Министерство науки и высшего образования РФ

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №2

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. С помощью внедрения зависимостей связать разработанный класс ProductService с интерфейсом IProductService (builder.Service.AddSingleton).

2. В конструктор контроллера ProductController передать интерфейс IProductService и присвоить его внутренней переменной.

3. Усовершенствовать класс ProductService. Конструктор класса принимает на вход параметр IConfiguration. Присвоить его внутренней переменной. Конфигурация по умолчанию считывает из файла appsettings.json. Внутри файла объявить поле "DataBaseFilePath" : "database.txt".

4. Внутри ProductService объявить приватный метод InitFromFile, который из файла читает сериализованный список продуктов. Путь до файла запрашивать из конфигурации.

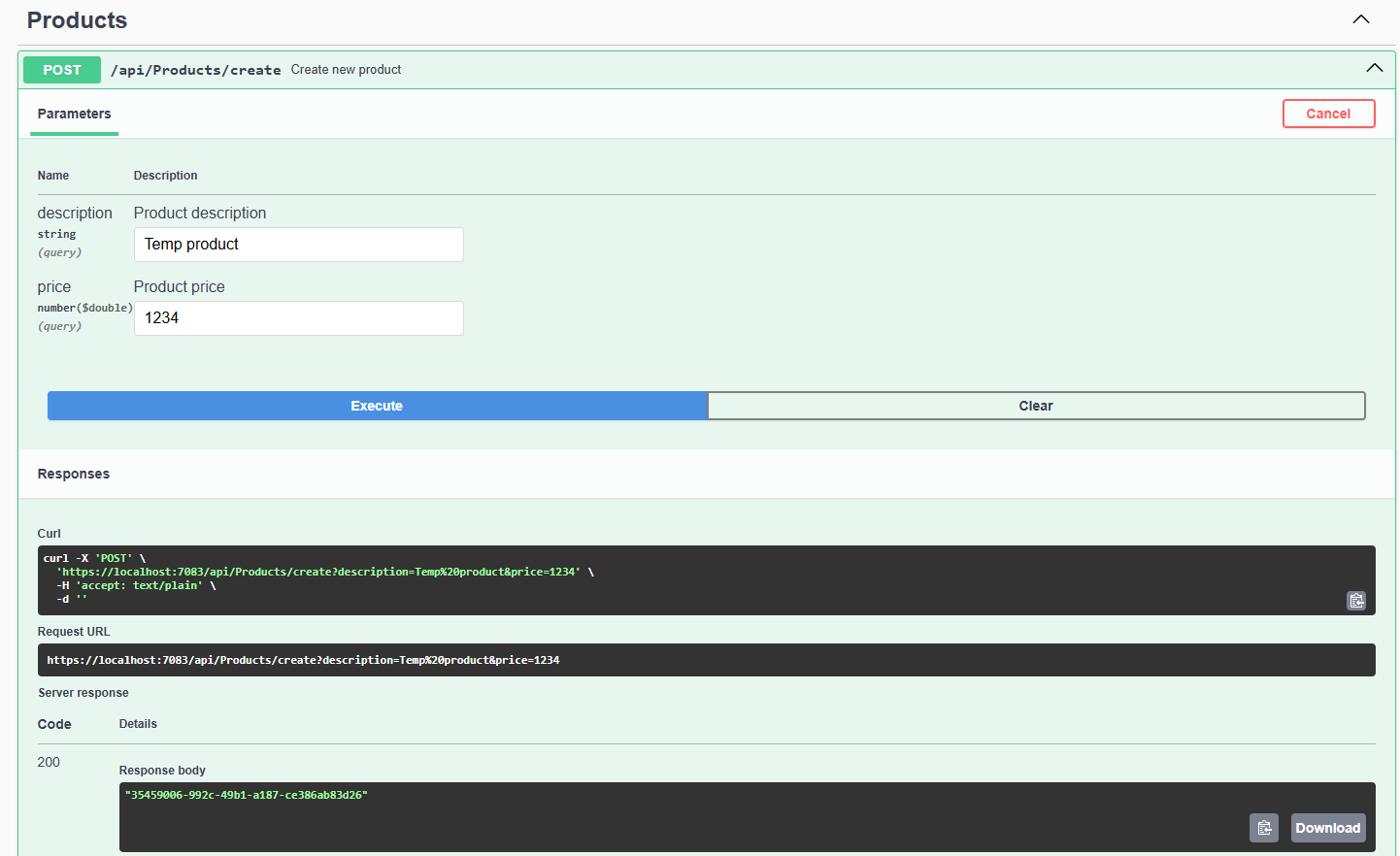
5. Десериализовать считанный файл в словарь ключ – значение. Данный словарь будет служить копией базы данных в памяти.

