**Министерство транспорта Российской Федерации**

**Федеральное государственное автономное образовательное  
учреждение высшего образования  
«РУТ (МИИТ)»**

**Институт транспортной техники и систем управления**

**Кафедра «Управление и защита информации»**

ПРАКТИЧЕСКАЯ РАБОТА

по дисциплине

**«Основы построения защищенных баз данных»**

**на тему  
«Архив документов»**

Выполнил: ст. гр. ТКИ-541,  
Николаев А. М.  
Проверил: доц., к.т.н.,

Васильева М.А.

Москва 2023

**Оглавление**

[Цель работы 6](#_Toc149055120)

[Задание 6](#_Toc149055121)

[Выбор стека технологий 7](#_Toc149055122)

[UNL-диаграмма классов 8](#_Toc149055123)

[Код программы 8](#_Toc149055124)

[1.1. Файл Program.cs 8](#_Toc149055125)

[1.2. Файл Document.cs 12](#_Toc149055126)

[1.3. Файл Inventory.cs 14](#_Toc149055127)

[1.4. Файл ReadingRoom.cs 15](#_Toc149055128)

[1.5. Файл DocumentTests.cs 17](#_Toc149055129)

[1.6. Файл InventoryTests.cs 19](#_Toc149055130)

[1.7. Файл ReadingRoomTests.cs 22](#_Toc149055131)

[1.8. Файл DocumentMap.cs 24](#_Toc149055132)

[1.9. Файл InventoryMap.cs 25](#_Toc149055133)

[1.10. Файл ReadingRoomMap.cs 25](#_Toc149055134)

[1.11. Файл DocumentRepository.cs 26](#_Toc149055135)

[1.12. Файл InventoryRepository.cs 28](#_Toc149055136)

[1.13. Файл ReadingRoomRepository.cs 30](#_Toc149055137)

[1.14. Файл AppDbContext.cs 32](#_Toc149055138)

[1.15. Файл DesignTimeDbContextFactory.cs 33](#_Toc149055139)

[1.16. Файл EFCoreTestsConfigurator.cs 34](#_Toc149055140)

[1.17. Файл IRepository.cs 34](#_Toc149055141)

[1.18. Файл DocumentMapTests.cs 35](#_Toc149055142)

[1.19. Файл InventoryMapTests.cs 36](#_Toc149055143)

[1.20. Файл ReadingRoomMapTests.cs 38](#_Toc149055144)

[Прохождение тестов 40](#_Toc149055145)

[Полученная ERD 41](#_Toc149055146)

[Вывод 42](#_Toc149055147)

[Список литературы 43](#_Toc149055148)

Цель работы

Познакомиться с механизмами и инструментами, которые позволяют предотвратить SQL-инъекции при взаимодействии с базами данных на языке программирования C# с использованием технологии ORM (Object-Relational Mapping).

Задание

1. Реализовать три класса, описывающих структуру Архива:
   * Document - класс, описывающий документ;
   * Inventory - класс, описывающий опись;
   * ReadingRoom - класс, описывающий читальный зал.
2. Реализовать методы для каждого из классов, необходимые для выполнения следующих операций:
   * добавление элемента данных;
   * поиск элемента данных;
   * изменение элемента данных;
   * удаление элемента данных.
3. Реализовать маппинги для классов Document, Inventory и ReadingRoom, чтобы обеспечить корректное отображение данных в базе данных.
4. Реализовать репозитории для управления сущностями каждого из классов.
5. Написать тесты для проверки корректности работы классов Document, Inventory и ReadingRoom.
6. Написать тесты для проверки корректности работы маппингов.
7. Реализовать консольный интерфейс с функциями CRUD (создание, чтение, обновление, удаление) для управления данными в Архиве.
8. Предоставить тестовый набор данных для демонстрации работы программного обеспечения.

Выбор стека технологий

1. Язык программирования - C#
2. Doxygen - используется для автоматической генерации документации из исходного кода.
3. StyleCop - используется для проверки соответствия кода стандартам кодирования.
4. NUnit - фреймворк для модульного тестирования в среде .NET.
5. Entity Framework - фреймворк для работы с базами данных.
6. PostgreSQL - система управления базами данных с открытым исходным кодом.
7. Visual Studio 2022 Enterprise - среда разработки, которая предоставляет разработчикам все необходимые инструменты для эффективной разработки, отладки и тестирования приложений.
8. DBeaver - инструмент (графическое окружение) для работы с базами данных, который поддерживает большинство популярных СУБД, включая PostgreSQL.

UNL-диаграмма классов

На рисунке 1 приведена UML-диаграмма.

|  |
| --- |
|  |
| Рисунок 1 – UML-диаграмма классов |

Код программы

* 1. Файл Program.cs

// <copyright file="Program.cs" company="Николаев А.М.">

// Copyright (c) Николаев А.М.. All rights reserved.

// </copyright>

namespace ArchiveDemo

{

using System;

using Domain;

using Microsoft.EntityFrameworkCore;

using ORM.Repositories;

/// <summary>

/// Главный класс программы.

/// </summary>

internal class Program

{

/// <summary>

/// Точка входа в программу.

/// </summary>

private static void Main()

{

var optionsBuilder = new DbContextOptionsBuilder<AppDbContext>();

optionsBuilder.UseNpgsql("Host=localhost;Username=postgres;Password=;Database=testbd1;");

using var context = new AppDbContext(optionsBuilder.Options);

var documentRepository = new DocumentRepository(context);

var inventoryRepository = new InventoryRepository(context);

var readingRoomRepository = new ReadingRoomRepository(context);

while (true)

{

Console.WriteLine("Выберите действие:");

Console.WriteLine("1: Создать документ");

Console.WriteLine("2: Показать все документы");

Console.WriteLine("3: Обновить документ");

Console.WriteLine("4: Удалить документ");

Console.WriteLine("0: Выход");

var choice = Console.ReadLine();

switch (choice)

{

case "1":

CreateDocument(documentRepository, inventoryRepository, readingRoomRepository);

break;

case "2":

ReadDocuments(documentRepository);

break;

case "3":

UpdateDocument(documentRepository);

break;

case "4":

DeleteDocument(documentRepository);

break;

case "0":

return;

default:

Console.WriteLine("Неверный ввод. Попробуйте еще раз.");

break;

}

}

}

/// <summary>

/// Создает новый документ.

/// </summary>

/// <param name="documentRepository">Репозиторий документов.</param>

/// <param name="inventoryRepository">Репозиторий описей.</param>

/// <param name="readingRoomRepository">Репозиторий читальных залов.</param>

private static void CreateDocument(DocumentRepository documentRepository, InventoryRepository inventoryRepository, ReadingRoomRepository readingRoomRepository)

{

Console.WriteLine("Введите название документа:");

var title = Console.ReadLine();

var inventory = CreateOrGetInventory(inventoryRepository);

var readingRoom = CreateOrGetReadingRoom(readingRoomRepository);

#pragma warning disable CS8604 // Возможно, аргумент-ссылка, допускающий значение NULL.

var document = new Document(0, title, inventory, readingRoom);

#pragma warning restore CS8604 // Возможно, аргумент-ссылка, допускающий значение NULL.

documentRepository.Save(document);

Console.WriteLine("Документ успешно создан.");

}

/// <summary>

/// Создает или получает опись.

