

FilmiSub

документация

Изработен от: Васил Пищялов,

Деница Тренчева,

Йордан Горанов

Съдържание

1. Цели

2. Описание

3. Разпределение на ролите

4. Ниво на сложност на проекта

4.1 Entity Framework

4.2 Бизнес логика

5. Filmisub

6. Визуализация на проекта

7. Инсталация и деинсталация

8. Използвани технологии

9. Обновяване и развитие

10.Nunit тестове

11.ER Diagram

1.Цели

Проектът представлява каталог за филми. Целта на програмата е да предостави удобен и интуитивен начин за управление на филмова база данни. Целевата аудитория може да включва администратори на видеотеки, киномани или разработчици, нуждаещи се от опростена система за каталогизиране на филми.

2.Описание

Проектът е реализиран в Visual Studio на C#. Основни функционалности са показване на филми, добавяне на нови записи и редактиране на съществуващи, както и интеграция с база данни за съхранение на данните- Microsoft SQL server.

Използваме различни библиотеки за работа с база данни (Entity Framework), както и библиотеки за тестове като Nunit - рамка за модулно тестване.

Проектът следва основния модел за приложение с конзолен интерфейс, който комуникира със сървърната част, отговаряща за базата данни.

Структората на проектът е подходяща за обновяване и развиване. Примери за това са раширяване на интерфейса и преминаване към веб базиран интерфейс с цел да се използва мрежова среда.

3. Разпределение на ролите

Обработката на софтуера - базата данни, функционалността, документацията, тестовете и презентацията е създадена от Васил Пищялов, Деница Тренчева и Йордан Горанов

4. Ниво на сложност на проекта

Определяме нивото на сложност на проекта ни като умерен. Сблъскахме се проблеми и грешки, които изчистихме в хода на работа.

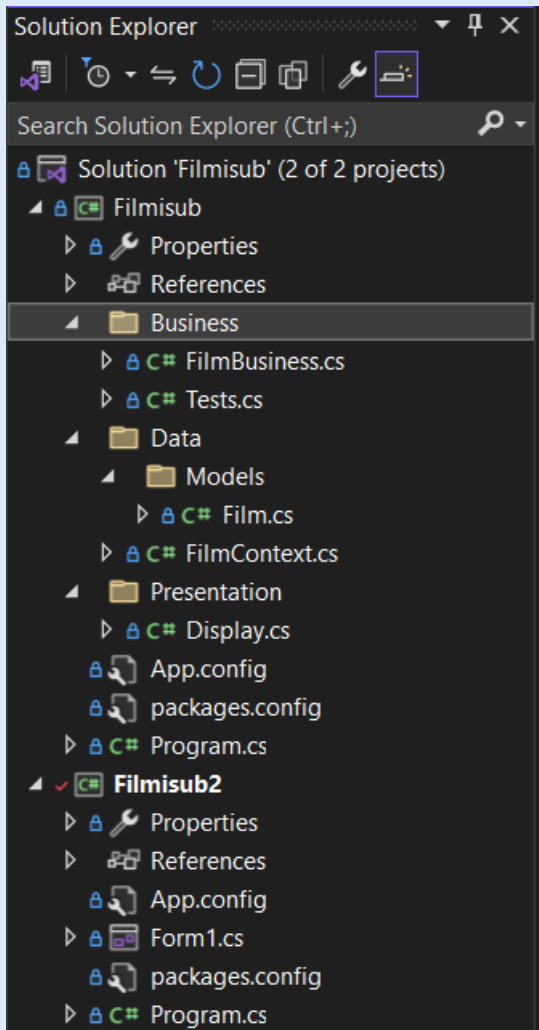
4.1 Entity Framework

За първи път се сблъскахме с този инструмент и трябваше да го разучим как работи. В крайна сметка се справихме и преминахме трудностите.

4.2 Бизнес логика

Внедряването на бизнес логиката също беше предизвикателна задача. Трябваше да отстраним доста възникнали бъгове и да тестваме кода си, за да се уверим, че работи безпроблемно.

5.Filmisub



Business – тук се съдържа бизнес логиката. Реализира се връзка с базата данни и модифициране на информация

- FilmBusiness - бизнес логиката, която осигурява интерфейс за основни операции с база данни за филми чрез Entity Framework.
- Tests – тук се намират Nunit тестовете за приложението.

