|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Министерство науки и высшего образования РФ | | | | | | | | | | | | |  |
|  | | | | | |  | | |  | | | | | |
|  | | | Пермский государственный национальный исследовательский университет | | | | | | | | |  | | |
|  | | | | | |  | | |  | | | | | |
|  | | | | ОтчЕт  по лабораторной работе №8 «Автоматизированное рабочее место»  по дисциплине «Учебная практика по базам данных» | | | | | | |  | | | |
|  | | | | | |  | | |  | | | | | |
|  | | Работу выполнил  студент гр. ПМИ-1,2  Барсамов Е.С. \_\_\_\_\_  «\_\_\_» \_\_\_\_\_\_\_\_ 2019 | | | | |  | Проверил  \_\_\_\_\_\_\_\_ кафедры МОВС  Постаногов И.С. \_\_\_\_\_\_\_\_\_  «\_\_\_» \_\_\_\_\_\_\_\_ 2019 | | | | |  | |
|  | | | | |  |  | | |  |  | | | | |
|  | | | | |  | Пермь 2019 | | |  |  | | | | |

Оглавление

[1 Постановка задачи 5](#_Toc19368869)

[2 Анализ предметной области, предлагаемая схема БД 6](#_Toc19368870)

[3 Решение задачи 10](#_Toc19368871)

[3.1 Выбранные средства решения задачи 10](#_Toc19368872)

[3.2 Описание графического интерфейса пользователя 10](#_Toc19368873)

[3.3 Создание графического интерфейса 15](#_Toc19368874)

[3.3 Объекты приложения, ORM 17](#_Toc19368875)

[3.4 Работа с данными, использование ORM 18](#_Toc19368876)

[3.5 Алгоритм добавления сущностей 21](#_Toc19368877)

[3.6 Алгоритм удаления сущностей 22](#_Toc19368878)

[3.7 Алгоритм обновления сущностей 22](#_Toc19368879)

[4 Тестирование 24](#_Toc19368880)

[4.1 Попытка входа в несуществующий аккаунт 24](#_Toc19368881)

[4.2 Попытка входа с неправильным паролем 24](#_Toc19368882)

[4.3 Вход под учётной записью оператора 24](#_Toc19368883)

[4.4 Смена учётной записи на учётную запись гостя 24](#_Toc19368884)

[4.5 Вход с помощью регистрации нового пользователя 24](#_Toc19368885)

[4.6 Смена учётной записи на учётную запись администратора 24](#_Toc19368886)

[4.7 Регистрация нового пользователя через таблицу пользователей 25](#_Toc19368887)

[4.8 Редактирование данных пользователя через таблицу пользователей 25](#_Toc19368888)

[4.9 Удаление пользователя через таблицу пользователей 25](#_Toc19368889)

[4.10 Попытка удаления своей учётной записи 25](#_Toc19368890)

[4.11 Попытка сменить роль своей учётной записи 25](#_Toc19368891)

[4.12 Неправильный ввод при добавлении (обновлении) пользователей 25](#_Toc19368892)

[4.13 Отображение пустой БД 25](#_Toc19368893)

[4.14 Добавление кинокомпании 26](#_Toc19368894)

[4.15 Добавление фильма с существующей кинокомпанией 26](#_Toc19368895)

[4.16 Редактирование кинокомпании 26](#_Toc19368896)

[4.17 Попытка удалить кинокомпанию, на которую ссылаются фильмы 26](#_Toc19368897)

[4.18 Добавление актёра 26](#_Toc19368898)

[4.19 Добавление контракта с существующими фильмом и актёром 26](#_Toc19368899)

[4.20 Редактирование актёра 26](#_Toc19368900)

[4.21 Редактирование фильма 26](#_Toc19368901)

[4.22 Попытка удалить фильм, на который ссылаются контракты 27](#_Toc19368902)

[4.23 Попытка удалить актёра, на которого ссылаются контракты 27](#_Toc19368903)

[4.24 Редактирование контракта с существующими фильмом и актёром 27](#_Toc19368904)

[4.25 Удаление контракта 27](#_Toc19368905)

[4.26 Удаление фильма 27](#_Toc19368906)

[4.27 Удаление кинокомпании 27](#_Toc19368907)

[4.28 Удаление актёра 27](#_Toc19368908)

[4.29 Контекстное добавление кинокомпании при добавлении (обновлении) фильма 28](#_Toc19368909)

[4.30 Контекстное добавление фильма при добавлении (обновлении) контракта 28](#_Toc19368910)

[4.31 Контекстное добавление актёра при добавлении (обновлении) контракта 28](#_Toc19368911)

[4.32 Неправильный ввод при добавлении (обновлении) фильма 28](#_Toc19368912)

[4.33 Неправильный ввод при добавлении (обновлении) актёра 28](#_Toc19368913)

[4.34 Неправильный ввод при добавлении (обновлении) кинокомпании 28](#_Toc19368914)

[4.35 Неправильный ввод при добавлении (обновлении) контракта 29](#_Toc19368915)

# 1 Постановка задачи

Написать программу, которая на графическом интерфейсе пользователя позволяет просматривать, добавлять, удалять, изменять записи и осуществлять поиск минимум 2 стандартных запросов в 4 связанных таблицах из спроектированной ранее базы данных предметной области без необходимости задания и отображения пользователю искусственных ключей. Между указанными таблицами должно присутствовать как отношение типа 1:М, так и отношение типа М:М.

При запуске программы пользователю должна отображаться форма аутентификации по логину и паролю, допускающая гостевой вход. Из формы аутентификации должна быть возможность перейти на форму регистрации нового логина и пароля для оператора АРМ.

Пользователь, зашедший в приложение под учётной записью администратора АРМ, должен иметь возможность просматривать, добавлять, удалять, изменять записи в 4 используемых таблицах из спроектированной ранее базы данных предметной области и в таблице учётных записей пользователей. Пользователь, зашедший в приложение под учётной записью оператора АРМ, должен иметь возможность просматривать, добавлять, удалять, изменять записи только в 4 используемых таблицах из спроектированной ранее базы данных предметной области. Пользователь, зашедший в приложение под гостевой учётной записью АРМ, должен иметь возможность только просматривать записи в 4 используемых таблицах из спроектированной ранее базы данных предметной области.

