**Министерство транспорта Российской Федерации**

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

**Институт транспортной техники и систем управления**

**Кафедра «Управление и защита информации»**

ОТЧЁТ

по дисциплине

**«Основы построения защищенных баз данных»**

на тему

**«Архив»**

Выполнил: ст. гр. ТКИ-541  
Ситников Д.Ю.  
Проверил: Васильева М.А.

**Цель работы:**

Научится использовать структуру ORM (Object-Relational Mapping, объектно-реляционное отображение) на языке программирования высокого уровня.

**Постановка задачи:**

Написать на языке С# базу данных для информационного обслуживания сотрудников архива уголовного розыска. В архиве лежат дела за разные годы.

1. Сформулировать предметную область (цель задачи)
2. Реализовать классы предметной области
3. Написать тесты на классы предметной области
4. Подключить stylecop
5. Создать решение DAL(Data access layer)
6. Создать БД
7. Написать конфигурацию создания сессий
8. Написать mock
9. Написать тесты на DAL
10. Создать объекты классов
11. Поместить объекты через ORM в БД

**Выполнение работы:**

**Код программы Programm (основной блок)**

using ClassLibrary;

using NHibernate.Cfg;

namespace ConsoleApp

{

/// Точка входа в программу

internal class Program

{

// private static readonly string? middleName1;

public static object? Settings { get; private set; }

private static void Main()

{

var NameInvestigator = new Investigator(1, "Рижимин", "Петр", " ");

var NameInvestigator2 = new Investigator(2, "Викторов", "Константин", " ");

var NameInvestigator3 = new Investigator(3, "Героев", "Аркадий", " ");

var NameInvestigator4 = new Investigator(4, "Римский", "Даниил", " ");

var NameInvestigator5 = new Investigator(5, "Прямовидов", "Игнат", " ");

var NameInvestigator6 = new Investigator(6, "Гордеев", "Виталий", " ");

var NameConvicted = new Convicted(1, "Ридискин", "Павел", " ");

var NameConvicted2 = new Convicted(2, "Воров", "Кирил", " ");

var NameConvicted3 = new Convicted(3, "Горин", "Артемий", " ");

var NameConvicted4 = new Convicted(4, "Ридискин", "Даниил", " ");

var NameConvicted5 = new Convicted(5, "Воровитов", "Игнат", " ");

var NameConvicted6 = new Convicted(6, "Горлин", "Витя", " ");

var CourtCase = new CourtCase(1, 1208, "Горбушечка", "Убийство", 1977, 1978, 105, middleName1);

var CourtCase2 = new CourtCase(2, 1380, "Хлопушка", "Порча имущества", 1978, 1979, 168, middleName1);

var CourtCase3 = new CourtCase(3, 1204, "Хлебозавод", "Кража со взломом", 1979, 1980, 158, middleName1);

var CourtCase4 = new CourtCase(4, 1206, "Горб", "Убийство", 1977, 1978, 105, middleName1);

var CourtCase5 = new CourtCase(5, 1388, "Хлеба и зрелищ", "Организованный разбой", 1978, 1979, 162, middleName1);

var CourtCase6 = new CourtCase(6, 1111, "Хлебоводы", "Угон", 1979, 1980, 166, middleName1);

Console.WriteLine($"{CourtCase} {NameInvestigator} {NameConvicted}");

ClassLibrary.Settings Settings = new();

Settings.AddDatabaseServer("DESKTOP-3011201");

Settings.AddDatabase("Arhiv");

}

}

}

**Код программы Convicted (блок Convicted)**

using System;

namespace ConsoleApp

{

/// <summary>

/// Осужденный

/// </summary>

public class Convicted : ConvictedBase, IEquatable<Convicted>

{

/// <summary>

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

/// </summary>

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

/// <param name="lastName">Фамилия</param>

/// <param name="firstName">Имя</param>

/// <exception cref="ArgumentOutOfRangeException">В случае если <paramref name="lastName"/> или <paramref name="firstName"/>

/// <see langword="null"/>, пустая строка или строка, содержащая только пробельные символы.

