Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧËТ**  
по лабораторной работе №1  
по курсу «Программные технологии проектирования ПО ВС»  
на тему «Создание простого Web-приложения»

Выполнили студенты группы 22ВВП1:

Хоссейни Нежад С.А.С.М.

Захаров А.С.

Приняли:

Деев М.В.

Патунин Д.В.

Пенза 2025

**Название**

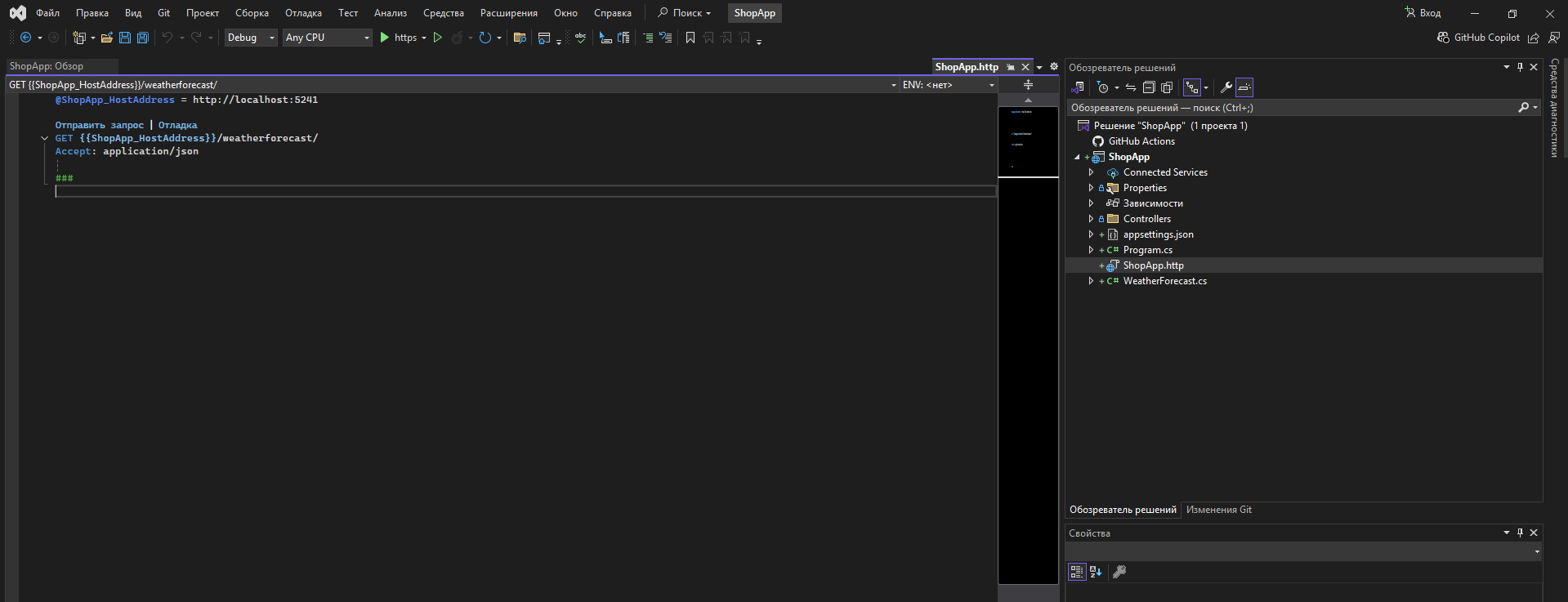
Создание простого Web-приложения

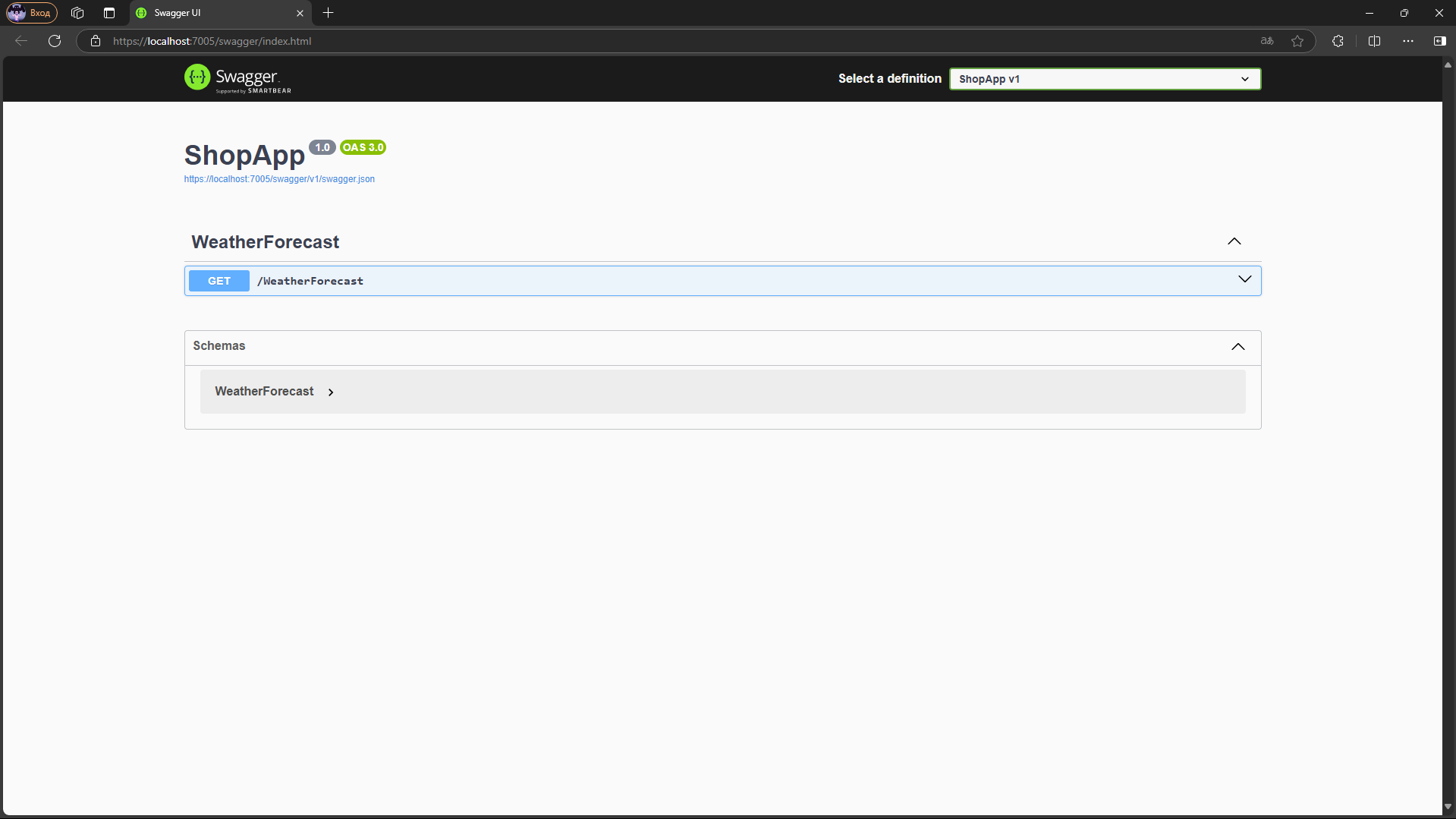
**Лабораторное задание**

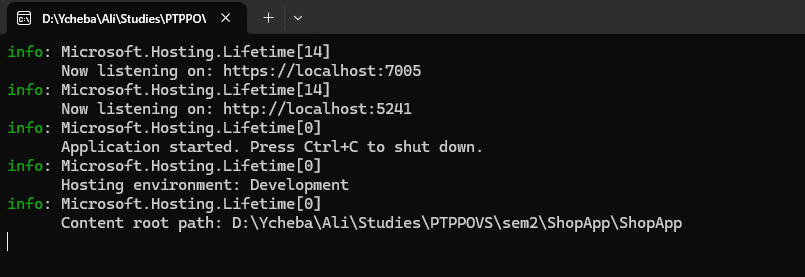
1. Установить Visual Studio 2022.
2. Установить .NET версии 6.0.
3. Создать проект по шаблону – web api
4. Запустить проект и продемонстрировать открывшуюся страницу в браузере
5. Инициализировать директорию с проектом для системы контроля версий и добавить ваш проект в созданный репозиторий.
6. Установить 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**

