ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»  
(РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

КУРСОВАЯ РАБОТА

По дисциплине «Компьютерные технологии управления в технических системах»

Выполнил: ст. гр. ТУУ – 171

Гаврилов Е.А.

Проверил: к.т.н., доц.

Васильева М.А.

Москва 2023

Оглавление

[Цель курсового проекта 2](#_Toc124503117)

[Предметная область 2](#_Toc124503118)

[Логическое проектирование реляционной БД 2](#_Toc124503119)

[Создание классов, описывающих предметную область 2](#_Toc124503120)

[Тестирование классов из проекта MusicStore.Core 7](#_Toc124503121)

[Создание слоя для взаимодействия с данными 10](#_Toc124503122)

[Тестирование классов из проекта DataAccessLayer 16](#_Toc124503123)

[Проверка работоспособности 19](#_Toc124503124)

[Вывод 21](#_Toc124503125)

## **Цель курсового проекта**

Целью курсового проекта является построение ORM, закрепление навыков программирования.

## Предметная область

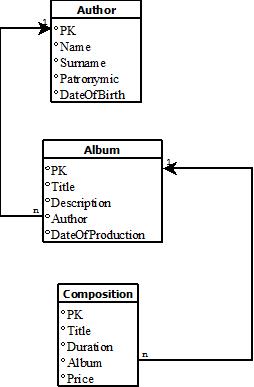
В качестве предметной области был выбран музыкальный магазин. База данных будет состоять из 3 таблиц(сущностей): альбом, автор, композиция.

Альбом – название, описание, автор, дата выпуска.

Автор – имя, фамилия, отчество, дата рождения.

Композиция – название, продолжительность, альбом, цена.

## Логическое проектирование реляционной БД



**Рисунок 1** – ER диаграмма будущей БД

## Создание классов, описывающих предметную область

Создадим проект MusicStore.Core и дальнейшую работу будем проводить в нём.

Листинг файла Album.cs:

// <copyright file="Album.cs" company="Команда">

// Copyright (c) Команда 2022.

// </copyright>

namespace MusicStore.Core

{

/// <summary>

/// Альбом.

/// </summary>

public class Album : IEquatable<Album>

{

/// <summary>

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

/// </summary>

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

/// <param name="description"> Описание. </param>

/// <param name="author"> Автор. </param>

/// <param name="dateOfProduction"> Дата производства. </param>

public Album(string title, string description, Author author, DateTime dateOfProduction)

{

this.Id = Guid.NewGuid();

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

this.Description = description ?? throw new ArgumentNullException(nameof(description));

this.Author = author ?? throw new ArgumentNullException(nameof(author));

this.DateOfProduction = dateOfProduction;

}

/// <summary>

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

/// Пустой конструктор для работы с ORM.

/// </summary>

[Obsolete("For ORM", true)]

protected Album()

{

}

/// <summary>

/// Идентификатор.

/// </summary>

public virtual Guid Id { get; protected set; }

/// <summary>

/// Название.

/// </summary>

public virtual string Title { get; protected set; }

/// <summary>

/// Описание.

/// </summary>

public virtual string Description { get; protected set; }

/// <summary>

/// Автор.

/// </summary>

public virtual Author Author { get; protected set; }

/// <summary>

/// Дата производства.

/// </summary>

public virtual DateTime DateOfProduction { get; protected set; }

/// <summary>

/// Композиции.

/// </summary>

private ISet<Composition> Compositions { get; set; } = new HashSet<Composition>();

/// <summary>

/// Добавление композиции альбому.

/// </summary>

/// <param name="composition"> Композиция. </param>

public virtual void AddCompositionToAlbum(Composition composition)

{

this.Compositions.Add(composition);

composition.Album = this;

}

/// <inheritdoc/>

public override int GetHashCode()

{

return this.Id.GetHashCode();

}

/// <inheritdoc/>

public virtual bool Equals(Album? other)

{

return Equals(this.Id, other?.Id);

}

}

}

Листинг файла Author.cs:

// <copyright file="Author.cs" company="Команда">

// Copyright (c) Команда 2022.