/// </summary>

/// <param name="inventoryRepository">Репозиторий описей.</param>

/// <returns>Опись.</returns>

private static Domain.Inventory CreateOrGetInventory(InventoryRepository inventoryRepository)

{

Console.WriteLine("Введите название описи:");

var title = Console.ReadLine();

var inventory = inventoryRepository.Find(i => i.Title == title);

if (inventory == null)

{

#pragma warning disable CS8604 // Возможно, аргумент-ссылка, допускающий значение NULL.

inventory = new Inventory(0, title);

#pragma warning restore CS8604 // Возможно, аргумент-ссылка, допускающий значение NULL.

inventoryRepository.Save(inventory);

}

return inventory;

}

/// <summary>

/// Создает или получает читальный зал.

/// </summary>

/// <param name="readingRoomRepository">Репозиторий читальных залов.</param>

/// <returns>Читальный зал.</returns>

private static Domain.ReadingRoom CreateOrGetReadingRoom(ReadingRoomRepository readingRoomRepository)

{

Console.WriteLine("Введите название читального зала:");

var name = Console.ReadLine();

var readingRoom = readingRoomRepository.Find(r => r.Name == name);

if (readingRoom == null)

{

#pragma warning disable CS8604 // Возможно, аргумент-ссылка, допускающий значение NULL.

readingRoom = new ReadingRoom(0, name);

#pragma warning restore CS8604 // Возможно, аргумент-ссылка, допускающий значение NULL.

readingRoomRepository.Save(readingRoom);

}

return readingRoom;

}

/// <summary>

/// Выводит все документы.

/// </summary>

/// <param name="documentRepository">Репозиторий документов.</param>

private static void ReadDocuments(DocumentRepository documentRepository)

{

var documents = documentRepository.GetAll();

foreach (var document in documents)

{

Console.WriteLine(document);

}

}

/// <summary>

/// Обновляет документ.

/// </summary>

/// <param name="documentRepository">Репозиторий документов.</param>

private static void UpdateDocument(DocumentRepository documentRepository)

{

Console.WriteLine("Введите ID документа, который вы хотите обновить:");

if (int.TryParse(Console.ReadLine(), out var id))

{

var document = documentRepository.Get(id);

if (document != null)

{

Console.WriteLine("Введите новое название документа:");

var newTitle = Console.ReadLine();

#pragma warning disable CS8604 // Возможно, аргумент-ссылка, допускающий значение NULL.

document.SetName(newTitle);

#pragma warning restore CS8604 // Возможно, аргумент-ссылка, допускающий значение NULL.

documentRepository.Save(document);

Console.WriteLine("Документ успешно обновлен.");

}

else

{

Console.WriteLine("Документ с таким ID не найден.");

}

}

else

{

Console.WriteLine("Неверный ввод. Попробуйте еще раз.");

}

}

/// <summary>

/// Удаляет документ.

/// </summary>

/// <param name="documentRepository">Репозиторий документов.</param>

private static void DeleteDocument(DocumentRepository documentRepository)

{

Console.WriteLine("Введите ID документа, который вы хотите удалить:");

if (int.TryParse(Console.ReadLine(), out var id))

{

var document = documentRepository.Get(id);

if (document != null)

{

documentRepository.Delete(document);

Console.WriteLine("Документ успешно удален.");

}

else

{

Console.WriteLine("Документ с таким ID не найден.");

}

}

else

{

Console.WriteLine("Неверный ввод. Попробуйте еще раз.");

}

}

}

}

* 1. Файл Document.cs

// <copyright file="Document.cs" company="Николаев А.М.">

// Copyright (c) Николаев А.М.. All rights reserved.

// </copyright>

namespace Domain

{

using System;

using Staff.Extensions;

/// <summary>

/// Документ.

/// </summary>

public class Document

{

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="Document"/>.

/// </summary>

/// <param name="documentCode">Код документа.</param>

/// <param name="title">Название.</param>

/// <param name="inventory">Опись.</param>

/// <param name="readingRoom">Читальный зал.</param>

public Document(int documentCode, string title, Inventory inventory, ReadingRoom readingRoom)

{

this.DocumentCode = documentCode;

this.Title = title.TrimOrNull() ?? throw new ArgumentOutOfRangeException(nameof(title));

this.Inventory = inventory ?? throw new ArgumentNullException(nameof(inventory));

this.ReadingRoom = readingRoom ?? throw new ArgumentNullException(nameof(readingRoom));

this.InventoryCode = inventory.InventoryCode;

this.ReadingRoomCode = readingRoom.ReadingRoomCode;

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="Document"/>.

/// </summary>

[Obsolete("For ORM only", true)]

#pragma warning disable CS8618 // Поле, не допускающее значения NULL, должно содержать значение, отличное от NULL, при выходе из конструктора. Возможно, стоит объявить поле как допускающее значения NULL.

protected Document()

#pragma warning restore CS8618 // Поле, не допускающее значения NULL, должно содержать значение, отличное от NULL, при выходе из конструктора. Возможно, стоит объявить поле как допускающее значения NULL.

{

}

/// <summary>

/// Уникальный код документа.

/// </summary>

public virtual int DocumentCode { get; protected set; }

/// <summary>

/// Название.

/// </summary>

public virtual string Title { get; protected set; }

/// <summary>

/// Опись.

/// </summary>

public virtual Inventory Inventory { get; protected set; }

/// <summary>

/// Читальный зал.

/// </summary>

public virtual ReadingRoom ReadingRoom { get; protected set; }

/// <summary>

/// Код описи.

/// </summary>

public virtual int InventoryCode { get; protected set; }

/// <summary>

/// Код читального зала.

/// </summary>

public virtual int ReadingRoomCode { get; protected set; }

/// <summary>

/// Устанавливает новое название документа.

/// </summary>

/// <param name="newName">Новое название документа.</param>

public virtual void SetName(string newName)

{

this.Title = newName.TrimOrNull() ?? throw new ArgumentOutOfRangeException(nameof(newName));

}

/// <summary>

/// Сравнивает текущий объект с другим объектом того же типа.

/// </summary>

/// <param name="obj">Объект для сравнения.</param>

/// <returns>Возвращает true, если объекты равны, иначе false.</returns>

public override bool Equals(object? obj)

{

if (obj is Document other)

{

return this.DocumentCode == other.DocumentCode &&

this.Title == other.Title &&

this.InventoryCode == other.InventoryCode &&

this.ReadingRoomCode == other.ReadingRoomCode;

}

return false;

}

/// <summary>

/// Возвращает хеш-код для текущего объекта.

/// </summary>

/// <returns>Хеш-код текущего объекта.</returns>

public override int GetHashCode()

{

return HashCode.Combine(this.DocumentCode, this.Title, this.InventoryCode, this.ReadingRoomCode);

}

/// <summary>

/// Возвращает строковое представление текущего объекта.