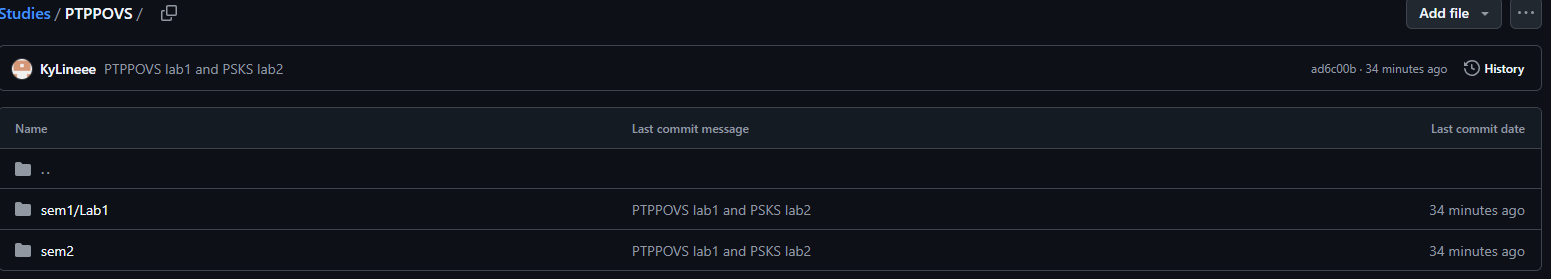
**1 Создали проект по шаблону и запустили его**

