**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра ВТ**

**ОТЧЕТ**

**по лабораторной работе №2**

**по дисциплине «Основы разработки корпоративных систем на платформе .NET»**

**Тема: Разработка слоя бизнес-логики приложения**

Студент гр. 6306 Гордиенко М. Е.

Преподаватель Пешехонов К. А.

Санкт-Петербург

2020

**Цель работы**

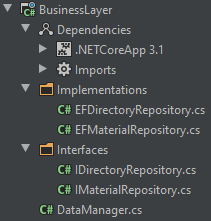
Разработка слоя бизнес-логики Web-приложения.

**Задачи**

1. Реализация бизнес-логики приложения
2. Тестирование методов бизнес-логики приложения

**Описание работы и примеры кода**

**Структура слоя бизнес-логики**

****

**Файл DataManager.cs**

using BusinessLayer.Interfaces;  
  
namespace BusinessLayer **{** public class DataManager {  
 public IDirectoryRepository DirectoryRepository { get; }  
 public IMaterialRepository MaterialRepository { get; }  
  
 public DataManager(IDirectoryRepository directoryRepository, IMaterialRepository materialRepository) {  
 DirectoryRepository = directoryRepository;  
 MaterialRepository = materialRepository;  
 }  
 }  
**}**

**Файл IDirectoryRepository.cs**

using System.Collections.Generic;  
using DataLayer.Entityes;  
  
namespace BusinessLayer.Interfaces **{** public interface IDirectoryRepository {  
 IEnumerable<Directory> GetAllDirectories();  
 Directory GetDirectoryById(int directoryId);  
 int SaveDirectory(Directory directory);  
 void DeleteDirectory(Directory directory);  
 }  
**}**

**Файл IMaterialRepository.cs**

using System.Collections.Generic;  
using DataLayer.Entityes;  
  
namespace BusinessLayer.Interfaces **{** public interface IMaterialRepository {  
 IEnumerable<Material> GetAllMaterials();  
 Material GetMaterialById(int materialId);  
 int SaveMaterial(Material material);  
 void DeleteMaterial(Material material);  
 }  
**}**

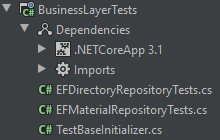
**Файл EFDirectoryRepository**

using System.Collections.Generic;  
using System.Diagnostics.CodeAnalysis;  
using System.Linq;  
using BusinessLayer.Interfaces;  
using DataLayer;  
using DataLayer.Entityes;  
using Microsoft.EntityFrameworkCore;  
  
namespace BusinessLayer.Implementations **{** [SuppressMessage("ReSharper", "InconsistentNaming")]  
 public class EFDirectoryRepository : IDirectoryRepository {  
 private readonly EFDBContext \_context;  
  
 public EFDirectoryRepository(EFDBContext context) {  
 this.\_context = context;  
 }  
   
 public IEnumerable<Directory> GetAllDirectories() {  
 return \_context.Set<Directory>().Include(x => x.Materials).ToList();  
 }  
  
 public Directory GetDirectoryById(int directoryId) {  
 return \_context.Set<Directory>().Include(x => x.Materials).  
 FirstOrDefault(x => x.Id == directoryId);  
 }  
  
 public int SaveDirectory(Directory directory) {  
 if (directory.Id == 0) {  
 \_context.Directory.Add(directory);  
 }  
 else {  
 \_context.Entry(directory).State = EntityState.*Modified*;  
 }  
  
 \_context.SaveChanges();  
 return directory.Id;  
 }  
  
 public void DeleteDirectory(Directory directory) {  
 try {  
 \_context.Directory.Remove(directory);  
 \_context.SaveChanges();  
 }  
 catch (DbUpdateConcurrencyException) { }  
 }  
 }  
**}**

**Файл EFMaterialRepository.cs**

using System.Collections.Generic;  
using System.Diagnostics.CodeAnalysis;  
using System.Linq;  
using BusinessLayer.Interfaces;  
using DataLayer;  
using DataLayer.Entityes;  
using Microsoft.EntityFrameworkCore;  
  
namespace BusinessLayer.Implementations {  
 [SuppressMessage("ReSharper", "InconsistentNaming")]  
 public class EFMaterialRepository : IMaterialRepository {  
 private readonly EFDBContext \_context;  
  
 public EFMaterialRepository(EFDBContext context) {  
 this.\_context = context;  
 }  
   
 public IEnumerable<Material> GetAllMaterials() {  
 return \_context.Set<Material>().Include(x => x.Directory).ToList();  
 }  
  
 public Material GetMaterialById(int materialId) {  
 return \_context.Set<Material>().Include(x => x.Directory)  
 .FirstOrDefault(x => x.Id == materialId);  
 }  
  
 public int SaveMaterial(Material material) {  
 if (material.Id == 0) {  
 \_context.Material.Add(material);  
 }  
 else {  
 \_context.Entry(material).State = EntityState.*Modified*;  
 }  
  
 \_context.SaveChanges();  
 return material.Id;  
 }  
  
 public void DeleteMaterial(Material material) {  
 try {  
 \_context.Material.Remove(material);  
 \_context.SaveChanges();  
 } catch (DbUpdateConcurrencyException) {}  
 }  
 }  
}

