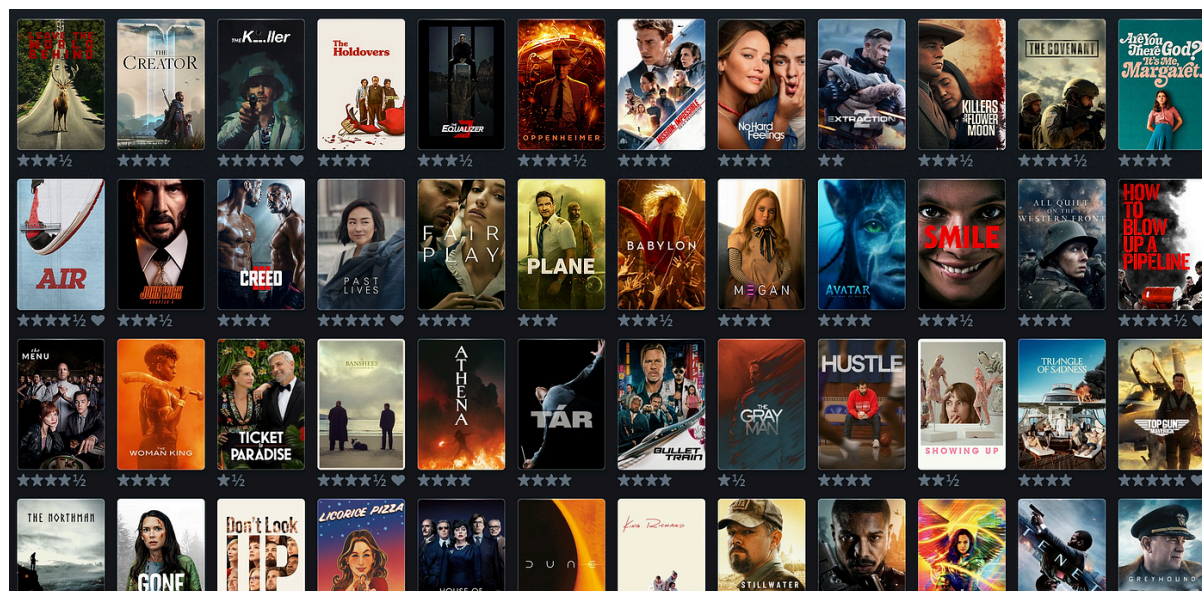


# Catalog for Movies

Курсов проект - IT Kariera Module 7



**Математическа гимназия „Академик Кирил Попов“**

**4001 Пловдив, ул. „Чемшир“ 11**

**тел.: + 359 32 643 157,**

**E – mail: <https://omg-bg.com/>**

**Ръководител: Кристиан Ташев**

**Изготвили проекта:**

**Анастасия Хронева Хронева**

**Дара Томова Георгиева**

**Диана Найденова Начева**

**Ева Николаева Вълкова**

# **Съдържание:**

## **1. Цели**

## **2. Разпределение на ролите**

## **3. Ниво на сложност на проекта**

### **3.1. Многослойна архитектура**

### **3.2. Работа с база данни**

### **3.3 Интерфейс и потребителско изживяване**

### **3.4. Работа с Git и GitHub**

### **3.5. Разширяемост на проекта**

## **4. Инсталация и деинсталация**

## **5. Основни етапи в реализирането на проект**

### **5.1 Потребители**

### **6. Реализация**

#### **6.1 Архитектура**

#### **6.2 База данни**

#### **6.3 Описание на приложението**

#### **6.4 Използвани технологии**

#### **6.5 Използвани езици за програмиране**

#### **6.6 Описание на приложението**

## **7. Развитие и нововъведения**

## **8. Заключение**

## **9. Използвана литература**

# 1. Цели на нашия проект

Целта на проекта е да се разработи стабилно настолно приложение, което опростява управлението и търсенето на филми и сериали по зададени критерии. Чрез осигуряване на бърз и лесен достъп до информация за заглавието, жанра, годината и описанието, приложението улеснява потребителите в откриването на желаното съдържание. То е проектирано с фокус върху удобството на крайния потребител, с интуитивен интерфейс и ясен дизайн, който ангажира вниманието и подобрява потребителския опит.

