Лабораторная работа №14
Вызов методов web-API для доступа к данным

1 **Цель работы**

1.1 Научиться вызывать методы web-API для доступа к БД.

2Литература

2.1 Взаимодействие HttpClient с Web API | Полное руководство. metanit.com – Текст : электронный // metanit.com, 2025. – URL: https://metanit.com/sharp/net/2.7.php

3Подготовка к работе

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

3.2 Изучить описание практической работы.

4Основное оборудование

4.1 Персональный компьютер.

5Задание

5.1 Создание структуры проектов

5.1.1 Создать проект «Веб-API ASP.NET Core (Майкрософт)». В окне «Дополнительные сведения» выбрать платформу .NET 9 и поставить флажки «Включить поддержку OpenAPI» и «Использовать контроллеры». Добавить в проект поддержку Scalar для тестирования web-API.

5.1.2 Настроить запуск браузера и открытие страницы со Scalar при старте приложения.

5.1.3 Создать проект DatabaseLibrary типа «Библиотека классов (Майкрософт)», выбрать платформу .NET 9. Добавить в проект пакеты для работы с БД, создать контекст данных и модели данных для таблиц предметной области «Кинотеатр».

5.1.4 Подключить в сервисы приложения веб-API контекст БД и сгенерировать контроллеры для классов Фильм, Жанр, Посетитель, Билет.

5.1.5 Создать консольный проект для тестирования работы сервисов для работы с API.

5.1.6 Настроить параметры запуска проектов: запуск проекта веб-API и консольного проекта (в конце в консольного проекта указать Console.ReadLine();).

5.2 Создание http-клиента с автоматической сериализацией/десериализацией

5.2.1 Создать проект ApiServicesLibrary типа «Библиотека классов (Майкрософт)», выбрать платформу .NET 9.

5.2.2 Добавить в проект ApiServicesLibrary класс для вызова методов на чтение и запись одного из контроллеров, используя автоматическую сериализацию/десериализацию:

var response = await _client.МетодJsonAsync(…);

В методы, возвращающие коды ошибок, добавить генерацию исключений:

response.EnsureSuccessStatusCode();

При вызове метода POST возвращать созданный объект.

5.2.3 Проверить работу методов созданного класса в консольном приложении.

5.3 Создание http-клиента с явной сериализацией/десериализацией

Для явной сериалзации/десериализации использовать опции:

PropertyNamingPolicy = JsonNamingPolicy.CamelCase

5.3.1 Добавить в проект ApiServicesLibrary класс для вызова методов на чтение одного из контроллеров, используя явную десериализацию:

var response = await _client.GetAsync(…);

var content = await response.Content.ReadAsStringAsync();

return JsonSerializer.Deserialize<тип>(content, _jsonOptions);

В методы, возвращающие коды ошибок, добавить генерацию исключений:

response.EnsureSuccessStatusCode();

5.3.2 Добавить в проект ApiServicesLibrary класс для вызова методов на запись одного из контроллеров, используя явную сериализацию:

var json = JsonSerializer.Serialize(объект, _jsonOptions);

var content = new StringContent(json, Encoding.UTF8, "application/json");

var response = await _client.МетодAsync(…, content);

В методы, возвращающие коды ошибок, добавить генерацию исключений:

response.EnsureSuccessStatusCode();

При вызове метода POST возвращать созданный объект.

5.3.3 Проверить работу методов созданного класса в консольном приложении.

5.4 Обработка разных статус-кодов при чтении данных

Варианты обработки кодов ответа:

- if (response.IsSuccessStatusCode) // для обработки успешного ответа
- if (!response.IsSuccessStatusCode) // для обработки неуспешного ответа
- if (response.StatusCode == HttpStatusCode.Код1) // для проверки ответа с кодом
- response.StatusCode switch // перебор ответов с разными кодами

```
{
    HttpStatusCode.Код1 => результат1,
    HttpStatusCode.Код2 => результат1,
    HttpStatusCode.Код3 => результат2,
    HttpStatusCode.Код4 => throw ТипException(…),
    _ when response.IsSuccessStatusCode => результат1,
    _ => throw Exception(…)
};
```

5.4.1 Добавить в проект ApiServicesLibrary класс для вызова методов на чтение одного из контроллеров. В методе получения объекта реализовать:

- возврат null, если код ответа 404,
- генерацию исключения, если код ответа неуспешный,
- возврат объекта в остальных случаях.

При необходимости внести изменения в методы веб-API.

5.4.2 В методе получения списка объектов одного из классов сервиса:

- возврат списка объектов, если код ответа 200,
- возврат пустого списка, если код ответа 204,
- генерацию исключения с сообщением «Ресурс не найден», если код ответа 404,
- генерацию исключения, если код ответа неуспешный.

При необходимости внести изменения в методы веб-API.

5.4.3 Проверить работу методов созданного класса в консольном приложении.

5.5 Обработка разных статус-кодов при записи данных
5.5.1 Добавить в метод вставки объекта класса из п.5.4:
- тип возврата ТипОбъекта (возвращает созданный ресурс)
- возврат объекта, если код ответа 201 или другой успешный,
- генерацию исключения, если код ответа 400,
- генерацию исключения, если код ответа 409,
- генерацию исключения, если код ответа неуспешный.
При необходимости внести изменения в методы веб-API.
5.5.2 Добавить в метод изменения объекта класса из п.5.4:
- тип возврата bool (удалось/не удалось обновить ресурс)
- возврат true, если код ответа 200, 204, другой успешный,
- возврат false, если код ответа 404,
- генерацию исключения, если код ответа 400,
- генерацию исключения, если код ответа неуспешный.
При необходимости внести изменения в методы веб-API.
5.5.3 Проверить работу методов созданного класса в консольном приложении.

6Порядок выполнения работы

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

7Содержание отчета

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

8Контрольные вопросы

8.1 Что такое json и для чего он используется?
8.2 Что такое HttpClient и для чего он используется?
8.3 Как вызывать методы веб-API без явной сериализации/десериализации?
8.4 Как вызывать методы веб-API с явной сериализацией/десериализацией?
8.5 Как добавить генерацию исключений в случае неуспешных кодов ответа?
8.6 Как реализовать обработку разных кодов ответа?