> Напишите ответы на задания в любом удобном вам формате и пришлите нам. Мы будем оценивать правильность и точность ответов, а не оформление.
> Вы можете использовать любые источники информации.

Задание 1

Есть некий документ, который описывает вызов GET-метода API для разработчика клиентского приложения. Метод позволяет получать события календаря, которые создал пользователь, например записи о днях рождения или встречах. Формат ответа — JSON.

Документ

> Возвращает список записей календаря. Фильтрует записи по дате старта.

`GET` https://domain.com/Calendar

Запрос

Тип	Имя	Обязательность	Значение по умолчанию	Описание
int	tfDays			
int	afterDays			

Пример

```
curl -location -request GET 'https://domain.com/Calendar'
```

Разработчик клиентского приложения хочет получить:

- все события, дата начала которых с "завтра*" - до "сегодня* + 7 дней";
- все события за вчера;
- все события за сегодня.

*Сегодня = текущее время сервера. Время клиента и время сервера совпадает.

У разработчика нет доступа к определению метода, он есть только у вас и у вашей команды.

Составьте GET-запросы (если такие возможны) и дополните описание запроса в документе. Опишите Query-параметры tfDays и afterDays. Укажите обязательность параметров и значения по умолчанию.

Вот метод контроллера, который принимает параметры из запроса и возвращает выборку:

Метод (представлен в C#)

```
public class Calendar
{
    public Response<CalendarEvent[]> GetCalendar(int tfDays = 0, int afterDays = 0)
    {
        try
        {
            var after = DateTime.Now.AddDays(afterDays == 0 ? 7 : afterDays);
            var start = DateTime.Now.AddDays(tfDays == 0 ? 1 : tfDays);

            /* Фильтр событий по дате старта */
            IEnumerable<Event> eventList = CacheWrap.GetAllEvents(e => e.Start > start && e.Start < after);

            var events = eventList.ToArray();
            return new Response<CalendarEvent[]>(events);
        }
        catch (Exception ee)
        {
            Log.Error(ee);
            return new Response<CalendarEvent>("Нет событий за указанную дату!");
        }
    }
}
```

Где:

public class CashWrap

```
public IEnumerable<Event> GetAllEvents(Func<Event, bool> filter)
{
 return GetAll().Where(filter);  // GetAll() – получаем события Event из кэша
}
```

Где:

Where()— https://docs.microsoft.com/en-us/dotnet/api/system.linq.enumerable.where?view=net-6.0

Func<T,TResult> — https://docs.microsoft.com/en-us/dotnet/api/system.func-2?view=net-6.0

Event

```
[DataContract(Name = "Event", Namespace = "http://www.domain.com")]
public class Event
{
    [DataMember(Name = "Start", EmitDefaultValue = false)]
    public DateTime Start { get; set; }

    [DataMember(Name = "Id", EmitDefaultValue = false)]
    public int ID { get; set; }

    [DataMember(Name = "Name", EmitDefaultValue = false)]
    public string Name { get; set; }

    [DataMember(Name = "Note", EmitDefaultValue = false)]
    public string Note { get; set; }

    [DataMember(Name = "Type", EmitDefaultValue = false)]
    public int Type { get; set; }

}
```

Задание 2

Составьте примеры HTTP-запросов к API, используя описание API ниже. Запросы должны быть информативными:

- укажите в них все перечисленные параметры, если это возможно.
- составить несколько (минимум 2) примеров вызова GET-метода, чтобы показать его в действии. Хотя бы один из них должен вернуть как можно больше событий разных типов.

Вы можете использовать любой синтаксис (cURL, HTTP, JavaScript: Fetch, Shell: Wget и другие).

API не требует авторизации и дополнительных заголовков запросов.

GET http://domain.com/Events

Тип	Имя	Обязательность	Описание
int[]	ids	-	Идентификаторы событий. Если не указаны, вернет все события согласно фильтрам.

bool	isArchive	-	Включать ли события из архива. По умолчанию: false.
int[]	type	-	Типы событий: 0 — Все; 1 — Личное; 2 — Семья; 3 — Работа; 4 — Нет типа.

POST http://domain.com/Event

Тип	Имя	Обязательность	Описание
int	Start	-	Дата "начала" события в формате Unix Timestamp (в секундах). Если не указана, будет установлено текущее время.
string	Name	✓	Название события
string	Note	-	Описание события
int	Type	-	Типы событий: 1 — Личное; 2 — Семья; 3 — Работа; 4 — Нет типа.

Задание 3

Используя следующее определение базы данных, составьте SQL-запрос, который вернет количество всех пользователей с фамилией "Иванов" или "Иванова"

TABLE users

```
id INTEGER PRIMARY KEY,
firstName VARCHAR(30) NOT NULL,
lastName VARCHAR(30) NOT NULL
```