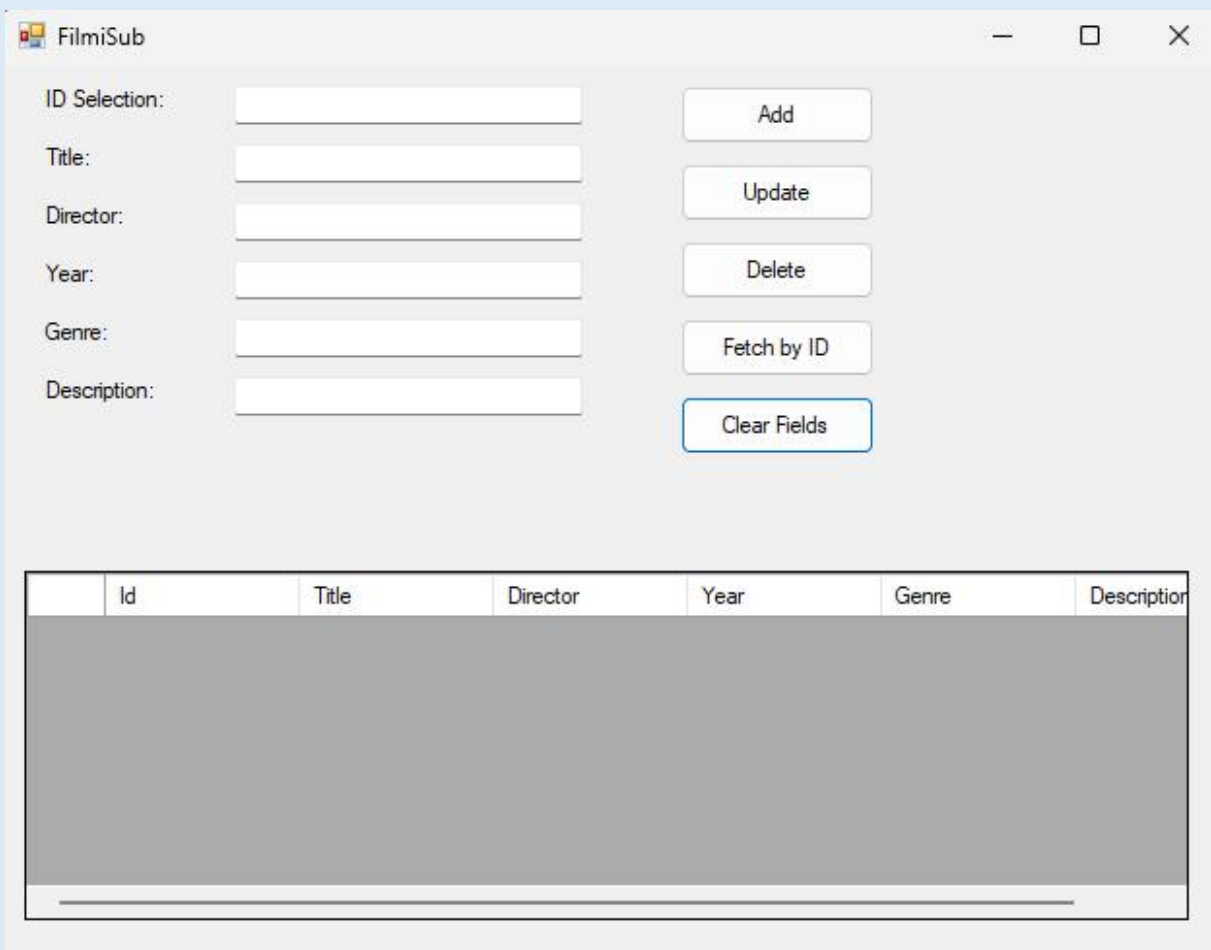
Data - работи с базата данни чрез Entity Framework.

Models – Представя структурата на данните

- Film.cs -представява таблица от базата: съдържа свойства като Id, Title, Genre, Year, Description.

- FilmContext - осъществява връзка с базата
- Presentation** - Отговаря за потребителския интерфейс
- Display.cs - е главният интерфейс между потребителя и системата. Извежда менюта, получава вход от потребителя.

6. Визуализация на проекта



The screenshot shows a Windows application window titled "FilmiSub". The window contains a form with the following fields and buttons:

- Input fields: ID Selection, Title, Director, Year, Genre, Description.
- Buttons: Add, Update, Delete, Fetch by ID, Clear Fields.

Below the form is a table with the following columns: Id, Title, Director, Year, Genre, Description. The table is currently empty, showing only the header row.

