Министерство образования и науки Российской Федерации

**Муромский институт (филиал)**

федерального государственного бюджетного образовательного учреждения высшего образования

**«Владимирский государственный университет**

**Имени Александра Григорьевича и Николая Григорьевича Столетовых»**

**(МИ ВлГУ)**

Факультет ИТР

Кафедра: ПИн

Курсовая работа

По: *Системы управления базами данных*

Тема: Информационная система хранения и просмотра фильмов

Руководитель



Колпаков А.А.

(Оценка)

(фамилия, инициалы)

(подпись) (дата)

Члены комиссии

Студент ПИН-122

(группа)

Сергеев Д.Д.

(Подпись Ф.И.О.)

(фамилия, инициалы)

(Подпись Ф.И.О.)

(подпись) (дата)

Муром 2025

**Содержание**

[**Введение** 4](#_Toc186159678)

[**1. Анализ технического задания** 5](#_Toc186159679)

[**2. Разработка моделей данных** 8](#_Toc186159680)

[**3. Разработка и реализация АИС** 12](#_Toc186159681)

[**3.1 Структура базы данных** 12](#_Toc186159682)

[**3.2 Взаимодействие с базой данных** 13](#_Toc186159683)

[**3.3 Структура программы** 13](#_Toc186159684)

[**3.4 Инструкция пользователя** 33](#_Toc186159685)

[**4. Тестирование АИС** 38](#_Toc186159686)

[**Заключение** 51](#_Toc186159689)

[**Список литературы** 52](#_Toc186159690)

[**Приложения** 53](#_Toc186159691)

[**Приложение А** 53](#_Toc186159692)

[**Приложение Б** 54](#_Toc186159693)

[**Приложение В** 55](#_Toc186159694)

**Введение**

Вступление: В современном мире библиотеки должны адаптироваться к быстро меняющимся условиям и ожиданиям пользователей. Внедрение автоматизированных информационных систем (АИС) становится важным для обеспечения удобного доступа к ресурсам. АИС оптимизируют процессы учета, что повышает качество библиотечных услуг и улучшает организационную структуру библиотек.

Актуальность работы: Современные пользователи ожидают оперативного доступа к фильмам, удобного поиска и возможности сортировки по различным критериям. Поэтому оптимизация процессов хранения и поиска информации о фильмах с использованием АИС является актуальной задачей, позволяющей улучшить взаимодействие пользователей с системой и повысить эффективность управления коллекцией видеоматериалов.

Цель работы: АИС для хранения и поиска фильмов.

Задачи:

* Проанализировать существующие АИС для хранения и поиска фильмов, выявить их преимущества и недостатки;
* Определить функциональные требования к разрабатываемой системе;
* Разработать архитектуру автоматизированной информационной системы для хранения и поиска фильмов;
* Описать основные алгоритмы работы системы.

Объект исследования: АИС для хранения и поиска фильмов.

Предмет исследования: Проектирование АИС для хранения и поиска фильмов.

Методы, используемые в работе:

* Анализ литературы по теме автоматизации процессов хранения и поиска информации о фильмах;
* Сравнительный анализ существующих АИС для фильмов;
* Метод проектирования для разработки архитектуры системы;
* Метод моделирования для описания алгоритмов работы системы.

Теоретическая важность: Теоретическая значимость работы заключается в систематизации знаний о современных подходах к автоматизации процессов хранения и поиска фильмов и разработке эффективных АИС.

Практическая важность: Практическая значимость работы заключается в применении разработанной АИС в киноархивах, библиотеках, а также на платформах потокового видео, что улучшит качество обслуживания пользователей, оптимизирует внутренние процессы и повысит общую эффективность работы с видеоматериалами.

Структура работы: В первой части работы представлен анализ требований к системе и ее описание. Во второй части работы содержатся модели АИС и примеры её работы.

**1. Анализ технического задания**

Согласно техническому заданию на курсовую работу необходимо разработать автоматизированную информационную систему (АИС) для хранения и поиска фильмов.