**Основните цели на проекта се обобщават в следните направления:**

- **Функционалност за търсене и филтриране на филми и сериали:**

Приложението ще позволява на потребителите да търсят заглавия по име, жанр или година на издаване. Това значително ще съкрати времето за търсене и ще предостави по-прецизни резултати, отговарящи на предпочитанията на потребителя.

- **Визуализация на съдържанието в различни категории:**

Приложението ще представя визуално наличните филми и сериали, разделени по жанрове или други характеристики. Това ще улесни разглеждането и ориентирането в каталога, като ще предоставя ясен и структуриран преглед.

- **Лекота на използване и достъпност:**

Интерфейсът е разработен така, че да бъде максимално лесен за използване, дори от потребители без технически опит. Това гарантира по-широк обхват и възможност за използване в различни среди – училища, библиотеки, клубове и лични колекции.

С реализирането на горните цели, приложението ще предложи ефективен и удобен инструмент за организиране и откриване на филми и сериали, като елиминира нуждата от дълги и неструктурирани търсения в интернет или на физически носители.

## **2. Разпределение на ролите**

Обработката на софтуера - базата данни, функционалността и дизайна, създаването на документацията и тестовете е разпределена между Анастасия Хронева Хронева, Дара Томова Георгиева, Диана Найденова Начева и Ева Николаева Вълкова.

## **3. Ниво на сложност на проекта**

Проектът има **средно ниво на сложност**, като комбинира основни и междинни принципи на обектно-ориентираното програмиране, работа с база данни, и изграждане на потребителски интерфейс. Сложността се определя от необходимостта за добра организация на кода, съгласуване

между слоевете в архитектурата, както и ефективна визуализация на информацията. Основните аспекти на сложността включват:

### 3.1. Многослойна архитектура:

Приложението е реализирано с ясно разделение между слоевете:

- **Data слой (модели и бази данни)** - достъп до базата данни
- **Business слой** - съдържа самата бизнес логика (например изчисления, валидации).
- **Service слой** - координира как и кога се използва тази логика например комбинира няколко бизнес операции
- **Presentation слой (интерфейс)**

Това налага планиране на взаимодействията между компонентите и спазване на добри практики.

### 3.2. Работа с база данни:

Използва се **Entity Framework Core** за управление на връзката между приложението и базата данни, което включва създаване на миграции, заявки, навигационни свойства и валидиране на данни.

### 3.3 Интерфейс и потребителско изживяване:

Интерфейсът е проектиран така, че да е интуитивен и лесен за използване, но в същото време да предлага визуална подредба, филтри, търсачка и детайлни изгледи на филми и сериали.

### **3.4. Работа с Git и GitHub:**

Проектът се развива в колаборация с други участници, като се използва GitHub за контрол на версиите. Това включва създаване на клонове, pull requests и сливане на промени – умения, необходими за реална разработка в екип.

### **3.5. Разширяемост на проекта:**

Приложението е създадено с мисъл за бъдещи разширения – като например добавяне на потребителски акаунти, оценки, ревюта или API интеграции. Това изисква предварително планиране на структурата.

## **4. Инсталация и Деинсталация**

Както всеки един софтуерен продукт преди да бъде използван трябва да се премине през последователността от стъпки за инсталация и настройки за експлоатация. Изисквания:

- Инсталирайте Visual Studio 2022
- Уверете се, че .NET Framework 8.0 е инсталиран:  
<https://dotnet.microsoft.com/en-us/download/dotnet/8.0>
- Инсталирайте MySQL Community Server -  
<https://dev.mysql.com/downloads/mysql/8.0.html>
- Инсталирайте MySQL Workbench или SQL Management Studio -

<https://dev.mysql.com/downloads/workbench/>

1. Клонирайте кода, като използвате следната команда:

git clone <https://github.com/Zzznai/BBMS.git>

2. Отворете проекта във Visual Studio

3. С помощта на SQL Management Studio създайте две нови бази данни със следните имена: а. CatalogforMoviesData

б. CatalogForMoviesTestData (за тестване)