// </copyright>

namespace MusicStore.Core

{

/// <summary>

/// Автор.

/// </summary>

public class Author : IEquatable<Author>

{

/// <summary>

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

/// </summary>

/// <param name="name"> Имя. </param>

/// <param name="surname"> Фамилия. </param>

/// <param name="patronymic"> Отчество. </param>

/// <param name="dateOfBirth"> Дата рождения. </param>

public Author(string name, string surname, string patronymic, DateTime dateOfBirth)

{

this.Id = Guid.NewGuid();

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

this.Surname = surname ?? throw new ArgumentNullException(nameof(surname));

this.Patronymic = patronymic; // Может не быть

this.DateOfBirth = dateOfBirth;

}

/// <summary>

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

/// Пустой конструктор для работы с ORM.

/// </summary>

[Obsolete("For ORM", true)]

protected Author()

{

}

/// <summary>

/// Идентификатор.

/// </summary>

public virtual Guid Id { get; protected set; }

/// <summary>

/// Имя.

/// </summary>

public virtual string Name { get; protected set; }

/// <summary>

/// Фамилия.

/// </summary>

public virtual string Surname { get; protected set; }

/// <summary>

/// Отчество.

/// </summary>

public virtual string Patronymic { get; protected set; }

/// <summary>

/// Дата рождения.

/// </summary>

public virtual DateTime DateOfBirth { get; protected set; }

/// <inheritdoc/>

public override int GetHashCode()

{

return this.Id.GetHashCode();

}

/// <inheritdoc/>

public virtual bool Equals(Author? other)

{

return Equals(this.Id, other?.Id);

}

}

}

Листинг файла Composition.cs:

// <copyright file="Composition.cs" company="Команда">

// Copyright (c) Команда 2022.

// </copyright>

namespace MusicStore.Core

{

/// <summary>

/// Композиция.

/// </summary>

public class Composition : IEquatable<Composition>

{

/// <summary>

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

/// </summary>

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

/// <param name="duration"> Продолжительность. </param>

/// <param name="album"> Альбом. </param>

/// <param name="price"> Цена. </param>

public Composition(string title, int duration, Album album, decimal price)

{

if (duration <= 0)

{

throw new ArgumentOutOfRangeException(nameof(duration));

}

else

{

this.Duration = duration;

}

if (price <= 0)

{

throw new ArgumentOutOfRangeException(nameof(price));

}

else

{

this.Price = price;

}

this.Id = Guid.NewGuid();

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

this.Duration = duration;

this.Album = album;

this.Price = price;

}

/// <summary>

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

/// Пустой конструктор для работы с ORM.

/// </summary>

[Obsolete("For ORM", true)]

protected Composition()

{

}

/// <summary>

/// Серийный номер.

/// </summary>

public virtual Guid Id { get; protected set; }

/// <summary>

/// Название.

/// </summary>

public virtual string Title { get; protected set; }

/// <summary>

/// Продолжительность.

/// </summary>

public virtual int Duration { get; protected set; }

/// <summary>

/// Альбом.

/// </summary>

public virtual Album Album { get; set; }

/// <summary>

/// Цена.

/// </summary>

public virtual decimal Price { get; protected set; }

/// <inheritdoc/>

public virtual bool Equals(Composition? other)

{

return Equals(this.Id, other?.Id);

}

/// <inheritdoc/>

public override int GetHashCode()

{

return this.Id.GetHashCode();

}

}

}

## Тестирование классов из проекта MusicStore.Core

Создадим проект MusicStore.Core.Tests и дальнейшую работу будем проводить в нём.

Листинг файла AlbumTests.cs:

// <copyright file="AlbumTests.cs" company="Команда">

// Copyright (c) Команда 2022.

// </copyright>

namespace MusicStore.Core.Tests

{

using MusicStore.Core;

/// <summary>

/// Тесты для класса Album.

/// </summary>

[TestFixture]

public class AlbumTests

{

/// <summary>

/// Тест на конструктор с пустым названием.