Информационная система хранения и поиска фильмов предназначена для автоматизации процессов управления коллекцией фильмов, учета пользователей и реализации функций поиска и бронирования контента. Основными пользователями системы являются администраторы и зрители. Система должна обеспечивать эффективное взаимодействие между этими группами пользователей, способствуя улучшению пользовательского опыта.

Для успешного проектирования информационной системы необходимо провести исследование потребностей потенциальных пользователей. Администраторы нуждаются в удобном интерфейсе для учета фильмов, а также в автоматизации процессов добавления, обновления и удаления записей о фильмах. Зрители хотят иметь возможность быстро находить и бронировать фильмы, а также создавать запросы на добавление новых фильмов в коллекцию.

Ожидается, что система позволит снизить трудозатраты на выполнение рутинных операций, повысит качество обслуживания зрителей и оптимизирует управление коллекцией фильмов.

Требования к базе данных:

* должны содержаться сведения о фильмах, жанрах, актерах, пользователях;
* должно быть предусмотрено хранение изображений;
* должны быть реализованы хранимые процедуры или триггеры;
* должно быть не менее 50 записей.

Требования к приложению:

* должно обеспечивать надежное и безопасное соединение с базой данных;
* должно иметь понятный и удобный графический интерфейс;
* должно включать механизмы для автоматической диагностики ошибок, предоставляя пользователю понятные сообщения об ошибках;
* должны быть реализованы классы и методы, позволяющие эффективно извлекать, изменять и удалять информацию из таблиц базы данных;
* должно вести временной учёт просмотра фильмов.

Для данной работы была выбрана база данных PostgreSQL[2]. Рассмотрим основные критерии выбора:

* поддерживает реляционную модель данных, что обеспечит адекватное представление информации о книгах, читателях и операциях;
* обеспечивает высокую производительность обработки запросов даже при увеличении объема данных (например, при добавлении новых книг или читателей);
* позволяет легко масштабировать систему и добавлять новые функции в будущем (например, интеграция с электронными ресурсами);
* является бесплатной и открытой СУБД, что снижает затраты на лицензирование.

Для более обоснованного выбора рассмотрим также две альтернативные базы данных: MySQL и Microsoft SQL Server. MySQL:

* реляционная модель данных: MySQL также поддерживает реляционную модель данных, однако в некоторых случаях может иметь ограничения по функциональности по сравнению с PostgreSQL, особенно в области поддержки сложных запросов и транзакций;
* производительность: MySQL известен своей высокой производительностью при чтении данных, что делает его отличным выбором для приложений, ориентированных на операции чтения. Однако при сложных запросах и больших объемах данных его производительность может уступать PostgreSQL;
* лицензирование: MySQL также имеет бесплатную версию (Community Edition), но для доступа к полному набору функций может потребоваться платная версия, что увеличивает затраты на лицензирование;
* реляционная модель данных: Microsoft SQL Server поддерживает реляционную модель данных и предлагает широкий спектр функций для работы с данными, включая продвинутые аналитические возможности;
* SQL Server обеспечивает высокую производительность и оптимизацию запросов, но в некоторых случаях может требовать значительных ресурсов сервера, особенно при больших объемах данных;
* Microsoft SQL Server является коммерческой СУБД, что подразумевает значительные затраты на лицензирование.
* В заключение, выбор PostgreSQL обоснован его высокой производительностью, масштабируемостью, поддержкой реляционной модели данных и отсутствием затрат на лицензирование. Эти факторы делают PostgreSQL идеальным выбором для разработки системы управления библиотекой.

При выборе технологий для создания приложения важно учитывать их особенности и преимущества по сравнению с альтернативами.

Windows Forms против WPF. Хотя WPF (Windows Presentation Foundation) является более современным решением для создания настольных приложений с поддержкой MVVM и богатых графических интерфейсов, Windows Forms[3] предоставляет более простую и интуитивно понятную среду разработки. Для небольших или средних приложений, которые требуют быстрого прототипирования и разработки, Windows Forms может оказаться более подходящим выбором. Кроме того, наличие большого количества готовых компонентов и библиотек для Windows Forms позволяет ускорить процесс разработки.