4. Отидете на Visual Studio

5. Отидете на App.config и в двата проекта

а. За тестовите, във файла App.config променете връзката, за да сочи към базата данни CatalogForMoviesTestData

б. Променете низовете за връзка (като използвате вашия SQL низ за връзка)

7. Отворете Package Manager Console

8. Въведете следната команда, за да актуализирате базата данни на CatalogForMoviesData:

а. Update-Database

9. Актуализирайте базата данни за целите на тестването:  
Update-Database -ConnectionString "yourTestConnectionString" -  
ConnectionProviderName "System.Data.SqlClient"

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

## **5. Основни етапи при създаване на приложението**

Софтуерът “CatalogForMovies” съдържа единствено модул за вече регистрирани потребители

### **5.1. Потребители**

Потребителите имат възможност да търсят филми според заглавие, жанр и режисьор.

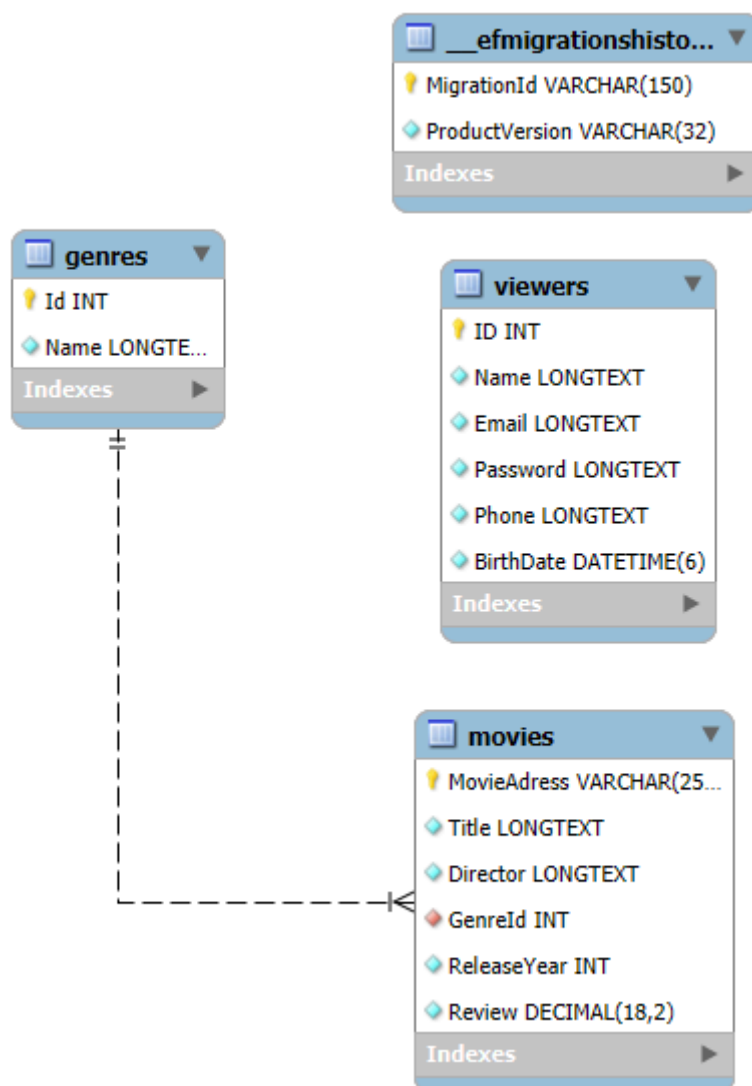
## **6. Реализация**

### **6.1. Архитектура**

Софтуерът е изграден от един проект.



## 6.2. Бази данни



## 6.3. CatalogForMovies

### 1.BusinessLayer (Бизнес слой)

Тук се дефинират основните модели (ентитети), отразяващи реалните обекти в приложението. Всеки от тях използва атрибути за валидация и съдържа подходящи конструктори.

- **Movie.cs** – Представа филм с характеристики като **Title**, **Director**, **ReleaseDate** и **Genre**.

- Genre.cs – Описва жанр с Name и списък от свързани филми.
- Viewer.cs – Представя потребител (зрител) с лични данни като Name, Email, Password и BirthDate.