/// </summary>

/// <returns>Строковое представление текущего объекта.</returns>

public override string ToString()

{

return $"Код документа: {this.DocumentCode}, Название: {this.Title}, Код описи: {this.InventoryCode}, Код читального зала: {this.ReadingRoomCode}";

}

}

}

* 1. Файл Inventory.cs

// <copyright file="Inventory.cs" company="Николаев А.М.">

// Copyright (c) Николаев А.М.. All rights reserved.

// </copyright>

namespace Domain

{

using System;

using Staff.Extensions;

/// <summary>

/// Опись.

/// </summary>

public class Inventory

{

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="Inventory"/>.

/// </summary>

/// <param name="inventoryCode">Код описи.</param>

/// <param name="title">Название описи.</param>

public Inventory(int inventoryCode, string title)

{

this.InventoryCode = inventoryCode;

this.Title = title.TrimOrNull() ?? throw new ArgumentOutOfRangeException(nameof(title));

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="Inventory"/>.

/// </summary>

[Obsolete("For ORM only", true)]

#pragma warning disable CS8618 // Поле, не допускающее значения NULL, должно содержать значение, отличное от NULL, при выходе из конструктора. Возможно, стоит объявить поле как допускающее значения NULL.

protected Inventory()

#pragma warning restore CS8618 // Поле, не допускающее значения NULL, должно содержать значение, отличное от NULL, при выходе из конструктора. Возможно, стоит объявить поле как допускающее значения NULL.

{

}

/// <summary>

/// Уникальный код описи.

/// </summary>

public virtual int InventoryCode { get; protected set; }

/// <summary>

/// Название описи.

/// </summary>

public virtual string Title { get; protected set; }

/// <summary>

/// Устанавливает новое название описи.

/// </summary>

/// <param name="newName">Новое название описи.</param>

public virtual void SetName(string newName)

{

this.Title = newName.TrimOrNull() ?? throw new ArgumentOutOfRangeException(nameof(newName));

}

/// <summary>

/// Сравнивает текущий объект с другим объектом того же типа.

/// </summary>

/// <param name="obj">Объект для сравнения.</param>

/// <returns>Возвращает true, если объекты равны, иначе false.</returns>

public override bool Equals(object? obj)

{

if (obj is Inventory other)

{

return this.InventoryCode == other.InventoryCode &&

this.Title == other.Title;

}

return false;

}

/// <summary>

/// Возвращает хеш-код для текущего объекта.

/// </summary>

/// <returns>Хеш-код текущего объекта.</returns>

public override int GetHashCode()

{

return HashCode.Combine(this.InventoryCode, this.Title);

}

/// <summary>

/// Возвращает строковое представление текущего объекта.

/// </summary>

/// <returns>Строковое представление текущего объекта.</returns>

public override string ToString()

{

return $"Код описи: {this.InventoryCode}, Название: {this.Title}";

}

}

}

* 1. Файл ReadingRoom.cs

// <copyright file="ReadingRoom.cs" company="Николаев А.М.">

// Copyright (c) Николаев А.М.. All rights reserved.

// </copyright>

namespace Domain

{

using System;

using Staff.Extensions;

/// <summary>

/// Читальный зал.

/// </summary>

public class ReadingRoom

{

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="ReadingRoom"/>.

/// </summary>

/// <param name="code">Код читального зала.</param>

/// <param name="name">Название читального зала.</param>

#pragma warning disable CS8618 // Поле, не допускающее значения NULL, должно содержать значение, отличное от NULL, при выходе из конструктора. Возможно, стоит объявить поле как допускающее значения NULL.

public ReadingRoom(int code, string name)

#pragma warning restore CS8618 // Поле, не допускающее значения NULL, должно содержать значение, отличное от NULL, при выходе из конструктора. Возможно, стоит объявить поле как допускающее значения NULL.

{

this.ReadingRoomCode = code;

this.SetName(name);

}

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="ReadingRoom"/>.

/// </summary>

[Obsolete("For ORM only", true)]

#pragma warning disable CS8618 // Поле, не допускающее значения NULL, должно содержать значение, отличное от NULL, при выходе из конструктора. Возможно, стоит объявить поле как допускающее значения NULL.

protected ReadingRoom()

#pragma warning restore CS8618 // Поле, не допускающее значения NULL, должно содержать значение, отличное от NULL, при выходе из конструктора. Возможно, стоит объявить поле как допускающее значения NULL.

{

}

/// <summary>

/// Уникальный код читального зала.

/// </summary>

public virtual int ReadingRoomCode { get; protected set; }

/// <summary>

/// Название читального зала.

/// </summary>

public virtual string Name { get; protected set; }

/// <summary>

/// Устанавливает новое значение для свойства Name.

/// </summary>

/// <param name="name">Новое значение для свойства Name.</param>

public void SetName(string name)

{

this.Name = name.TrimOrNull() ?? throw new ArgumentOutOfRangeException(nameof(name));

}

/// <summary>

/// Сравнивает текущий объект с другим объектом того же типа.

/// </summary>

/// <param name="obj">Объект для сравнения.</param>

/// <returns>Возвращает true, если объекты равны, иначе false.</returns>

public override bool Equals(object? obj)

{

if (obj is ReadingRoom other)

{

return this.ReadingRoomCode == other.ReadingRoomCode &&

this.Name == other.Name;

}

return false;

}

/// <summary>

/// Возвращает хеш-код для текущего объекта.

/// </summary>

/// <returns>Хеш-код текущего объекта.</returns>

public override int GetHashCode()

{

return HashCode.Combine(this.ReadingRoomCode, this.Name);

}

/// <summary>

/// Возвращает строковое представление текущего объекта.

/// </summary>

/// <returns>Строковое представление текущего объекта.</returns>

public override string ToString()

{

return $"Код читального зала: {this.ReadingRoomCode}, Название: {this.Name}";

}

}

}

* 1. Файл DocumentTests.cs

// <copyright file="DocumentTests.cs" company="Николаев А.М.">

// Copyright (c) Николаев А.М.. All rights reserved.

// </copyright>

namespace Domain.Tests

{

using System;

using Domain;

using NUnit.Framework;

/// <summary>

/// Тесты для класса <see cref="Document"/>.

/// </summary>

[TestFixture]

public class DocumentTests

{

private Inventory inventory;

private ReadingRoom readingRoom;

/// <summary>

/// Настройка тестового окружения.

/// </summary>

[SetUp]

public void Setup()

{

this.inventory = new Inventory(1, "Опись 1");

this.readingRoom = new ReadingRoom(1, "Читальный зал 1");

}

/// <summary>

/// Тестирование метода <see cref="Document.ToString"/>.

/// </summary>

[Test]

public void ToString\_ValidData\_Success()

{

// Arrange

var document = new Document(1, "Паспорт", this.inventory, this.readingRoom);

// Act

var result = document.ToString();

// Assert

Assert.That(result, Is.EqualTo("Код документа: 1, Название: Паспорт, Код описи: 1, Код читального зала: 1"));

}

/// <summary>

/// Тестирование метода <see cref="Document.SetName"/> с корректными данными.

