

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

ОТЧЁТ
по лабораторной работе №2
по дисциплине «Программирование в среде dotNET»
Тема: “Разработка слоя бизнес-логики приложения”

Студент гр. 6305

Силинский А.С.

Преподаватель

Пешехонов К.А.

Санкт-Петербург

2020

Оглавление

Цель работы	3
Задание	3
Ход работы.....	3
Выводы.....	4
Приложение А.....	5
Приложение Б	9

Цель работы

Целью данной лабораторной работы является разработка слоя бизнес-логики приложения.

Задание

1. Сформулировать тему проекта приложения ASP.NET Core 3 WebAPI.
2. Реализовать слой бизнес-логики.
3. Покрыть слой бизнес-логики модульными тестами.

Ход работы

В качестве темы была выбрана база данных фестивалей. В этой базе находится сведения о различных фестивалях (их название, описание и дата проведения). Также у каждого фестиваля есть свои участники и основные сведения о них (имя, фамилия, возраст).

В данном приложении должен быть реализован следующий функционал:

- Добавление, изменение, а также удаление фестивалей из БД;
- Добавление, изменение, а также удаление участников из БД;
- Просмотр информации о фестивалях/фестивале;
- Просмотр информации об участниках/участнике;
- Просмотр всех участников какой-либо фестиваля.

Основные интерфейсы сервисов, репозиториях лежат в их же проектах. Для хранения сущностей создается отдельный модуль Domain. В нём будут храниться сущности (так как используют все слои)

Реализация бизнес-логики была вынесена в отдельный модуль “BLL”.

Структура модуля выглядит следующим образом:

- /Exceptions (классы исключений)
- /Contracts (интерфейсы сервисов)
- FestivalService.cs
- ParticipantService.cs

В ходе проведения данной лабораторной работы были написаны юнит-тесты для классов FestivalService и ParticipantService, все тесты прошли и в дальнейшем облегчат процесс разработки приложения.

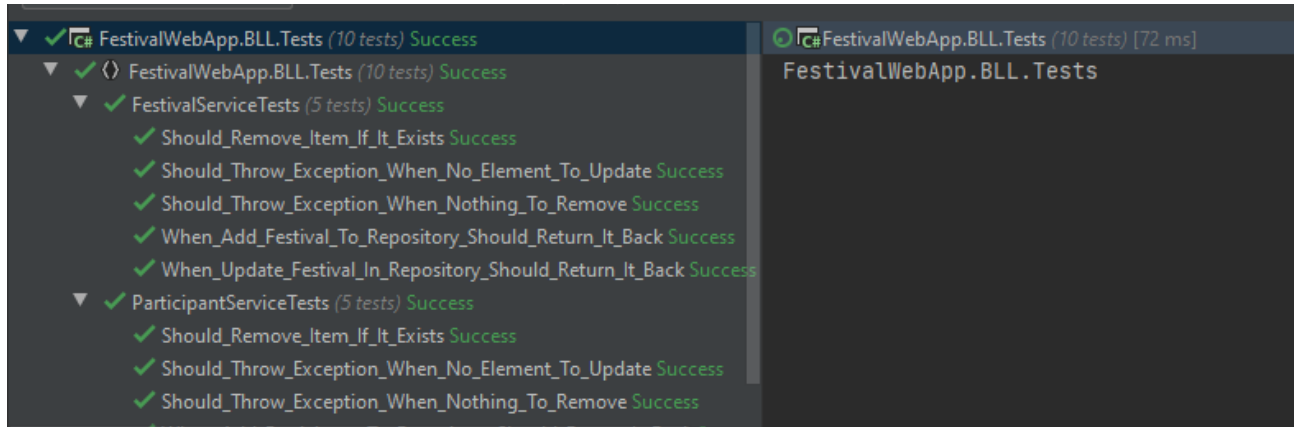


Рис. 1. Пройденные юнит-тесты

Код исходных файлов данного модуля можно посмотреть в приложении к данной лабораторной работе.

Выводы

В ходе выполнения данной лабораторной работы был реализован слой бизнес-логики, который был разбит на два модуля: “Domain” и “BLL”. Были написаны юнит-тесты для слоя “BLL”.

Приложение А

Festival.cs

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;

namespace FestivalWebApp.Core.Models
{
    public class Festival
    {
        public int Id { get; set; }

        public ICollection<Participant> Participants { get; set; } = new
        Collection<Participant>();
        public string Name { get; set; }
        public string Description { get; set; }
        public DateTime Date { get; set; }

        public override string ToString()
        {
            return "Festival: " + Id + " "
                + Name + " "
                + Description + " "
                + Date;
        }
    }
}
```

Participant.cs

```
namespace FestivalWebApp.Core.Models
{
    public class Participant
    {
        public int Id { get; set; }
        public int Age { get; set; }
        public int FestivalId { get; set; }
        public string Name { get; set; }
        public string SecondName { get; set; }

        public Festival Festival { get; set; }
    }
}
```

```

    public override string ToString()
    {
        return "Participant: " + Id + " "
            + Age + " "
            + FestivalId + " "
            + Name + " "
            + SecondName + " "
            + Festival;
    }
}

```

IFestivalRepository.cs

