**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра Вычислительной Техники**

**ОТЧЕТ**

**по лабораторной работе №2**

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

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

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6307 |  | Пякшина О.С. |
| Преподаватель |  | Пешехонов К.А. |

Санкт-Петербург

2020

**Цель работы**

Реализовать слой бизнес-логики приложения.

**Задание**

1) Написать слой бизнес-логики приложения.

2) Покрыть слой бизнес-логики приложения тестами.

Для разработки было выбрано приложение по ведению базы магазина книг.

**Код программы**

Примеры кода программы даны для сущности «книга», другие сущности («покупатель» и «заказ») реализованы похожим образом.

1) Пример сервиса (BookGetService)

- интерфейс IBookGetService

public interface IBookGetService

{

Task<IEnumerable<Book>> GetAsync();

Task<Book> GetAsync(IBookIdentity book);

Task ValidateAsync(IBookContainer storeContainer);

}

- класс BookGetService

public class BookGetService : IBookGetService

{

private IBookDataAccess BookDataAccess { get; }

public BookGetService(IBookDataAccess bookDataAccess)

{

this.BookDataAccess = bookDataAccess;

}

public Task<IEnumerable<Book>> GetAsync()

{

return this.BookDataAccess.GetAsync();

}

public Task<Book> GetAsync(IBookIdentity book)

{

return this.BookDataAccess.GetAsync(book);

}

public async Task ValidateAsync(IBookContainer bookContainer)

{

if (bookContainer == null)

{

throw new ArgumentNullException(nameof(bookContainer));

}

var book = await this.GetBy(bookContainer);

if (bookContainer.BookId.HasValue && book == null)

{

throw new InvalidOperationException($"Book not found by id {bookContainer.BookId}");

}

}

private Task<Book> GetBy(IBookContainer bookContainer)

{

return this.BookDataAccess.GetByAsync(bookContainer);

}

}

2) Пример теста (BookGetServiceTests)

public class BookGetServiceTests

{

[Test]

public async Task ValidateAsync\_BookExists\_DoesNothing()

{

// Arrange

var bookContainer = new Mock<IBookContainer>();

var book = new Book();

var bookDataAccess = new Mock<IBookDataAccess>();

bookDataAccess.Setup(x => x.GetByAsync(bookContainer.Object)).ReturnsAsync(book);

var bookGetService = new BookGetService(bookDataAccess.Object);

// Act

var action = new Func<Task>(() => bookGetService.ValidateAsync(bookContainer.Object));

// Assert

await action.Should().NotThrowAsync<Exception>();

}

[Test]

public async Task ValidateAsync\_BookNotExists\_ThrowsError()

{

// Arrange

var fixture = new Fixture();

var id = fixture.Create<int>();

var bookContainer = new Mock<IBookContainer>();

bookContainer.Setup(x => x.BookId).Returns(id);

var book = new Book();

var bookDataAccess = new Mock<IBookDataAccess>();

bookDataAccess.Setup(x => x.GetByAsync(bookContainer.Object)).ReturnsAsync((Book)null);

var bookGetService = new BookGetService(bookDataAccess.Object);

// Act

var action = new Func<Task>(() => bookGetService.ValidateAsync(bookContainer.Object));

// Assert

await action.Should().ThrowAsync<InvalidOperationException>($"Book not found by id {id}");

}

}

**Выводы**

В процессе выполнения лабораторной работы был реализован слой бизнес-логики приложения в среде .NET, а также были реализованы тесты для этого слоя. Были получены навыки по написанию бизнес-логики приложения и его тестированию.