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 .NET»

Тема: “Разработка слоя бизнес-логики приложения”

Студент гр. 6305 _____ Буракаев Д. А.

Преподаватель _____ Пешехонов К. А.

Санкт-Петербург

2020

Содержание

Цель работы

Целью данной лабораторной работы является формулирование темы проекта, реализация слоя бизнес-логики.

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

Тема приложения

В качестве модели проекта для реализации приложения была выбрана «*Билетная касса*».

Приложение должно реализовывать функционал редактирования списков текущих премьер, заказов от пользователя.

Транспортные объекты

В ходе выполнения лабораторной работы была написана программа на языке *C#*, вычисления и обработка данных в которой происходит несколькими функциональными слоями. В качестве объектов, транспортируемых между этими слоями (**DTO**) были реализованы следующие классы:

1. *QuestDTO.cs*:

```
namespace BLL.DTO
{
    public class QuestDTO
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public int PlayersLimit { get; set; }
        public double Price { get; set; }
        public int Duration { get; set; }
    }
}
```

Класс содержит описание актуального киносеанса.

2. ReservDTO.cs

```
namespace BLL.DTO
{
    public class ReservDTO
    {
        public int Id { get; set; }
        public QuestDTO Quest { get; set; }
        public string DateTimeSections { get; set; }
        public double Cost { get; set; }
    }
}
```

Класс содержит описание заказа билета на пермьеру.

Экземпляры объектов размечаются от доменных объектов *Quest* и *Reserv* в модуле *AutoMapperConfig.cs*:

```
namespace BLL.DTO
{
    public class ReservDTO
    {
        public int Id { get; set; }
        public QuestDTO Quest { get; set; }
        public string DateTimeSections { get; set; }
        public double Cost { get; set; }
    }
}
```

Помимо этого, в директории *Infrostructure* присутствует вспомогательный модуль *ConnectionModule.cs*, позволяющий более ясно осуществлять связку интерфейсов (инъекцию зависимостей) с **EF**-слоем:

```
using DAL.Interfaces;
using DAL.Repositories;
using Ninject.Modules;
```

```
namespace BLL.Infrastructure
{
    public class ConnectionModule : NinjectModule
    {
        private string connectionString;

        public ConnectionModule(string connection)
        {
            connectionString = connection;
        }

        public override void Load()
        {

Bind<IUnitOfWork>().To<EFUnitOfWork>().WithConstructorArgument(connectionString);
        }
    }
}
```

Бизнес-логика

Слой **BLL** ответственен за основную логику разрабатываемого приложения, а также за работу со слоем доступа к данным (Data Access Layer, **DAL**) и слоем представления данных.

Реализуем два интерфейса сервисов, ответственных за обработку ранее обозначенных ключевых объектов:

QuestSevice.cs

```
using AutoMapper;
using BLL.DTO;
using BLL.Interfaces;
using DAL.Entities;
using DAL.Interfaces;
using System;
using System.Collections.Generic;
```

```
using System.Linq;

namespace BLL.Services
{
    public class QuestService : IQuestService
    {
        IUnitOfWork Database { get; set; }

        public QuestService(IUnitOfWork uow)
        {
            Database = uow;
        }

        public QuestDTO Find(int id)
        {
            return Mapper.Map<Quest, QuestDTO>(Database.Quests.Get(id));
        }

        public IEnumerable<QuestDTO> GetAll()
        {
            return                                          Mapper.Map<IEnumerable<Quest>,
List<QuestDTO>>(Database.Quests.GetAll());
        }

        public void Dispose()
        {
            Database.Dispose();
        }

        public IEnumerable<QuestDTO> Find(Func<QuestDTO, bool> predicate)
        {
            return GetAll().Where(predicate);
        }

        public  IEnumerable<QuestDTO>  FilterResults(int  players,  int  duration,  int
price)
        {
            IEnumerable<QuestDTO> quests = GetAll();
                quests = quests.Where(x => x.PlayersLimit <= players);
                quests = quests.Where(x => x.Price <= price);
            quests = quests.Where(x => x.Duration <= duration);
```

```
                return quests;
            }
        }
    }
```