Id	Title	Director	Year	Genre	Description
----	-------	----------	------	-------	-------------

Командите представляват:

Add - Добавя нов филм към базата

Update – Обновява вече съществуващ филм

Delete – Изтрива филм по ID.

Fetch by ID - Извлича и показва филм по ID.

7.Инсталация и деинсталация

Преди да бъде използван разработения софтуер е необходимо да се премина през последователност от стъпки за неговата инсталация
Инсталирайте Visual Studio 2022

Уверете се, че .NET Framework 4.8 е инсталиран
Инсталирайте Microsoft SQL Server

1. Клонирайте кода, като използвате следната команда: `git clone`

2. Отворете проекта във Visual Studio

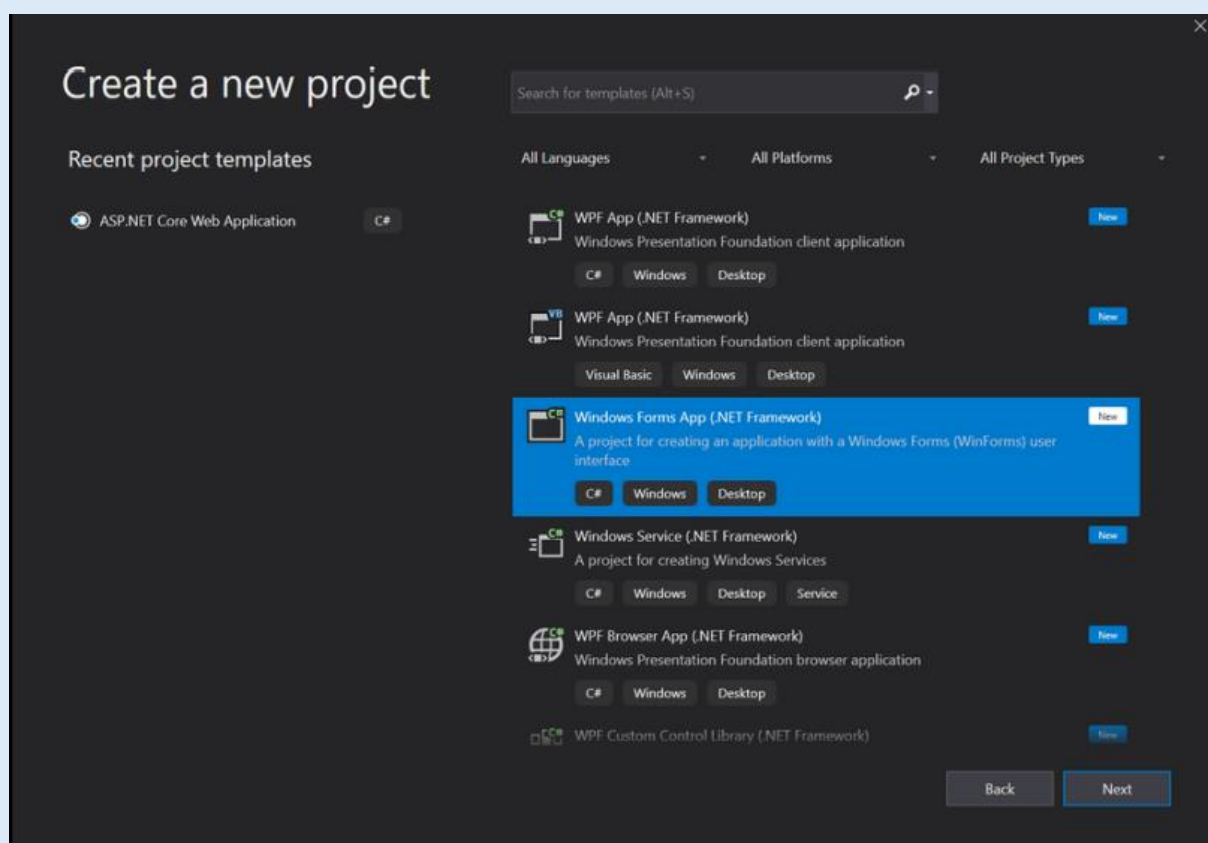
3. Програмата се стартира от основният клас Program.cs

За деинсталация, изтрийте папката, в която се намира проекта и изтрийте и локалната база от данни.

8. Използвани технологии

1).NETFramework - NET Framework е безплатна и open source софтуерна платформа, която работи предимно на Microsoft Windows .Тя включва голяма библиотека с класове, наречена Framework Class Library (FCL), и осигурява езикова оперативна съвместимост.