В пользовательском интерфейсе модификации должна предоставляться возможность контекстного пополнения связанных сущностей.

Средства стандартного поиска доступны всем категориям пользователей.

Провести тестирование корректности работы программы.

# 2 Анализ предметной области, предлагаемая схема БД

В своей работе приложение использует шесть таблиц из базы данных: пять из них были взяты из спроектированной при выполнении предыдущих лабораторных работ базы данных, ещё одна таблица – таблица пользователей – была создана. Из шести таблиц одна таблица – таблица стран - непосредственно не отображается пользователю, и используется только для организации удобного ввода страны посредством выбора нужной из раскрывающегося списка.

Таблица “companies” хранит в себе данные о кинокомпаниях, является одной из таблиц, которые будут отображаться в программе. С данной таблицей связана таблица “films”. Структура таблицы “companies” представлена ниже (таблица 2.1).

|  |  |
| --- | --- |
| companies | |
| id | Уникальный идентификатор компании, натуральное число. Обязательное поле. |
| name | Название компании, строка символов. Обязательное поле. |
| foundation\_year | Год основания компании, целое число. |
| address | Адрес кинокомпании, строка символов. |

Таблица 2.1 – Структура таблицы “companies”

Таблица “countries” предназначена для хранения списка стран, является вспомогательной и не отображается пользователю. Она изначально заполнена названиями стран, и именно из неё приложение получает список допустимых для ввода стран. При необходимости таблица может быть обновлена. Представляем её структуру ниже (таблица 2.2).

|  |  |
| --- | --- |
| countries | |
| id | Уникальный идентификатор страны, натуральное число. Обязательное поле. |
| name | Название страны, строка символов. |

Таблица 2.2 – Структура таблицы “countries”

Таблица “films” хранит в себе данные о фильмах, является ещё одной таблицей, которая будет отображаться в программе. Некоторые её поля были заменены: вместо идентификаторов страны, жанра, режиссёра и композитора были вставлены непосредственно названия страны, жанра, имена режиссёра и композитора. Это было сделано для большей информативности отображаемых данных. Структура данной таблицы представлена ниже (таблица 2.3).

|  |  |
| --- | --- |
| films | |
| id | Уникальный идентификатор фильма, натуральное число. Обязательное поле. |
| name | Название фильма, строка символов. Обязательное поле. |
| year | Год выпуска фильма, целое число. |
| country | Название страны-производителя, строка символов. |
| budget | Бюджет фильма, вещественное число. |
| age\_limit | Возрастное ограничение, целое число. |
| duration | Продолжительность фильма, целое число. |
| rating | Рейтинг фильма, целое число. |
| genre | Жанр фильма, строка символов. |
| company\_id | Идентификатор кинокомпании, выпустившей фильм – натуральное число. |
| director | Имя режиссёра, строка символов. |
| composer | Имя композитора, строка символов. |

Таблица 2.3 – Структура таблицы “films”

Таблица “actors” хранит в себе данные об актёрах, является ещё одной таблицей, которая будет отображаться в программе. Структура данной таблицы представлена ниже (таблица 2.4).

|  |  |
| --- | --- |
| actors | |
| id | Уникальный идентификатор актёра, натуральное число. Обязательное поле. |
| name | Имя актёра, строка символов. Обязательное поле. |
| birthday | Дата рождения актёра, в виде даты. |
| country | Название страны актёра, строка символов. |
| description | Описание актёра, строка символов. |

Таблица 2.4 – Структура таблицы “actors”

Таблица “films\_and\_actors” - связующая между таблицами “films” и “actors”, одна из отображаемых в приложении таблиц. Фактически, она предоставляет данные о контракте актёров на съёмки в конкретных фильмах. Структура данной таблицы представлена ниже (таблица 2.5).

|  |  |
| --- | --- |
| films\_and\_actors | |
| film\_id | Идентификатор фильма, натуральное число. Обязательное поле. |
| actor\_id | Идентификатор актёра, натуральное число. Обязательное поле. |
| actor\_honorarium | Гонорар данного актёра за съёмки в данном фильме, вещественное число. |

Таблица 2.5 – Структура таблицы “films\_and\_actors”

Отметим, что значения пар столбцов “film\_id” и “actor\_id” должны быть уникальными: данные о нескольких контрактах между одними и теми же субъектами храниться не могуть.

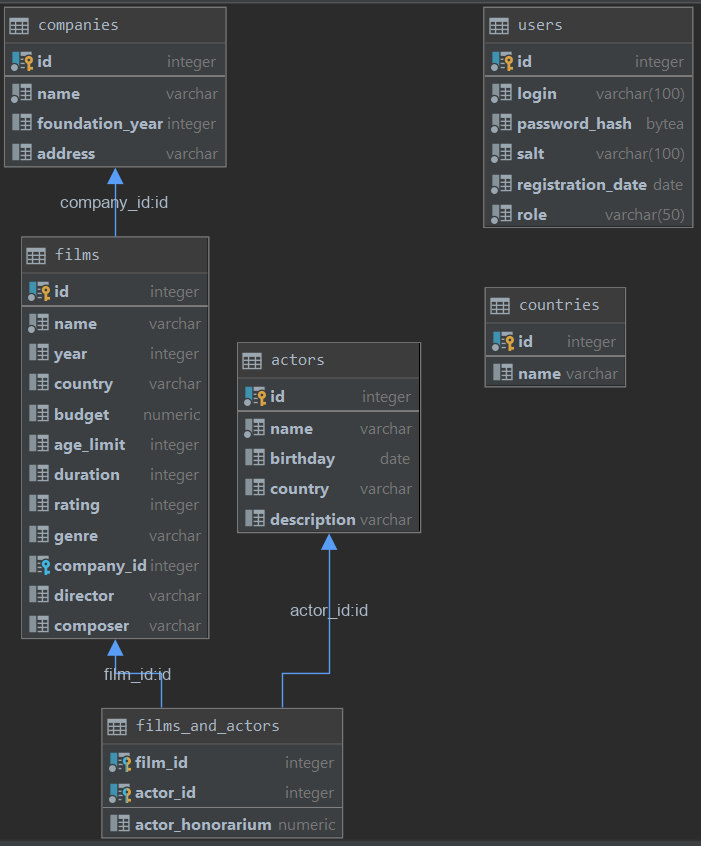
Таблица “users” – ещё одна отображаемая таблица. Данная таблица предоставляет данные о пользователях нашего приложения. Её структура представлена ниже (таблица 2.6).