## 2. DataLayer (Достъп до база данни)

Съдържа логиката за взаимодействие с базата данни чрез Entity Framework Core. Всеки ентитет си има съответен DBManager, който имплементира CRUD операциите чрез интерфейса IDB<T, K>.

- MoviesDBManager.cs
- GenresDBManager.cs
- ViewersDBManager.cs

Тези класове използват CatalogforMoviesDBContext, който представлява контекстът на базата данни.

## 3. ServiceLayer

Предоставя статични фасадни класове, които обединяват логиката от DataLayer и улесняват употребата им в потребителския интерфейс.

- MoviesManager.cs

- **GenresManager.cs**
- **ViewersManager.cs**

Този слой служи за изолация на бизнес логиката от визуалната част и опростява използването на DBManager-ите.

#### **4. PresentationLayer (Windows Forms)**

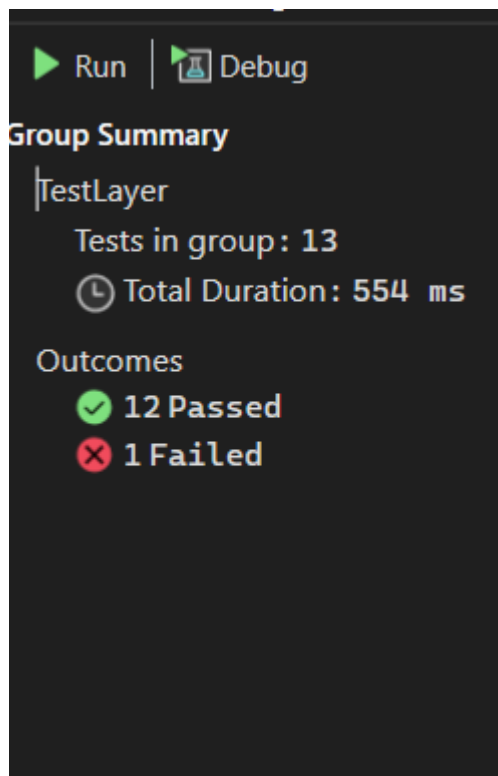
Това е визуалният интерфейс на приложението, разработен чрез WinForms. Всеки ентитет има собствена форма с елементи за въвеждане, избор и визуализация:

- **GenresForm** – Управлява жанровете. Позволява добавяне, редактиране и изтриване на жанрове.
- **ViewersForm** – Съдържа форма за управление на зрители: добавяне, редакция, изтриване.
- **MoviesForm** – Позволява работа с филми: задаване на заглавие, режисьор, жанр (чрез ComboBox), дата на излизане, както и преглед през DataGridView.
- **MainForm** – Главен прозорец на приложението, чрез който се отварят останалите форми. Представява навигационна точка.

## 5. TestingLayer (Unit Testing)

Съдържа тестове за основните класове от DataLayer. Тестовите са писани чрез NUnit и включват операции по създаване, четене, обновяване и изтриване (CRUD). За всеки мениджър има отделен тестов клас, използващ in-memory база или тестова база.

- MoviesDBManagerTests.cs
- GenresDBManagerTests.cs
- ViewersDBManagerTests.cs



## 6.4. Използвани технологии

- .NET Framework & .NET Core

- .NET 8.0 — използван за основните проекти (BusinessLayer, DataLayer, ServiceLayer, TestingLayer).

## 2. Entity Framework Core

- Позволява взаимодействие с база данни чрез C# обекти.
- Включва DbContext (CatalogforMoviesDbContext), миграции, навигационни свойства и валидации с [Required], [Key] и др.

## 3. NUnit

- Използван за писане на unit тестове в TestingLayer.
- Позволява валидиране на CRUD операциите на всеки мениджър от DataLayer.