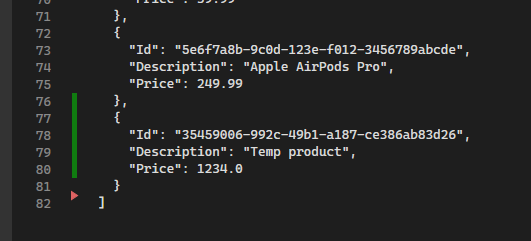
6. Явно реализовать методы класса ProductService. Все методы связанные с изменением состояния базы данных должны записывать изменения как в оперативную память, так и на диск. Использовать мьютекс для записи изменений.

7. Для сохранения изменений на диск реализовать приватный метод WriteToFile внутри класса ProductService.

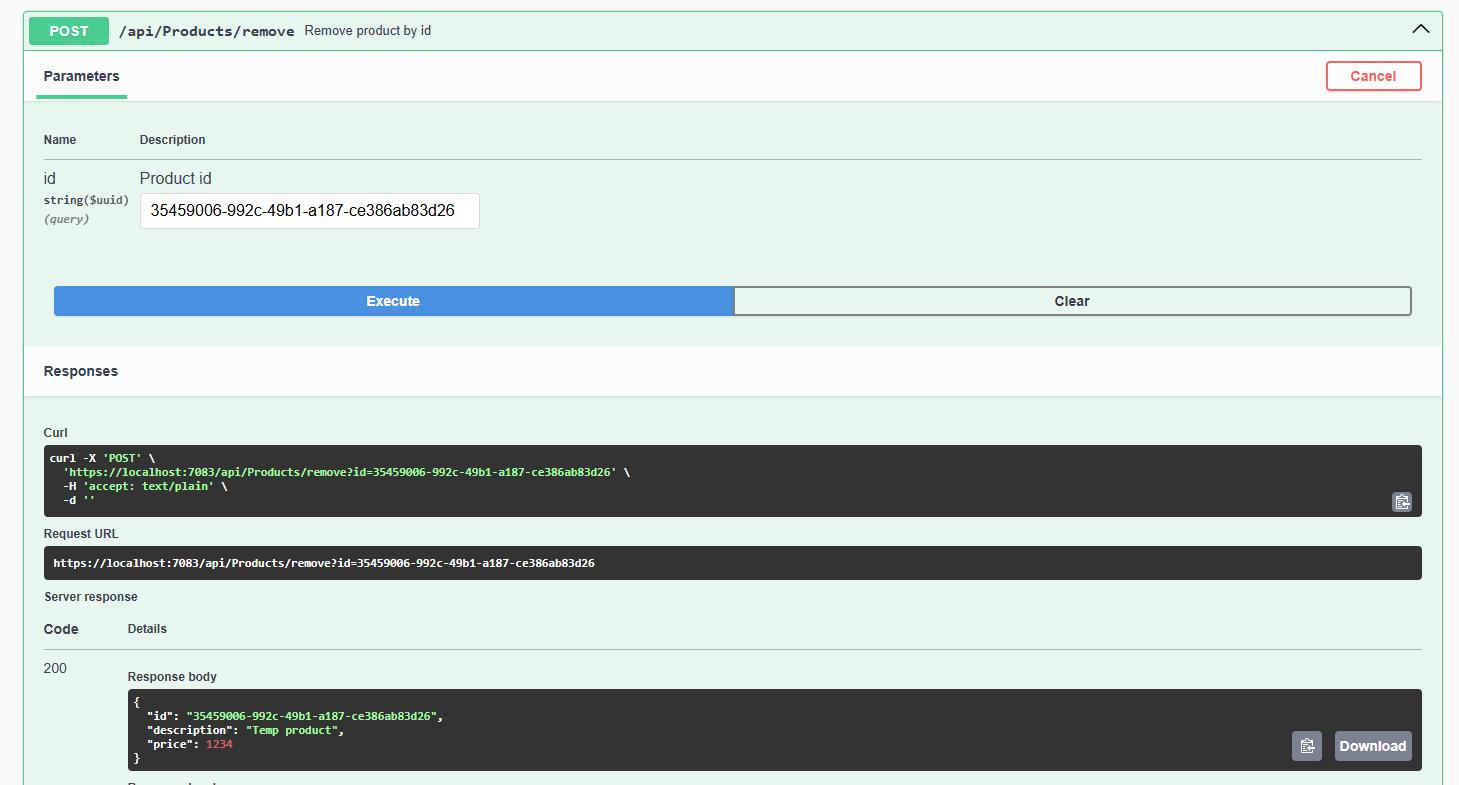
**Ход работы**

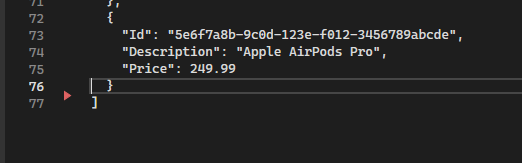
Добавление продукта



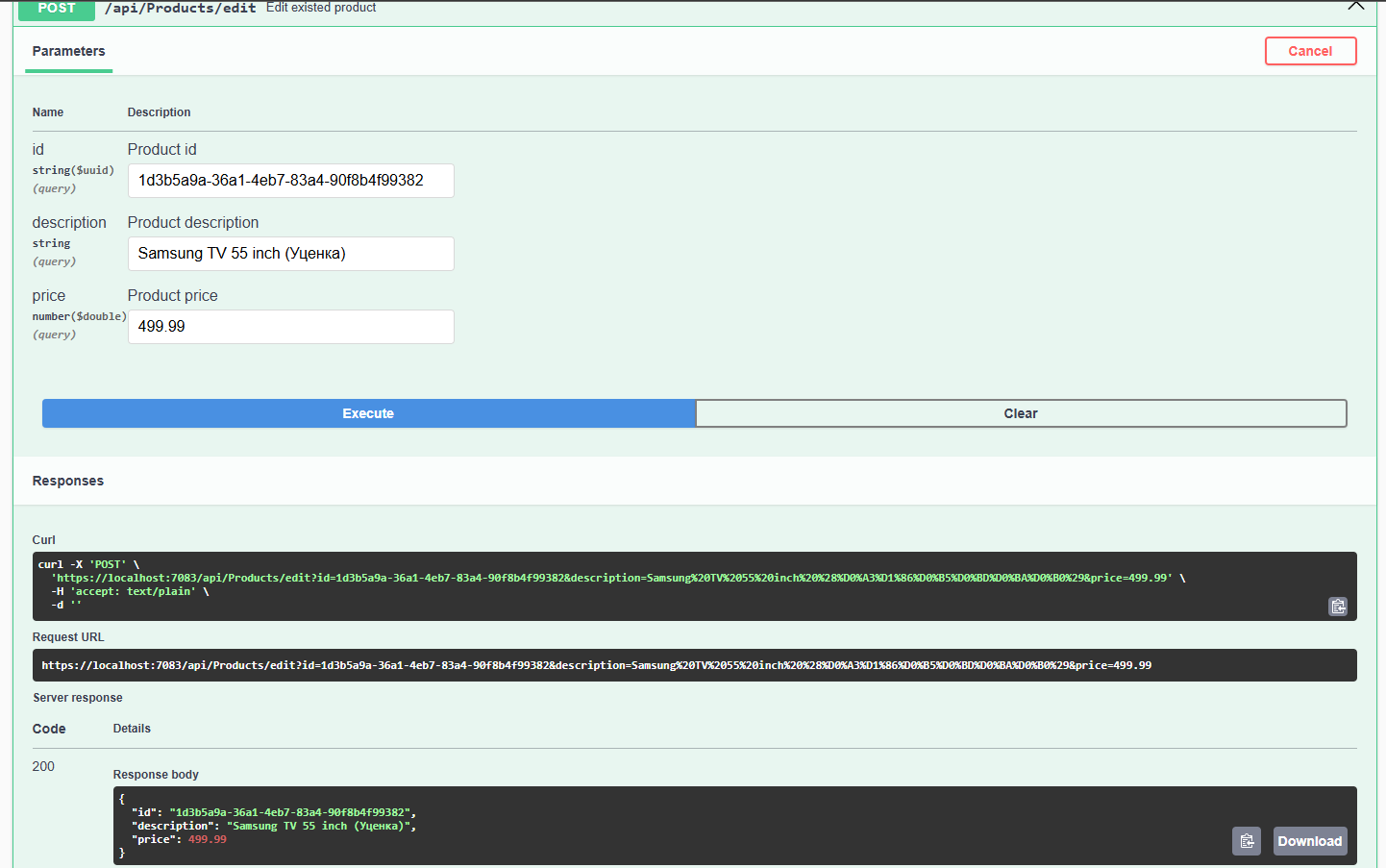


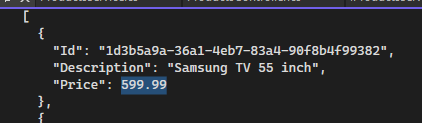
Удаление продукта

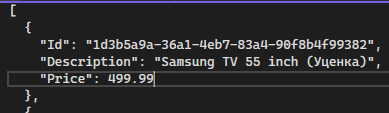




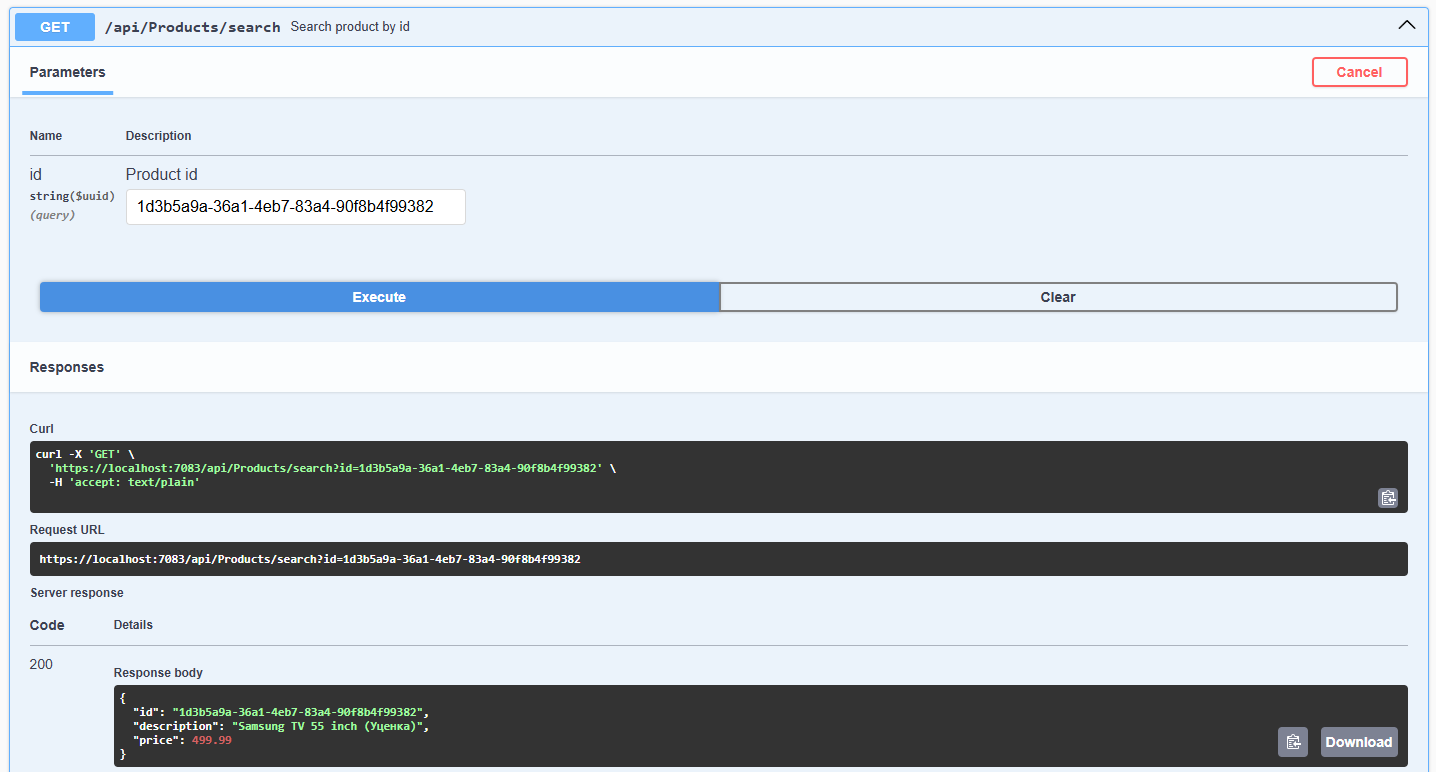
Редактирование продукта



Было: 

Стало: 

Поиск



**Листинг**

*appsettings.json*

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*",