/// </summary>

[Test]

public void ThrowAnExceptionOnNullTitle()

{

Author author = new Author("nornal name", "nornal surname", "nornal patronymic", new DateTime(1999, 12, 4));

Assert.Throws<ArgumentNullException>(() => \_ = new Album(null, "normal description", author, new DateTime(1922, 2, 5)));

}

/// <summary>

/// Тест на конструктор с пустым автором.

/// </summary>

[Test]

public void ThrowAnExceptionOnNullAuthor()

{

Assert.Throws<ArgumentNullException>(() => \_ = new Album("normal title", "normal description", null, new DateTime(1922, 2, 5)));

}

/// <summary>

/// Тест на конструктор с пустым описанием.

/// </summary>

[Test]

public void ThrowAnExceptionOnNullDescription()

{

Author author = new Author("nornal name", "nornal surname", "nornal patronymic", new DateTime(1999, 12, 4));

Assert.Throws<ArgumentNullException>(() => \_ = new Album("normal title", null, author, new DateTime(1922, 2, 5)));

}

/// <summary>

/// Тест на конструктор с нормальным данными.

/// </summary>

[Test]

public void NormalAlbumCreation()

{

Author author = new Author("nornal name", "nornal surname", "nornal patronymic", new DateTime(1999, 12, 4));

Assert.DoesNotThrow(() => \_ = new Album("normal title", "normal description", author, new DateTime(1900, 2, 1)));

}

}

}

Листинг файла AuthorTests.cs:

// <copyright file="AuthorTests.cs" company="Команда">

// Copyright (c) Команда 2022.

// </copyright>

namespace MusicStore.Core.Tests

{

using MusicStore.Core;

/// <summary>

/// Тесты для класса Author.

/// </summary>

[TestFixture]

public class AuthorTests

{

/// <summary>

/// Тест на конструктор с пустым именем.

/// </summary>

[Test]

public void ThrowAnExceptionOnNullName()

{

Assert.Throws<ArgumentNullException>(() => \_ = new Author(null, "normal surname", "normal patronymic", new DateTime(1922, 2, 5)));

}

/// <summary>

/// Тест на конструктор с пустой фамилий.

/// </summary>

[Test]

public void ThrowAnExceptionOnNullSurname()

{

Assert.Throws<ArgumentNullException>(() => \_ = new Author("nornal name", null, "normal patronymic", new DateTime(1922, 2, 5)));

}

/// <summary>

/// Тест на конструктор с нормальными данными.

/// </summary>

[Test]

public void NormalAuthorCreation()

{

Assert.DoesNotThrow(() => \_ = new Author("nornal name", "nornal surname", "normal patronymic", new DateTime(1922, 2, 5)));

}

}

}

Листинг файла CompositionTests.cs:

// <copyright file="CompositionTests.cs" company="Команда">

// Copyright (c) Команда 2022.

// </copyright>

namespace MusicStore.Core.Test

{

using MusicStore.Core;

/// <summary>

/// Тесты для класса Composition.

/// </summary>

[TestFixture]

public class CompositionTests

{

/// <summary>

/// Тест на конструктор с пустым названием.

/// </summary>

[Test]

public void ThrowAnExceptionOnNullTitle()

{

Assert.Throws<ArgumentNullException>(() => \_ = new Composition(null, 123, null, 12.1m));

}

/// <summary>

/// Тест на конструктор с нулевой ценой.

/// </summary>

[Test]

public void ThrowAnExceptionOnZeroPrice()

{

Assert.Throws<ArgumentOutOfRangeException>(() => \_ = new Composition("ok title", 123, null, 0));

}

/// <summary>

/// Тест на конструктор с ценой меньше нуля.

/// </summary>

[Test]

public void ThrowAnExceptionOnLessZeroPrice()

{

Assert.Throws<ArgumentOutOfRangeException>(() => \_ = new Composition("ok title", 123, null, -12.1m));

}

/// <summary>

/// Тест на конструктор с нормальными данными.