## **6.5. Използвани езици за програмиране**

проекта Catalog for Movies са използвани следните езици за програмиране:

### 1. C#

- Основният език за разработка на всички части от проекта:

- BusinessLayer (модели и бизнес логика)
- DataLayer (достъп до база данни с Entity Framework)
- ServiceLayer (услуги и фасаден достъп)
- PresentationLayer (Windows Forms GUI)
- TestingLayer (unit тестове с NUnit)

## 2. SQL (чрез Entity Framework)

- Използва се индиректно за взаимодействие с базата данни (чрез миграции, заявки и LINQ към SQL).
- Няма ръчно написан SQL код

### **6.6. Описание на приложението**

#### **6.6.1. MainForm**

При стартиране на приложението се отваря началната страница, която съдържа четири бутона - Movies, Genres, Viewers. При натискането на който и да е от тях, потребителят е препратен към страницата на определеният компонент.

MainForm

Movies

Genres

Viewers

## 6.6.2 Movie Form

MovieForm

Title

Director

Release Date

Genre

Review

Create

Read

Update

Delete

Clear

Формата `MovieForm` в приложението `Catalog for Movies` служи за управление на филмите в базата данни. Тя предоставя графичен интерфейс, чрез който потребителите могат да създават, преглеждат, редактират и изтриват филми. Ето как работи всяка част:

Компоненти на `MovieForm`:

**1. TextBoxes:**

- **Title:** потребителят въвежда заглавието на филма.
- **Director:** потребителят въвежда името на режисьора.

**2. DateTimePicker:**

- **ReleaseDate:** потребителят избира датата на премиера на филма.

**3. ComboBox:**

- Избира се жанрът на филма от предварително зареден списък с жанрове (прочетени от базата чрез `GenresManager.ReadAll()`).

**4. DataGridView:**



- Показва списък с всички филми, записани в базата данни. При избор на ред, данните се зареждат обратно във формата за редакция.

## **5. Бутони:**

- **Create:** създава нов филм и го записва в базата чрез `MoviesManager.Create()`.
- **Read:** презарежда списъка с всички филми.
- **Update:** актуализира избрания филм в базата.
- **Delete:** изтрива избрания филм от базата.
- **Clear:** изчиства всички полета от формата.

## **Логика на работа (code-behind, съкратено):**

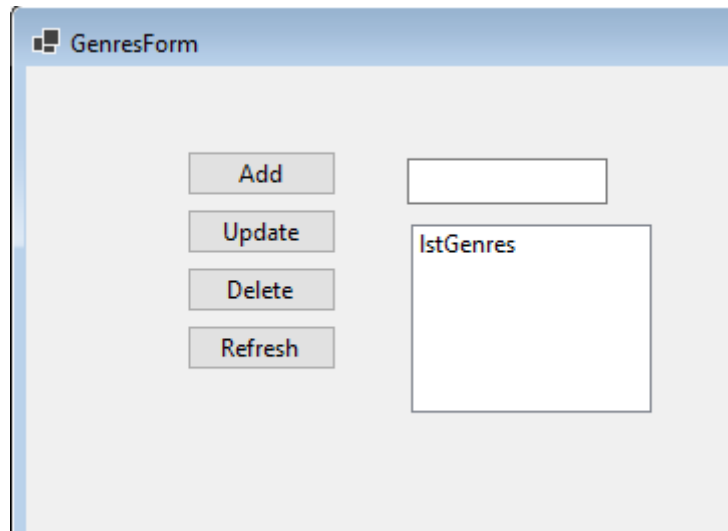
- При зареждане на формата се извикват методи за:
  - Зареждане на жанровете в `ComboBox`.
  - Зареждане на всички филми в `DataGridView`.

- При натискане на Create:
  - Валидират се полетата.
  - Създава се обект `Movie` с въведените данни.
  - Извиква се `MoviesManager.Create(movie)`.
  - Обновява се `DataGridView`.
- При избор на ред от `DataGridView`:
  - Данните на избрания филм се зареждат обратно във формата.

Допълнително:

- Формата комуникира само със `ServiceLayer`, който скрива `DataLayer`.
- `MovieForm` е напълно отделена от базата, като използва мениджърите за всички действия

### **6.6.3. GenresForm**



GenresForm във формата Catalog for Movies е отговорна за създаването и управлението на жанровете, които се използват от филмите. Това е една от най-основните форми, тъй като жанровете са свързани с други обекти в системата, включително Movie.