ReservationService.cs

```
using AutoMapper;
using BLL.DTO;
using BLL.Interfaces;
using DAL.EF;
using DAL.Entities;
using DAL.Interfaces;
using DAL.Repositories;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BLL.Services
{
    public class ReservationService : IReservationService
    {
        IUnitOfWork Database { get; set; }

        public ReservationService(IUnitOfWork uow)
        {
            Database = uow;
        }

        public IEnumerable<string> GetTimeSections(QuestDTO quest, DateTime date)
        {
            List<string> list = new List<string>();
            List<string> exTime = new List<string>();

            int startTime = 8;
            int endTime = 20;

            for (int i = startTime; i < endTime; i += quest.Duration)
```

```
            {
                string time = DateTime.Parse(i + ":00").ToShortTimeString();
                string newDateTimeSection = date.ToShortDateString() + " " + time;
                var find = Database.Reserves.Find(x => x.Quest.Id == quest.Id).
                    Intersect(Database.Reserves.Find(x      =>      x.DateTimeSection      ==
newDateTimeSection));

                if (find != null)
                    foreach (var el in find)
                        exTime.Add(el.DateTimeSection.Split(' ')[1]);

                list.Add(time);
            }

            return list.Except(exTime);
        }

        public IEnumerable<ReservDTO> GetAll()
        {
            return                                          Mapper.Map<IEnumerable<Reserv>,
List<ReservDTO>>(Database.Reserves.GetAll());
        }

        public void Add(ReservDTO entity)
        {
            if (entity == null || entity.Quest == null)
                throw new ArgumentNullException();

            Database.Reserves.Create(new Reserv
            {
                Quest = Database.Quests.Get(entity.Quest.Id),
                DateTimeSection = entity.DateTimeSections,
                Cost = entity.Cost
            });

            Database.Save();
        }

        public void Dispose()
        {
            Database.Dispose();
        }
```

```
        public bool VerifyCertificateCode(string code)
        {
            List<string> codes = new List<string>
            {
                "QWERTY",
                "DFHDSE",
                "Q6N8DF",
                "FF7KK0",
                "ANNNNA"
            };

            return codes.Contains(code.ToUpper()) ? true : false;
        }
    }
}
```

Модуль-тесты

Для создания модульных тестов (unit tests) используется фрэймворк *NUnit* - наиболее популярный фрэймворк для unit-тестирования приложений на платформе .NET, позволяющий быстро написать и автоматически проверить модульные тесты. Помимо этого, для создания мок-объектов при тестировании используется фреймворк Moq.

QuestServiceTest.cs

```
        using AutoMapper;
        using BLL.Interfaces;
        using BLL.Services;
        using DAL.Entities;
        using DAL.Interfaces;
        using Moq;
        using NUnit.Framework;
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;

namespace QuestRoomBLL.Tests
{
    [TestFixture]
    public class QuestServiceTest
    {
        private IQuestService questService;
        private Mock<IUnitOfWork> uow;
        private Mock<IRepository<Quest>> questRepository;

        static QuestServiceTest()
        {
            try
            {
                BLL.Infrastructure.AutoMapperConfig.Initialize();
            }
            catch { }
        }

        [SetUp]
        public void Load()
        {
            uow = new Mock<IUnitOfWork>();
            questRepository = new Mock<IRepository<Quest>>();

            uow.Setup(x => x.Quests).Returns(questRepository.Object);

            questService = new QuestService(uow.Object);
        }

        [Test]
        public                                                                  void
GetAll_TryToGetSomeList_ShouldRepositoryCallOnce_ShouldReturnNotNullList()
        {
            //arrange
            questRepository.Setup(x => x.GetAll()).Returns(new List<Quest>() { });
```

```
                //act & assert
                Assert.IsNotNull(questService.GetAll());
                questRepository.Verify(x => x.GetAll(), Times.Once);
            }

            [Test]
            public void Dispose()
            {
                Assert.DoesNotThrow(questService.Dispose);
            }

            [Test]
            public void Find_TryToGetNullValue_ShouldThrowException()
            {
                //arrange
                questRepository.Setup(x => x.Get(It.IsAny<int>())).Returns<Quest>(null);

                // act & assert
                Assert.IsNull(questService.Find(It.IsAny<int>()));
            }

            [Test]
            public void Find_TryToGetValue_ShouldReturnSomeValue()
            {
                //arrange
                var lot = new Quest { Name = It.IsAny<string>(), Price =
It.IsAny<double>(), Description = It.IsAny<string>() };
                uow.Setup(x => x.Quests.Get(It.IsAny<int>())).Returns(lot);

                // act & assert
                Assert.IsNotNull(questService.Find(It.IsAny<int>()));
            }

            [Test]
            public void Find()
            {
                // act & assert
                Assert.IsNotNull(questService.FilterResults(It.IsAny<int>(),
It.IsAny<int>(), It.IsAny<int>()));
            }
```

```
            [Test]
            public void Find_TryToGetValueByPredicate_ShouldReturnSomeValue()
            {
                questRepository.Setup(s => s.Find(It.IsAny<Func<Quest, bool>>()))
                .Returns((Func<Quest, bool> expr) => new List<Quest>());

                questService.Find(x => x.Name == "name");

                uow.VerifyAll();
            }
        }
    }
```

