Министерство транспорта Российской Федерации

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

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

КУРСОВОЙ ПРОЕКТ

по дисциплине

**«Основы построения защищенных баз данных»**

**на тему**  
**«Кинотеатр»**

Выполнили: ст. гр. ТКИ-441

Бабич И.С.

Кирюшин Н.A.

Шибзухова Д.А.

Проверил: доц., к.т.н.

Васильева М. А.

Москва 2024

Оглавление

[Цель курсового проекта 2](#_Toc854028219)

[Задание на курсовой проект 3](#_Toc1141465586)

[UML-Диаграмма 3](#_Toc457321642)

[Код программы 4](#_Toc1404026013)

[Прохождение тестов 16](#_Toc1876381724)

[Настройка миграции 17](#_Toc1638469354)

[ER-диаграмма 17](#_Toc441963184)

[Список литературы 17](#_Toc1668853084)

# Цель курсового проекта

Изучить современные технологии ORM, разработать приложение с базой данных, умеющее отрабатывать операции CRUD.

Стек технологий: язык программирования C#, ORM – EF Core, NUnit, PostgreSQL. stylecop

# Задание на курсовой проект

1. Выбрать предметную область.

В соответствии с заданием (выбрать предметную область самостоятельно)описать предметную область, выделить основные сущности, формализовать задачу будущего приложения.

2. Разработать приложение с учетом выбранной технологии ORM. Покрыть все сущностные классы и классы отображения тестами.

3. Разработать репозиторий для работы с БД.

**Описание предметной области.**

База данных создаётся для информационного хранения информации о сеансах в кинотеатре. БД должна содержать информацию о фильмах, сеансах, о залах для показа, билетах и зрителях.