Компоненти на GenresForm:

1. TextBox:

- Name: потребителят въвежда името на жанра (напр. „Драма“, „Комедия“, „Научна фантастика“ и т.н.).

2. DataGridView:

- Показва списък с всички жанрове от базата данни.

- При избор на ред, информацията за жанра се зарежда обратно в TextBox-а за редакция.

### 3. Бутони:

- Create: създава нов жанр чрез `GenresManager.Create()`.
- Read: презарежда списъка с жанрове от базата.
- Update: редактира избрания жанр.
- Delete: изтрива избрания жанр от базата.
- Clear: изчиства текстовото поле и премахва селекцията от таблицата.

### Логика на работа:

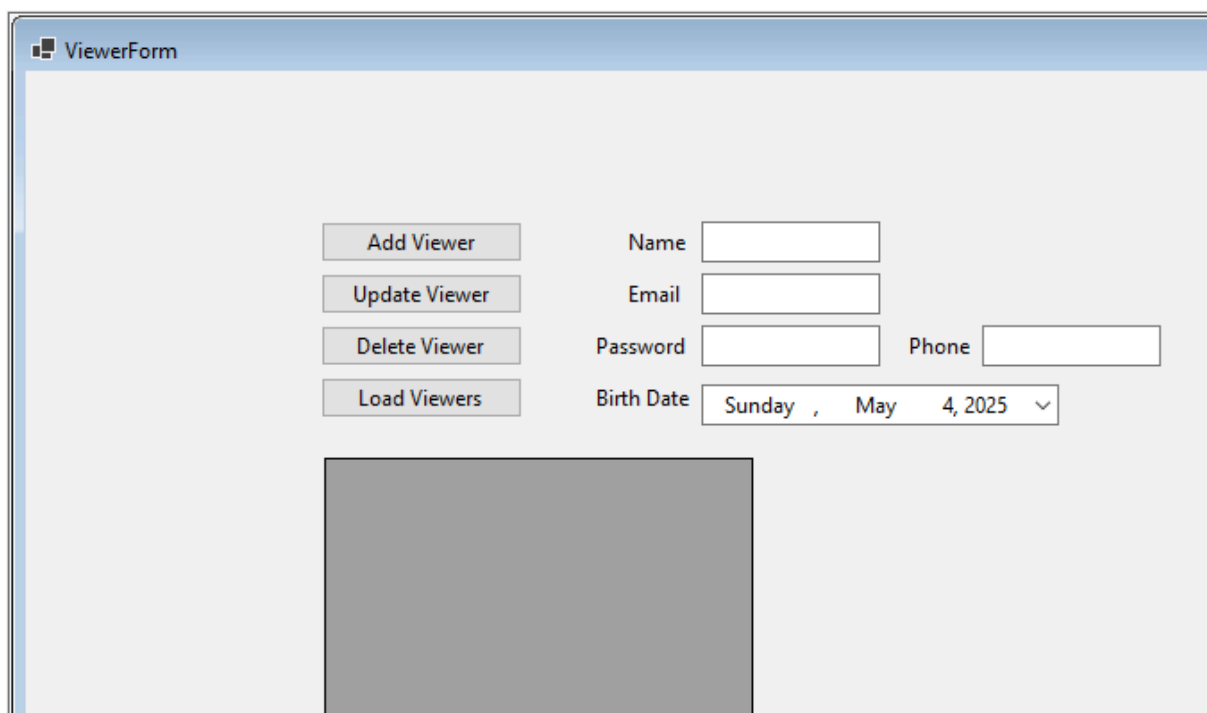
- При зареждане на формата:
  - `GenresManager.ReadAll()` извлича всички жанрове и ги зарежда в `DataGridView`.
- При натискане на бутона Create:

- Проверява се дали полето за име не е празно.
- Създава се нов обект Genre и се подава на GenresManager.Create().
- Обновява се таблицата (DataGridView) със GenresManager.ReadAll().
- При избор на ред от DataGridView:
  - Данните от реда се зареждат в полето за име, което позволява редакция или изтриване.
- При натискане на Update:
  - Жанрът се актуализира с новото име и GenresManager.Update() го записва в базата.
- При натискане на Delete:
  - Избраният жанр се премахва от базата и таблицата се обновява.

