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

**ОТЧËТ**  
по лабораторной работе №2  
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. 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**

1 С помощью внедрения зависимостей связали разработанный класс ProductService с интерфейсом IProductService

builder.Services.AddSingleton<IProductService, ProductService>();

2 В конструктор контроллера ProductController передали интерфейс IProductService и присвоили его внутренней переменной.

private readonly IProductService \_products;

public ProductsController(IProductService products)

{

\_products = products;

}

3 Усовершенствовали класс ProductService. Конструктор класса принимает на вход параметр IConfiguration. Присвоили его внутренней переменной.

private Dictionary<Guid, Product> \_products;

private readonly IConfiguration \_configuration;

private readonly string \_databaseFilePath;

private readonly Mutex \_fileMutex = new Mutex();

/// <summary>

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

/// </summary>

public ProductService(IConfiguration config)

{

\_configuration = config;

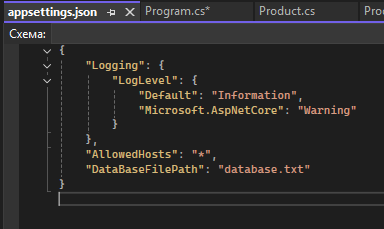
\_databaseFilePath = \_configuration["DataBaseFilePath"] ?? "database.txt";

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

InitFromFile();

}

Внутри файла объявили поле "DataBaseFilePath" : "database.txt".



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

private void InitFromFile()

{

if (!File.Exists(\_databaseFilePath))

{

return; // Если файла нет, просто выходим

}

try

{

\_fileMutex.WaitOne();

string json = File.ReadAllText(\_databaseFilePath);

var products = JsonSerializer.Deserialize<List<Product>>(json);

if (products != null)

{

foreach (var product in products)

{

\_products[product.Id] = product;

}

}

}

catch (Exception ex)

{

Console.WriteLine($"Error reading from file: {ex.Message}");

}

finally

{

\_fileMutex.ReleaseMutex();

}

}

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

private void InitFromFile()

{

if (!File.Exists(\_databaseFilePath))

{

return; // Если файла нет, просто выходим

}

try

{

\_fileMutex.WaitOne();

string json = File.ReadAllText(\_databaseFilePath);

var products = JsonSerializer.Deserialize<List<Product>>(json);

if (products != null)

{

foreach (var product in products)

{

\_products[product.Id] = product;

}

}

}

catch (Exception ex)

{

Console.WriteLine($"Error reading from file: {ex.Message}");

}

finally

{

\_fileMutex.ReleaseMutex();

}

}

6 Явно реализовали методы класса ProductService. Все методы связанные с изменением состояния базы данных записывают изменения как в оперативную память, так и на диск. Использовали мьютекс для записи изменений.

/// <summary>

/// Adds a new product to the collection.

/// </summary>

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

/// <returns>Adds a new product to the collection and saves the update to disk if successful; otherwise, <c>null</c>.</returns>

public Product? Add(Product product)

{

if (\_products.TryAdd(product.Id, product))

{

WriteToFile();

return product;

}

return 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 and updates the database if found; otherwise, <c>null</c>.</returns>

public Product? Remove(Guid id)

{

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

{

\_products.Remove(id);

WriteToFile();

return product;

}

return null;

}

/// <summary>

/// Edits an existing product.

/// </summary>

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

/// <returns>The edited product and updates the database if found; otherwise, <c>null</c>.</returns>

public Product? Edit(Product product)

{

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

{

return null;

}

\_products[product.Id] = product;

WriteToFile();

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

/// <summary>

/// Saves the current product list to the database file.