/// </summary>

[Test]

public void SetName\_ValidData\_Success()

{

// Arrange

var document = new Document(1, "Паспорт", this.inventory, this.readingRoom);

var newTitle = "Свидетельство о рождении";

// Act

document.SetName(newTitle);

var result = document.ToString();

// Assert

Assert.That(result, Is.EqualTo("Код документа: 1, Название: Свидетельство о рождении, Код описи: 1, Код читального зала: 1"));

}

/// <summary>

/// Тестирование метода <see cref="Document.SetName"/> с пустым названием.

/// </summary>

[Test]

public void SetName\_EmptyTitle\_Fail()

{

// Arrange

var document = new Document(1, "Паспорт", this.inventory, this.readingRoom);

// Act & Assert

Assert.Throws<ArgumentOutOfRangeException>(() => document.SetName(string.Empty));

}

/// <summary>

/// Тест создания двух объектов Document с одинаковыми значениями.

/// </summary>

[Test]

public void Equals\_WithSameValues\_ReturnsTrue()

{

// Arrange

var document1 = new Document(1, "Паспорт", this.inventory, this.readingRoom);

var document2 = new Document(1, "Паспорт", this.inventory, this.readingRoom);

// Act

bool areEqual = document1.Equals(document2);

// Assert

Assert.That(areEqual, Is.True);

}

/// <summary>

/// Тест создания двух объектов Document с разными значениями Title.

/// </summary>

[Test]

public void Equals\_WithDifferentTitle\_ReturnsFalse()

{

// Arrange

var document1 = new Document(1, "Паспорт", this.inventory, this.readingRoom);

var document2 = new Document(1, "Свидетельство о рождении", this.inventory, this.readingRoom);

// Act

bool areEqual = document1.Equals(document2);

// Assert

Assert.That(areEqual, Is.False);

}

/// <summary>

/// Тестирование конструктора <see cref="Document"/> с пустым названием.

/// </summary>

[Test]

public void Ctor\_EmptyTitle\_Fail()

{

// Act & Assert

Assert.Throws<ArgumentOutOfRangeException>(() => \_ = new Document(1, string.Empty, this.inventory, this.readingRoom));

}

}

}

* 1. Файл InventoryTests.cs

// <copyright file="InventoryTests.cs" company="Николаев А.М.">

// Copyright (Николаев А.М.. All rights reserved.

// </copyright>

namespace Domain.Tests

{

using System;

using Domain;

using NUnit.Framework;

/// <summary>

/// Класс, содержащий тесты для <see cref="Inventory"/>.

/// </summary>

[TestFixture]

public class InventoryTests

{

/// <summary>

/// Объект документа для тестирования.

/// </summary>

private Document document;

/// <summary>

/// Метод, который выполняется перед каждым тестом.

/// </summary>

[SetUp]

public void Setup()

{

var inventory = new Inventory(1001, "Опись 1");

var readingRoom = new ReadingRoom(101, "Читальный зал 1");

this.document = new Document(1, "Договор", inventory, readingRoom);

}

/// <summary>

/// Тестирование метода <see cref="Inventory.ToString"/> на корректность возвращаемой строки.

/// </summary>

[Test]

public void ToString\_ValidData\_Success()

{

// Arrange

var inventory = new Inventory(1001, "Опись 1");

// Act

var result = inventory.ToString();

// Assert

Assert.That(result, Does.Contain("Опись 1"));

}

/// <summary>

/// Тестирование конструктора <see cref="Inventory"/> на выброс исключения при передаче пустого названия.

/// </summary>

[Test]

public void Ctor\_WrongData\_EmptyTitle\_Fail()

{

Assert.Throws<ArgumentOutOfRangeException>(() => \_ = new Inventory(1001, string.Empty));

}

/// <summary>

/// Тестирование метода <see cref="Inventory.SetName"/> на корректность изменения названия описи.

/// </summary>

[Test]

public void SetName\_ValidData\_Success()

{

// Arrange

var inventory = new Inventory(1001, "Опись 1");

// Act

inventory.SetName("Опись 2");

// Assert

Assert.That(inventory.ToString(), Does.Contain("Опись 2"));

}

/// <summary>

/// Тестирование метода <see cref="Inventory.SetName"/> на выброс исключения при передаче пустого значения.

/// </summary>

[Test]

public void SetName\_EmptyTitle\_Fail()

{

// Arrange

var inventory = GetInventory("Опись 1");

// Act & Assert

Assert.Throws<ArgumentOutOfRangeException>(() => inventory.SetName(string.Empty));

}

/// <summary>

/// Тест создания двух объектов Inventory с одинаковыми значениями.

/// </summary>

[Test]

public void Equals\_WithSameValues\_ReturnsTrue()

{

// Arrange

var inventory1 = new Inventory(1, "Опись 1");

var inventory2 = new Inventory(1, "Опись 1");

// Act

bool areEqual = inventory1.Equals(inventory2);

// Assert

Assert.That(areEqual, Is.True);

}

/// <summary>

/// Тест создания двух объектов Inventory с разными значениями Title.

/// </summary>

[Test]

public void Equals\_WithDifferentTitle\_ReturnsFalse()

{

// Arrange

var inventory1 = new Inventory(1, "Опись 1");

var inventory2 = new Inventory(1, "Опись 2");

// Act

bool areEqual = inventory1.Equals(inventory2);

// Assert

Assert.That(areEqual, Is.False);

}

/// <summary>

/// Получение объекта <see cref="Inventory"/> для тестирования.

/// </summary>

/// <param name="title">Название описи.</param>

/// <returns>Объект <see cref="Inventory"/>.</returns>

private static Inventory GetInventory(string title = null)

{

return new Inventory(1001, title ?? "Тестовая опись");

}

}

}

* 1. Файл ReadingRoomTests.cs

// <copyright file="ReadingRoomTests.cs" company="Николаев А.М.">

// Copyright (c) Николаев А.М.. All rights reserved.

// </copyright>

namespace Domain.Tests

{

using System;

using Domain;

using NUnit.Framework;

/// <summary>

/// Тесты для класса <see cref="ReadingRoom"/>.

/// </summary>

[TestFixture]

public class ReadingRoomTests

{

/// <summary>

/// Тестовый читальный зал.

/// </summary>

private ReadingRoom readingRoom;

/// <summary>

/// Установка тестового окружения.

/// </summary>

[SetUp]

public void Setup()

{

this.readingRoom = new ReadingRoom(1, "Главный зал");

}

/// <summary>

/// Тест метода ToString с валидными данными.

/// </summary>

[Test]

public void ToString\_ValidData\_Success()

{

// Act

var result = this.readingRoom.ToString();

// Assert

Assert.That(result, Is.EqualTo("Код читального зала: 1, Название: Главный зал"));

}

/// <summary>

/// Тест конструктора с неверными данными.