"DataBaseFilePath": "resources/database.txt"

}

*Program.cs*

using ShopApp.Models;

using ShopApp.Services;

WebApplicationBuilder builder = WebApplication.CreateBuilder(args);

builder.Services.AddSingleton<IProductsService<Product>, ProductsService>();

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(options =>

{

string basePath = AppContext.BaseDirectory;

string xmlPath = Path.Combine(basePath, "ShopApp.xml");

options.IncludeXmlComments(xmlPath);

});

WebApplication app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

\_ = app.UseSwagger();

\_ = app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}");

app.Run();

ProductsService.cs

using Newtonsoft.Json;

using ShopApp.Models;

namespace ShopApp.Services

{

/// <summary>

/// Implementation of the product service for managing products.

/// Provides methods to add, edit, remove, and search products.

/// The products are stored in a JSON file whose path is specified in the configuration.

/// </summary>

public class ProductsService : IProductsService<Product>

{

private const string CONFIGURATION\_DATA\_BASE\_FILE\_PATH = "DataBaseFilePath";

private readonly Dictionary<Guid, Product> \_products;

private readonly IConfiguration \_configuration;

private readonly SemaphoreSlim \_semaphore;

private readonly CancellationTokenSource \_cancellationTokenSource;

private readonly string \_dataBasePath;

/// <summary>

/// Base constructor. Initializes the products collection by loading data from the file.

/// Throws an exception if the file path is not specified in the configuration.

/// </summary>

/// <param name="configuration">Application configuration containing the database file path.</param>

/// <exception cref="Exception">Thrown if the required configuration key is missing.</exception>

public ProductsService(IConfiguration configuration)

{

\_cancellationTokenSource = new CancellationTokenSource();

\_semaphore = new SemaphoreSlim(1, 1);

\_configuration = configuration;

\_dataBasePath = \_configuration[CONFIGURATION\_DATA\_BASE\_FILE\_PATH];

if (string.IsNullOrEmpty(\_dataBasePath))

{

throw new Exception($"ERROR: Can't find {CONFIGURATION\_DATA\_BASE\_FILE\_PATH} in configuration file");

}

\_products = [];

Task.Run(() => InitFromFileAsync(\_cancellationTokenSource.Token)).Wait();

}

/// <summary>

/// Finalizer for the service. Cancels any pending operations.

