МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра ВТ

ОТЧЕТ

по лабораторной работе №4
по дисциплине «Программирование в среде .NET»
Тема: РАЗРАБОТКА ПРОГРАММНОГО ИНТЕРФЕЙСА ДЛЯ ВЗАИМОДЕЙСТВИЯ С ПРИЛОЖЕНИЕМ

Студент гр. 6306 Милакин А.Д.

Преподаватель Пешехонов К. А.

Санкт-Петербург
2020

Цель работы

Исследовать вопрос реализации программного интерфейса для взаимодействия с приложением средствами языка C#.

Задание

Реализовать WebAPI слой приложения.

Код программы

Код программы целиком выложен на гитхаб. Приведем пример WebAPI контроллера.

Код контроллера

```
namespace Media.WebAPI.Controllers
{
  [ApiController]
  [Route("api/podcast")]
  public class PodcastController
  {
    private ILogger<PodcastController> Logger { get; }
    private IPodcastCreateService PodcastCreateService { get; }
    private IPodcastGetService PodcastGetService { get; }
    private IPodcastUpdateService PodcastUpdateService { get; }
    private IMapper Mapper { get; }

    public PodcastController(ILogger<PodcastController> logger, IMapper mapper, IPodcastCreateService podcastCreateService, IPodcastGetService podcastGetService, IPodcastUpdateService podcastUpdateService)
    {
      this.Logger = logger;
      this.PodcastCreateService = podcastCreateService;
      this.PodcastGetService = podcastGetService;
      this.PodcastUpdateService = podcastUpdateService;
      this.Mapper = mapper;
    }

    [HttpPut]
    [Route("")]
    public async Task<PodcastDTO> PutAsync(PodcastCreateDTO podcast)
    {
      this.Logger.LogTrace($"{nameof(this.PutAsync)} called");

      var result = await this.PodcastCreateService.CreateAsync(this.Mapper.Map<PodcastUpdateModel>(podcast));

      return this.Mapper.Map<PodcastDTO>(result);
    }
```

```
    [HttpPatch]
    [Route("")]
    public async Task<PodcastDTO> PatchAsync(PodcastUpdateDTO podcast)
    {
        this.Logger.LogTrace($"{nameof(this.PutAsync)} called");

        var result = await this.PodcastUpdateService.UpdateAsync(this.Mapper.Map<PodcastUpdateModel>(podcast));

        return this.Mapper.Map<PodcastDTO>(result);
    }

    [HttpGet]
    [Route("")]
    public async Task<IEnumerable<PodcastDTO>> GetAsync()
    {
        this.Logger.LogTrace($"{nameof(this.GetAsync)} called");

        return this.Mapper.Map<IEnumerable<PodcastDTO>>(await this.PodcastGetService.GetAsync());
    }

    [HttpGet]
    [Route("{podcastId}")]
    public async Task<PodcastDTO> GetAsync(int podcastId)
    {
        this.Logger.LogTrace($"{nameof(this.GetAsync)} called for {podcastId}");

        return this.Mapper.Map<PodcastDTO>(await this.PodcastGetService.GetAsync(new PodcastIdentityModel(podcastId)));
    }
  }
}
```

Пример тестирования.

Тестирование происходит с помощью программы Postman.