/// </summary>

[Test]

public void Ctor\_WrongData\_EmptyName\_Fail()

{

// Assert

Assert.Throws<ArgumentOutOfRangeException>(() => \_ = new ReadingRoom(1, string.Empty));

}

/// <summary>

/// Тест создания двух объектов ReadingRoom с одинаковыми значениями.

/// </summary>

[Test]

public void Equals\_WithSameValues\_ReturnsTrue()

{

// Arrange

var readingRoom1 = new ReadingRoom(1, "Главный зал");

var readingRoom2 = new ReadingRoom(1, "Главный зал");

// Act

bool areEqual = readingRoom1.Equals(readingRoom2);

// Assert

Assert.IsTrue(areEqual);

}

/// <summary>

/// Тест создания двух объектов ReadingRoom с разными значениями Name.

/// </summary>

[Test]

public void Equals\_WithDifferentName\_ReturnsFalse()

{

// Arrange

var readingRoom1 = new ReadingRoom(1, "Главный зал");

var readingRoom2 = new ReadingRoom(1, "Малый зал");

// Act

bool areEqual = readingRoom1.Equals(readingRoom2);

// Assert

Assert.IsFalse(areEqual);

}

/// <summary>

/// Тест конструктора с неверными данными.

/// </summary>

[Test]

public void Ctor\_WrongData\_NullName\_Fail()

{

// Assert

Assert.Throws<ArgumentOutOfRangeException>(() => \_ = new ReadingRoom(1, null));

}

/// <summary>

/// Тест метода SetName с валидными данными.

/// </summary>

[Test]

public void SetName\_ValidData\_Success()

{

// Arrange

var newName = "Малый зал";

// Act

this.readingRoom.SetName(newName);

// Assert

Assert.That(this.readingRoom.Name, Is.EqualTo(newName));

}

/// <summary>

/// Тест метода SetName с неверными данными.

/// </summary>

[Test]

public void SetName\_WrongData\_EmptyName\_Fail()

{

// Assert

Assert.Throws<ArgumentOutOfRangeException>(() => this.readingRoom.SetName(string.Empty));

}

/// <summary>

/// Тест метода SetName с неверными данными.

/// </summary>

[Test]

public void SetName\_WrongData\_NullName\_Fail()

{

// Assert

Assert.Throws<ArgumentOutOfRangeException>(() => this.readingRoom.SetName(null));

}

}

}

* 1. Файл DocumentMap.cs

// <copyright file="DocumentMap.cs" company="Николаев А.М.">

// Copyright (c) Николаев А.М.. All rights reserved.

// </copyright>

namespace ORM.Mappings

{

using Domain;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata.Builders;

/// <summary>

/// Класс, описывающий правила отображения <see cref="Document"/> на таблицу и наоборот.

/// </summary>

public class DocumentMap : IEntityTypeConfiguration<Document>

{

/// <summary>

/// Конфигурирует схему таблицы для <see cref="Document"/>.

/// </summary>

/// <param name="builder">Построитель сущности.</param>

public void Configure(EntityTypeBuilder<Document> builder)

{

// Указание имени таблицы

builder.ToTable("documents");

// Указание первичного ключа

builder.HasKey(d => d.DocumentCode)

.HasName("document\_code");

builder.Property(d => d.Title)

.IsRequired()

.HasMaxLength(255)

.HasColumnName("title");

builder.HasOne(d => d.Inventory)

.WithMany()

.HasForeignKey(d => d.InventoryCode)

.IsRequired()

.OnDelete(DeleteBehavior.Cascade)

.HasConstraintName("fk\_documents\_inventories");

builder.HasOne(d => d.ReadingRoom)

.WithMany()

.HasForeignKey(d => d.ReadingRoomCode)

.IsRequired()

.OnDelete(DeleteBehavior.Cascade)

.HasConstraintName("fk\_documents\_reading\_rooms");

}

}

}

* 1. Файл InventoryMap.cs

// <copyright file="InventoryMap.cs" company="Николаев А.М.">

// Copyright (c) Николаев А.М.. All rights reserved.

// </copyright>

namespace ORM.Mappings

{

using Domain;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata.Builders;

/// <summary>

/// Класс, описывающий правила отображения объектов <see cref="Inventory"/> на таблицу базы данных и обратно.

/// </summary>

public class InventoryMap : IEntityTypeConfiguration<Inventory>

{

/// <summary>

/// Конфигурирует схему таблицы для <see cref="Inventory"/>.

/// </summary>

/// <param name="builder">Построитель сущности.</param>

public void Configure(EntityTypeBuilder<Inventory> builder)

{

// Указание имени таблицы

builder.ToTable("inventories");

// Указание первичного ключа

builder.HasKey(i => i.InventoryCode)

.HasName("inventory\_code");

builder.Property(i => i.Title)

.IsRequired()

.HasMaxLength(255)

.HasColumnName("title");

}

}

}

* 1. Файл ReadingRoomMap.cs

// <copyright file="ReadingRoomMap.cs" company="Николаев А.М.">

// Copyright (c) Николаев А.М.. All rights reserved.

// </copyright>

namespace ORM.Mappings

{

using Domain;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata.Builders;

/// <summary>

/// Класс, описывающий правила отображения <see cref="ReadingRoom"/> на таблицу и наоборот.

/// </summary>

public class ReadingRoomMap : IEntityTypeConfiguration<ReadingRoom>

{

/// <summary>

/// Конфигурирует схему, необходимую для отображения сущности <see cref="ReadingRoom"/>.

/// </summary>

/// <param name="builder">Построитель сущности.</param>

public void Configure(EntityTypeBuilder<ReadingRoom> builder)

{

// Указание имени таблицы

builder.ToTable("reading\_rooms");

// Указание первичного ключа

builder.HasKey(r => r.ReadingRoomCode)

.HasName("reading\_room\_code");

builder.Property(r => r.Name)

.IsRequired()

.HasColumnName("name");

}

}

}

* 1. Файл DocumentRepository.cs

// <copyright file="DocumentRepository.cs" company="Николаев А.М.">

// Copyright (c) Николаев А.М.. All rights reserved.

// </copyright>

namespace ORM.Repositories

{

using System;

using System.Linq;

using System.Linq.Expressions;

using Domain;

/// <summary>

/// Репозиторий для управления сущностями <see cref="Document"/>.

/// </summary>

public class DocumentRepository : IRepository<Document>

{

private readonly AppDbContext context;

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="DocumentRepository"/>.

/// </summary>

/// <param name="context">Контекст базы данных.</param>

/// <exception cref="ArgumentNullException">Выбрасывается, когда контекст равен null.</exception>

public DocumentRepository(AppDbContext context)

{

this.context = context ?? throw new ArgumentNullException(nameof(context));

}

/// <summary>

/// Удаляет документ из базы данных.