|  |  |
| --- | --- |
| users | |
| id | Уникальный идентификатор пользователя, натуральное число. Обязательное поле. |
| login | Имя пользователя (логин), строка символов (максимум – 100). Обязательное, уникальное поле. |
| password\_hash | Хэш пароля, массив байт. Обязательное поле. |
| salt | “Соль” пароля, строка символов (максимум – 100). Обязательное поле. |
| registration\_date | Дата регистрации актёра, в виде даты. Обязательное поле. Значение по умолчанию – текущая дата. |
| role | Роль пользователя, определяющая его права (“admin”, “operator”, “guest”), строка символов (максимум – 50). Обязательное поле. |

Таблица 2.6 – Структура таблицы “users”

Ниже представляем схему описанной базы данных (рисунок 2.1).

Рисунок 2.1 – Схема используемой базы данных



# 3 Решение задачи

Задачей создаваемой программы является организация удобной работы с данными из базы данных, спроектированной в одной из предыдущих лабораторных работ. Кроме того, работать с приложением могут разные пользователи, возможности и права которых также могут различаться. Ниже приводим описание программы.

## 3.1 Выбранные средства решения задачи

Для решения задачи была выбрана среда программирования Microsoft Visual Studio 2017, язык программирования – C#. В качестве СУБД используется PostgreSQL.

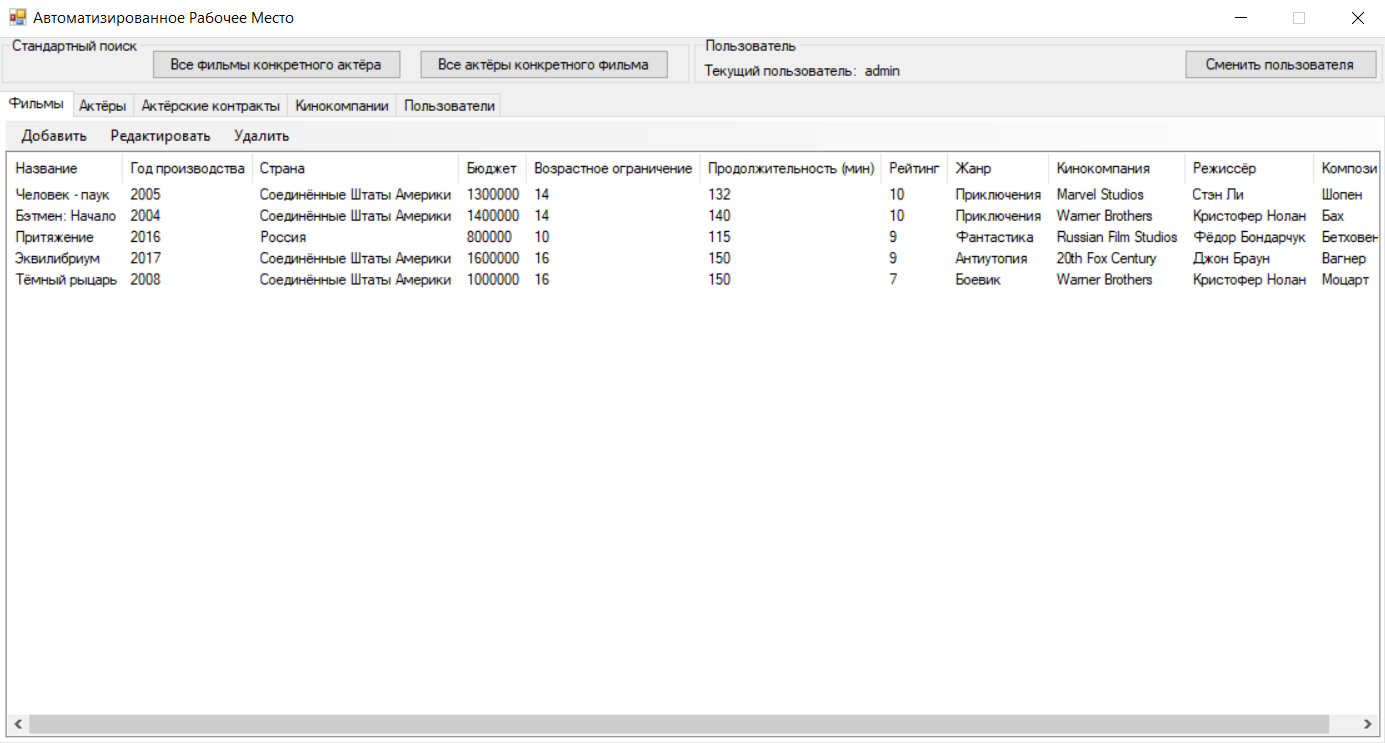
В проекте используется дополнительная зависимость “EntityFramework6.Npgsql”, необходимая для использования ORM-технологии доступа к данным “ADO.NET Entity Framework” вместе с СУБД PostgreSQL, “BouncyCastle” для хеширования пароля пользователя и “zxcvbn Core” для проверки сложности паролей. Данные компоненты были загружены с помощью встроенного менеджера пакетов NuGet. Кроме того, необходимо установить плагин PostgreSQL для Visual Studio.

Для создания графического интерфейса пользователя был выбран интерфейс программирования Windows Forms (WF).

Для работы с базой данных было использовано ORM-решение для .Net Framework от Microsoft - “ADO.NET Entity Framework”.