/// </exception>

public Convicted(int id, string lastName, string firstName, string? middleName = null)

{

this.Id = id;

this.MiddleName = middleName ?? throw new ArgumentOutOfRangeException(nameof(lastName));

this.LastName = lastName ?? throw new ArgumentOutOfRangeException(nameof(lastName));

this.FirstName = firstName ?? throw new ArgumentOutOfRangeException(nameof(firstName)); ;

}

/// <summary>

/// Уникальный идентификатор

/// </summary>

public virtual int Id { get; protected set; }

public string MiddleName { get; }

/// <summary>

/// Имя

/// </summary>

public virtual string LastName { get; protected set; }

/// <summary>

/// Фамилия

/// </summary>

public virtual string FirstName { get; protected set; }

bool IEquatable<Convicted>.Equals(Convicted? other)

{

throw new NotImplementedException();

}

public override int GetHashCode()

{

throw new NotImplementedException();

}

public override bool Equals(object obj)

{

return ((IEquatable<Convicted>)this).Equals (obj as Convicted);

}

}

public class ConvictedBase

{

}

}

**Код программы Investigator (блок Investigator)**

using System;

namespace ConsoleApp

{

/// <summary>

/// Следователь

/// </summary>

public class Investigator : IEquatable<Investigator>

{

/// <summary>

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

/// </summary>

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

/// <param name="lastName">Фамилия</param>

/// <param name="firstName">Имя</param>

/// <exception cref="ArgumentOutOfRangeException">В случае если <paramref name="lastName"/> или <paramref name="firstName"/>

/// <see langword="null"/>, пустая строка или строка, содержащая только пробельные символы.

/// </exception>

public Investigator(int id, string lastName, string firstName, string middleName = null)

{

this.Id = id;

MiddleName = middleName;

this.LastName = lastName ?? throw new ArgumentOutOfRangeException(nameof(lastName));

this.FirstName = firstName ?? throw new ArgumentOutOfRangeException(nameof(firstName)); ;

}

/// <summary>

/// Уникальный идентификатор

/// </summary>

public virtual int Id { get; protected set; }

public string MiddleName { get; }

/// <summary>

/// Имя

/// </summary>

public virtual string LastName { get; protected set; }

/// <summary>

/// Фамилия

/// </summary>

public virtual string FirstName { get; protected set; }

//public virtual string FullName == $"{this.Last.Name} {this.FirstName}.".Trim();

public bool Equals(Investigator other)

{

throw new NotImplementedException();

}

public override bool Equals(object obj) => Equals(obj as Investigator);

public override int GetHashCode()

{

throw new NotImplementedException();

}

}

}

**Код программы CourtCase (блок Дело)**

using System;

using System.Collections.Generic;

namespace ConsoleApp

{

/// <summary>

/// Уголовные дела

/// </summary>

public class CourtCase

{

/// <summary>

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

/// </summary>

/// 1, "1208", "Горбушечка", "Убийство", "1977", "1978", "105"

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

/// <param name="article">Номер статьи</param>

/// <param name="nameCase">Название дела</param>

/// <param name="content">Содержание</param>

/// <param name="YearOfOpening">Год открытия</param>

/// <param name="YearOfClose">Год закрытия</param>

/// <param name="NamberCase">Номер</param>

/// <exception cref="ArgumentOutOfRangeException">В случае если <paramref name="lastName"/> или <paramref name="firstName"/>

/// <see langword="null"/>, пустая строка или строка, содержащая только пробельные символы.