/// </summary>

/// <param name="entity">Документ, который необходимо удалить.</param>

/// <exception cref="ArgumentNullException">Выбрасывается, если переданный документ равен null.</exception>

public void Delete(Document entity)

{

if (entity == null)

{

throw new ArgumentNullException(nameof(entity));

}

this.context.Documents.Remove(entity);

this.context.SaveChanges();

}

/// <summary>

/// Фильтрует документы на основе предиката.

/// </summary>

/// <param name="predicate">Предикат для фильтрации.</param>

/// <returns>Отфильтрованные документы.</returns>

public IQueryable<Document> Filter(Expression<Func<Document, bool>> predicate)

{

return this.GetAll().Where(predicate);

}

/// <summary>

/// Находит документ на основе предиката.

/// </summary>

/// <param name="predicate">Предикат для поиска.</param>

/// <returns>Найденный документ.</returns>

public Document Find(Expression<Func<Document, bool>> predicate)

{

#pragma warning disable CS8603 // Возможно, возврат ссылки, допускающей значение NULL.

return this.GetAll().FirstOrDefault(predicate);

#pragma warning restore CS8603 // Возможно, возврат ссылки, допускающей значение NULL.

}

/// <summary>

/// Получает документ по ID.

/// </summary>

/// <param name="id">ID документа.</param>

/// <returns>Документ с указанным ID.</returns>

public Document Get(int id)

{

#pragma warning disable CS8603 // Возможно, возврат ссылки, допускающей значение NULL.

return this.context.Documents.Find(id);

#pragma warning restore CS8603 // Возможно, возврат ссылки, допускающей значение NULL.

}

/// <summary>

/// Получает все документы.

/// </summary>

/// <returns>Все документы.</returns>

public IQueryable<Document> GetAll()

{

return this.context.Documents;

}

/// <summary>

/// Сохраняет документ.

/// </summary>

/// <param name="entity">Документ для сохранения.</param>

/// <returns>True, если документ был успешно сохранен; в противном случае, false.</returns>

public bool Save(Document entity)

{

if (entity == null)

{

return false;

}

this.context.Documents.Add(entity);

this.context.SaveChanges();

return true;

}

}

}

* 1. Файл InventoryRepository.cs

// <copyright file="InventoryRepository.cs" company="Николаев А.М.">

// Copyright (c) Николаев А.М.. All rights reserved.

// </copyright>

namespace ORM.Repositories

{

using System;

using System.Linq;

using System.Linq.Expressions;

using Domain;

/// <summary>

/// Репозиторий для управления сущностями <see cref="Inventory"/>.

/// </summary>

public class InventoryRepository : IRepository<Inventory>

{

private readonly AppDbContext context;

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="InventoryRepository"/>.

/// </summary>

/// <param name="context">Контекст базы данных.</param>

/// <exception cref="ArgumentNullException">Выбрасывается, когда контекст равен null.</exception>

public InventoryRepository(AppDbContext context)

{

this.context = context ?? throw new ArgumentNullException(nameof(context));

}

/// <summary>

/// Удаляет опись из базы данных.

/// </summary>

/// <param name="entity">Опись, который необходимо удалить.</param>

/// <exception cref="ArgumentNullException">Выбрасывается, если переданная опись равен null.</exception>

public void Delete(Inventory entity)

{

if (entity == null)

{

throw new ArgumentNullException(nameof(entity));

}

this.context.Inventories.Remove(entity);

this.context.SaveChanges();

}

/// <summary>

/// Фильтрует опись на основе предиката.

/// </summary>

/// <param name="predicate">Предикат для фильтрации.</param>

/// <returns>Отфильтрованная опись.</returns>

public IQueryable<Inventory> Filter(Expression<Func<Inventory, bool>> predicate)

{

return this.GetAll().Where(predicate);

}

/// <summary>

/// Находит опсиь на основе предиката.

/// </summary>

/// <param name="predicate">Предикат для поиска.</param>

/// <returns>Найденная опись.</returns>

public Inventory Find(Expression<Func<Inventory, bool>> predicate)

{

#pragma warning disable CS8603 // Возможно, возврат ссылки, допускающей значение NULL.

return this.GetAll().FirstOrDefault(predicate);

#pragma warning restore CS8603 // Возможно, возврат ссылки, допускающей значение NULL.

}

/// <summary>

/// Получает опись по ID.

/// </summary>

/// <param name="id">ID описи.</param>

/// <returns>Опись с указанным ID.</returns>

public Inventory Get(int id)

{

#pragma warning disable CS8603 // Возможно, возврат ссылки, допускающей значение NULL.

return this.context.Inventories.Find(id);

#pragma warning restore CS8603 // Возможно, возврат ссылки, допускающей значение NULL.

}

/// <summary>

/// Получает все описи.

/// </summary>

/// <returns>Все описи.</returns>

public IQueryable<Inventory> GetAll()

{

return this.context.Inventories;

}

/// <summary>

/// Сохраняет описи.

/// </summary>

/// <param name="entity">Опись для сохранения.</param>

/// <returns>True, если опись была успешно сохранена; в противном случае, false.</returns>

public bool Save(Inventory entity)

{

if (entity == null)

{

return false;

}

this.context.Inventories.Add(entity);

this.context.SaveChanges();

return true;

}

}

}

* 1. Файл ReadingRoomRepository.cs

// <copyright file="ReadingRoomRepository.cs" company="Николаев А.М.">

// Copyright (c) Николаев А.М.. All rights reserved.

// </copyright>

namespace ORM.Repositories

{

using System;

using System.Linq;

using System.Linq.Expressions;

using Domain;

/// <summary>

/// Репозиторий для управления сущностями <see cref="ReadingRoom"/>.

/// </summary>

public class ReadingRoomRepository : IRepository<ReadingRoom>

{

private readonly AppDbContext context;

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="ReadingRoomRepository"/>.

/// </summary>

/// <param name="context">Контекст базы данных.</param>

/// <exception cref="ArgumentNullException">Выбрасывается, когда контекст равен null.</exception>

public ReadingRoomRepository(AppDbContext context)

{

this.context = context ?? throw new ArgumentNullException(nameof(context));

}

/// <summary>

/// Удаляет читальный зал из базы данных.

/// </summary>

/// <param name="entity">Читальный зал, который необходимо удалить.</param>

/// <exception cref="ArgumentNullException">Выбрасывается, если переданный читальный зал равен null.</exception>

public void Delete(ReadingRoom entity)

{

if (entity == null)

{

throw new ArgumentNullException(nameof(entity));

}

this.context.ReadingRooms.Remove(entity);

this.context.SaveChanges();

}

/// <summary>

/// Фильтрует читальные залы на основе предиката.

/// </summary>

/// <param name="predicate">Предикат для фильтрации.</param>

/// <returns>Отфильтрованные читальные залы.</returns>

public IQueryable<ReadingRoom> Filter(Expression<Func<ReadingRoom, bool>> predicate)

{

return this.GetAll().Where(predicate);

}

/// <summary>

/// Находит читальный зал на основе предиката.