# UML-Диаграмма

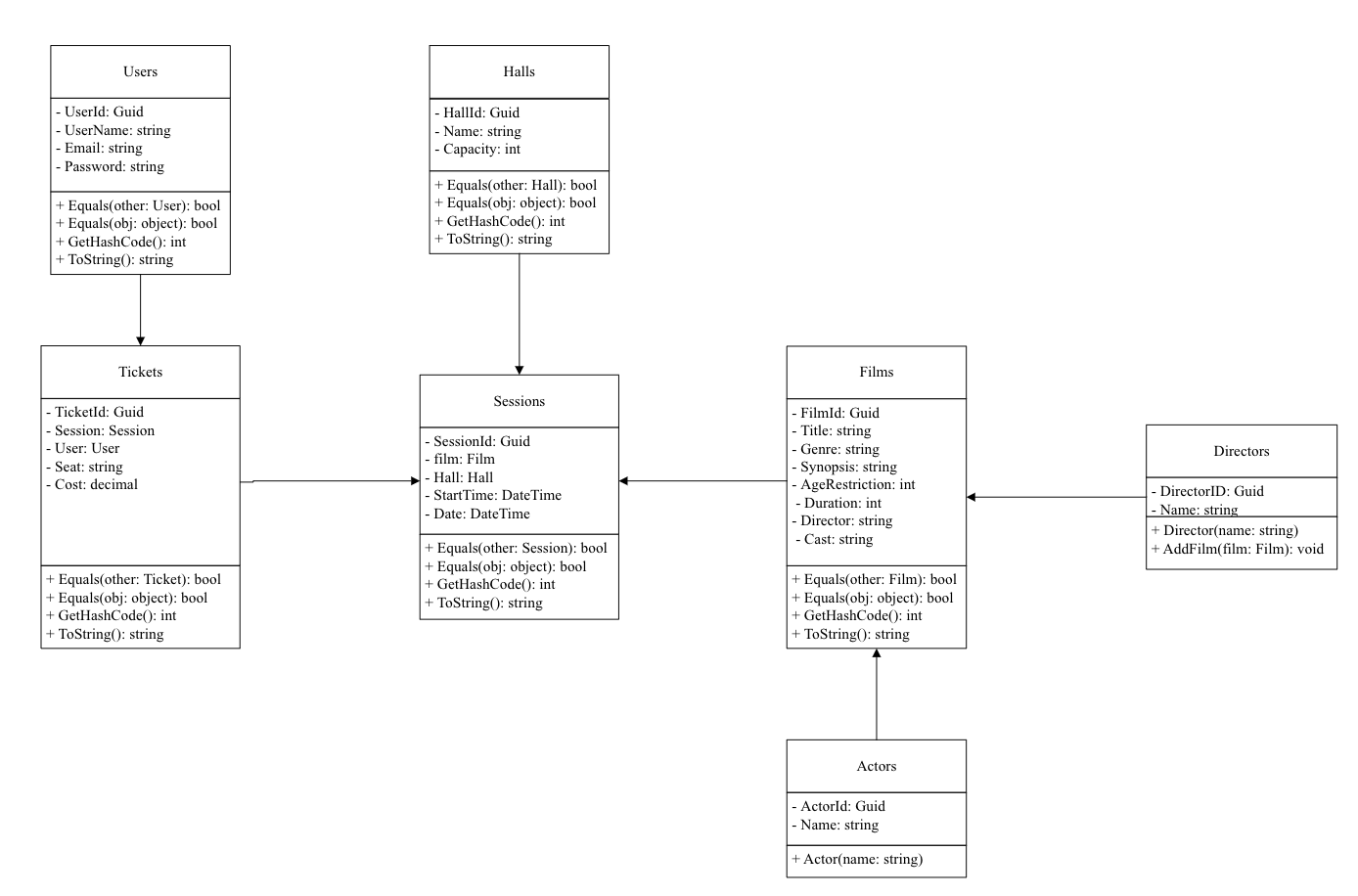


Рисунок 1 – UML - диаграмма

# Код программы

**Film.cs**

namespace Cinema

{

using Staff;

/// Класс Фильмы.

public sealed class Film : IEquatable<Film>

{

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

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

/// <param name="genre"> Жанр фильма. </param>

/// <param name="synopsis"> Краткий обзор. </param>

/// <param name="ageRestriction"> Возрастное ограничение. </param>

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

/// <param name="director"> Режиссер. </param>

/// <param name="cast"> Актерский состав. </param>

public Film(

string title,

string genre,

string synopsis,

int ageRestriction,

int duration,

string director,

string cast)

{

this.FilmId = Guid.NewGuid();

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

this.Genre = genre ?? throw new ArgumentNullException(nameof(genre));

this.Synopsis = synopsis ?? throw new ArgumentNullException(nameof(synopsis));

this.AgeRestriction = ageRestriction;

this.Duration = duration;

this.Director = director ?? throw new ArgumentNullException(nameof(director));

this.Cast = cast ?? throw new ArgumentNullException(nameof(cast));

}

public Guid FilmId { get; }

public string Title { get; }

public string Genre { get; }

public string Synopsis { get; }

public int AgeRestriction { get; }

public int Duration { get; }

public string Director { get; }

public string Cast { get; }

public bool Equals(Film? other)

{

if (other is null)

{

return false;

}

if (ReferenceEquals(this, other))

{

return true;

}

return this.Title == other.Title &&

this.Genre == other.Genre &&

this.Synopsis == other.Synopsis &&

this.AgeRestriction == other.AgeRestriction &&

this.Duration == other.Duration &&

this.Director == other.Director &&

this.Cast == other.Cast;

}

public override bool Equals(object? obj)

{

return this.Equals(obj as Film);

}

public override int GetHashCode()

{

return HashCode.Combine(this.Title, this.Genre, this.Synopsis, this.AgeRestriction, this.Duration, this.Director, this.Cast);

}

public override string ToString() =>

$"{this.Title} {this.Genre} {this.Synopsis}{this.AgeRestriction} {this.Duration} {this.Director} {this.Cast}";

}

}

**Hall.cs**

namespace Cinema

{

using Staff;

/// Класс Зал

public sealed class Hall : IEquatable<Hall>

{

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

/// <param name="name"> Имя зала.</param>

/// <param name="capacity"> Вместимость зала. </param>

public Hall(

string name,

int capacity)

{

this.HallId = Guid.NewGuid();

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

this.Capacity = capacity;

}

public Guid HallId { get; }

public string Name { get; }

public int Capacity { get; }

public bool Equals(Hall? other)

{

if (other is null)

{

return false;

}

if (ReferenceEquals(this, other))

{

return true;

}

return this.Name == other.Name &&

this.Capacity == other.Capacity;

}

public override bool Equals(object? obj)

{

return this.Equals(obj as Hall);

}

public override int GetHashCode()

{

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

}

public override string ToString() =>

$"{this.Name} (Capacity: {this.Capacity})";

}

}

**Session.cs**

namespace Cinema

{

using Staff;

/// Класс Сеанс.

public sealed class Session : IEquatable<Session>

{

/// <summary>

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

/// </summary>

/// <param name="film"> Фильм.</param>

/// <param name="hall"> Зал. </param>

/// <param name="startTime"> Начало сеанса. </param>

/// <param name="date"> Дата сеанса. </param>

public Session(

Film film,

Hall hall,

DateTime startTime,

DateTime date)

{

this.SessionId = Guid.NewGuid();