/// </summary>

[Test]

public void NormalCompositionCreation()

{

Author author = new Author("nornal name", "nornal surname", "nornal patronymic", new DateTime(1999, 12, 4));

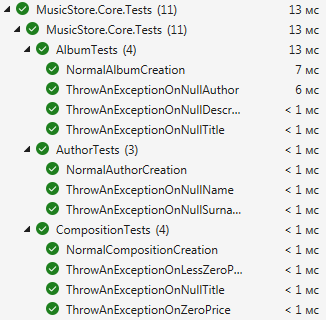
Album album = new Album("normal title", "nornal author", author, new DateTime(1900, 2, 1));

Assert.DoesNotThrow(() => \_ = new Composition("ok title", 123, album, 12.1m));

}

}

}



**Рисунок 2** – Результат запуска тестов

Все тесты успешно закончили свою работу. Дальше разработаем классы для взаимодействия со слоем данных.

## Создание слоя для взаимодействия с данными

Создадим проект DataAccessLayer и дальнейшую работу будем проводить в нём.

Листинг файла Settings.cs, хранящего данные для подключения к БД:

// <copyright file="Settings.cs" company="Команда">

// Copyright (c) Команда 2022.

// </copyright>

namespace DataAccessLayer

{

/// <summary>

/// Настройки для СУБД.

/// </summary>

public sealed class Settings

{

private string databaseHost;

private string user;

private string password;

private string databaseName;

/// <summary>

/// Сеттер для databaseHost.

/// </summary>

/// <param name="databaseHost"> Хост СУБД. </param>

public void AddDatabaseHost(string databaseHost)

{

this.databaseHost = databaseHost;

}

/// <summary>

/// Геттер для databaseHost.

/// </summary>

/// <returns>Строку хоста СУБД.</returns>

public string GetDatabaseHost()

{

return this.databaseHost;

}

/// <summary>

/// Сеттер для user.

/// </summary>

/// <param name="user">Пользователь СУБД.</param>

public void AddUser(string user)

{

this.user = user;

}

/// <summary>

/// Геттер для user.

/// </summary>

/// <returns>Пользователь СУБД.</returns>

public string GetUser()

{

return this.user;

}

/// <summary>

/// Сеттер для password.

/// </summary>

/// <param name="password">Пароль пользователя СУБД.</param>

public void AddPassword(string password)

{

this.password = password;

}

/// <summary>

/// Геттер для password.

/// </summary>

/// <returns>Пароль пользователя СУБД.</returns>

public string GetPassword()

{

return this.password;

}

/// <summary>

/// Сеттер для databaseName.

/// </summary>

/// <param name="databaseName">Название БД.</param>

public void AddDatabaseName(string databaseName)

{

this.databaseName = databaseName;

}

/// <summary>

/// Геттер для databaseName.

/// </summary>

/// <returns>Название БД.</returns>

public string GetDatabaseName()

{

return this.databaseName;

}

}

}

Листинг файла FluentNHibernateConfigurator.cs, содержащий конфигуратор FluentNHibernate и фабрику сессий:

// <copyright file="FluentNHibernateConfigurator.cs" company="Команда">

// Copyright (c) Команда 2022.

// </copyright>

namespace DataAccessLayer.ORM

{

using System.Reflection;

using FluentNHibernate.Cfg;

using FluentNHibernate.Cfg.Db;

using NHibernate;

using NHibernate.Tool.hbm2ddl;

/// <summary>

/// FluentNHibernate конфигуратор.

/// </summary>

public static class FluentNHibernateConfigurator

{

private static FluentConfiguration fluentConfiguration;

/// <summary>

/// Создание фабрики сессий.

/// </summary>

/// <param name="settings"> Параметры СУБД. </param>

/// <param name="assembly"> Целевая сборка. </param>

/// <param name="showSql"> Показывать генерируемый SQL-код. </param>

/// <returns> Фабрику сессий. </returns>

public static ISessionFactory GetSessionFactory(

Settings settings,

Assembly assembly = null,

bool showSql = false)

{

return GetConfiguration(settings, assembly ?? Assembly.GetExecutingAssembly(), showSql)

.BuildSessionFactory();

}

/// <summary>

/// Создание конфигурации FluentNHibernate.

