Практическая работа №10
Оформление документации на программные средства с использованием инструментальных средств

1 Цель работы

1.1 Познакомиться с процессом оформления документации на программные средства с использованием инструментальных средств.

2 Литература

2.1 Гагарина, Л. Г. Технология разработки программного обеспечения : учебное пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Сидорова-Виснадул ; под ред. Л.Г. Гагариной. — Москва : ФОРУМ : ИНФРА-М, 2025. — 400 с. — (Среднее профессиональное образование). - ISBN 978-5-8199-0812-9. - Текст : электронный. - URL: https://znanium.ru/catalog/product/2183867 – Режим доступа: по подписке.

3 Подготовка к работе

3.1 Повторить теоретический материал (см. п.2).

3.2 Изучить описание лабораторной работы.

4 Основное оборудование

4.1 Персональный компьютер.

5 Задание

5.1 Создание проекта API

5.1.1 Создать проект Web-API ASP.NET Core (Майкрософт)

5.1.1.1 Убрать галочку с пункта «Настроить для Https», установить галочку у пунктов «Включить поддержку OpenApi» и «Использовать контроллеры»

5.1.2 Добавить в проект контроллер для работы со списком задач: добавление задач, удаление задач, получение всех задач, получение задачи по id. Модель задачи должна содержать: id, название, описание, срок и статус (Enum).

5.1.3 Добавить в проект MinimalApi-Endpoint (app.MapGet в Program.cs) «/health» для проверки состояния Api.

5.1.4 Добавить в проект MinimalApi-Endpoint для поиска задач, срок которых находится в пределах промежутка переданных в параметрах дат.

В ответах использовать различные варианты http-ответов в запросах (Ok, Created, Not Found, Bad Request)

5.2 Документирование XML

5.2.1 Добавить XML-комментарии для всех методов в контроллерах и моделей данных, а также всех публичных элементов.

5.2.2 Включить сохранение файла XML-документации в параметрах проекта (Свойства – Сборка – Выходные данные)

5.3 Документирование Api

5.3.1 Изменить код добавления сервиса Swagger, добавив туда настройки:

```
builder.Services.AddSwaggerGen(options =>
{
    options.SwaggerDoc("v1", new OpenApiInfo
    {
        Title = "Наименование",
        Version = "v1",
        Description = "Описание",
        Contact = new OpenApiContact
        {
            Name = "Разработчик",
            Email = "dev@example.com",
            Url = new Uri("https://example.com/contact")
        },
        License = new OpenApiLicense
        {
            Name = "Пример лицензии"
            Url = new Uri("https://example.com/license")
        }
    });

};
```

5.3.2 Изменить описание Api в соответствии с предметной областью, укажите информацию о контакте и лицензии.

5.3.3 Добавить в настройки Swagger использование сгенерированного XML-файла документации:

```
var xmlPath = Path.Combine(AppContext.BaseDirectory,
"Название проекта.xml");
options.IncludeXmlComments(xmlPath);
```

5.3.4 Запустить проект, проверить, что в Swagger отображается XML-документация

5.4 Расширенные XML-комментарии

5.4.1 Добавить в комментарии к методам запросов дополнительную информацию при помощи тегов <remarks>, например:

```
/// <remarks>
/// Пример запроса:
///
///     POST /Todo
///     {
///        "id" : 1,
///        "name" : "Сделать дело",
///        ...
///     }
/// </remarks>
```

5.4.2 Добавить в комментарии к методам запросов информацию о

возможных вариантах ответов при помощи тегов <response>, например:

```
/// <response code="200">Успешное выполнение</response>
/// <response code="400">Ошибка API</response>
```

5.5 Документирование MinimalApi

5.5.1 Добавить документацию к методам MinimalApi при помощи методов, например:

```
.WithName("CreateToDo")
.WithTags("ToDos")
.Accepts<CreateToDoRequest>("application/json")
.Produces<ToDo>(StatusCodes.Status200OK)
.Produces(StatusCodes.Status404NotFound)
.WithOpenApi(operation =>
{
    operation.Summary = "Название";
    operation.Description = "Описание";
    operation.Parameters.Add(
        new Microsoft.OpenApi.Models.OpenApiParameter()
        {
            Name = "Имя параметра",
            Description = "Описание"
        }
    );
    return operation;
});
```

5.5.2 Запустить Api, проверить отображение документации.

5.6 Документирование в Bruno Api

5.6.1 Импортировать Swagger-коллекцию в Bruno

5.6.2 Добавить описания методам Api, сохранить примеры запросов и ответов. При документировании использовать разметку MarkDown.

5.6.3 Экспортировать коллекцию Bruno в виде файла JSON

5.7 Генерация документации к коду

5.7.1 Установить инструмент DocFx: dotnet tool update -g docfx

5.7.2 Инициализировать DocFx в папке с проектом: docfx init

5.7.3 Отредактировать файл docfx.json, указав в нем корректное расположение файлов проекта.

5.7.4 Скомпилировать документацию выполнив команду: docfx docfx.json

5.7.5 Запустить сайт документации: docfx serve _site

5.7.6 Откройте в браузере http://localhost:8080, проверьте работу документации

5.8 Настройка страниц документации

5.8.1 Заполнить описание проекта в файле docs/introduction.md

5.8.2 Заполнить инструкцию по старту работы с Api (установка и т.д.) в файле docs/getting-started.md

5.8.3 Добавить файл с описанием и примерами основных возможностей Api, добавить его в структуру документации в файле toc.yml

6 Порядок выполнения работы

6.1 Повторить теоретический материал п. 3.1;
6.2 Выполнить задание п. 5.1;
6.3 Ответить на контрольные вопросы п. 8;
6.4 Заполнить отчет п. 7.

7 Содержание отчета

7.1 Титульный лист;
7.2 Цель работы;
7.3 Ответы на контрольные вопросы п. 6.3;
7.4 Вывод по проделанной работе.

8 Контрольные вопросы

8.1 Какие инструментальные средства документирования существуют?
8.2 Какие преимущества предоставляет использование инструментальных средств документирования?