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

this.Hall = hall ?? throw new ArgumentNullException(nameof(hall));

this.StartTime = startTime;

this.Date = date;

}

public Guid SessionId { get; }

public Film film { get; }

public Hall Hall { get; }

public DateTime StartTime { get; }

public DateTime Date { get; }

public bool Equals(Session? other)

{

if (other is null)

{

return false;

}

if (ReferenceEquals(this, other))

{

return true;

}

return this.film.Equals(other.film) &&

this.Hall.Equals(other.Hall) &&

this.StartTime == other.StartTime &&

this.Date == other.Date;

}

public override bool Equals(object? obj)

{

return this.Equals(obj as Session);

}

public override int GetHashCode()

{

return HashCode.Combine(this.film, this.Hall, this.StartTime);

}

public override string ToString() =>

$"{this.film.Title} in {this.Hall.Name} on {this.Date.ToShortDateString()} at {this.StartTime.ToShortDateString()}";

}

}

**Ticket.cs**

namespace Cinema

{

using Staff;

using System.Xml.Linq;

/// Класс Билет.

public sealed class Ticket : IEquatable<Ticket>

{

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

/// <param name="session"> Сеанс.</param>

/// <param name="user"> Клиент.</param>

/// <param name="seat"> Место в зале.</param>

/// <param name="cost"> Цена билета.</param>

public Ticket(

Session session,

User user,

string seat,

decimal cost)

{

this.TicketId = Guid.NewGuid();

this.Session = session ?? throw new ArgumentNullException(nameof(session));

this.User = user ?? throw new ArgumentNullException(nameof(user));

this.Seat = seat ?? throw new ArgumentNullException(nameof(seat));

this.Cost = cost;

}

public Guid TicketId { get; }

public Session Session { get; }

public User User { get; }

public string Seat { get; }

public decimal Cost { get; }

public bool Equals(Ticket? other)

{

if (other is null)

{

return false;

}

if (ReferenceEquals(this, other))

{

return true;

}

return this.Session.Equals(other.Session) &&

this.User.Equals(other.User) &&

this.Seat == other.Seat &&

this.Cost == other.Cost;

}

public override bool Equals(object? obj)

{

return this.Equals(obj as Ticket);

}

public override int GetHashCode()

{

return HashCode.Combine(this.Session, this.User, this.Seat, this.Cost);

}

public override string ToString() =>

$"Ticket for {this.User.UserName} - sear: {this.Seat}, Cost: {this.Cost:C}";

}

}

**Users.cs**

namespace Cinema

{

using Staff;

/// Класс Клиент.

public sealed class User : IEquatable<User>

{

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

/// <param name="userName"> Имя клиента.</param>

/// <param name="email"> Электронная почта. </param>

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

public User(

string userName,

string email,

string password)

{

this.UserId = Guid.NewGuid();

this.UserName = userName.TrimOrNull() ?? throw new ArgumentNullException(nameof(userName));

this.Email = email.TrimOrNull() ?? throw new ArgumentNullException(nameof(email));

this.Password = password.TrimOrNull() ?? throw new ArgumentNullException(nameof(password));

}

public Guid UserId { get; }

public string UserName { get; }

public string Email { get; }

public string Password { get; }

public bool Equals(User? other)

{

if (other is null)

{

return false;

}

if (ReferenceEquals(this, other))

{

return true;

}

return this.Email == other.Email &&

this.UserName == other.UserName &&

this.Password == other.Password;

}

public override bool Equals(object? obj)

{

return this.Equals(obj as User);

}

public override int GetHashCode()

{

return HashCode.Combine(this.UserName, this.Email, this.Password);

}

public override string ToString() =>

$"{this.UserName} {this.Email} {this.Password}";

}

}

**Actor.cs**

// <copyright file="Actor.cs" company="Кирюшин Н.А.">

// Copyright (c) Кирюшин Н.А.. All rights reserved.

// </copyright>

namespace Cinema

{

/// <summary>

/// Абстрактный класс Актер.

/// </summary>

public class Actor : Person

{

/// <summary>

/// ID актера.

/// </summary>

public Guid ActorId { get; set; }

/// <summary>

/// Имя.

/// </summary>

public new string Name { get; }

/// <summary>

/// Инициализирует новый экземпляр класса Actor.

/// </summary>

/// <param name="name">Имя актера.</param>

public Actor(string name)

: base(name)

{

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

}

[Obsolete("For EF Core only", true)]

private Actor()

: base(string.Empty)

{

this.Name = string.Empty;

}

}

}

**Directos.cs**

namespace Cinema

{

/// <summary>

/// Режисер.