ReservationServiceTest.cs

```
    using BLL.DTO;
    using BLL.Interfaces;
    using BLL.Services;
    using DAL.Entities;
    using DAL.Interfaces;
    using Moq;
    using NUnit.Framework;
    using System;
    using System.Collections.Generic;

    namespace QuestRoomBLL.Tests
    {
        [TestFixture]
        public class ReservationServiceTest
        {
            private IReservationService reservationService;
            private Mock<IUnitOfWork> uow;
            private Mock<IRepository<Reserv>> reservRepository;

            static ReservationServiceTest()
            {
                try
                {
                    BLL.Infrastructure.AutoMapperConfig.Initialize();
                }
                catch { }
```

```
        }

        [SetUp]
        public void Load()
        {
            uow = new Mock<IUnitOfWork>();
            reservRepository = new Mock<IRepository<Reserv>>();

            uow.Setup(x => x.Reserves).Returns(reservRepository.Object);

            reservationService = new ReservationService(uow.Object);
        }

        [Test]
        public                                                                  void
GetAll_TryToGetSomeList_ShouldRepositoryCallOnce_ShouldReturnNotNullList()
        {
            //arrange
            reservRepository.Setup(x => x.GetAll()).Returns(new List<Reserv>() { });

            //act & assert
            Assert.IsNotNull(reservationService.GetAll());
            reservRepository.Verify(x => x.GetAll(), Times.Once);
        }

        [Test]
        public void Add_TryToCreateNullValue_ShouldThrowException()
        {
            // act & assert
            Assert.Throws<ArgumentNullException>(() => reservationService.Add(null));
        }

        [Test]
        public void Add_TryToCreateElementWithNullQuest()
        {
            //arrange
            var reserv = new ReservDTO { Quest = null };
            uow.Setup(x => x.Quests.Get(It.IsAny<int>())).Returns<Quest>(null);

            //assert
            Assert.Throws<ArgumentNullException>(()                             =>
reservationService.Add(reserv));
```

```
            }

            [Test]
            public void Add_TryToAdd_ShouldRepositoryCreateOnce()
            {
                //arrange
                var reserv = new ReservDTO { Cost = It.IsAny<double>(),  Quest = new
QuestDTO { Name = It.IsAny<string>() } };
                uow.Setup(x => x.Quests.Get(It.IsAny<int>())).Returns(new Quest { Name =
It.IsAny<string>() });

                // act
                reservationService.Add(reserv);

                //assert
                reservRepository.Verify(x => x.Create(It.IsAny<Reserv>()), Times.Once);
            }

            [Test]
            public void Dispose()
            {
                Assert.DoesNotThrow(reservationService.Dispose);
            }
        }
    }
```

Убедимся в том, что тестирование происходит корректно:

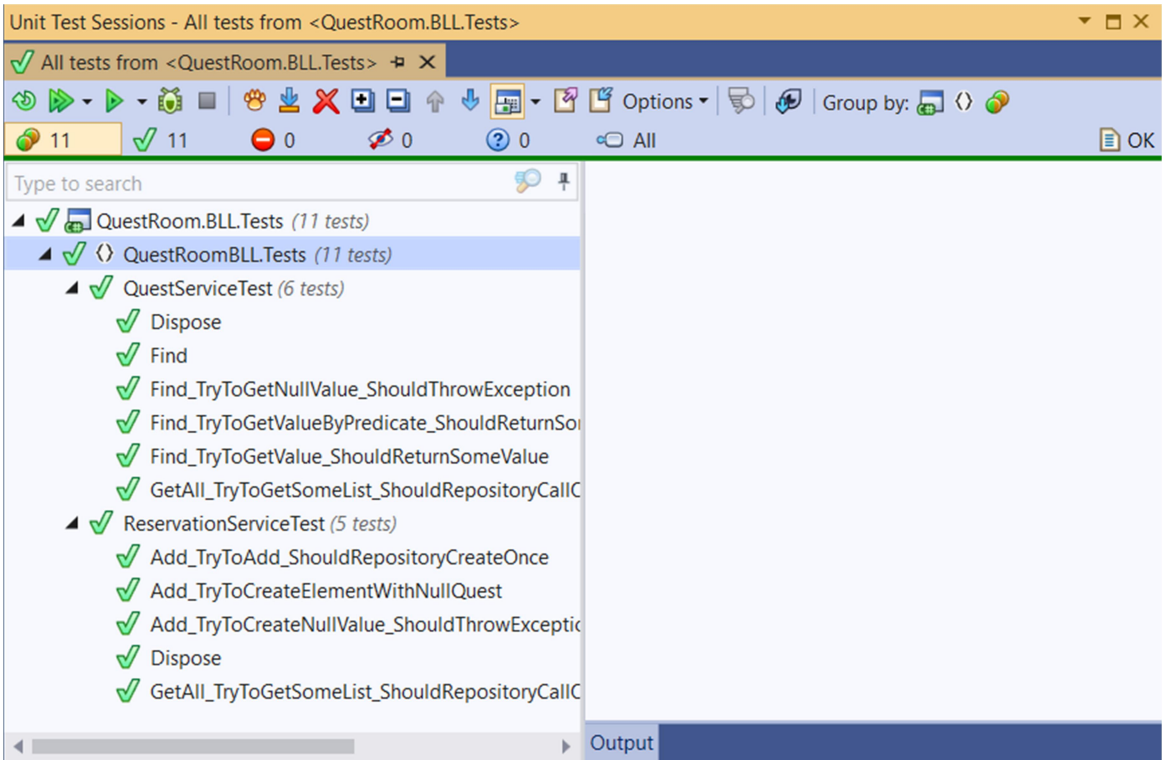

Рис 1. Результаты тестирования модулей

Вывод

В ходе выполнения лабораторной работы была сформулирована тема проекта, реализован слой бизнес-логики приложения на языке *C#*.