/// </summary>

~ProductsService()

{

\_cancellationTokenSource?.Cancel();

}

/// <summary>

/// Adds a product to the collection.

/// </summary>

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

/// <returns>True if the product was successfully added, otherwise false.</returns>

public bool Add(Product product)

{

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

{

return false;

}

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

// Fire-and-forget async write to file

\_ = WriteToFileAsync(\_cancellationTokenSource.Token);

return true;

}

/// <summary>

/// Edits (updates) an existing product in the collection.

/// </summary>

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

/// <returns>The updated product if successful, otherwise null.</returns>

public Product? Edit(Product product)

{

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

{

return null;

}

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

\_ = WriteToFileAsync(\_cancellationTokenSource.Token);

return \_products[product.Id];

}

/// <summary>

/// Removes a product from the collection by its identifier.

/// </summary>

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

/// <returns>The removed product if successful, otherwise null.</returns>

public Product? Remove(Guid productId)

{

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

{

return null;

}

\_ = \_products.Remove(productId);

\_ = WriteToFileAsync(\_cancellationTokenSource.Token);

return removedProduct;

}

/// <summary>

/// Searches for a product in the collection by its identifier.

/// </summary>

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

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

public Product? Search(Guid productId)

{

return \_products.TryGetValue(productId, out Product? foundProduct) ? foundProduct : null;

}

/// <summary>

/// Asynchronously initializes the products collection by loading data from the JSON file.

/// If the file does not exist, a new file with an empty JSON array is created.

/// </summary>

/// <param name="cancellationToken">A cancellation token for the operation.</param>

/// <returns>A task representing the asynchronous operation.</returns>

private async Task InitFromFileAsync(CancellationToken cancellationToken)

{

if (!File.Exists(\_dataBasePath))

{

await File.WriteAllTextAsync(\_dataBasePath, "[]", cancellationToken);

return;

}

string json = await File.ReadAllTextAsync(\_dataBasePath, cancellationToken);

if (string.IsNullOrEmpty(json))

{

return;

}

List<Product>? productsFromFile = JsonConvert.DeserializeObject<List<Product>>(json);

if (productsFromFile == null)

{

return;

}

\_products.Clear();

foreach (Product product in productsFromFile)

{

\_products[product.Id] = product;

}

}

/// <summary>

/// Asynchronously writes the current products collection to the JSON file.

/// This operation is performed in a thread-safe manner using SemaphoreSlim.

/// </summary>

/// <param name="cancellationToken">A cancellation token for the operation.</param>

/// <returns>A task representing the asynchronous operation.</returns>

private async Task WriteToFileAsync(CancellationToken cancellationToken)

{

await \_semaphore.WaitAsync(cancellationToken);

try

{

// Serialize the collection of products.

string json = JsonConvert.SerializeObject(\_products.Values.ToList(), Formatting.Indented);

await File.WriteAllTextAsync(\_dataBasePath, json, cancellationToken);

}

catch (Exception ex)

{

Console.WriteLine($"ERROR: can't write data to file. {ex.Message}");

}

finally

{

\_ = \_semaphore.Release();

}

}

}

}

*IProductsService.cs*

namespace ShopApp.Services

{

/// <summary>

/// Abstract product service

/// </summary>

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

public interface IProductsService<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="productId">Product id</param>

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

T? Remove(Guid productId);

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

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

T? Search(Guid productId);

}

}

*Product.cs*

namespace ShopApp.Models

{

/// <summary>

/// Represents a product in the store.

/// </summary>

public class Product : ICloneable

{

/// <summary>

/// Gets or sets the unique identifier of the product.

/// </summary>

public Guid Id { get; set; }

/// <summary>

/// Gets or sets the description of the product.

/// </summary>

public string? Description { get; set; }

/// <summary>

/// Gets or sets the price of the product.

/// </summary>

public double Price { get; set; }

/// <summary>

/// Creates a deep copy of the current product instance.