/// </summary>

public sealed class Director : Person

{

public Director(string name)

: base(name)

{

}

public void AddFilm(Film film)

{

if (film is null)

{

throw new ArgumentNullException(nameof(film));

}

film.Director = this;

this.Films.Add(film);

}

}

}

**Person.cs**

// <copyright file="Person.cs" company="Кирюшин Н.А.">

// Copyright (c) Кирюшин Н.А.. All rights reserved.

// </copyright>

namespace Cinema

{

using System.Collections.Generic;

/// <summary>

/// Абстрактный класс Person, представляющий человека.

/// </summary>

public abstract class Person

{

/// <summary>

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

/// </summary>

/// <param name="name">Имя.</param>

protected Person(string name)

{

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

}

/// <summary>

/// Идентификатор.

/// </summary>

public Guid PersonId { get; }

/// <summary>

/// Имя.

/// </summary>

public string Name { get; }

/// <summary>

/// Список фильмов.

/// </summary>

public ISet<Film> Films { get; } = new HashSet<Film>();

}

}

# Прохождение тестов

**FilmTests.cs**

namespace CinemaTest { using System; using System.Collections.Generic; using Cinema; using NUnit.Framework;

/// Тесты на полку <see cref="Cinema.Film"/>.  
[TestFixture]  
public sealed class FilmTests  
{  
 [TestCase("Inception", "Sci-Fi", "A mind-bending thriller.", 13, 148, "Christopher Nolan", "Leonardo DiCaprio Joseph, Gordon-Levitt")]  
 [TestCase("The Matrix", "Action", "A hacker discovers the nature of his reality.", 16, 136, "Lana Wachowski, Lilly Wachowski", "Keanu Reeves, Laurence Fishburne")]  
 public void Ctor\_ValidData\_DoesNotThrow(string title, string genre, string synopsis, int ageRestriction, int duration, string director, string cast)  
 {  
 Assert.DoesNotThrow(() => \_ = new Film(title, genre, synopsis, ageRestriction, duration, director, cast));  
 }  
  
 [TestCase(null, "Sci-Fi", "A mind-bending thriller.", 13, 148, "Christopher Nolan", "Leonardo DiCaprio")]  
 [TestCase("Inception", null, "A mind-bending thriller.", 13, 148, "Christopher Nolan", "Leonardo DiCaprio")]  
 [TestCase("Inception", "Sci-Fi", null, 13, 148, "Christopher Nolan", "Leonardo DiCaprio" )]  
 [TestCase("Inception", "Sci-Fi", "A mind-bending thriller.", 13, -1, "Christopher Nolan", "Leonardo DiCaprio")]  
 [TestCase("Inception", "Sci-Fi", "A mind-bending thriller.", 13, 148, null, "Leonardo DiCaprio")]  
 public void Ctor\_InvalidData\_ThrowsArgumentNullException(string title, string genre, string synopsis, int ageRestriction, int duration, string director, string cast)  
 {  
 Assert.Throws<ArgumentNullException>(() => \_ = new Film(title, genre, synopsis, ageRestriction, duration, director, cast));  
 }  
  
 [Test]  
 public void Equals\_DifferentFilms\_NotEqual()  
 {  
 var film1 = new Film("Inception", "Sci-Fi", "A mind-bending thriller.", 13, 148, "Christopher Nolan", "Leonardo DiCaprio, Joseph Gordon-Levitt");  
 var film2 = new Film("Interstellar", "Sci-Fi", "Exploration of space and time.", 13, 169, "Christopher Nolan", "Matthew McConaughey, Anne Hathaway");  
  
 Assert.That(film1, Is.Not.EqualTo(film2));  
 }  
  
 [Test]  
 public void Equals\_SimilarFilms\_Success()  
 {  
 var film1 = new Film("Inception", "Sci-Fi", "A mind-bending thriller.", 13, 148, "Christopher Nolan", "Leonardo DiCaprio, Joseph Gordon-Levitt");  
 var film2 = new Film("Inception", "Sci-Fi", "A mind-bending thriller.", 13, 148, "Christopher Nolan", "Leonardo DiCaprio, Joseph Gordon-Levitt");  
  
 Assert.That(film1, Is.EqualTo(film2));  
 }  
  
 [Test]  
 public void ToString\_ReturnsExpectedString()  
 {  
 var film = new Film("Inception", "Sci-Fi", "A mind-bending thriller.", 13, 148, "Christopher Nolan", "Leonardo DiCaprio, Joseph Gordon-Levitt");  
 var expectedString = "Inception (Sci-Fi) - Directed by Christopher Nolan";  
  
 Assert.That(film.ToString(), Is.EqualTo(expectedString));  
 }  
}

}

**HallTests.cs**

namespace CinemaTest { using System; using System.Collections.Generic; using Cinema; using NUnit.Framework;