/// </summary>

/// <param name="settings"> Параметры СУБД. </param>

/// <param name="assembly"> Целевая сборка. </param>

/// <param name="showSql"> Показывать генерируемый SQL-код. </param>

/// <returns> Конфигурация FluentNHibernate. </returns>

private static FluentConfiguration GetConfiguration(

Settings settings,

Assembly assembly,

bool showSql = false)

{

if (fluentConfiguration is null)

{

var databaseConfiguration = MsSqlConfiguration.MsSql2012.ConnectionString(

x => x

.Server(settings.GetDatabaseHost())

.Username(settings.GetUser())

.Password(settings.GetPassword())

.Database(settings.GetDatabaseName()));

if (showSql)

{

databaseConfiguration = databaseConfiguration.ShowSql().FormatSql();

}

fluentConfiguration = Fluently.Configure()

.Database(databaseConfiguration)

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

.ExposeConfiguration(BuildSchema);

}

return fluentConfiguration;

}

/// <summary>

/// Создание схемы по конфигурации.

/// </summary>

/// <param name="configuration"> Конфигурация FluentNHibernate. </param>

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

{

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

}

}

}

Листинг файла TestConfigurator.cs, содержащий конфигуратор сессий для тестов и фабрику сессий:

// <copyright file="TestsConfigurator.cs" company="Команда">

// Copyright (c) Команда 2022.

// </copyright>

namespace Library.DataAccess

{

using System.Reflection;

using FluentNHibernate.Cfg;

using FluentNHibernate.Cfg.Db;

using NHibernate;

using NHibernate.Cfg;

using NHibernate.Tool.hbm2ddl;

/// <summary>

/// Класс для настройки соединения для тестов.

/// </summary>

public class TestsConfigurator

{

private static Configuration configuration;

/// <summary>

/// Создание фабрики сессий.

/// </summary>

/// <param name="assembly"> Целевая сборка. </param>

/// <param name="showSql"> Показывать генерируемый SQL-код. </param>

/// <returns> Фабрику сессий. </returns>

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();

}

/// <summary>

/// Генерируется сессия для модульных тестов.

/// </summary>

/// <param name="showSql"> Показывать генерируемый SQL-код. </param>

/// <returns> Сессия подключения к тестовой БД. </returns>

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;

}

}

}

Листинг файла AlbumMap.cs, содержащий маппинги класса Album:

// <copyright file="AlbumMap.cs" company="Команда">

// Copyright (c) Команда 2022.

// </copyright>

namespace DataAccessLayer.ORM

{

using FluentNHibernate.Mapping;

using MusicStore.Core;

/// <summary>

/// Mapping на <see cref="Author"/>.

/// </summary>

public class AlbumMap : ClassMap<Album>

{

/// <summary>

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

/// </summary>

public AlbumMap()

{

this.Table("Album");

this.Id(x => x.Id);

this.Map(x => x.Title)

.Not.Nullable().Length(50);

this.Map(x => x.Description)

.Not.Nullable().Length(50);

this.References(x => x.Author)

.Not.Nullable();

this.Map(x => x.DateOfProduction);

}

}

}

Листинг файла AuthorMap.cs, содержащий маппинги класса Author:

// <copyright file="AuthorMap.cs" company="Команда">

// Copyright (c) Команда 2022.

// </copyright>

namespace DataAccessLayer.ORM

{

using FluentNHibernate.Mapping;

using MusicStore.Core;

/// <summary>

/// Mapping на <see cref="Author"/>.

/// </summary>

public class AuthorMap : ClassMap<Author>

{

/// <summary>

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

/// </summary>

public AuthorMap()

{

this.Table("Author");

this.Id(x => x.Id);

this.Map(x => x.Name)

.Not.Nullable().Length(50);

this.Map(x => x.Surname)

.Not.Nullable().Length(50);

this.Map(x => x.Patronymic)

.Length(50);

this.Map(x => x.DateOfBirth);

}

}

}

Листинг файла CompositionMap.cs, содержащий маппинги класса Composition:

// <copyright file="CompositionMap.cs" company="Команда">

// Copyright (c) Команда 2022.