## 3.2 Описание графического интерфейса пользователя

Графический интерфейс пользователя представляет собой одно главное окно и несколько вспомогательных. В главном окне имеются переключаемые вкладки, каждая из которых соответствует таблице БД: “Фильмы”, “Кинокомпании”, “Актёры”, “Актёрские контракты” и “Пользователи”. В зависимости от активной в данный момент вкладки отображается нужная таблица. Выше области, содержащей вкладки с таблицами, расположены ещё две именованные области: “Стандартный поиск” и “Пользователь”. Первая предоставляет возможности стандартного поиска по данным, вторая – отображает имя текущего пользователя, а также позволяет сменить пользователя. (рисунок 3.1). Таблица “Пользователи”, отметим, доступна только пользователям-администраторам.

  
Рисунок 3.1 – Главное окно приложения

В каждой вкладке выше таблицы с данными расположено три кнопки. Кнопка “Удалить” удаляет выделенные элементы. Кнопка “Добавить” вызывает вспомогательное окно для заполнения данных о добавляемом элементе (пример добавления компании на рисунке 3.2), и в случае подтверждения ввода и прохождения проверки его правильности добавляет описанный элемент.

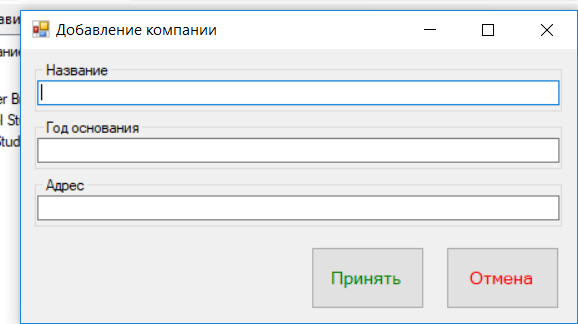
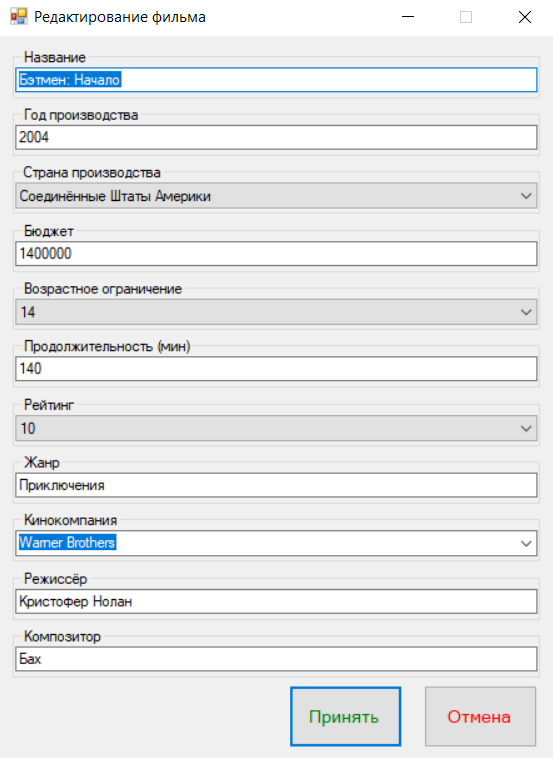


Рисунок 3.2 – Добавление новой кинокомпании

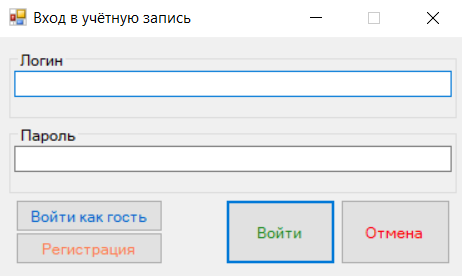
Кнопка “Изменить” вызывает вспомогательное окно для заполнения данных об изменяемом элементе, при этом заполняя поля ввода данными редактируемого элемента. (пример редактирования фильма представлен нарисунке 3.3), и в случае подтверждения ввода и прохождения проверки его правильности изменяет выбранный элемент. Заметим, что изменить можно сразу несколько элементов, выделив их и нажав кнопку “Изменить” (при этом окно для редактирования данных будет поочерёдно открыто для каждого элемента).

  
Рисунок 3.3 – Редактирование существующего фильма

Редактирование некоторых данных о фильме (страна производства, возрастное ограничение, рейтинг, кинокомпания) сделано в виде выпадающих списков. Это делает пользовательский ввод удобнее. Однако, в выпадающие списки, отображающие связанные данные (кинокомпания для фильма, актёр и фильм для контракта) допускается ввод – таким образом реализовано контекстное добавление новых связанных элементов. В целом, все вспомогательные окна выполнены в одном стиле, предоставляют те или иные формы ввода, назначение каждой из которых подписано рядом.

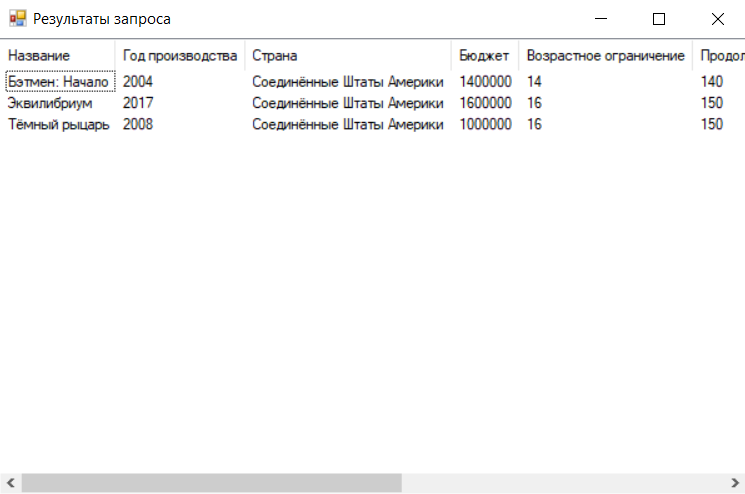
При изменении данных в таблицах автоматически подбирается размер колонок по их содержимому.

Кроме основного окна и окон добавления и редактирования отдельно выделим окно аутентификации пользователя (рисунок 3.4).