/// Тесты для класса <see cref="Cinema.Hall"/>.  
[TestFixture]  
public sealed class HallTests  
{  
 [TestCase("Main Hall", 100)]  
 [TestCase("VIP Hall", 50)]  
 public void Ctor\_ValidData\_DoesNotThrow(string name, int capacity)  
 {  
 Assert.DoesNotThrow(() => \_ = new Hall(name, capacity));  
 }  
  
 [TestCase(null, 100)]  
 [TestCase("Main Hall", -1)]  
 public void Ctor\_InvalidData\_ThrowsException(string name, int capacity)  
 {  
 Assert.Throws<ArgumentException>(() => \_ = new Hall(name, capacity));  
 }  
  
 [Test]  
 public void Equals\_DifferentHalls\_NotEqual()  
 {  
 var hall1 = new Hall("Hall1", 100);  
 var hall2 = new Hall("Hall2", 100);  
  
 Assert.That(hall1, Is.Not.EqualTo(hall2));  
 }  
  
 [Test]  
 public void Equals\_SimilarHalls\_Success()  
 {  
 var hall1 = new Hall("Hall", 100);  
 var hall2 = new Hall("Hall", 100);  
  
 Assert.That(hall1, Is.EqualTo(hall2));  
 }

}

**SessionTests.cs**

namespace CinemaTest { using System; using System.Collections.Generic; using Cinema; using NUnit.Framework;

/// Тесты для класса <see cref="Cinema.Session"/>.  
[TestFixture]  
public sealed class SessionTests  
{  
 [Test]  
 public void Ctor\_ValidData\_DoesNotThrow()  
 {  
 var film = new Film("Inception", "Sci-Fi", "A mind-bending thriller.", 13, 148, "Christopher Nolan", "Leonardo DiCaprio, Joseph Gordon-Levitt");  
 var hall = new Hall("Hall", 100);  
 var date = new DateTime(2023, 10, 31, 19, 0, 0);  
 var startTime = new DateTime(2024, 10, 31, 19, 0, 0);  
  
 Assert.DoesNotThrow(() => \_ = new Session(film, hall, date, startTime));  
 }  
  
 [Test]  
 public void Equals\_DifferentSessions\_NotEqual()  
 {  
 var film = new Film("Inception", "Sci-Fi", "A mind-bending thriller.", 13, 148, "Christopher Nolan", "Leonardo DiCaprio, Joseph Gordon-Levitt");  
 var hall1 = new Hall("Hall1", 100);  
 var hall2 = new Hall("Hall2", 100);  
 var date = new DateTime(2021, 7, 21, 19, 0, 0);  
 var startTime = new DateTime(2022, 10, 31, 19, 0, 0);  
  
 var session1 = new Session(film, hall1, date, startTime);  
 var session2 = new Session(film, hall2, date, startTime);  
  
 Assert.That(session1, Is.Not.EqualTo(session2));  
 }  
}

}

**TicketTests.cs**

namespace CinemaTest { using System; using System.Collections.Generic; using Cinema; using NUnit.Framework; using static System.Runtime.InteropServices.JavaScript.JSType;

/// Тесты для класса <see cref="Cinema.Ticket"/>.  
[TestFixture]  
public sealed class TicketTests  
{  
 [Test]  
 public void Ctor\_ValidData\_DoesNotThrow()  
 {  
 var user = new User("username", "[email@example.com](mailto:email@example.com)", "password");  
 var film = new Film("Inception", "Sci-Fi", "A mind-bending thriller.", 13, 148, "Christopher Nolan", "Leonardo DiCaprio, Joseph Gordon-Levitt");  
 var hall = new Hall("Hall", 100);  
 var date = new DateTime(2023, 10, 31);  
 var startTime = new DateTime(19, 0);  
  
 var session = new Session(film, hall, startTime, date);  
  
 Assert.DoesNotThrow(() => \_ = new Ticket(session, user, "5", 15.99m));  
 }  
  
 [Test]  
 public void Equals\_DifferentTickets\_NotEqual()  
 {  
 var user = new User("username", "[email@example.com](mailto:email@example.com)", "password");  
 var film = new Film("Inception", "Sci-Fi", "A mind-bending thriller.", 13, 148, "Christopher Nolan", "Leonardo DiCaprio, Joseph Gordon-Levitt");  
 var hall = new Hall("Hall", 100);  
 var date1 = new DateTime(2022, 11, 21);  
 var startTime1 = new DateTime(19, 0);  
 var date2 = new DateTime(2022, 12, 23);  
 var startTime2 = new DateTime(19, 0);  
  
 var session1 = new Session(film, hall, startTime1, date1);  
 var session2 = new Session(film, hall, startTime2, date2);  
  
 var ticket1 = new Ticket(session1, user, "5", 15.99m);  
 var ticket2 = new Ticket(session2, user, "5", 15.99m);  
  