// </copyright>

namespace DataAccessLayer.ORM

{

using FluentNHibernate.Mapping;

using MusicStore.Core;

/// <summary>

/// Mapping на <see cref="Author"/>.

/// </summary>

public class CompositionMap : ClassMap<Composition>

{

/// <summary>

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

/// </summary>

public CompositionMap()

{

this.Table("Composition");

this.Id(x => x.Id);

this.Map(x => x.Title)

.Not.Nullable().Length(50);

this.Map(x => x.Duration);

this.References(x => x.Album);

this.Map(x => x.Price);

}

}

}

## Тестирование классов из проекта DataAccessLayer

Создадим проект ORM.Tests и дальнейшую работу будем проводить в нём.

Листинг файла BaseMapTests.cs, содержащий класс от которого будут наследовать все остальные классы:

// <copyright file="BaseMapTests.cs" company="Команда">

// Copyright (c) Команда 2022.

// </copyright>

namespace ORM.Tests

{

using Library.DataAccess;

using NHibernate;

using NUnit.Framework;

/// <summary>

/// Базовый класс для создания всех последующих тестов, связанных с работой с бд(sqlite).

/// </summary>

public class BaseMapTests

{

/// <summary>

/// Сессия для работы с sqlite.

/// </summary>

protected ISession Session { get; private set; }

/// <summary>

/// Создание сессии sqlite с конфигурацией по умолчанию.

/// </summary>

[SetUp]

public void SetUp()

{

this.Session = TestsConfigurator.BuildSessionForTest();

}

/// <summary>

/// Закрытие сессии, когда тестирование окончено.

/// </summary>

[TearDown]

public void TearDown()

{

this.Session?.Dispose();

}

}

}

Листинг файла AlbumMapTests.cs, который содержит тестирование маппингов AlbumMap:

// <copyright file="AlbumMapTests.cs" company="Команда">

// Copyright (c) Команда 2022.

// </copyright>

namespace ORM.Tests

{

using FluentNHibernate.Testing;

using MusicStore.Core;

using NUnit.Framework;

/// <summary>

/// Тесты маппингов для класса Album.

/// </summary>

[TestFixture]

internal class AlbumMapTests : BaseMapTests

{

/// <summary>

/// Тест правильной работы маппинга.

/// </summary>

[Test]

public void PersistenceSpecification\_ValidData\_Success()

{

// arrange

Author author = new Author("nornal name", "nornal surname", "nornal patronymic", new DateTime(1999, 12, 4));

this.Session.Save(author);

var album = new Album("normal title", "normal description", author, new DateTime(1900, 2, 1));

// act & assert

new PersistenceSpecification<Album>(this.Session)

.VerifyTheMappings(album);

}

}

}

Листинг файла AuthorMapTests.cs, который содержит тестирование маппингов AuthorMap:

// <copyright file="AuthorTests.cs" company="Команда">

// Copyright (c) Команда 2022.

// </copyright>

namespace MusicStore.Core.Tests

{

using MusicStore.Core;

/// <summary>

/// Тесты для класса Author.

/// </summary>

[TestFixture]

public class AuthorTests

{

/// <summary>

/// Тест на конструктор с пустым именем.

/// </summary>

[Test]

public void ThrowAnExceptionOnNullName()

{

Assert.Throws<ArgumentNullException>(() => \_ = new Author(null, "normal surname", "normal patronymic", new DateTime(1922, 2, 5)));

}

/// <summary>

/// Тест на конструктор с пустой фамилий.

/// </summary>

[Test]

public void ThrowAnExceptionOnNullSurname()

{

Assert.Throws<ArgumentNullException>(() => \_ = new Author("nornal name", null, "normal patronymic", new DateTime(1922, 2, 5)));

}

/// <summary>

/// Тест на конструктор с нормальными данными.