/// </summary>

private void WriteToFile()

{

try

{

\_fileMutex.WaitOne(); // Блокируем доступ к файлу

string json = JsonSerializer.Serialize(\_products, new JsonSerializerOptions { WriteIndented = true });

File.WriteAllText(\_databaseFilePath, json);

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка при сохранении базы данных: {ex.Message}");

}

finally

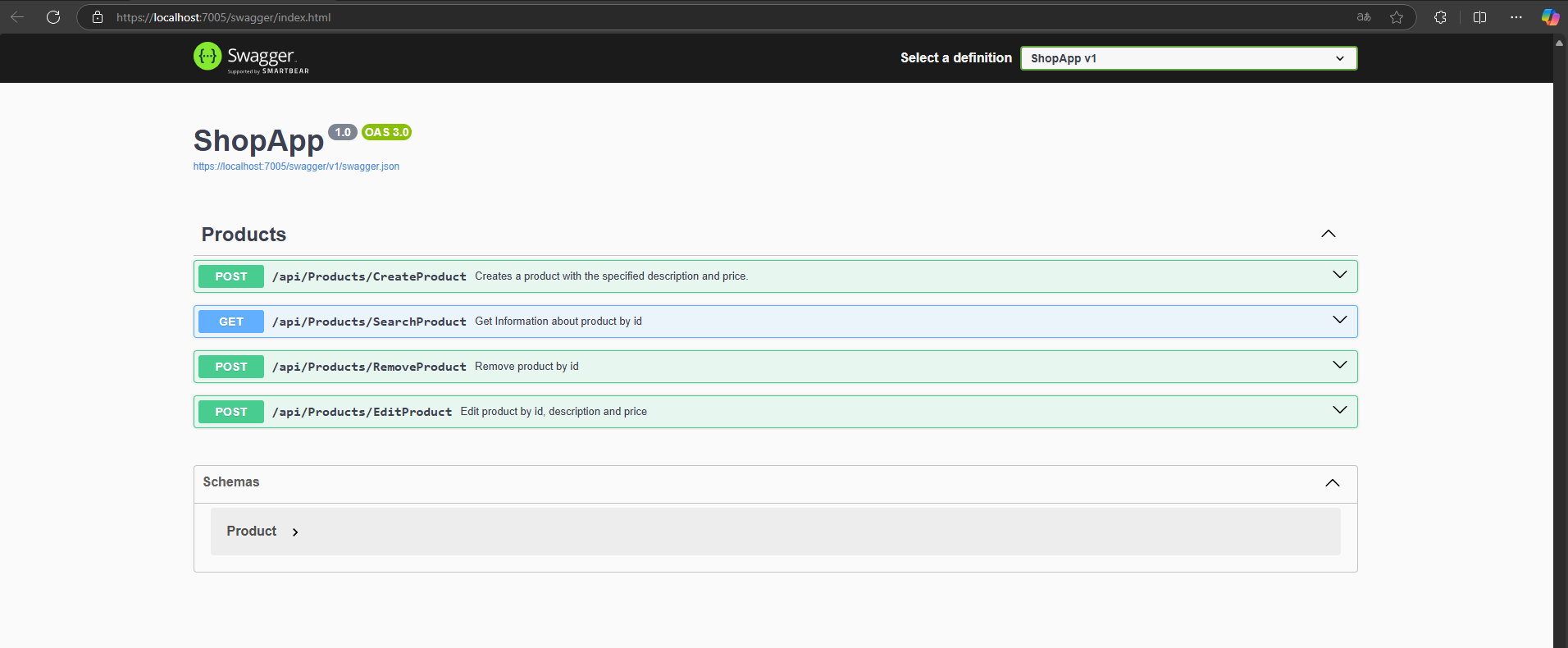
{

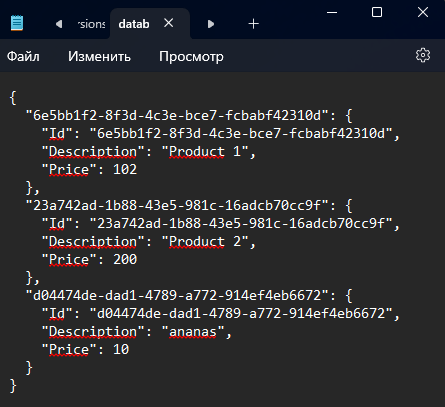
\_fileMutex.ReleaseMutex(); // Разблокируем доступ

}

}

8 Проверка

****

****

**Вывод**

Создали простое Web-приложения