 Assert.That(ticket1, Is.Not.EqualTo(ticket2));  
 }  
  
 [Test]  
 public void Equals\_SimilarTickets\_Success()  
 {  
 var user = new User("username", "[email@example.com](mailto:email@example.com)", "password");  
 var film = new Film("Inception", "Sci-Fi", "A mind-bending thriller.", 13, 148, "Christopher Nolan", "Leonardo DiCaprio, Joseph Gordon-Levitt");  
 var hall = new Hall("Hall", 100);  
 var date1 = new DateTime(2022, 11, 21);  
 var startTime1 = new DateTime(19, 0);  
  
 var session = new Session(film, hall, startTime1, date1);  
  
 var ticket1 = new Ticket(session, user, "5", 15.99m);  
 var ticket2 = new Ticket(session, user, "5", 15.99m);  
  
 Assert.That(ticket1, Is.EqualTo(ticket2));  
 }  
}

}

**UserTests.cs**

namespace CinemaTest { using System; using System.Collections.Generic; using Cinema; using NUnit.Framework;

/// Тесты для класса <see cref="Cinema.User"/>.  
[TestFixture]  
public sealed class UserTests  
{  
 [TestCase("username", "[email@example.com](mailto:email@example.com)", "password")]  
 [TestCase("testUser", "[test@example.com](mailto:test@example.com)", "securePassword")]  
 public void Ctor\_ValidData\_DoesNotThrow(string userName, string email, string password)  
 {  
 Assert.DoesNotThrow(() => \_ = new User(userName, email, password));  
 }  
  
 [TestCase(null, "[email@example.com](mailto:email@example.com)", "password")]  
 [TestCase("username", null, "password")]  
 [TestCase("username", "[email@example.com](mailto:email@example.com)", null)]  
 public void Ctor\_InvalidData\_ThrowsArgumentNullException(string userName, string email, string password)  
 {  
 Assert.Throws<ArgumentNullException>(() => \_ = new User(userName, email, password));  
 }  
  
 [Test]  
 public void Equals\_ValidDataDifferentUserName\_NotEqual()  
 {  
 var user1 = new User("User1", "[email@example.com](mailto:email@example.com)", "password");  
 var user2 = new User("User2", "[email@example.com](mailto:email@example.com)", "password");  
  
 Assert.That(user1, Is.Not.EqualTo(user2));  
 }  
  
 [Test]  
 public void Equals\_ValidDataSameUserNameAndEmail\_Success()  
 {  
 var user1 = new User("User", "[email@example.com](mailto:email@example.com)", "password1");  
 var user2 = new User("User", "[email@example.com](mailto:email@example.com)", "password2");  
  