2)WindowsFormsApp - Windows форми е графична (GUI) библиотека от класове в състава на Microsoft .NET Framework, която предоставя платформа за писане на клиентски приложения за настолни компютри, лаптопи и таблети.



3) Visual Studio 2022 - интегрирана среда за разработка (IDE), разработена от Microsoft. Visual Studio предоставя мощна интегрирана среда за писане на код, компилиране, изпълнение, дебъгване (както за високо така и за машинно ниво), тестване на приложения, дизайн на потребителски интерфейс, моделиране на данни, моделиране на класове, изпълнение на тестове, пакетиране на приложения и стотици други функции. Visual Studio поддържа 36 различни езика за програмиране и позволява на редактора на код и дебъгера да поддържат почти всеки език за програмиране.

4) Entity Framework (EF) - е обектно-релационно преобразуване ,библиотека за .NET. Тя ти позволява да работиш с база данни, използвайки C# обекти, без да пишеш директно SQL заявки.

5) Microsoft SQL Server - Microsoft SQL Server е сървърна система за управление на бази от данни на компанията Microsoft.

9.Обновяване и развитие

Програмата е създадена със структура, която позволява лесно разширяване и усъвършенстване в бъдеще. Възможностите за развитие включват:

- Добавяне на потребителски акаунти и роли
- Въвеждане на различни нива на достъп (администратор, потребител), включително система за вход и защита.
- Търсене и филтриране на филми – Разширяване на интерфейса с опции за търсене по заглавие, режисьор, жанр и година.
- Преминаване към уеб базиран интерфейс – Миграция към ASP.NET или Blazor, с цел достъп

- през браузър и използване в мрежова среда.
- Автоматична валидация и по-добро потребителско изживяване – Подобрения в интерфейса като падащи менюта, подсказки, проверка на данни в реално време и по-информативни съобщения за грешки.
- Поддръжка на изображения и трейлъри – Добавяне на възможност за прикачване на плакати или видео линкове за всеки филм.

