Практическая работа №1
Разработка web-API для доступа к данным

1 Цель работы

1.1 Научиться выполнять разработку web-API для доступа к БД.

2 Литература

2.1 ASP.NET Core MVC | Полное руководство. metanit.com – Текст : электронный // metanit.com, 2023. – URL: https://metanit.com/sharp/aspnet6/ – гл.11-12.

3 Подготовка к работе

3.1 Повторить теоретический материал (см.п.2).
3.2 Изучить описание практической работы.

4 Основное оборудование

4.1 Персональный компьютер.

5 Задание

5.1 Создание проекта Web-API

5.1.1 Создать приложение «Веб-API ASP.NET Core (Майкрософт)». В окне «Дополнительные сведения» снять все флажки кроме «Включить поддержку OpenAPI» и «Использовать контроллеры». Версия .Net: 8.0.

5.1.2 Добавить в проект контекст данных и модель данных для фильма и жанров фильма.

5.1.3 Добавить в приложение строку:
builder.Services.AddDbContext<ИмяКонтекста>();

5.1.4 Сгенерировать контроллер для фильма на основе модели данных:
- GET /api/movies — получить все фильмы.
- GET /api/movies/{id} — получить фильм по идентификатору.
- POST /api/movies — добавить новый фильм.
- PUT /api/movies/{id} — обновить данные фильма.
- DELETE /api/movies/{id} — удалить фильм.

Изменить конечные точки так, чтобы отображалась версия API (v1).

5.1.4 Проверить работу созданных методов, используя Postman или Swagger.

5.2 Валидация входных данных

5.2.1 Требуется вернуть:

- 404 Not Found, если фильм с указанным Id не найден (в сообщение ошибки добавить текст с указанием идентификатора и того, что этот фильм не найден),

- 400 Bad Request с сообщением, если данные не прошли валидацию (добавить текст сообщения в параметры BadRequest). Добавить в методы Post и Put валидацию данных:

- название не может быть пустой строкой или состоять из пробельных символов,
- год выхода не может быть раньше 1900 и позже следующего года,
- длительность должна быть больше нуля.

5.2.2 Проверить работу созданных методов с учетом возникающих ошибок.

5.3 Фильтрация, сортировка и пагинация

5.3.1 Добавить новый метод для возврата списка фильмов, доступный по /api/v1/movies/filter.

5.3.2 Добавить в метод пагинацию (по умолчанию возвращать 3 фильма с первой страницы, но пользователь может передать свои параметры)

5.3.3 Добавить в метод сортировку по названию, убыванию даты проката, году выпуска по убыванию и году выпуска по возрастанию.

5.3.4 Добавить в метод фильтрацию:

- по фильмам, которые идут в прокате (дата проката не позже текущей, дата окончания – не указана или позже текущей),
- по фильмам, в названии которых есть указанный текст.

5.3.5 Проверить работу созданного метода.

5.4 Работа со связанными данными

5.4.1 Добавить новый метод для возврата списка фильмов и жанров фильма, доступный по /api/v1/movies/genres.

5.4.2 Добавить в метод фильтрацию по фильмам определенного жанра, указанного пользователем.

5.4.3 Проверить работу созданного метода.

5.5 Документирование API

5.5.1 Добавить в приложение файл документации:

- Свойства проекта – Сборка – Выходные данные – Файл документации (поставить флажок),
- добавить опции в Swagger:

вместо

```
builder.Services.AddSwaggerGen();
```

написать:

```
builder.Services.AddSwaggerGen(options =>
{
    var xmlFile = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";
    var xmlPath = Path.Combine(AppContext.BaseDirectory, xmlFile);
    options.IncludeXmlComments(xmlPath);
});
```

5.5.2 Добавить в методы контроллера документацию по образцу:

```
/// <summary>
/// Создать новую категорию.
/// </summary>
/// <param name="category">Объект категории.</param>
/// <returns>Созданная категория.</returns>
/// <response code="201">категория успешно создана.</response>
/// <response code="400">Входные данные неверны.</response>
[HttpPost]
[ProducesResponseType(StatusCodes.Status201Created)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
```

- `[ProducesResponseType(StatusCodes.КодОтвета)]` требуется для указания типа возвращаемого объекта и HTTP-кода ответа в документации,
- `<response code="код">комментарий</response>` требуется для указания HTTP-кода ответа и текста комментария к ответу в документации.

6 Порядок выполнения работы

6.1 Выполнить все задания из п.5.
6.2 Ответить на контрольные вопросы.

7 Содержание отчета

7.1 Титульный лист
7.2 Цель работы
7.3 Ответы на контрольные вопросы
7.4 Вывод

8 Контрольные вопросы

8.1 Что такое REST-запрос?
8.2 Что такое RESTful?
8.3 Для чего используется метод GET?
8.4 Для чего используется метод POST?
8.5 Для чего используется метод PUT?
8.6 Для чего используется метод DELETE?