 Assert.That(user1, Is.EqualTo(user2));  
 }  
}  
 }

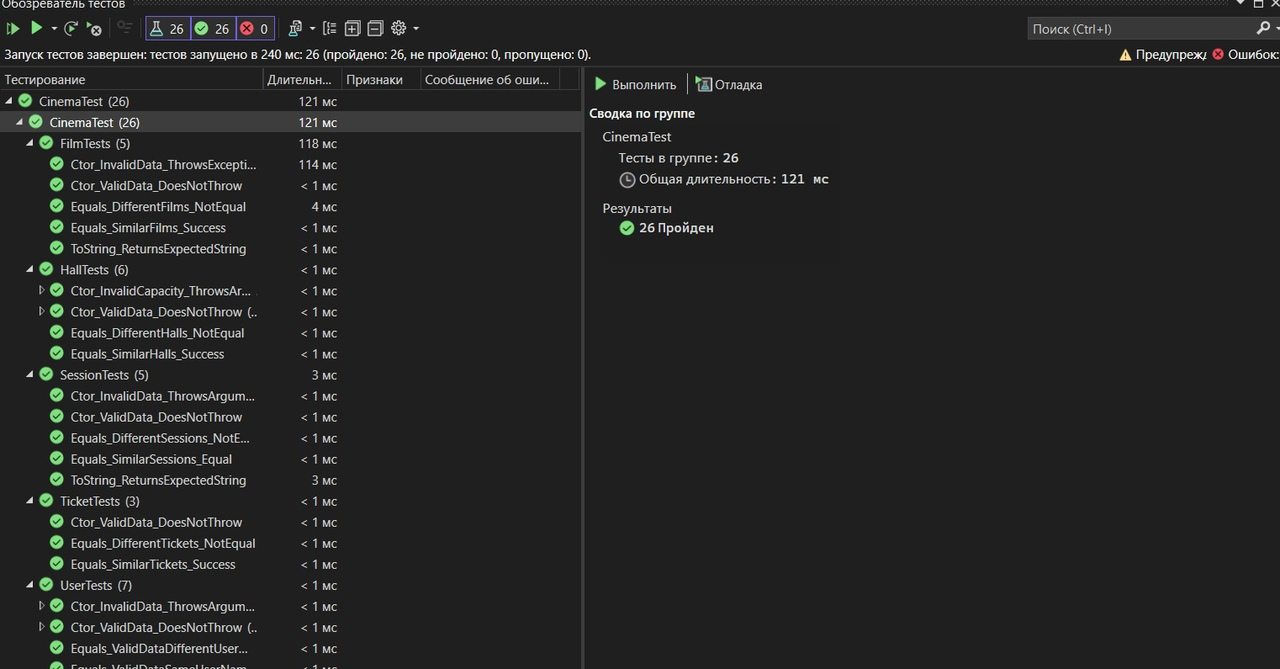


Рисунок 2 - Результат прохождения тестов

# Настройка миграции

**ActorConfiguration.cs**

// <copyright file="ActorConfiguration.cs" company="Кирюшин Н.А.">

// Copyright (c) Кирюшин Н.А.. All rights reserved.

// </copyright>

using Cinema;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata.Builders;

/// <summary>

/// ActorConfiguration.

/// </summary>

public class ActorConfiguration : IEntityTypeConfiguration<Actor>

{

/// <inheritdoc/>

public void Configure(EntityTypeBuilder<Actor> builder)

{

builder.HasKey(a => a.ActorId);

builder.Property(a => a.Name)

.IsRequired()

.HasMaxLength(200)

.HasComment("Имя актера");

builder.ToTable("Actors");

}

}

**DirectorConfiguration.cs**

// <copyright file="DirectorConfiguration.cs" company="Кирюшин Н.А.">

// Copyright (c) Кирюшин Н.А.. All rights reserved.

// </copyright>

using Cinema;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata.Builders;

/// <summary>

/// DirectorConfiguration.

/// </summary>

public class DirectorConfiguration : IEntityTypeConfiguration<Director>

{

/// <inheritdoc/>

public void Configure(EntityTypeBuilder<Director> builder)

{

builder.HasKey(d => d.PersonId);

builder.Property(d => d.Name).IsRequired().HasMaxLength(200).HasComment("Имя режиссера");

builder.ToTable("Directors");

}

}

**FilmConfiguration.cs**

using Cinema;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata.Builders;

/// <summary>

/// FilmConfiguration.

/// </summary>

public class FilmConfiguration : IEntityTypeConfiguration<Film>

{

/// <inheritdoc/>

public void Configure(EntityTypeBuilder<Film> builder)

{

builder.HasKey(f => f.FilmId);

builder.Property(f => f.Title)

.IsRequired()

.HasMaxLength(200);

builder.Property(f => f.Genre)

.IsRequired()

.HasMaxLength(100);

builder.Property(f => f.Synopsis)

.IsRequired();

builder.Property(f => f.AgeRestriction)

.IsRequired();

builder.Property(f => f.Duration)

.IsRequired();

builder.HasOne(f => f.Director)

.WithMany(d => d.Films)

.HasForeignKey(f => f.DirectorId);

builder.HasMany(f => f.Cast)

.WithMany(a => a.Films)

.UsingEntity(j => j.ToTable("FilmActors"));

builder.ToTable("Films");

}

}

**HallConfiguration.cs**

using Cinema;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata.Builders;

/// <summary>

/// HallConfiguration.

/// </summary>

public class HallConfiguration : IEntityTypeConfiguration<Hall>

{

/// <inheritdoc/>

public void Configure(EntityTypeBuilder<Hall> builder)

{

builder.HasKey(h => h.HallId);

builder.Property(h => h.Name).IsRequired().HasMaxLength(200).HasComment("Название зала");

builder.Property(h => h.Capacity).IsRequired().HasComment("Вместимость зала");

builder.ToTable("Halls");

}

}

**SessionConfiguration.cs**

using Cinema;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata.Builders;

/// <summary>

/// SessionConfiguration.