  
Рисунок 3.4 – Окно аутентификации

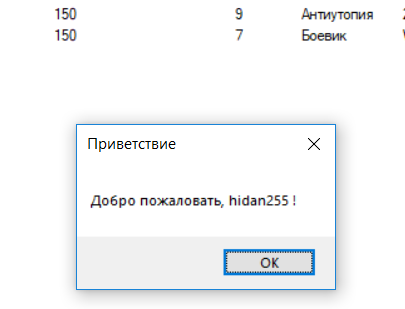
Окно аутентификации пользователя предоставляет формы для ввода имени пользователя и его пароля, а также даёт возможность войти как гость или зарегистрировать нового пользователя. Отметим, что возможность входа в качестве гостя без указания логина и пароля возможна только в том случае, если в БД существует пользователь с именем “guest”. Таким образом, администратор приложения может включать и выключать эту функцию при необходимости. Окно регистрации представляет собой простую форму ввода логина и пароля новой учётной записи.

Ещё одним особенным окном является окно вывода результатов поиска. Оно представляет собой большую таблицу с результатами (рисунок 3.5).

  
Рисунок 3.5 – Окно результатов поиска

Отметим также, что все окна, предназначенные для ввода данных, интерпретируют нажатие клавиши “Enter” как попытку принять введённые данные. Таким образом, вместо нажатия кнопки “Принять” при редактировании данных или кнопки “Войти” при смене пользователя можно просто нажимать клавишу “Enter”.

Окончание большинства действий программы (вход в учётную запись, удаление, редактирование, добавление данных) сопровождаются сообщениями, отмечающими успех или неудачу того или иного действия (пример входа в учётную запись представлен на рисунке 3.6).

  
Рисунок 3.6 – Вход в учётную запись

Размеры всех окон приложения фиксированы, их ручное масштабирование и разворачивание на весь экран запрещено.

## 3.3 Создание графического интерфейса

Размеры окон приложения были заданы при конструировании форм. Запрет на ручное масштабирование был установлен с помощью значения “FixedSingle” свойства “FormBorderStyle” (запрещает масштабирование с помощью перетаскивания границ формы). Запрет на разворачивание окон был установлен с помощью значения “False” свойства “MaximizeBox”. Переключаемые вкладки были созданы с помощью элемента управления “TabControl”. С помощью свойства “Name” данный элемент управления был переименован. С помощью редактора свойства “TabPages” (коллекции вкладок, каждая из которых имеет набор свойств) были настроены вкладки – поименованы и озаглавлены. Настройка вкладок представлена на рисунке 3.5.

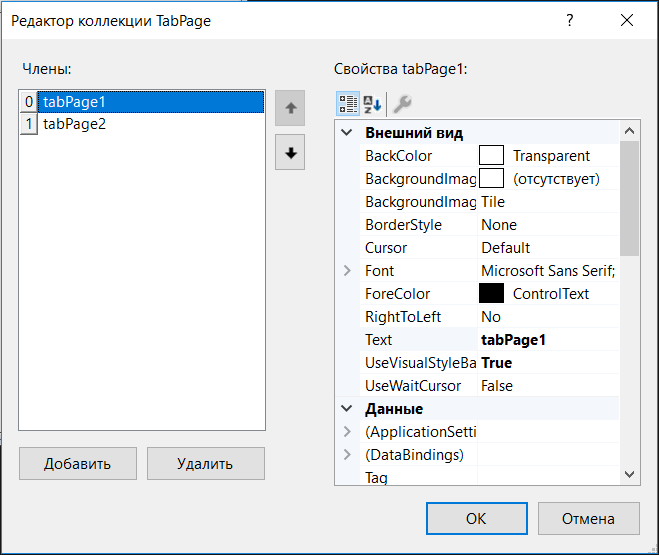


Рисунок 3.5 – Настройка вкладок элемента управления “TabControl”

В каждую из созданных вкладок был помещён элемент управления “MenuStrip” – меню, на котором расположились кнопки “Добавить”, “Изменить” и “Удалить”.

С помощью свойства “Name” эти меню были соответствующим образом поименованы. С помощью значения “Hidden” свойства “GripStyle” в меню был скрыт захват для изменения его положения, а с помощью значения “Top” свойства “Dock” меню было расположено сверху. Элементы меню были созданы в редакторе (рисунок 3.6), а затем с помощью свойства “Name” соответствующим образом поименованы.

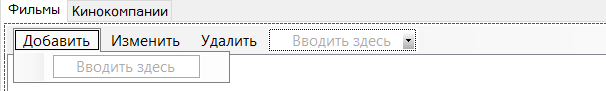


Рисунок 3.6 – Создание элементов меню в редакторе

После размещения во вкладках меню были добавлены элементы управления “ListView”. Каждый из них был закреплён в родительском контейнере (и после этого занял оставшееся место во вкладке. Сделано это с помощью всплывающего меню, вызываемого нажатием кнопки в форме треугольника над активным элементом управления – рисунок 3.7), и с помощью свойства “Name” соответствующим образом поименован. Также было выбрано представление “Details”.

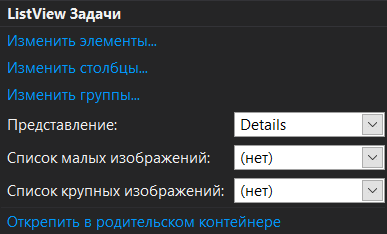


Рисунок 3.7 – Меню “ListView Задачи”

Остальная настройка элементов управления “ListView” была проведена в коде приложения.

Области “Стандартный поиск” и “Пользователь” были созданы с помощью элемента управления “GroupBox”.

Настойка вспомогательных окон представляет собой размещение нескольких кнопок и полей ввода. Все эти элементы управления так же были поименованы с помощью свойства “Name”.

Всплывающие списки “ComboBox” были использованы двух стилей: “DropDown” (ввод и выбор из выпадающего списка) и “DropDownList” (только выбор из выпадающего списка).

Ассоциация нажатия клавиши “Enter” с нажатием кнопки принятия введённых данных была задана с помощью свойства форм “AcceptButton”, который принимает в качестве значения кнопку, с которой ассоциировано нажатие клавиши.

## 3.3 Объекты приложения, ORM

ORM (англ. Object-Relational Mapping, рус. объектно-реляционное отображение) – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования и позволяет работать с данными из БД в терминах классов и объектов.