```

using System.Collections.Generic;
using System.Threading.Tasks;
using FestivalWebApp.Core.Models;

namespace FestivalWebApp.Core.Repositories
{
    public interface IFestivalRepository
    {
        Task<Festival> GetFestivalById(int id);
        Task<IEnumerable<Festival>> GetAllFestivals();
        Task<Festival> AddFestival(Festival festival);
        Task UpdateFestival(Festival festival);
        Task RemoveFestival(Festival festival);
        Task<bool> IsExist(int id);
    }
}

```

IParticipantRepository.cs

```

using System.Collections.Generic;
using System.Threading.Tasks;
using FestivalWebApp.Core.Models;

namespace FestivalWebApp.Core.Repositories
{
    public interface IParticipantRepository
    {
        Task<Participant> GetParticipantById(int id);
        Task<IEnumerable<Participant>> GetAllParticipants();
    }
}

```

```

        Task<IEnumerable<Participant>> GetParticipantsByFestivalId(int festivalId);
        Task<Participant> AddParticipant(Participant participant);
        Task UpdateParticipant(Participant participant);
        Task RemoveParticipant(Participant participant);
        Task<bool> IsExist(int id);
    }
}

```

IFestivalService.cs

```

using System.Collections.Generic;
using System.Threading.Tasks;
using FestivalWebApp.Core.Models;

namespace FestivalWebApp.Core.Services
{
    public interface IFestivalService
    {
        Task<Festival> GetFestivalById(int id);

        Task<IEnumerable<Festival>> GetAllFestivals();

        Task<Festival> AddFestival(Festival festival);

        Task UpdateFestival(Festival festival);

        Task RemoveFestival(int id);
    }
}

```

IParticipantService.cs

```

using System.Collections.Generic;
using System.Threading.Tasks;
using FestivalWebApp.Core.Models;

namespace FestivalWebApp.Core.Services
{
    public interface IParticipantService
    {
        Task<Participant> GetParticipantById(int id);
    }
}

```

```
Task<IEnumerable<Participant>> GetAllParticipants();

Task<IEnumerable<Participant>> GetParticipantsByFestivalId(int festivalId);

Task<Participant> AddParticipant(Participant participant);

Task UpdateParticipant(Participant participant);

Task RemoveParticipant(int id);
    }
}
```


Приложение Б

FestivalService.cs

```
using System.Collections.Generic;
using System.Threading.Tasks;
using FestivalWebApp.BLL.Exceptions;
using FestivalWebApp.Core.Models;
using FestivalWebApp.Core.Repositories;
using FestivalWebApp.Core.Services;

namespace FestivalWebApp.BLL
{
    public class FestivalService : IFestivalService
    {
        private readonly IFestivalRepository _repository;

        public FestivalService(IFestivalRepository repository)
        {
            _repository = repository;
        }

        public async Task<Festival> GetFestivalById(int id)
        {
            return await _repository.GetFestivalById(id);
        }

        public async Task<IEnumerable<Festival>> GetAllFestivals()
        {
            return await _repository.GetAllFestivals();
        }

        public async Task<Festival> AddFestival(Festival festival)
        {
            return await _repository.AddFestival(festival);
        }

        public async Task UpdateFestival(Festival festival)
        {
            var isExist = await _repository.IsExist(festival.Id);
            if (!isExist) throw new ElementNotFoundException(festival);
        }
    }
}
```

```

        await _repository.UpdateFestival(festival);
    }

    public async Task RemoveFestival(int id)
    {
        var isExist = await _repository.IsExist(id);
        if (!isExist) throw new ElementNotFoundException(id);

        var festival = await GetFestivalById(id);
        await _repository.RemoveFestival(festival);
    }
}

```

ParticipantService.cs

```

using System.Collections.Generic;
using System.Threading.Tasks;
using FestivalWebApp.BLL.Exceptions;
using FestivalWebApp.Core.Models;
using FestivalWebApp.Core.Repositories;
using FestivalWebApp.Core.Services;

namespace FestivalWebApp.BLL
{
    public class ParticipantService : IParticipantService
    {
        private readonly IParticipantRepository _repository;

        public ParticipantService(IParticipantRepository repository)
        {
            _repository = repository;
        }

        public async Task<Participant> GetParticipantById(int id)
        {
            return await _repository.GetParticipantById(id);
        }

        public async Task<IEnumerable<Participant>> GetAllParticipants()
        {

```

```

        return await _repository.GetAllParticipants();
    }

    public async Task<IEnumerable<Participant>> GetParticipantsByFestivalId(int
festivalId)
    {
        return await _repository.GetParticipantsByFestivalId(festivalId);
    }

    public async Task<Participant> AddParticipant(Participant participant)
    {
        return await _repository.AddParticipant(participant);
    }

    public async Task UpdateParticipant(Participant participant)
    {
        var isExist = await _repository.IsExist(participant.Id);
        if (!isExist) throw new ElementNotFoundException(participant);

        await _repository.UpdateParticipant(participant);
    }

    public async Task RemoveParticipant(int id)
    {
        var isExist = await _repository.IsExist(id);
        if (!isExist) throw new ElementNotFoundException(id);

        var participant = await GetParticipantById(id);
        await _repository.RemoveParticipant(participant);
    }
}

```