/// </summary>

public class SessionConfiguration : IEntityTypeConfiguration<Session>

{

/// <inheritdoc/>

public void Configure(EntityTypeBuilder<Session> builder)

{

builder.HasKey(s => s.SessionId);

builder.Property(s => s.StartTime).IsRequired().HasComment("Время начала сеанса");

builder.Property(s => s.Date).IsRequired().HasComment("Дата сеанса");

builder.HasOne(s => s.Film)

.WithMany()

.HasForeignKey("FilmId")

.IsRequired()

.OnDelete(DeleteBehavior.Cascade);

builder.HasOne(s => s.Hall)

.WithMany(h => h.Sessions)

.HasForeignKey("HallId")

.IsRequired()

.OnDelete(DeleteBehavior.Cascade);

builder.ToTable("Sessions");

}

}

**TicketConfiguration.cs**

// <copyright file="TicketConfiguration.cs" company="Кирюшин Н.А.">

// Copyright (c) Кирюшин Н.А.. All rights reserved.

// </copyright>

using Cinema;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata.Builders;

/// <summary>

/// TicketConfiguration.

/// </summary>

public class TicketConfiguration : IEntityTypeConfiguration<Ticket>

{

/// <inheritdoc/>

public void Configure(EntityTypeBuilder<Ticket> builder)

{

builder.HasKey(t => t.TicketId);

builder.Property(t => t.Seat).IsRequired().HasMaxLength(10).HasComment("Место");

builder.Property(t => t.Cost).IsRequired().HasComment("Стоимость билета");

builder.HasOne(t => t.Session)

.WithMany()

.HasForeignKey("SessionId")

.IsRequired()

.OnDelete(DeleteBehavior.Cascade);

builder.HasOne(t => t.User)

.WithMany()

.HasForeignKey("UserId")

.IsRequired()

.OnDelete(DeleteBehavior.Cascade);

builder.ToTable("Tickets");

}

}

**UserConfiguration.cs**

// <copyright file="UserConfiguration.cs" company="Кирюшин Н.А.">

// Copyright (c) Кирюшин Н.А.. All rights reserved.

// </copyright>

using Cinema;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata.Builders;

/// <summary>

/// UserConfiguration.

/// </summary>

public class UserConfiguration : IEntityTypeConfiguration<User>

{

/// <inheritdoc/>

public void Configure(EntityTypeBuilder<User> builder)

{

builder.HasKey(u => u.UserId);

builder.Property(u => u.UserName)

.IsRequired()

.HasMaxLength(100)

.HasComment("Имя пользователя");

builder.Property(u => u.Email)

.IsRequired()

.HasMaxLength(200)

.HasComment("Электронная почта");

builder.Property(u => u.Password)

.IsRequired()

.HasMaxLength(200)

.HasComment("Пароль");

builder.ToTable("Users");

}

}

# ER-диаграмма

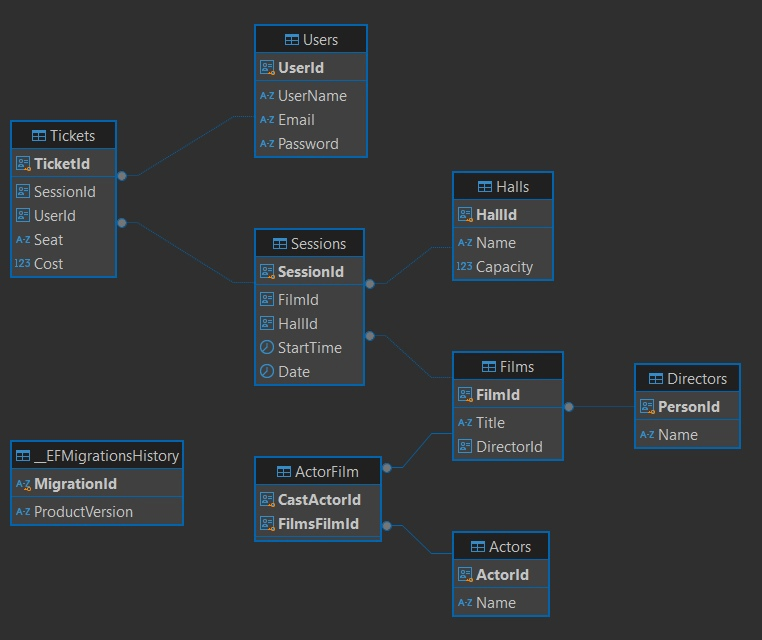


Рисунок 3 – ER-диаграмма

# 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.
      5. SQLAlchemy [Электронный ресурс] // SQLAlchemy [сайт]. URL: <https://www.sqlalchemy.org/> (дата обращения 24.10.2023).
      6. PostgreSQL [Электронный ресурс] // PostgreSQL [сайт]. — URL: <https://www.postgresql.org/> (дата обращения 24.10.2023).