/// </exception>

public CourtCase(int id,

int article,

string NameCase,

string Content,

int YearOfOpening,

int YearOfClose,

int NamberCase,

string middleName = null)

{

Id = id;

Article = article;

this.NameCase = NameCase;

this.Content = Content;

this.YearOfOpening = YearOfOpening;

this.YearOfClose = YearOfClose;

this.NamberCase = NamberCase;

MiddleName1 = middleName;

}

public int Id { get; private set; }

public int Article { get; private set; }

public object NameCase { get; private set; }

public object Content { get; private set; }

public object YearOfOpening { get; private set; }

public object YearOfClose { get; private set; }

public int NamberCase { get; private set; }

public string MiddleName { get; }

public string MiddleName1 { get; }

public override bool Equals(object obj)

{

return !(!(obj is CourtCase @case) ||

Id != @case.Id ||

Article != @case.Article ||

!EqualityComparer<object>.Default.Equals(NameCase, @case.NameCase) ||

!EqualityComparer<object>.Default.Equals(Content, @case.Content) ||

!EqualityComparer<object>.Default.Equals(YearOfOpening, @case.YearOfOpening) ||

!EqualityComparer<object>.Default.Equals(YearOfClose, @case.YearOfClose) ||

NamberCase != @case.NamberCase);

}

public override int GetHashCode()

{

throw new NotImplementedException();

}

}

public class StringNameCase

{

}

}

**Код программы Configuration (блок Configuration)**

using System.Reflection;

using FluentNHibernate.Cfg;

using FluentNHibernate.Cfg.Db;

using NHibernate;

using NHibernate.Linq;

using NHibernate.Cfg;

using NHibernate.Tool.hbm2ddl;

namespace Arhiv;

public static class Configuration

{

private static readonly FluentConfiguration? fluentConfiguration;

private static ISessionFactory GetSessionFactory(

Settings settings,

Assembly? assembly = null, //сборка

bool showSql = false)

{

return GetConfiguration();

}

private static ISessionFactory GetConfiguration()

{

throw new NotImplementedException();

}

private static FluentConfiguration GetConfiguration(

Settings settings,

Assembly assembly,

FluentConfiguration fluentConfiguration, bool showSql = false)

{

if (fluentConfiguration is null)

{

var databaseConfiguration =

MsSqlConfiguration.MsSql2012.ConnectionString(

x => x

.Server(settings.GetDatabaseServer())

.Database(settings.GetDatabaseName())

.TrustedConnection());

if (showSql)

{

databaseConfiguration = databaseConfiguration

.ShowSql()

.FormatSql();

}

fluentConfiguration = Fluently.Configure()

.Database(databaseConfiguration)

.Mappings(m => m.FluentMappings.AddFromAssembly(assembly))

.ExposeConfiguration(BuildSchema);

}

return fluentConfiguration;

}

private static void BuildSchema(NHibernate.Cfg.Configuration configuration)

{

new SchemaExport(configuration).Execute(true, true, false);

}

}

**Код программы Settings (блок Settings)**

using System;

///<summary>

///описние класса

/// </summary>

namespace ClassLibrary

{

public sealed class Settings

{

private string databaseServerName;

private string databaseName;

public void AddDabaseServer(string serverName)

{

this.databaseServerName = serverName;

}

public string GetDatabaseServer()

{

return this.databaseServerName;

}

public void AddDatabase(string databaseName)

{

this.databaseName = databaseName;

}

public string GetDatabaseName()

{

return this.databaseName;

}

public void AddDatabaseServer(string v)

{

throw new NotImplementedException();

}

}

}

**Код программы TestsMappings (блок Mappings)**

namespace ArhivTests

{

using NHibernate;

using NUnit.Framework;

public class MappingTests

{

protected ISession Session { get; private set; }

[SetUp]

public void Setup()

{

this.Session = ConfiguratorTests.BuildSessionForTest();

}

[TearDown]

public void TearDown()

{

this.Session?.Dispose();

}

}

}

**Код программы ConfiguratorTests (блок Configurator)**

namespace ArhivTests

{

using System.Reflection;

using FluentNHibernate.Cfg;

using FluentNHibernate.Cfg.Db;

using NHibernate;

using NHibernate.Cfg;

using NHibernate.Tool.hbm2ddl;

public class ConfiguratorTests

{

private static Configuration? configuration;

public static ISessionFactory GetSessionFactory(

Assembly? assembly = null,

bool showSql = false)

{

var databaseConfiguration =

SQLiteConfiguration.Standard.InMemory();

if (showSql)

{

databaseConfiguration = databaseConfiguration

.ShowSql()

.FormatSql();

}

return Fluently.Configure()

.Database(databaseConfiguration)

.Mappings(m => m.FluentMappings.AddFromAssembly(assembly ??