В данном приложении используется ORM-решение для .Net Framework от Microsoft - “ADO.NET Entity Framework”.

Так как база данных уже создана, и нам необходимо сгенерировать программный код для работы с сущностями, которые она хранит, мы будем использовать функцию ORM “Code First из БД”. Подробное описание всех проводимых действий приведено в отчёте к предыдущей лабораторной работе.

В результате применения функции “Code First из БД” в приложении будет сгенерирован весь необходимый программный код. Все созданные файлы отобразятся в окне обозревателя решений.

Напомним, что все поля сгенерированных классов объявлены в виде свойств. Помимо полей, объявленных в БД, классы содержат т.н. навигационные свойства – ссылки на сущности, связанные с ними. Класс “film” содержит навигационное свойство “company” – ссылка на кинокомпанию, связанную с данным фильмом. Класс “company”, напротив, содержит навигационное свойство “films” – коллекцию фильмов, выпущенных данной кинокомпанией.

Также классы “company”, “film” и “actor” были дополнены реализацией метода-заменителя для виртуального метода “ToString”. Это было сделано для того, чтобы классы нужным образом отображались в элементе управления “ComboBox”.

Кроме классов сущностей был автоматически создан ещё один класс – класс контекста данных. Данный класс является своего рода моделью БД, которая связывает все классы сущностей в один контекст и организует работу непосредственно с БД. Данный класс имеет имя, данное при создании модели – “DataContext”.

Помимо автоматически созданных классов, для работы построенной модели с СУБД PostgreSQL нужно создать дополнительный конфигурационный класс, код которого можно найти на официальной странице Npgsql, посвящённой Entity Framework 6. После его определения модель будет являться полностью работоспособной, и можно приступать к написанию логики приложения.

Для управления интерфейсом был создан класс “InterfaceManager”. Он содержит методы для инициализации таблиц, удаления и обновления строк, автоматического подбора ширины столбцов. Для работы с базой данных был создан класс “DataManager”. Он содержит методы для получения, вставки, обновления и удаления данных из БД. Ещё один класс – “UsersManager” – содержит методы, необходимые для работы с пользователями: получение соли пароля, генерация хэша и т.д. Все эти классы для удобства были помещены в папку MyWorkerClasses (“Добавить” => “Создать папку” в проекте в обозревателе решения). Папки образуют отдельные пространства имён, и для удобного использования их содержимого в коде нужно подключить эти пространства имён директивой “using”.

В двух словах опишем, как объекты взаимодействуют с таблицами интерфейса.

Фактически, в данной программе интерфейс отделён от всей остальной программной части, в том числе от объектов. Таблицы интерфейса заполняются строковыми представлениями полей объектов, а в соответствие каждой строке ставится идентификатор (первичный ключ) данной сущности. По идентификатору в последствии и происходит удаление данных из БД, их изменение: все методы, работающие с существующими данными из БД, принимают в качестве указателя на выбранный объект именно идентификаторы.

## 3.4 Работа с данными, использование ORM

Построенная модель даёт нам возможность использовать БД как совокупность связанных сущностей. Рассмотрим это подробнее.

Как было сказано выше, моделью БД в программном коде является класс контекста данных. Таким образом, для того, чтобы использовать данные из БД, необходимо создать объект данного контекста. Важно отметить, что после использования контекст необходимо закрывать. Именно поэтому контекст лучше объявлять и использовать в конструкции “using” (рисунок 3.8).

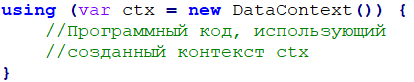


Рисунок 3.8 – Объявление контекста в конструкции “using”

Объект контекста данных имеет несколько типизированных коллекций типа “DBSet”, которые хранят сущности из таблиц базы данных. Каждой такой коллекции соответствует таблица в базе данных.

Данные коллекции можно обрабатывать с помощью цикла по коллекции “foreach”, или использовать метод “Find” для того, чтобы по первичному ключу найти нужный объект (метод принимает в качестве аргументов массив значений первичного ключа).

Добавление в коллекции происходит с помощью метода “Add”, который принимает в качестве аргумента добавляемый объект соответствующего типа (рисунок 3.9).

  
Рисунок 3.9 – Добавление фильма в коллекцию контекста “films”

Удаление из коллекции происходит с помощью метода “Remove”, который принимает в качестве аргумента удаляемый объект соответствующего типа (рисунок 3.10).

  
Рисунок 3.10 – Удаление фильма из коллекции контекста “films”

Однако, первоначально объекты, с которыми мы работаем в коде приложения, никак не связаны с сущностями, хранящимися в базе данных – это обычные программные объекты. Для того, чтобы установить связь конкретного объекта с сущностью, хранящейся в БД, используется метод “Attach” коллекции, в которой хранится данный объект (рисунок 3.11). Данный метод в качестве аргумента принимает объект соответствующего типа, и предназначен для прикрепления аргумента-объекта и связанных с ним объектов к контексту данных. После проведения данной операции все действия, происходящие с этим объектом (и со связанными объектами), будут отражаться на сущности из базы данных (в том числе изменение значений атрибутов, её удаление и связывание с другой сущностью). Неприкреплённые к контексту объекты считаются новыми для БД, и, к примеру, в случае связывания существующего (и прикреплённого к контексту) объекта с таковым прежде всего будет создана новая сущность на основе неприкреплённого объекта, и только затем будет установлена связь между ними.

  
Рисунок 3.11 – Прикрепление фильма к контексту данных

Все действия, проводимые с контекстом данных, его коллекциями и прикреплёнными к нему объектами отражаются на самом контексте данных, и первоначально не сохраняются в базе данных. Для того, чтобы сохранить все проведённые изменения в базе данных, необходимо вызвать метод “SaveChanges” контекста данных (рисунок 3.12).

  
Рисунок 3.12 – Сохранение изменений в БД

Отметим одну особенность. Как правило, новая сущность вставляется в БД без идентификатора. При этом, идентификатор необходим нам для того, чтобы ассоциировать его со строкой соответствующей таблицы. Оказывается, при вызове метода контекста “SaveChanges” добавление сущности генерирует к БД не только запрос на вставку новых данных, но и на выборку первичного ключа вставленных данных. Таким образом, идентификатор вставленного объекта обновится сам, и появится в соответствующем поле того объекта, который был передан параметром в метод “Add”.