**Забележки:**

- GenresForm е форма без зависимости от други форми, но е критично свързана с MovieForm, тъй като жанровете се използват при създаване на филми.
- Формата работи чрез ServiceLayer, който осигурява изолация от DataLayer (GenresDBManager и CatalogforMoviesDBContext).
- Обикновено името на жанра трябва да бъде уникално, така че допълнителна валидация може да бъде полезна.

#### 6.6.4. ViewersForm



The screenshot shows a Windows-style application window titled "ViewerForm". Inside the window, there is a form for managing viewers. On the left side, there are four buttons stacked vertically: "Add Viewer", "Update Viewer", "Delete Viewer", and "Load Viewers". To the right of these buttons are input fields for "Name", "Email", "Password", and "Phone". Below these fields is a "Birth Date" field with a date picker showing "Sunday , May 4, 2025". At the bottom of the form is a large, empty rectangular box, likely intended for displaying a list of viewers or a detailed view of a selected viewer.

ViewersForm във формата Catalog for Movies е предназначена за създаване, преглед, редакция и изтриване на потребители (наричани "Viewers" в системата). Всеки Viewer представлява индивидуален потребител.

## Компоненти на ViewersForm:

### 1. TextBoxes:

- Name: име на потребителя.
- Email: имейл адрес.
- Password: парола.
- Phone: телефонен номер (незадължителен).

### 2. DateTimePicker:

- BirthDate: дата на раждане на потребителя.

### 3. DataGridView:

- Показва списък с всички регистрирани потребители.
- При избор на ред, данните се зареждат в съответните полета за редакция или изтриване.

### 4. Бутони:

- **Create:** създава нов Viewer с въведените данни.
- **Read:** презарежда таблицата с всички потребители от базата.
- **Update:** редактира избрания Viewer.
- **Delete:** изтрива избрания Viewer от базата.
- **Clear:** изчиства всички полета и премахва избора в таблицата.

#### Логика на работа:

- При зареждане на формата:
  - **ViewersManager.ReadAll()** извлича всички потребители и ги зарежда в **DataGridView**.
- При натискане на **Create**:
  - Събират се въведените данни от полетата.
  - Създава се нов **Viewer** обект и се подава към **ViewersManager.Create()**.



- Таблицата се обновява със всички потребители чрез `ReadAll()`.
- При избор на ред от `DataGridView`:
  - Данните се попълват обратно в текстовите полета и `DateTimePicker`-а.
  - Позволява редакция или изтриване на избрания `Viewer`.
- При натискане на `Update`:
  - Данните от формата се използват за актуализиране на съществуващ `Viewer`.
  - Извиква се `ViewersManager.Update()` и таблицата се обновява.
- При натискане на `Delete`:
  - Избраният `Viewer` се премахва от базата чрез `ViewersManager.Delete()`.

Допълнителна информация:

- Паролата не се криптира – добра идея е добавянето на криптиране на този етап.
- В бъдеще може да се добави валидация за уникалност на имейл адреса или защита от грешно форматиране.

#### Зависимости:

- Работи чрез ServiceLayer → ViewersManager, който използва DataLayer → ViewersDBManager.

## 6.7. Заключение

Проектът Catalog for Movies представлява настолно приложение, което успешно интегрира многослойна архитектура за ефективно управление на филми, жанрове, потребители и класации. С помощта на Entity Framework Core и Windows Forms, системата предлага стабилна база данни и интуитивен потребителски интерфейс, който улеснява търсенето, съхраняването и организирането на информация. Разделянето на проекта на слоеве – бизнес логика, достъп до данни, услуга и представяне – допринася за по-добра поддръжка, модулност и разширяемост на приложението. Чрез тази структура се осигурява ясно разграничение между отговорностите, което улеснява бъдещо развитие и добавяне на нови функционалности.

Приложението изпълнява своята основна цел – да предостави на потребителя бърз, лесен и удобен начин за откриване и категоризиране на филмово съдържание, като същевременно осигурява стабилна основа за бъдещи разширения и подобрения.