Assembly.GetExecutingAssembly()))

.ExposeConfiguration(c => configuration = c)

.BuildSessionFactory();

}

public static ISession BuildSessionForTest(bool showSql = true)

{

var session = GetSessionFactory(showSql: showSql).OpenSession();

new SchemaExport(configuration)

.Execute(true, true, false, session.Connection, null);

return session;

}

}

}

**Описание классов:**

**Таблица 1 – Схема отношения** **Сущность\_1** (**Entity\_1**)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Первичный ключ | ID | Целое число | первичный ключ |
| Следователь | ID\_Investigator | Целое число | обязательное поле |
| Осужденный | ID\_Convicted | Целое число | обязательное поле |
| Номер дела | Number | nvarchar(6) | уникальное, обязательное поле |
| Название | Name | nvarchar(10) | уникальное, обязательное поле |
| Содержание | Content | ntext | уникальное, обязательное поле |
| Год открытия дела | Yea case open | nvarchar(10) | обязательное поле |
| Год окончания дела | Yes case of close | nvarchar(10) | обязательное поле |

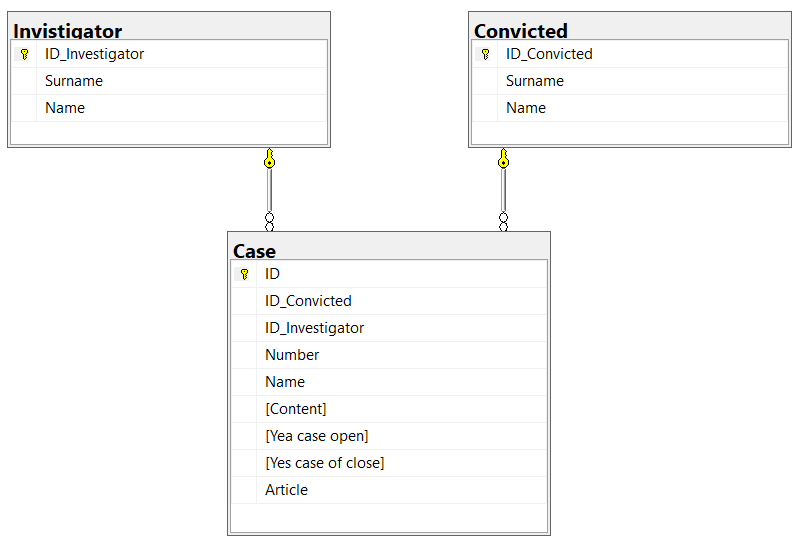
**Продолжение таблицы 1 – Схема отношения** **Сущность\_1** (**Entity\_1**)

|  |  |  |  |
| --- | --- | --- | --- |
| Год окончания дела | Yes case of close | nvarchar(10) | обязательное поле |
| Статья | Article | nvarchar(10) | обязательное поле |