/// </summary>

/// <param name="predicate">Предикат для поиска.</param>

/// <returns>Найденный читальный зал.</returns>

public ReadingRoom Find(Expression<Func<ReadingRoom, bool>> predicate)

{

#pragma warning disable CS8603 // Возможно, возврат ссылки, допускающей значение NULL.

return this.GetAll().FirstOrDefault(predicate);

#pragma warning restore CS8603 // Возможно, возврат ссылки, допускающей значение NULL.

}

/// <summary>

/// Получает читальный зал по ID.

/// </summary>

/// <param name="id">ID читального зала.</param>

/// <returns>Читальный зал с указанным ID.</returns>

public ReadingRoom Get(int id)

{

#pragma warning disable CS8603 // Возможно, возврат ссылки, допускающей значение NULL.

return this.context.ReadingRooms.Find(id);

#pragma warning restore CS8603 // Возможно, возврат ссылки, допускающей значение NULL.

}

/// <summary>

/// Получает все читальные залы.

/// </summary>

/// <returns>Все читальные залы.</returns>

public IQueryable<ReadingRoom> GetAll()

{

return this.context.ReadingRooms;

}

/// <summary>

/// Сохраняет читальный зал.

/// </summary>

/// <param name="entity">Читальный зал для сохранения.</param>

/// <returns>True, если читальный зал был успешно сохранен; в противном случае, false.</returns>

public bool Save(ReadingRoom entity)

{

if (entity == null)

{

return false;

}

this.context.ReadingRooms.Add(entity);

this.context.SaveChanges();

return true;

}

}

}

* 1. Файл AppDbContext.cs

// <copyright file="AppDbContext.cs" company="Николаев А.М.">

// Copyright (c) Николаев А.М.. All rights reserved.

// </copyright>

using Domain;

using Microsoft.EntityFrameworkCore;

using ORM.Mappings;

/// <summary>

/// Контекст базы данных для работы с Entity Framework Core.

/// </summary>

public class AppDbContext : DbContext

{

/// <summary>

/// Получает или задает набор документов в базе данных.

/// </summary>

public DbSet<Document> Documents { get; set; }

/// <summary>

/// Получает или задает набор описей в базе данных.

/// </summary>

public DbSet<Inventory> Inventories { get; set; }

/// <summary>

/// Получает или задает набор читальных залов в базе данных.

/// </summary>

public DbSet<ReadingRoom> ReadingRooms { get; set; }

/// <summary>

/// Инициализирует новый экземпляр класса <see cref="AppDbContext"/>.

/// </summary>

/// <param name="options">Параметры конфигурации контекста.</param>

#pragma warning disable SA1201 // Elements should appear in the correct order

public AppDbContext(DbContextOptions<AppDbContext> options)

#pragma warning restore SA1201 // Elements should appear in the correct order

: base(options)

{

}

/// <summary>

/// Настроить модель данных, которая будет использоваться для создания базы данных.

/// </summary>

/// <param name="modelBuilder">Строитель модели для конфигурации сущностей.</param>

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

// Применить конфигурации маппинга для каждой сущности

modelBuilder.ApplyConfiguration(new DocumentMap());

modelBuilder.ApplyConfiguration(new InventoryMap());

modelBuilder.ApplyConfiguration(new ReadingRoomMap());

}

}

* 1. Файл DesignTimeDbContextFactory.cs

// <copyright file="DesignTimeDbContextFactory.cs" company="Николаев А.М.">

// Copyright (c) Николаев А.М.. All rights reserved.

// </copyright>

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Design;

using ORM;

/// <summary>

/// Фабрика для создания контекста данных во время дизайна.

/// </summary>

public class DesignTimeDbContextFactory : IDesignTimeDbContextFactory<AppDbContext>

{

/// <summary>

/// Создает экземпляр контекста данных.

/// </summary>

/// <param name="args">Аргументы командной строки.</param>

/// <returns>Возвращает экземпляр контекста данных.</returns>

public AppDbContext CreateDbContext(string[] args)

{

var optionsBuilder = new DbContextOptionsBuilder<AppDbContext>();

var connectionString = "Host=localhost;Username=postgres;Password=;Database=testbd1;";

optionsBuilder.UseNpgsql(connectionString);

return new AppDbContext(optionsBuilder.Options);

}

}

* 1. Файл EFCoreTestsConfigurator.cs

// <copyright file="EFCoreTestsConfigurator.cs" company="Николаев А.М.">

// Copyright (c) Николаев А.М.. All rights reserved.

// </copyright>

using Microsoft.EntityFrameworkCore;

/// <summary>

/// Конфигуратор для создания контекста базы данных для тестов с использованием EF Core.

/// </summary>

public class EFCoreTestsConfigurator

{

/// <summary>

/// Создает и конфигурирует контекст базы данных для тестов.

/// </summary>

/// <returns>Контекст базы данных для тестов.</returns>

public static AppDbContext BuildContextForTest()

{

// Создание опций для контекста базы данных

var options = new DbContextOptionsBuilder<AppDbContext>()

.UseNpgsql("Host=localhost;Username=postgres;Password=;Database=testbd1;")

.Options;

// Создание и конфигурация контекста базы данных

var context = new AppDbContext(options);

context.Database.EnsureCreated();

return context;

}

}

* 1. Файл IRepository.cs

// <copyright file="IRepository.cs" company="Николаев А.М.">

// Copyright (c) Николаев А.М.. All rights reserved.

// </copyright>

namespace ORM.Repositories

{

using System;

using System.Linq;

using System.Linq.Expressions;

/// <summary>

/// Интерфейс для репозиториев.

/// </summary>

/// <typeparam name="TEntity">Тип сущности.</typeparam>

public interface IRepository<TEntity>

{

/// <summary>

/// Получает сущность по идентификатору.

/// </summary>

/// <param name="id">Идентификатор.</param>

/// <returns>Сущность.</returns>

TEntity Get(int id);

/// <summary>

/// Находит сущность по выражению.

/// </summary>

/// <param name="predicate">Выражение для поиска.</param>

/// <returns>Сущность.</returns>

TEntity Find(Expression<Func<TEntity, bool>> predicate);

/// <summary>

/// Получает все сущности.

/// </summary>

/// <returns>Все сущности.</returns>

IQueryable<TEntity> GetAll();

/// <summary>

/// Фильтрует сущности по выражению.

/// </summary>

/// <param name="predicate">Выражение для фильтрации.</param>

/// <returns>Отфильтрованные сущности.</returns>

IQueryable<TEntity> Filter(Expression<Func<TEntity, bool>> predicate);

/// <summary>

/// Сохраняет сущность.

/// </summary>

/// <param name="entity">Сущность для сохранения.</param>

/// <returns>Результат сохранения.</returns>

bool Save(TEntity entity);

}

}

* 1. Файл DocumentMapTests.cs

// <copyright file="DocumentMapTests.cs" company="Николаев А.М.">

// Copyright (c) Николаев А.М.. All rights reserved.

// </copyright>

namespace ORM.Tests

{

using System.Linq;

using Domain;

using Microsoft.EntityFrameworkCore;

using NUnit.Framework;

using ORM.Mappings;

/// <summary>

/// Тесты для проверки корректности отображения сущности <see cref="Document"/> на таблицу и наоборот.