/// </summary>

[Test]

public void NormalAuthorCreation()

{

Assert.DoesNotThrow(() => \_ = new Author("nornal name", "nornal surname", "normal patronymic", new DateTime(1922, 2, 5)));

}

}

}

Листинг файла CompositionMapTests.cs, который содержит тестирование маппингов CompositionMap:

// <copyright file="CompositionTests.cs" company="Команда">

// Copyright (c) Команда 2022.

// </copyright>

namespace MusicStore.Core.Tests

{

using MusicStore.Core;

/// <summary>

/// Тесты для класса Composition.

/// </summary>

[TestFixture]

public class CompositionTests

{

/// <summary>

/// Тест на конструктор с пустым названием.

/// </summary>

[Test]

public void ThrowAnExceptionOnNullTitle()

{

Assert.Throws<ArgumentNullException>(() => \_ = new Composition(null, 123, null, 12.1m));

}

/// <summary>

/// Тест на конструктор с нулевой ценой.

/// </summary>

[Test]

public void ThrowAnExceptionOnZeroPrice()

{

Assert.Throws<ArgumentOutOfRangeException>(() => \_ = new Composition("ok title", 123, null, 0));

}

/// <summary>

/// Тест на конструктор с ценой меньше нуля.

/// </summary>

[Test]

public void ThrowAnExceptionOnLessZeroPrice()

{

Assert.Throws<ArgumentOutOfRangeException>(() => \_ = new Composition("ok title", 123, null, -12.1m));

}

/// <summary>

/// Тест на конструктор с нормальными данными.

/// </summary>

[Test]

public void NormalCompositionCreation()

{

Author author = new Author("nornal name", "nornal surname", "nornal patronymic", new DateTime(1999, 12, 4));

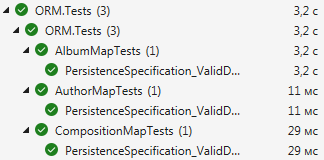
Album album = new Album("normal title", "nornal author", author, new DateTime(1900, 2, 1));

Assert.DoesNotThrow(() => \_ = new Composition("ok title", 123, album, 12.1m));

}

}

}



**Рисунок 3** – Результат запуска тестов

## Проверка работоспособности

Создадим проект Demo и дальнейшую работу будем проводить в нём.

Листинг файла Program.cs, в котором происходит взаимодействие с БД.

// <copyright file="Program.cs" company="Команда">

// Copyright (c) Команда 2022.

// </copyright>

namespace Demo

{

using System;

using DataAccessLayer;

using DataAccessLayer.ORM;

using MusicStore.Core;

/// <summary>

/// Исполняемый файл.

/// </summary>

internal class Program

{

/// <summary>

/// Точка входа в программу.

/// </summary>

private static void Main()

{

var author = new Author("Kanye", "West", "", new DateTime(1977, 6, 8));

var album = new Album("Donda", "new album", author, new DateTime(2021, 8, 29));

var composition = new Composition("ok title", 123, album, 12.1m);

Console.WriteLine(author);

Console.WriteLine(album);

Console.WriteLine(composition);

var settings = new Settings();

settings.AddDatabaseHost("127.0.0.1");

settings.AddUser("SA");

settings.AddPassword("Jekan1337.");

settings.AddDatabaseName("orm\_test");

using var sessionFactory = FluentNHibernateConfigurator

.GetSessionFactory(settings, showSql: true);

using (var session = sessionFactory.OpenSession())

{

session.Save(author);

session.Save(album); ;

session.Save(composition);

session.Flush();

}

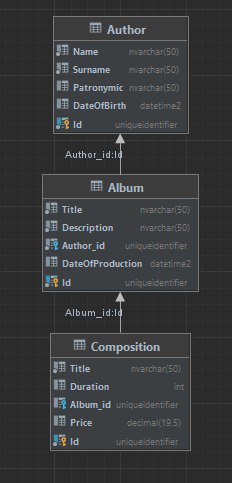
}

}

}

После запуска программы, в указанной БД были создана таблицы.

ER диаграмма полученной БД была получена с помощью ПО DataGrip.



**Рисунок 4** – ER диаграмма будущей БД

## Вывод

Была построена ORM, закреплены навыки программирования.