****

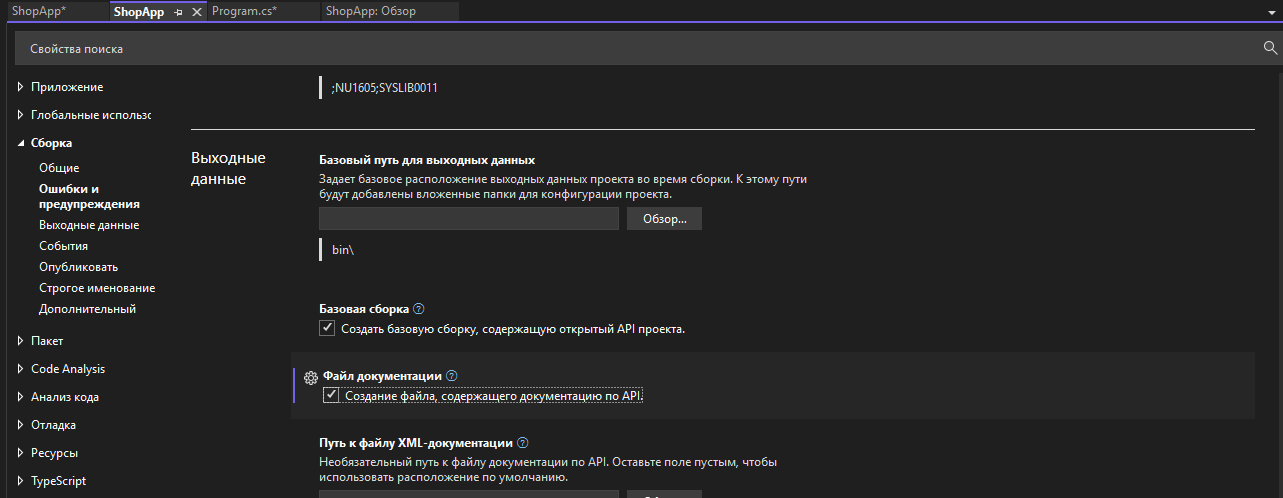
****

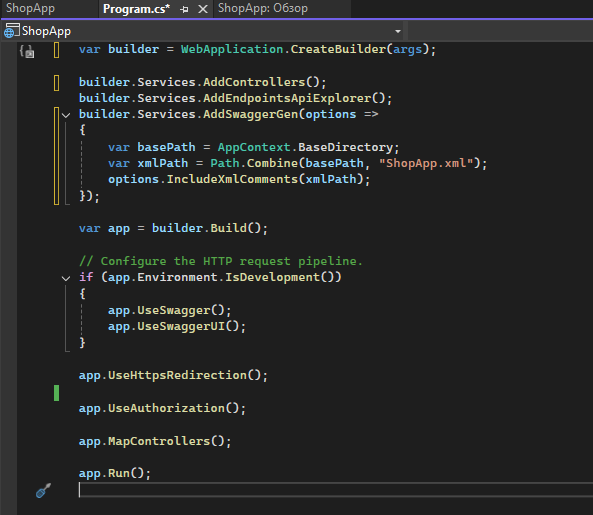
****

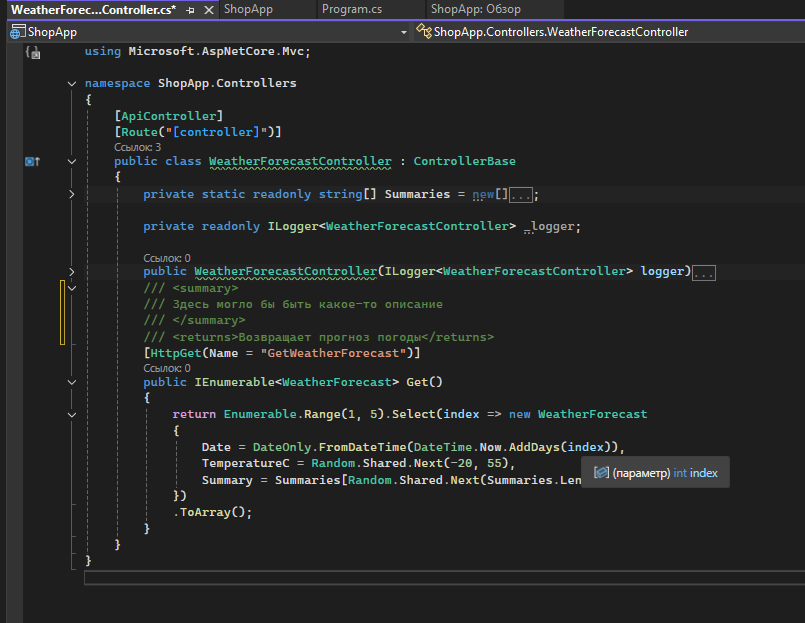
**2 Сделали коммит с изменениями**

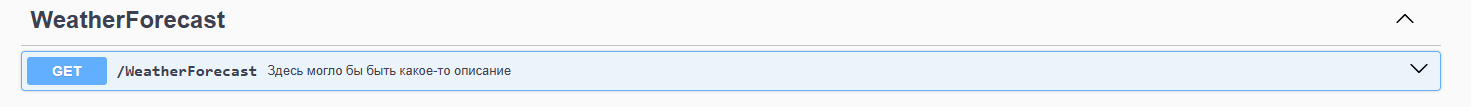
****

**3 Включили документацию**

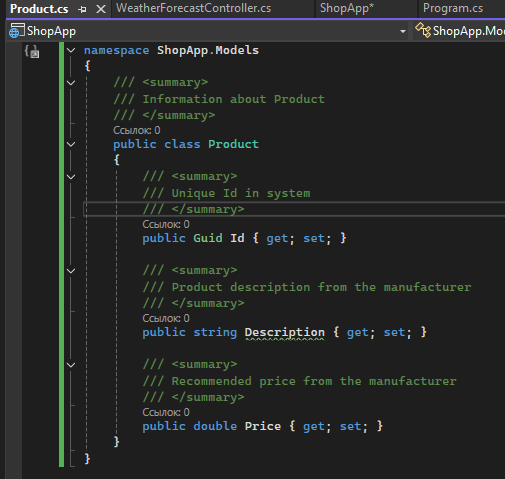
****

****

****

****

**4 Создали класс “Product”**

****

**5 Разработали интерфейс IProductService, определяющий операции с товаром (добавление, удаление, изменение, поиск)**

using ShopApp.Models;

namespace ShopApp.Services

{

public interface IProductService

{

/// <summary>

/// Adds a new product to the database.

/// </summary>

/// <param name="product">The product to add.</param>

/// <returns>The created product entity.</returns>

Product? Add(Product product);

/// <summary>

/// Removes a product from the database.