/// </summary>

[TestFixture]

public class DocumentMapTests

{

private AppDbContext context;

/// <summary>

/// Настройка тестового окружения.

/// </summary>

[SetUp]

public void SetUp()

{

this.context = EFCoreTestsConfigurator.BuildContextForTest();

}

/// <summary>

/// Очистка тестового окружения.

/// </summary>

[TearDown]

public void TearDown()

{

this.context.Database.EnsureDeleted();

this.context.Dispose();

}

/// <summary>

/// Тест проверяет, что сущность <see cref="Document"/> корректно сохраняется в базу данных с правильными связями.

/// </summary>

[Test]

public void PersistenceSpecification\_ValidData\_Success()

{

// Arrange

var inventory = new Inventory(1, "Опись 1");

var readingRoom = new ReadingRoom(1, "Читальный зал 1");

var document = new Document(1, "Документ 1", inventory, readingRoom);

this.context.Inventories.Add(inventory);

this.context.ReadingRooms.Add(readingRoom);

this.context.Documents.Add(document);

this.context.SaveChanges();

// Act

var savedDocument = this.context.Documents

.Include(d => d.Inventory)

.Include(d => d.ReadingRoom)

.FirstOrDefault(d => d.Title == "Документ 1");

// Assert

Assert.That(savedDocument, Is.Not.Null);

Assert.Multiple(() =>

{

Assert.That(savedDocument.Title, Is.EqualTo("Документ 1"));

Assert.That(savedDocument.Inventory.Title, Is.EqualTo("Опись 1"));

Assert.That(savedDocument.ReadingRoom.Name, Is.EqualTo("Читальный зал 1"));

});

}

}

}

* 1. Файл InventoryMapTests.cs

// <copyright file="InventoryMapTests.cs" company="Николаев А.М.">

// Copyright (c) Николаев А.М.. All rights reserved.

// </copyright>

namespace ORM.Tests

{

using System.Linq;

using Domain;

using Microsoft.EntityFrameworkCore;

using NUnit.Framework;

/// <summary>

/// Класс для тестирования маппинга <see cref="Inventory"/>.

/// </summary>

[TestFixture]

public class InventoryMapTests

{

/// <summary>

/// Контекст базы данных для тестов.

/// </summary>

private AppDbContext context;

/// <summary>

/// Настройка тестового окружения.

/// </summary>

[SetUp]

public void SetUp()

{

var options = new DbContextOptionsBuilder<AppDbContext>()

.UseInMemoryDatabase(databaseName: "TestDatabase")

.Options;

this.context = new AppDbContext(options);

}

/// <summary>

/// Очистка тестового окружения.

/// </summary>

[TearDown]

public void TearDown()

{

this.context.Dispose();

}

/// <summary>

/// Тест проверяет корректность маппинга для простого случая.

/// </summary>

[Test]

public void PersistenceSpecification\_ValidData\_Success()

{

// Arrange

var inventory = new Inventory(1, "w");

// Act

this.context.Inventories.Add(inventory);

this.context.SaveChanges();

// Assert

var savedInventory = this.context.Inventories.FirstOrDefault(i => i.Title == "w");

Assert.That(savedInventory, Is.Not.Null);

Assert.That(savedInventory.Title, Is.EqualTo("w"));

}

}

}

* 1. Файл ReadingRoomMapTests.cs

// <copyright file="ReadingRoomMapTests.cs" company="Николаев А.М.">

// Copyright (c) Николаев А.М.. All rights reserved.

// </copyright>

namespace ORM.Tests

{

using System.Linq;

using Domain;

using Microsoft.EntityFrameworkCore;

using NUnit.Framework;

/// <summary>

/// Класс для тестирования маппинга <see cref="ReadingRoom"/>.

/// </summary>

[TestFixture]

public class ReadingRoomMapTests

{

private AppDbContext context;

/// <summary>

/// Настройка тестового окружения.

/// </summary>

[SetUp]

public void SetUp()

{

var options = new DbContextOptionsBuilder<AppDbContext>()

.UseInMemoryDatabase(databaseName: "TestDatabase")

.Options;

this.context = new AppDbContext(options);

}

/// <summary>

/// Очистка тестового окружения.

/// </summary>

[TearDown]

public void TearDown()

{

this.context.Dispose();

}

/// <summary>

/// Тест проверяет корректность маппинга для простого случая.

/// </summary>

[Test]

public void PersistenceSpecification\_ValidData\_Success()

{

// Arrange

var readingRoom = new ReadingRoom(1, "Читальный зал 1");

// Act

this.context.ReadingRooms.Add(readingRoom);

this.context.SaveChanges();

// Assert

var savedReadingRoom = this.context.ReadingRooms.FirstOrDefault(r => r.Name == "Читальный зал 1");

Assert.That(savedReadingRoom, Is.Not.Null);

Assert.Multiple(() =>

{

Assert.That(savedReadingRoom.ReadingRoomCode, Is.EqualTo(1));

Assert.That(savedReadingRoom.Name, Is.EqualTo("Читальный зал 1"));

});

}

}

}

Прохождение тестов

Прохождение тестов представлено на рисунке 2.

|  |
| --- |
|  |
| Рисунок 2 – Результаты прохождения тестов |

Полученная ERD

На рисунке 3 приведен полученная ERD-диаграмма.

|  |
| --- |
|  |
| Рисунок 3 – ERD-диаграмма |

Вывод

В ходе выполнения работы были изучены и применены методы и инструменты, которые помогают предотвратить SQL-инъекции при работе с базами данных на языке C#. Особое внимание было уделено технологии ORM (Object-Relational Mapping), которая позволяет сопоставить объекты в программе с записями в базе данных, обеспечивая тем самым более безопасное и удобное взаимодействие с данными. ORM автоматизирует многие процессы, связанные с работой с базой данных, такие как создание запросов, обновление данных и их удаление, что снижает риск возникновения ошибок и уязвимостей, таких как SQL-инъекции.

Список литературы

* + - 1. Васильева М.А., Хобта Д.О., Фильтрация набора данных. Рекомендации по выполнению работы и перечень типовых заданий: Учебно-методическое пособие. Издание второе, исправленное и дополненное–М.:РУТ(МИИТ). 2023.–105с.
      2. Васильева М.А., Меркулов Д.А. Группировка и обобщение данных. Рекомендации по выполнению работы и перечень типовых заданий. Учебно-методическое пособие. М.:РУТ(МИИТ), 2023. 46–с.
      3. Васильева М.А., Ракинцев Н.А. Соединение данных из множества таблиц. Рекомендации по выполнению работы и перечень типовых заданий. Учебно-методическое пособие. М.:РУТ(МИИТ), 2023. 63–с.
      4. Балакина Е.П., Васильева М.А., Филипченко К.М. Информационное обеспечение систем управления. Методические указания к курсовому проектированию. Учебно-методическое пособие. Издание второе, исправленное и дополненное, 2023.102–с.