Полуфинал олимпиады «Я – профессионал» по направлению «Программная инженерия»

Задание №2

Введение:

С помощью веб-фреймворка ASP .NET Core 3.1 и языка C# было реализовано RESTful API приложение позволяющее совершать CRUD операции над моделью заметки, а также совершать операции поиска по введенному запросу. К проекту также добавлена документации API с помощью SwaggerUI

Основная часть:

Код приложения размещен в репозитории на Github в папке 2/CodeTask2: https://github.com/Hetfield96/Olympics-CS-Semifinal

Приложение создает базу данных PostgreSQL, в которой хранятся заметки, с помощью CodeFirst подхода и библиотек Npgsql.EntityFrameworkCore.PostgreSQL и Microsoft.EntityFrameworkCore.Tools. Для тестирования работы приложение необходимо выполнить следующие действия:

1) Запустить сервис postgresql
2) Настроить appsettings.json: указать строку соединения с БД
3) Выполнить следующие команды из папки проекта:
    a. dotnet ef migrations add init -o Migrations # Создает первоначальную миграцию
    b. dotnet ef database update # Создает базу данных

Для конфигурации числа N: число первых символов из строки Content, которыми заменяется пустой Header заметки, также следует использовать appsettings.json, а конкретно секцию «ConfigSection:ReplaceTitleN»

API запросы были протестированы с помощью приложения Postman.

При запуске проекта по умолчанию открывается вкладка с документацией по API – SwaggerUI в браузере

Заключение:

Преимущества решения:

1. **Использование БД:**

Быстрее было бы не использовать БД, а просто создать в контролере List заметок и работать с ним – тем не менее это решение не поддерживает персистентность данных и при каждому перезапуске сервера данные исчезают.

Можно было бы добиться персистентности сохраняя данные, например в json файл, сохраняемый на сервере, но это решение отличается низкой масштабируемостью в отличие от БД, куда можно добавить еще сколько угодно полей и как угодно усложнить логику работы приложения

При создании БД использовалась библиотека EntityFramework и CodeFirst approach, это решение позволяет поддерживать механизм миграций и понятную историю изменения структуры БД

2. **Кроссплатформенность:**

Для поддержки кроссплатформенности – возможности запуска приложения на любой ОС был выбран фреймворк ASP .NET Core, а также кроссплатформенная БД PostgreSQL. Это является преимуществом решения, т.к. приложение может быть запущено и база может быть создана на сервере управляемом любой из популярных ОС: linux, mac os, windows

Недостатки решения:

1. **Использование БД:**

Использование БД потребовало больших временных ресурсов при разработке в сравнении с другими вариантами решения.

2. **Поисковый запрос:**

При реализации поискового запроса с параметром query возникла проблема совпадения маршрутов методов GetAll и GetAllByQuery из-за одинакового пути к методам. Долго пытался решить эту проблему, в итоге немного отошел от поставленной задачи и реализовал метод GetAllByQuery по пути query и параметром query из маршрута запроса.