**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра ВТ**

**ОТЧЕТ**

**по лабораторной работе №2**

**по дисциплине «Программирование в среде .NET»**

**Тема: РАЗРАБОТКА СЛОЯ БИЗНЕС-ЛОГИКИ ПРИЛОЖЕНИЯ**

Студент гр. 6306 Милакин А.Д.

Преподаватель Пешехонов К. А.

Санкт-Петербург

2020

**Цель работы**

Исследовать вопрос разработки слоя бизнес-логики приложения средствами языка C#.

**Задание**

Разработать слой бизнес-логики приложения, написать тесты.

**Код программы**

Код программы целиком выложен на гитхаб. Приведем пример интерфейса, его реализации и теста.

**Пример интерфейса**

**namespace** Media.BLL.Contracts

{

**public interface** IPodcastCreateService

{

Task<Podcast> CreateAsync(PodcastUpdateModel podcast);

}

}

**Пример имплементации интерфейса**

**namespace** Media.BLL.Implementation

{

**public class** PodcastCreateService : IPodcastCreateService

{

**private** IPodcastDataAccess **PodcastDataAccess** { **get**; }

**private** IAlbumGetService **AlbumGetService** { **get**; }

**public** PodcastCreateService(IPodcastDataAccess podcastDataAccess, IAlbumGetService albumGetService)

{

**PodcastDataAccess** = podcastDataAccess;

**AlbumGetService** = albumGetService;

}

**public async** Task<Podcast> CreateAsync(PodcastUpdateModel podcast)

{

**await AlbumGetService**.ValidateAsync(podcast);

**return await PodcastDataAccess**.InsertAsync(podcast);

}

}

}

**Пример теста**

**namespace** Media.BLL.Tests

{

**public class** PodcastCreatServiceTests

{

**public class** ScreeningCreateServiceTests

{

[Test]

**public async** Task CreateAsync\_PodcastValidationSucceed\_CreatesScreening()

{

*// Arrange*

**var** podcast = **new** PodcastUpdateModel();

**var** expected = **new** Podcast();

**var** albumGetService = **new** Mock<IAlbumGetService>();

albumGetService.Setup(x => x.ValidateAsync(podcast));

**var** podcastDataAccess = **new** Mock<IPodcastDataAccess>();

podcastDataAccess.Setup(x => x.InsertAsync(podcast)).ReturnsAsync(expected);

**var** podcastGetService = **new** PodcastCreateService(podcastDataAccess.**Object**, albumGetService.**Object**);

*// Act*

**var** result = **await** podcastGetService.CreateAsync(podcast);

*// Assert*

result.Should().Be(expected);

}

[Test]

**public async** Task CreateAsync\_PodcastValidationFailed\_ThrowsError()

{

*// Arrange*

**var** fixture = **new** Fixture();

**var** podcast = **new** PodcastUpdateModel();

**var** expected = fixture.Create<**string**>();

**var** albumGetService = **new** Mock<IAlbumGetService>();

albumGetService

.Setup(x => x.ValidateAsync(podcast))

.Throws(**new** InvalidOperationException(expected));

**var** podcastDataAccess = **new** Mock<IPodcastDataAccess>();

**var** podcastGetService = **new** PodcastCreateService(podcastDataAccess.**Object**, albumGetService.**Object**);

*// Act*

**var** action = **new** Func<Task>(() => podcastGetService.CreateAsync(podcast));

*// Assert*

**await** action.Should().ThrowAsync<InvalidOperationException>().WithMessage(expected);

podcastDataAccess.Verify(x => x.InsertAsync(podcast), Times.Never);

}

}

}

}

**Выводы по работе**

В процессе выполнения лабораторной работы мы исследовали вопрос разработки слоя бизнес-логики приложения средствами языка C#.