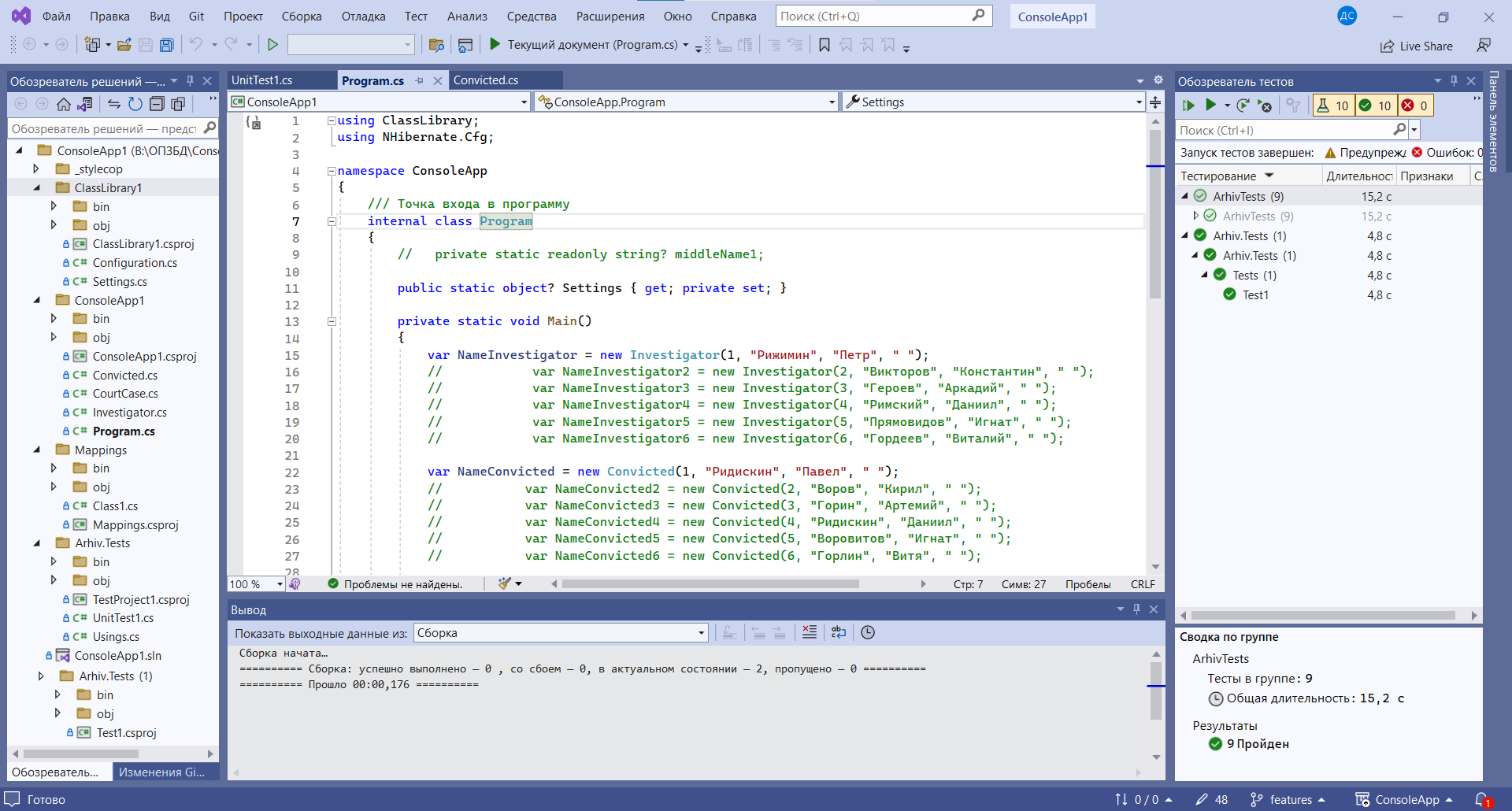
Таблица 2 – Схема отношения Сущность\_2 (Entity\_2)

|  |  |  |  |
| --- | --- | --- | --- |
| *Содержание поля* | *Имя поля* | *Тип, длина* | *Примечания* |
| Первичный ключ | ID | Целое число | первичный ключ |
| Фамилия | Surname | nvarchar(10) | обязательное поле |
| Имя | Name | nvarchar(10) | обязательное поле |

**Ход выполнения:**

**ER – диаграмма**

**Тесты**



**Вывод:**

Была произведена работа по созданию базы данных с помощью технологии ОРМ на языке C#.