <param name="price">Product price</param>

/// <returns>Created product</returns>

[HttpPost, Route("create")]

public Guid CreateProduct(string description, double price)

{

Product createdProduct = new Product()

{

Id = Guid.NewGuid(),

Description = description,

Price = price

};

if (\_productService.Add(createdProduct))

{

return createdProduct.Id;

}

return Guid.Empty;

}

/// <summary>

/// Remove product by id

/// </summary>

/// <param name="id">Product id</param>

/// <returns>Removed product if success, else - null</returns>

[HttpPost, Route("remove")]

public Product? RemoveProduct(Guid id)

{

return \_productService.Remove(id);

}

/// <summary>

/// Edit existed product

/// </summary>

/// <param name="id">Product id</param>

/// <param name="description">Product description</param>

/// <param name="price">Product price</param>

/// <returns>Edited product if success, else - null</returns>

[HttpPost, Route("edit")]

public Product? EditProduct(Guid id, string description, double price)

{

Product tempProduct = new Product()

{

Id = id,

Description = description,

Price = price

};

return \_productService.Edit(tempProduct);

}

/// <summary>

/// Search product by id

/// </summary>

/// <param name="id">Product id</param>

/// <returns>Found product if success, else - null</returns>

[HttpPost, Route("search")]

public Product? SearchProduct(Guid id)

{

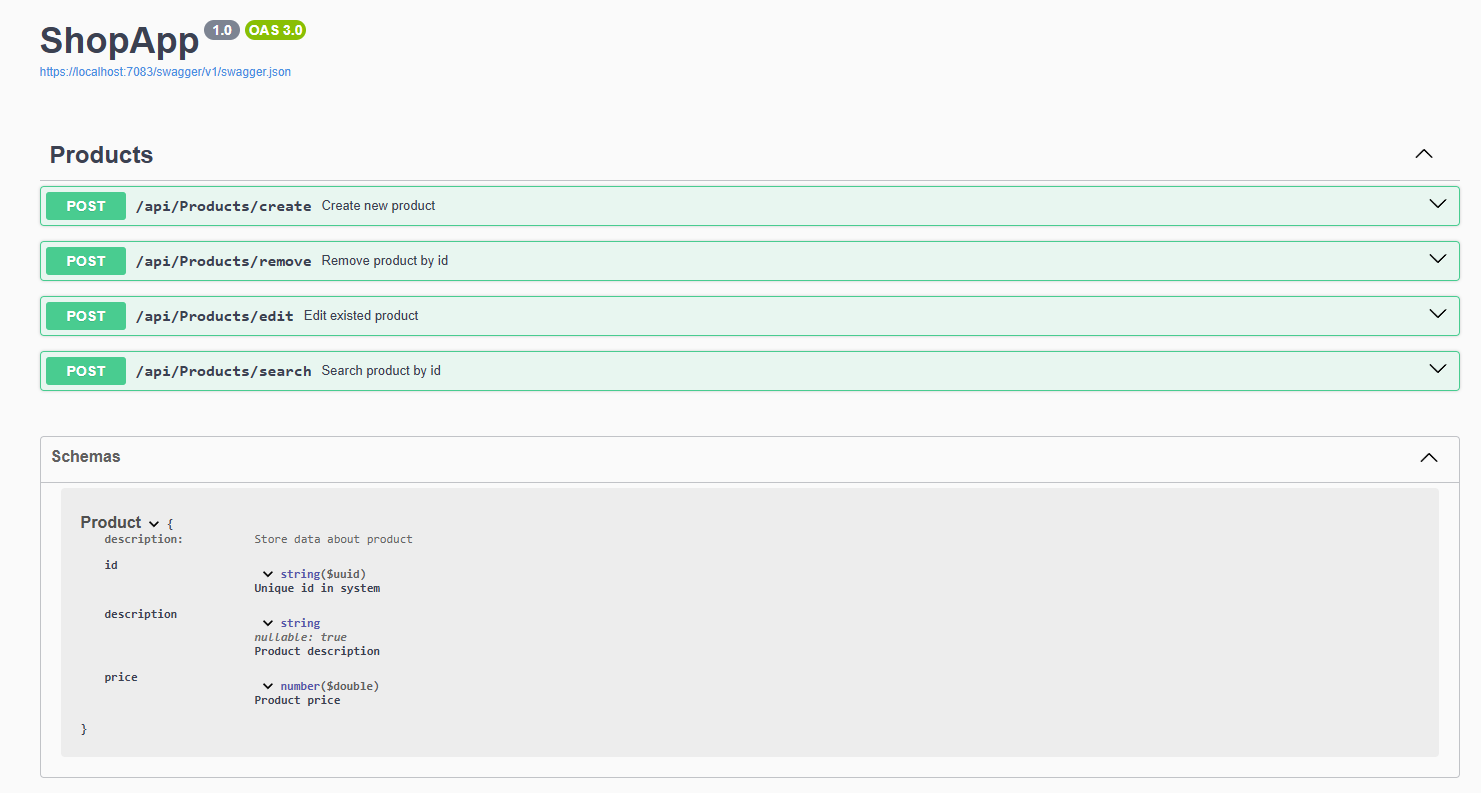
return \_productService.Search(id);

}

}

}

1. Проверка



**Вывод**

Создали простое Web-приложения