/// </summary>

/// <param name="id">The product identifier.</param>

/// <returns>The removed product.</returns>

Product? Remove(Guid id);

/// <summary>

/// Updates an existing product in the database.

/// </summary>

/// <param name="product">The product with updated data.</param>

/// <returns>The updated product entity.</returns>

Product? Edit(Product product);

/// <summary>

/// Get Information about product by id

/// </summary>

/// <param name="id">The ID of the product</param>

/// <returns>Returns product if the product exists. Otherwise, it returns null</returns>

Product? Search(Guid id);

}

}

**6 Разработали класс ProductService, реализующий интерфейс IProductService**

using ShopApp.Models;

namespace ShopApp.Services

{

/// <summary>

/// Provides services for managing products.

/// </summary>

public class ProductService : IProductService

{

private Dictionary<Guid, Product> \_products;

/// <summary>

/// Initializes a new instance of the <see cref="ProductService"/> class.

/// </summary>

public ProductService()

{

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

this.InitProducts();

}

private void InitProducts()

{

foreach (var item in Enumerable.Range(1, 5))

{

var product = new Product()

{

Description = $"Information about {item}",

Id = Guid.NewGuid(),

Price = Random.Shared.NextDouble()

};

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

}

}

/// <summary>

/// Adds a new product to the collection.

/// </summary>

/// <param name="product">The product to add.</param>

/// <returns>The added product if successful; otherwise, <c>null</c>.</returns>

public Product? Add(Product product)

{

return \_products.TryAdd(product.Id, product) ? product : null;

}

/// <summary>

/// Removes a product by its unique identifier.

/// </summary>

/// <param name="id">The unique identifier of the product to remove.</param>

/// <returns>The removed product if found; otherwise, <c>null</c>.</returns>

public Product? Remove(Guid id)

{

if (\_products.TryGetValue(id, out Product? product))

{

\_products.Remove(id);

return product;

}

return null;

}

/// <summary>

/// Edits an existing product.

/// </summary>

/// <param name="product">The product with updated information.</param>

/// <returns>The edited product if found; otherwise, <c>null</c>.</returns>

public Product? Edit(Product product)

{

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

{

return null;

}

\_products[product.Id] = product;

return product;

}

/// <summary>

/// Searches for a product by its unique identifier.

/// </summary>

/// <param name="id">The unique identifier of the product to search for.</param>

/// <returns>The found product if it exists; otherwise, <c>null</c>.</returns>

public Product? Search(Guid id)

{

\_products.TryGetValue(id, out var product);

return product;

}

}

}

**7 Создали контроллер ProductController, принимающий пользовательские запросы и возвращающий результат выполнения операции в формате json**

using Microsoft.AspNetCore.Mvc;

using ShopApp.Models;

using ShopApp.Services;

// <summary>

// Provides endpoints for interacting with products

// </summary>

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

[ApiController]

public class ProductsController : ControllerBase

{

private readonly IProductService \_products;

public ProductsController(IProductService products)

{

\_products = products;

}

/// <summary>

/// Creates a product with the specified description and price.

/// </summary>

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

/// <param name="price">The price of the product</param>

/// <returns>Returns the created product if successful. Otherwise, it returns <c>null</c></returns>

[HttpPost]

public Product? CreateProduct(string description, double price)

{

if (string.IsNullOrEmpty(description) || price < 0)

{

return null;

}

var product = new Product()

{

Id = Guid.NewGuid(),

Description = description,

Price = price

};

return \_products.Add(product);

}

/// <summary>

/// Get Information about product by id

/// </summary>

/// <param name="id">The ID of the product</param>

/// <returns>Returns product if the product exists. Otherwise, it returns <c>null</c></returns>

[HttpGet]

public Product? SearchProduct(Guid id)

{

return \_products.Search(id);

}

/// <summary>

/// Remove product by id

/// </summary>

/// <param name="id">The ID of the product</param>

/// <returns>Returns the removed product if it exists. Otherwise, it returns <c>null</c></returns>

[HttpPost]

public Product? RemoveProduct(Guid id)

{

return \_products.Remove(id);

}

/// <summary>

/// Edit product by id, description and price

/// </summary>

/// <param name="id">The ID of the product</param>

/// <param name="description">The new description</param>

/// <param name="price">The new price of the product</param>

/// <returns>Returns the edited product if it exists. Otherwise, it returns <c>null</c></returns>