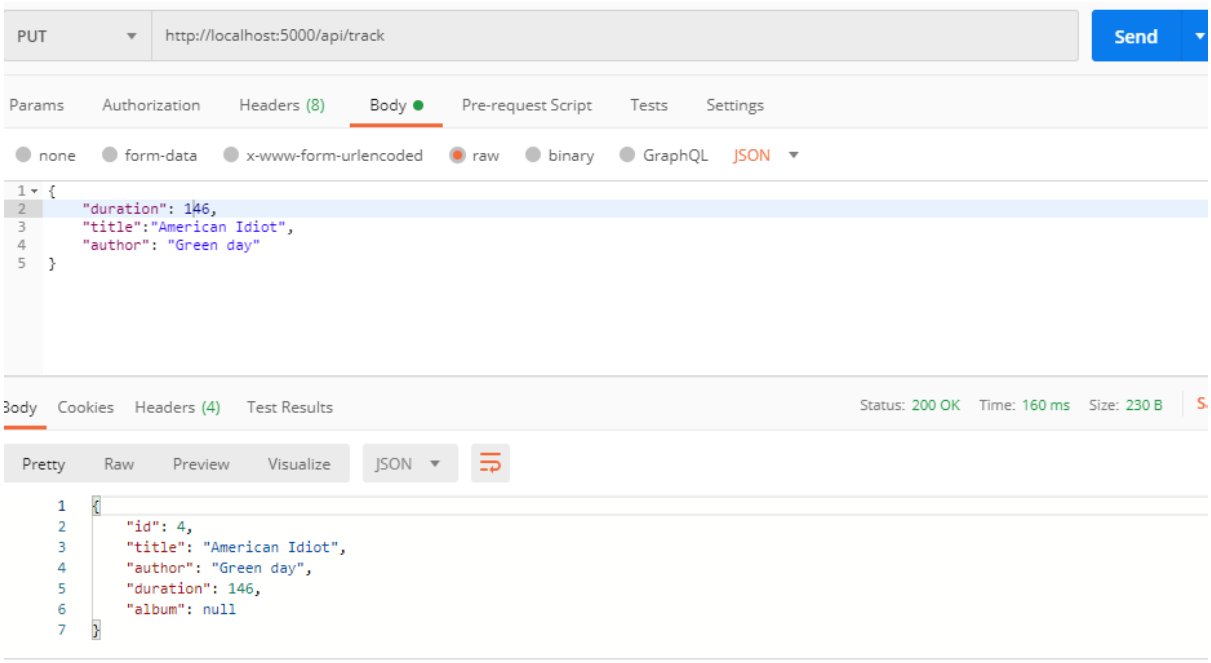

Рисунок 1. Пример PUT запроса

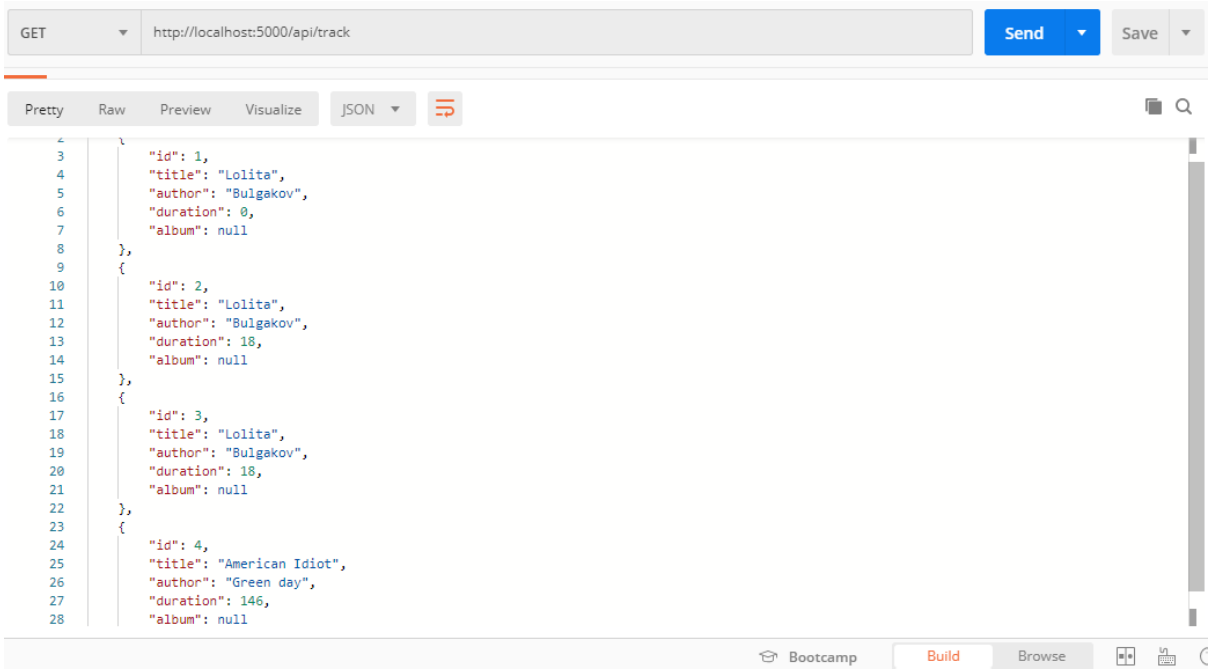

Рисунок 2. Пример GET запроса

Вывод

В процессе выполнения лабораторной работы мы исследовали вопрос реализации программного интерфейса для взаимодействия с приложением средствами языка C#.