**Структура проекта тестирования слоя бизнес-логики**

****

**Файл TestBaseInitializer.cs**

using System.Collections.Generic;  
using DataLayer;  
using DataLayer.Entityes;  
  
namespace BusinessLayerTests **{** public static class TestBaseInitializer {  
 public static void Initialize(EFDBContext context) {  
 var directories = new[] {  
 new Directory {Title = "FirstDirectory"},   
 new Directory {Title = "SecondDirectory"},  
 new Directory {Title = "Third empty Directory"}  
 };  
 var materials = new[] {  
 new Material {Directory = directories[0], DirectoryId = directories[0].Id, Title = "First material"},  
 new Material {Directory = directories[0], DirectoryId = directories[0].Id, Title = "Second material"},  
 new Material {Directory = directories[1], DirectoryId = directories[1].Id, Title = "Third material"}  
 };  
 directories[0].Materials = new List<Material> {materials[0], materials[1]};  
 directories[1].Materials = new List<Material> {materials[2]};  
  
 context.Directory.AddRange(directories);  
 context.Material.AddRange(materials);  
 context.SaveChanges();  
 }  
 }  
**}**

**Файл EFDirectoryRepositoryTests**

using System;  
using System.Diagnostics.CodeAnalysis;  
using System.Linq;  
using BusinessLayer.Implementations;  
using DataLayer;  
using DataLayer.Entityes;  
using Microsoft.EntityFrameworkCore;  
using Xunit;  
  
namespace BusinessLayerTests {  
 [SuppressMessage("ReSharper", "InconsistentNaming")]  
 public class EFDirectoryRepositoryTests : IDisposable{  
 private readonly EFDBContext \_context;  
 private readonly EFDBContext \_emptyContext;  
 private readonly EFDirectoryRepository \_directoryRepository;  
 private readonly EFDirectoryRepository \_emptyDirectoryRepository;  
  
 public EFDirectoryRepositoryTests() {  
 var optionsBuilder = new DbContextOptionsBuilder<EFDBContext>()  
 .UseInMemoryDatabase(Guid.NewGuid().ToString());  
 optionsBuilder.EnableSensitiveDataLogging();  
 var options = optionsBuilder.Options;  
   
 \_context = new EFDBContext(options);  
 \_context.Database.EnsureCreated();  
 \_directoryRepository = new EFDirectoryRepository(\_context);  
   
 TestBaseInitializer.Initialize(\_context);  
   
 optionsBuilder = new DbContextOptionsBuilder<EFDBContext>()  
 .UseInMemoryDatabase(Guid.NewGuid().ToString());  
 optionsBuilder.EnableSensitiveDataLogging();  
 options = optionsBuilder.Options;  
   
 \_emptyContext = new EFDBContext(options);  
 \_emptyContext.Database.EnsureCreated();  
 \_emptyDirectoryRepository = new EFDirectoryRepository(\_emptyContext);  
 }  
  
 [Fact]  
 public void GetAllDirectories\_BaseHaveData() {  
 var result = \_directoryRepository.GetAllDirectories();  
 var directories = result.ToList();  
 Assert.Equal(3, directories.Count);  
 Assert.Equal(2, directories[0].Materials.Count);  
 Assert.Single(directories[1].Materials);  
 Assert.Empty(directories[2].Materials);  
 }  
  
 [Fact]  
 public void GetAllDirectories\_EmptyBase() {  
 var result = \_emptyDirectoryRepository.GetAllDirectories();  
 var directories = result.ToList();  
 Assert.Empty(directories);  
 }  
  
 [Fact]  
 public void GetDirectoryById\_CurrentIdInDatabase() {  
 var result = \_directoryRepository.GetDirectoryById(2);  
 Assert.Equal(2, result.Id);  
 Assert.Single(result.Materials);  
 }  
  
 [Fact]  
 public void GetDirectoryById\_NoCurrentIdInDatabase() {  
 var result = \_directoryRepository.GetDirectoryById(4);  
 Assert.Null(result);  
 }  
  
 [Fact]  
 public void SaveDirectory\_DirectoryNotInBase() {  
 var directory = new Directory {Title = "Saved Directory"};  
 var result = \_directoryRepository.SaveDirectory(directory);  
 Assert.Equal("Saved Directory", \_directoryRepository.GetDirectoryById(result).Title);  
 }  
   
 [Fact]  
 public void SaveDirectory\_DirectoryInBase() {  
 var directory = \_directoryRepository.GetDirectoryById(1);  
 directory.Title = "Saved Directory";  
 var directoryId = \_directoryRepository.SaveDirectory(directory);  
 Assert.Equal("Saved Directory", \_directoryRepository.GetDirectoryById(directoryId).Title);  
 }  
  
 [Fact]  
 public void DeleteDirectory\_DirectoryInBase() {  
 var directory = \_directoryRepository.GetDirectoryById(1);  
 var directoryId = directory.Id;  
 Assert.Equal("FirstDirectory", directory.Title);  
 \_directoryRepository.DeleteDirectory(directory);  
 Assert.Null(\_directoryRepository.GetDirectoryById(directoryId));  
 }  
   
 [Fact]  
 public void DeleteDirectory\_DirectoryNotInBase() {  
 var directory = new Directory {Title = "Saved Directory", Id = 0};  
 \_directoryRepository.DeleteDirectory(directory);  
 Assert.Null(\_directoryRepository.GetDirectoryById(directory.Id));  
 }  
  
 public void Dispose() {  
 \_context.Database.EnsureDeleted();  
 \_context.Dispose();  
 \_emptyContext.Database.EnsureDeleted();  
 \_emptyContext.Dispose();  
 }  
 }  
}

**Вывод**

В ходе выполнения работы был создан слой бизнес-логики приложения, а также все его методы были протестированы с помощью фреймворка XUnit.