Отдельно стоит обсудить удаление сущности, на который ссылаются другие сущности. Для того, чтобы запретить удаление таких объектов, перед непосредственно удалением мы обращаемся к навигационному свойству, представляющему зависимых объектов, и проверяем их количество в коллекции. Удаление допускается только в случае, если зависимых элементов нет.

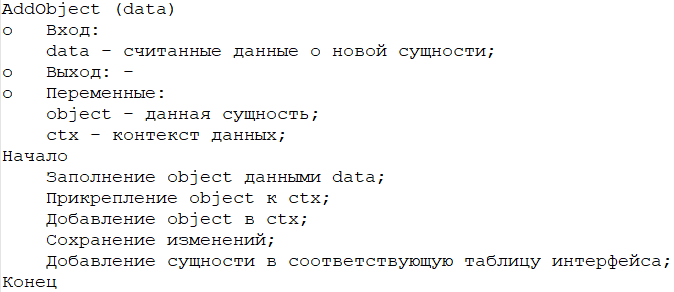
Также отметим, что Entity Framework по умолчанию не допускает проведение нескольких запросов к БД одновременно. Это может понадобиться при инициализации таблицы фильмов, когда во время чтения коллекции фильмов (выполняется в виде запроса) нужно получить название кинокомпании текущего фильма через навигационное свойство “company” класса “film” (также выполняется в виде запроса). Для этого первый запрос – коллекцию фильмов – можно материализовать (в данном случае – превратить в список) с помощью метода-расширения “ToList” для класса “DBSet” (находится в пространстве имён “System.Linq”).

Ещё одно замечание связано с использованием навигационных свойств. Отношения между сущностями в БД устанавливаются с помощью вставки соответствующих идентификаторов, без использования навигационных свойств. Это помогло избежать некоторых ошибок (к примеру, ошибки при повторном прикреплении к контексту объекта, который был прикреплён вместе с изменяемым объектом, т.к. уже связан с ним).

Важным замечанием также является то, что Entity Framework запрещает изменение значений первичного ключа объекта. Поэтому, для редактирования контрактов (которое подразумевает также и изменение значений первичного ключа) была выбрана следующая тактика: изменяемая запись удаляется из БД, и вместо неё вставляется новая, изменённая. Это никак не меняет суть операции, однако позволяет обойти существенное ограничение.

## 3.5 Алгоритм добавления сущностей

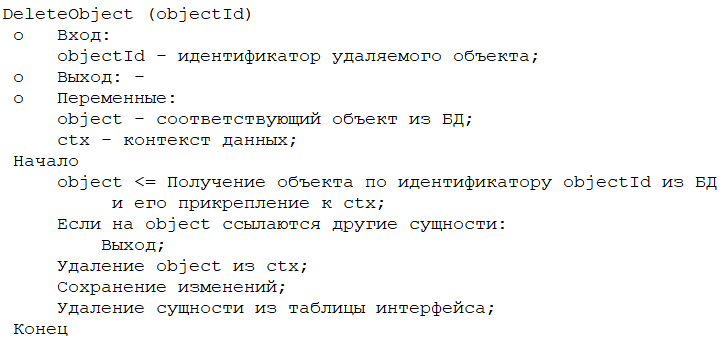
Приводим обобщённое описание алгоритма добавления сущностей на языке псевдокода (рисунок 3.13).

  
Рисунок 3.13 – Описание алгоритма добавления объекта

Отметим, что контекстное добавление связанных сущностей не входит в алгоритм: к моменту добавления основного объекта необходимые связанные объекты уже добавлены в БД, и всё, что необходимо сделать – это связать с ними добавляемый основной объект посредством указания идентификаторов в его соответствующих полях.

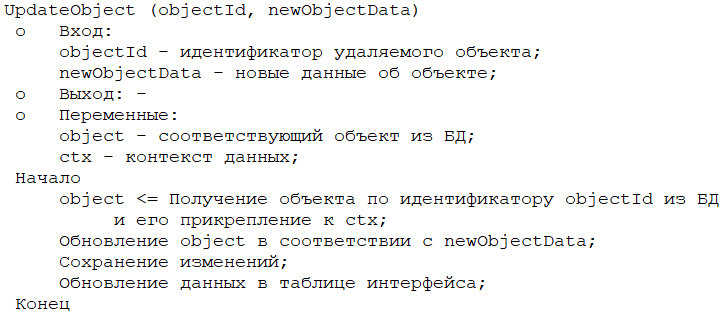
## 3.6 Алгоритм удаления сущностей

Опишем обобщённый алгоритм удаления сущностей на языке псевдокода (рисунок 3.14).

  
Рисунок 3.14 – Описание алгоритма удаления объекта

## 3.7 Алгоритм обновления сущностей

Приводим обобщённый алгоритм объектов на языке псевдокода (рисунок 3.15).

  
Рисунок 3.15 – Описание алгоритма обновления объекта

Контекстное добавление связанных сущностей также не входит в алгоритм обновления: к моменту обновления основного объекта необходимые для связи объекты уже созданы в БД, и необходимо только связать их.

# 4 Тестирование

Было проведено тестирование созданного приложения. Программа успешно прошла все тесты. Рассмотрим их.

## 4.1 Попытка входа в несуществующий аккаунт

Приложение корректно уведомило о том, что такого пользователя не существует.

## 4.2 Попытка входа с неправильным паролем

Приложение корректно уведомило о том, что был введён неправильный пароль.

## 4.3 Вход под учётной записью оператора

Вход был осуществлён, о чём уведомило приложение. На главном окне отобразилось четыре вкладки с таблицами (таблица “Пользователи” недоступна). Все кнопки действий активны, доступны средства стандартного поиска.

## 4.4 Смена учётной записи на учётную запись гостя

Вход под учётной записью гостя был произведён, о чём уведомило приложение. На главном окне отобразилось четыре вкладки с таблицами (таблица “Пользователи” недоступна). Кнопки действий с данными заблокированы, но доступны средства стандартного поиска.