/// </summary>

/// <returns>A new <see cref="Product"/> object with the same values.</returns>

public object Clone()

{

return new Product()

{

Id = Id,

Description = Description,

Price = Price

};

}

}

}

*ProductsController.cs*

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 IProductsService<Product> \_productService;

/// <summary>

/// Base constructor

/// </summary>

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

/// <param name="productsService">Products servise</param>

public ProductsController(ILogger<ProductsController> logger, IProductsService<Product> productsService)

{

\_logger = logger;

\_productService = productsService;

}

/// <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()

{

Id = Guid.NewGuid(),

Description = description,

Price = price

};

return \_productService.Add(createdProduct) ? createdProduct.Id : 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()

{

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>

[HttpGet, Route("search")]

public Product? SearchProduct(Guid id)

{

return \_productService.Search(id);

}

}

}

*HomeController.cs*

using Microsoft.AspNetCore.Mvc;

namespace ShopApp.Controllers

{

/// <summary>

/// Home controller for ShopApp

/// </summary>

public class HomeController : ControllerBase

{

/// <summary>

/// Home page for ShopApp

/// </summary>

/// <returns>Home string</returns>

public string Index()

{

return "Made by D. Belyaev and M. Demin";

}

}

}

*database.txt*

[

{

"Id": "1d3b5a9a-36a1-4eb7-83a4-90f8b4f99382",

"Description": "Samsung TV 55 inch",

"Price": 599.99

},

{

"Id": "2f3c6b1b-43d2-41f9-81ed-12d3d65e2cde",

"Description": "Lenovo ThinkPad Laptop",

"Price": 999.99

},

{

"Id": "3a9e5b8c-9d67-42f2-b63a-2f5e8f98c123",

"Description": "Apple iPhone 13",

"Price": 799.99

},

{

"Id": "4c6f7a12-56e3-45e1-9e27-8f3bfa1e2b5f",

"Description": "Xiaomi Mi Pad 4 Tablet",

"Price": 349.99

},

{

"Id": "5d8f9c34-7a4f-4d2a-a345-6f7d8c9e0bcd",

"Description": "Garmin Fenix 6 Smartwatch",

"Price": 449.99

},

{

"Id": "6e9f0d56-8b2e-4c3a-923f-7a1b8c9d0e1f",

"Description": "Sony WH-1000XM4 Wireless Headphones",

"Price": 349.99

},

{

"Id": "7f0a1b78-9c3d-4e5f-8a2b-3c4d5e6f7a8b",

"Description": "Sony PlayStation 5 Console",

"Price": 499.99

},

{

"Id": "8a1b2c3d-4e5f-6789-abcd-ef0123456789",

"Description": "Microsoft Xbox Series X",

"Price": 499.99

},

{

"Id": "9b2c3d4e-5f6a-789b-cdef-0123456789ab",

"Description": "Nintendo Switch Console",

"Price": 299.99

},

{

"Id": "0c1d2e3f-4a5b-6789-cdef-0123456789ab",

"Description": "Dell UltraSharp Monitor 27 inch",

"Price": 399.99

},

{

"Id": "1a2b3c4d-5e6f-7890-abcd-0123456789ab",

"Description": "HP LaserJet Pro Printer",

"Price": 199.99

},

{

"Id": "2b3c4d5e-6f7a-890b-cdef-0123456789ab",

"Description": "Bose SoundLink Bluetooth Speaker",

"Price": 149.99

},

{

"Id": "3c4d5e6f-7a8b-901c-def0-123456789abc",

"Description": "Amazon Echo Dot (4th Gen)",

"Price": 49.99

},

{

"Id": "4d5e6f7a-8b9c-012d-ef01-23456789abcd",

"Description": "Google Nest Mini",

"Price": 39.99

},

{

"Id": "5e6f7a8b-9c0d-123e-f012-3456789abcde",

"Description": "Apple AirPods Pro",

"Price": 249.99

}

]

**Вывод**

Создали простое Web-приложение