10.Nunit тестове

Направените от нас тестове към приложението.

```
// Tests whether fetching a valid film ID returns the correct object.
[Test]
0 references
public void Get_ExistingId_ShouldReturnCorrectFilm()
{
    int id = 2003;
    Film film = filmBusiness.Get(id);
    Assert.That(film, Is.Not.Null);
    Assert.That(film.Id, Is.EqualTo(id));
}
```

```
// Tests that requesting a non-existent ID returns null.
```

```
[Test]
```

```
0 references
```

```
public void Get_NonexistentId_ShouldReturnNull()
```

```
{
```

```
    int id = -1;
```

```
    Film film = filmBusiness.Get(id);
```

```
    Assert.That(film, Is.Null);
```

```
}
```

```
// Tests that adding a film with partial data increases count.
```

```
[Test]
```

```
0 references
```

```
public void Add_ShouldIncreaseFilmCount()
```

```
{
```

```
    int initialCount = filmBusiness.GetAll().Count;
```

```
    Film film = new Film
```

```
    {
```

```
        Title = "New Film",
```

```
        Description = "New Film Description"
```

```
    };
```

```
    filmBusiness.Add(film);
```

```
    int updatedCount = filmBusiness.GetAll().Count;
```

```
    Assert.That(updatedCount, Is.EqualTo(initialCount + 1));
```

```
}
```

```
// Tests that adding a fully populated film saves all properties.
[Test]
0 references
public void Add_FullFilm_ShouldSaveAllFields()
{
    Film film = new Film
    {
        Title = "Full Film",
        Director = "Test Director",
        Year = 2023,
        Genre = "Drama",
        Description = "Detailed film description."
    };

    filmBusiness.Add(film);

    Film saved = filmBusiness.GetAll().FirstOrDefault(f => f.Title == "Full Film");
    Assert.That(saved, Is.Not.Null);
    Assert.That(saved.Director, Is.EqualTo("Test Director"));
    Assert.That(saved.Year, Is.EqualTo(2023));
    Assert.That(saved.Genre, Is.EqualTo("Drama"));
    Assert.That(saved.Description, Is.EqualTo("Detailed film description.));
}
}
```

```
// Tests updating a film's title and director.
[Test]
0 references
public void Update_ExistingId_ShouldUpdateMultipleFields()
{
    int id = 2003;
    Film film = filmBusiness.Get(id);

    string newTitle = "Updated Title";
    string newDirector = "Updated Director";
    film.Title = newTitle;
    film.Director = newDirector;

    filmBusiness.Update(film);
    Film updated = filmBusiness.Get(id);

    Assert.That(updated.Title, Is.EqualTo(newTitle));
    Assert.That(updated.Director, Is.EqualTo(newDirector));
}
}
```

```
// Tests updating year and genre.
```

```
[Test]
```

```
0 references
```

```
public void Update_ShouldChangeYearAndGenre()
```

```
{
```

```
    var film = new Film
```

```
    {
```

```
        Title = "Year Genre Film",
```

```
        Director = "Dir",
```

```
        Year = 2000,
```

```
        Genre = "Old Genre",
```

```
        Description = "..."
```

```
    };
```

```
    filmBusiness.Add(film);
```

```
    var added = filmBusiness.GetAll().First(f => f.Title == "Year Genre Film");
```

```
    added.Year = 2024;
```

```
    added.Genre = "New Genre";
```

```
    filmBusiness.Update(added);
```

```
    var updated = filmBusiness.Get(added.Id);
```

```
    Assert.That(updated.Year, Is.EqualTo(2024));
```

```
    Assert.That(updated.Genre, Is.EqualTo("New Genre"));
```

```
}
```

```
// Tests that deleting an existing film decreases the total count.
```

```
[Test]
```

```
0 references
```

```
public void Delete_ExistingId_ShouldDecreaseFilmCount()
```

```
{
```

```
    var film = new Film { Title = "Delete Test", Description = "To be deleted" };
```

```
    filmBusiness.Add(film);
```

```
    int id = filmBusiness.GetAll().First(f => f.Title == "Delete Test").Id;
```

```
    int countBefore = filmBusiness.GetAll().Count;
```

```
    filmBusiness.Delete(id);
```

```
    int countAfter = filmBusiness.GetAll().Count;
```

```
    Assert.That(countAfter, Is.EqualTo(countBefore - 1));
```

```
}
```

```
// Tests that deleting a non-existent ID does not affect the count.
```

```
[Test]
```

```
0 references
```

```
public void Delete_NonexistentId_ShouldNotChangeFilmCount()
```

```
{
```

```
    int id = -99;
```

```
    int countBefore = filmBusiness.GetAll().Count;
```

```
    filmBusiness.Delete(id);
```

```
    int countAfter = filmBusiness.GetAll().Count;
```

```
    Assert.That(countAfter, Is.EqualTo(countBefore));
```

```
}
```

```
// Tests if the database connection string works.
[Test]
0 references
public void DatabaseConnection_ShouldBeOpen()
{
    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        connection.Open();
        Assert.That(connection.State, Is.EqualTo(System.Data.ConnectionState.Open));
    }
}
```

```
// Tests that GetAll returns at least one entry when films are present.
[Test]
0 references
public void GetAll_ShouldReturnNonEmptyListAfterAdd()
{
    filmBusiness.Add(new Film { Title = "Listed Film", Description = "For GetAll test" });
    var list = filmBusiness.GetAll();
    Assert.That(list.Count, Is.GreaterThan(0));
}
```

```
// Tests that films with empty fields still save correctly.
[Test]
0 references
public void Add_EmptyOptionalFields_ShouldStillSave()
{
    var film = new Film
    {
        Title = "No Genre or Director",
        Year = 2023,
        Description = "Edge case test"
    };

    filmBusiness.Add(film);
    var saved = filmBusiness.GetAll().FirstOrDefault(f => f.Title == "No Genre or Director");

    Assert.That(saved, Is.Not.Null);
    Assert.That(saved.Genre, Is.Null.Or.Empty);
    Assert.That(saved.Director, Is.Null.Or.Empty);
}
```

```
// Tests re-adding a deleted film with the same data.
[Test]
0 references
public void ReAdd_DeletedFilm_ShouldSucceed()
{
    var film = new Film { Title = "ReAdd Test", Description = "Will be deleted and re-added" };
    filmBusiness.Add(film);
    var saved = filmBusiness.GetAll().First(f => f.Title == "ReAdd Test");

    filmBusiness.Delete(saved.Id);

    filmBusiness.Add(film); // Re-add same data
    var readded = filmBusiness.GetAll().FirstOrDefault(f => f.Title == "ReAdd Test");
    Assert.That(readded, Is.Not.Null);
}
```

11.ER Diagram