[HttpPost]

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

{

var product = \_products.Search(id);

if (product == null || string.IsNullOrEmpty(description) || price < 0)

{

return null;

}

product.Description = description;

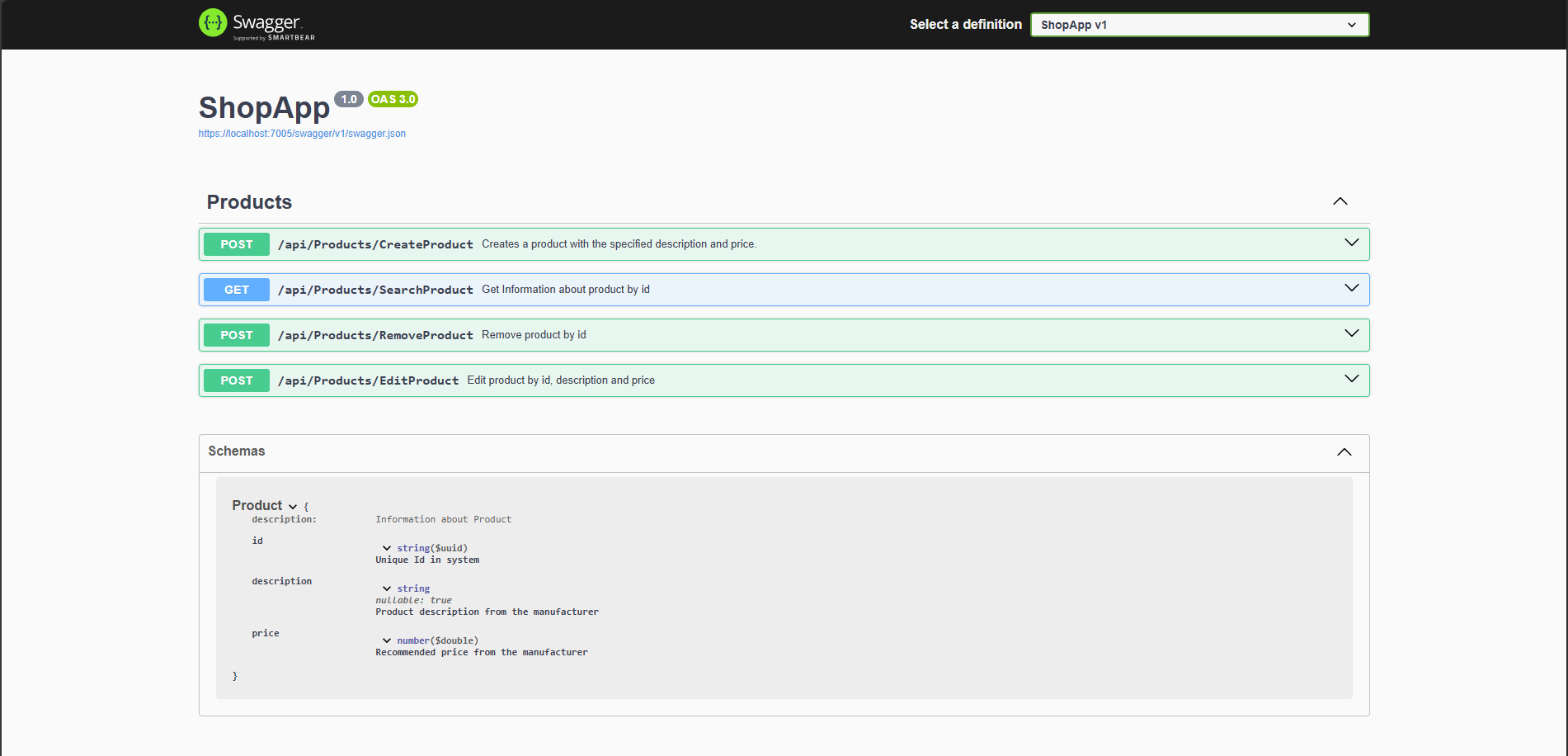
product.Price = price;

return \_products.Edit(product);

}

}

**8 Проверка**



**Вывод**

Создали простое Web-приложения