## 4.5 Вход с помощью регистрации нового пользователя

Вход через регистрацию нового пользователя был произведён, о чём уведомило приложение. На главном окне отобразилось четыре вкладки с таблицами (таблица “Пользователи” недоступна). Все кнопки действий активны, доступны средства стандартного поиска.

## 4.6 Смена учётной записи на учётную запись администратора

Вход под учётной записью администратора был произведён, о чём уведомило приложение. На главном окне отобразилось пять вкладок с таблицами (таблица “Пользователи” доступна). Кнопки действий с данными активны, доступны средства стандартного поиска.

## 4.7 Регистрация нового пользователя через таблицу пользователей

Добавление нового пользователя через таблицу пользователей прошло успешно, вся информация о нём отобразилась в таблице. Вход под его учётной записью прошёл успешно.

## 4.8 Редактирование данных пользователя через таблицу пользователей

Редактирование данных существующего пользователя через таблицу пользователей прошло успешно, вся новая информация о нём отобразилась в таблице. Вход под его изменённой учётной записью прошёл успешно.

## 4.9 Удаление пользователя через таблицу пользователей

Удаление существующего пользователя через таблицу пользователей прошло успешно, информация о нём была удалена из таблицы. Вход под его учётной записью окончился неудачей.

## 4.10 Попытка удаления своей учётной записи

Попытка удалить собственную учётную запись закончилась неудачей (это запрещено), о чём корректно уведомило приложение.

## 4.11 Попытка сменить роль своей учётной записи

Менять собственную роль пользователя запрещено – список ролей заблокирован.

## 4.12 Неправильный ввод при добавлении (обновлении) пользователей

Были совершены попытки неправильного ввода при редактировании данных пользователя: пустой логин, изменение соли без изменения пароля. Приложение корректно уведомило о неправильном вводе.

## 4.13 Отображение пустой БД

Пустая база данных корректно отображается в таблицах приложения.

## 4.14 Добавление кинокомпании

Добавление новой кинокомпании было успешно завершено. Все её данные отобразились в таблице приложения.

## 4.15 Добавление фильма с существующей кинокомпанией

Добавление фильма с существующей кинокомпанией прошло успешно. Все его данные отобразились в таблице приложения.

## 4.16 Редактирование кинокомпании

Редактирование существующей кинокомпании прошло успешно. Все её новые данные отобразились в таблицах приложения, в том числе в таблице фильмов.

## 4.17 Попытка удалить кинокомпанию, на которую ссылаются фильмы

Попытка удалить кинокомпанию, на которую ссылаются фильмы, закончилась неудачей, о чём корректно уведомило приложение.

## 4.18 Добавление актёра

Добавление нового актёра было успешно завершено. Все его данные отобразились в таблице приложения.

## 4.19 Добавление контракта с существующими фильмом и актёром

Добавление нового контракта было успешно завершено. Все его данные отобразились в таблице приложения.

## 4.20 Редактирование актёра

Редактирование существующего актёра прошло успешно. Все его новые данные отобразились в таблицах приложения, в том числе в таблице контрактов.

## 4.21 Редактирование фильма

Редактирование существующего фильма прошло успешно. Все его новые данные отобразились в таблицах приложения, в том числе в таблице контрактов.

## 4.22 Попытка удалить фильм, на который ссылаются контракты

Попытка удалить фильм, на который ссылаются контракты, закончилась неудачей, о чём корректно уведомило приложение.

## 4.23 Попытка удалить актёра, на которого ссылаются контракты

Попытка удалить актёра, на который ссылаются контракты, закончилась неудачей, о чём корректно уведомило приложение.

## 4.24 Редактирование контракта с существующими фильмом и актёром

Редактирование существующего контракта прошло успешно. Все его новые данные отобразились в таблице приложения.

## 4.25 Удаление контракта

Удаление контракта было успешно завешено. Все данные о нём были удалены из таблицы приложения.

## 4.26 Удаление фильма

Удаление фильма было успешно завешено. Все данные о нём были удалены из таблицы приложения.

## 4.27 Удаление кинокомпании

Удаление кинокомпании было успешно завешено. Все данные о ней были удалены из таблицы приложения.

## 4.28 Удаление актёра

Удаление актёра было успешно завешено. Все данные о нём были удалены из таблицы приложения.

## 4.29 Контекстное добавление кинокомпании при добавлении (обновлении) фильма

Фильм с новой компанией был успешно добавлен, все данные о нём были отображены в таблице приложения. Связанная компания с заданным названием была также добавлена, и информация о ней появилась в соответствующей таблице приложения.

## 4.30 Контекстное добавление фильма при добавлении (обновлении) контракта

Контракт с новым фильмом был успешно добавлен, все данные о нём были отображены в таблице приложения. Связанный фильм с заданным названием был также добавлен, и информация о нём появилась в соответствующей таблице приложения.

## 4.31 Контекстное добавление актёра при добавлении (обновлении) контракта

Контракт с новым актёром был успешно добавлен, все данные о нём были отображены в таблице приложения. Связанный актёр с заданным именем был также добавлен, и информация о нём появилась в соответствующей таблице приложения.

## 4.32 Неправильный ввод при добавлении (обновлении) фильма

Были совершены попытки неправильного ввода при редактировании данных фильма: пустые поля ввода, несоответствие введённых данных необходимым типам данных. Приложение корректно уведомило о неправильном вводе.

## 4.33 Неправильный ввод при добавлении (обновлении) актёра

Были совершены попытки неправильного ввода при редактировании данных актёра, а именно - пустые поля ввода. Приложение корректно уведомило о неправильном вводе.

## 4.34 Неправильный ввод при добавлении (обновлении) кинокомпании

Были совершены попытки неправильного ввода при редактировании данных кинокомпании: пустые поля ввода, несоответствие введённых данных необходимым типам данных. Приложение корректно уведомило о неправильном вводе.

## 4.35 Неправильный ввод при добавлении (обновлении) контракта

Были совершены попытки неправильного ввода при редактировании данных контракта, а именно - пустые поля ввода. Приложение корректно уведомило о неправильном вводе.