Npgsql против Entity Framework. Npgsql, как .NET Data Provider для PostgreSQL[1], предоставляет более низкоуровневый доступ к базе данных, чем Entity Framework (EF). Хотя EF облегчает работу с базами данных, используя ORM (Object-Relational Mapping), он может добавлять лишнюю абстракцию и снижение производительности в некоторых случаях. Если ваше приложение требует быстрого доступа к данным и высокой производительности, Npgsql позволяет тонко настраивать запросы и управлять транзакциями более эффективно.

Выбор C# Windows Forms, Npgsql для разработки приложения обусловлен их удобством, производительностью и необходимыми функциональными возможностями. Эти технологии обеспечивают эффективное сочетание простоты и мощности, что особенно важно для учета фильмов.

**2. Разработка моделей данных**

В данной работе инфологическое проектирование построено на функциональном подходе к проектированию БД. Этот метод реализует принцип "от задач" и применяется тогда, когда известны функции некоторой группы лиц и/или комплекса задач, для обслуживания информационных потребностей которых создаётся рассматриваемая БД.

Функциональные требований к системе описаны в формате UseCase (рисунок 1).

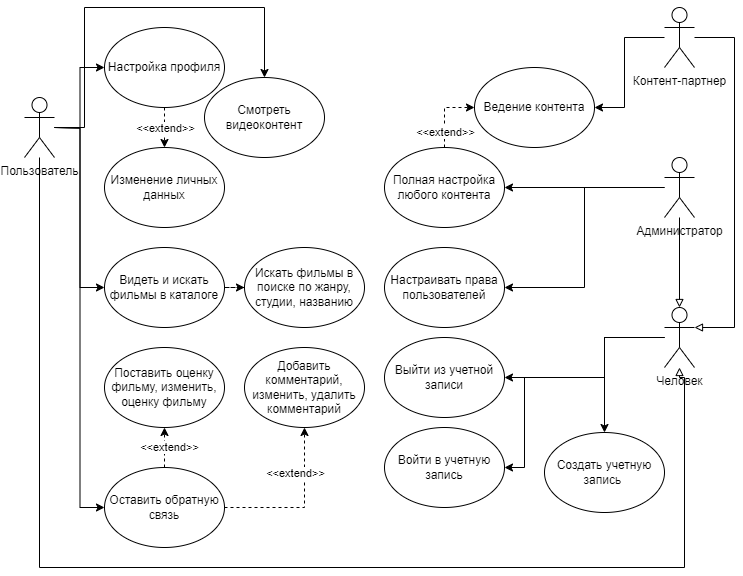


Рисунок 1 – UseCase диаграмма для АИС хранения и поиска фильмов.

На этапе логического проектирования была создана логическая структура базы данных, которая представлена на рисунке А.1.

На основе логической модели базы данных была создана физическая модель (рисунок Б.1).

Далее представлены таблицы, описывающие структуру таблиц базы данных.

Таблица 1. Пояснение к таблице Administrator

|  |  |
| --- | --- |
| Наименование столбца | Пояснение |

| 1 | 2 |
| --- | --- |
| administrator\_id | Идентификатор записи. Первичный ключ таблицы |
| login | Логин |
| hash\_password | Хэш-пароль |
| firstname | Имя администратора |
| lastname | Фамилия администратора |
| patronymic | Отчество администратора |
| birthday\_date | Дата рождения администратора |
| registration\_date | Дата создания записи |
| role | Роль пользователя |

Таблица 2. Пояснение к таблице Comment

|  |  |
| --- | --- |
| Наименование столбца | Пояснение |

| 1 | 2 |
| --- | --- |
| comment\_id | Идентификатор записи. Первичный ключ таблицы |
| content\_id | Идентификатор записи. Внешний ключ таблицы |
| comment\_text | Название книги |
| comment\_date | Дата создания записи |

Таблица 3. Пояснение к таблице Content

|  |  |
| --- | --- |
| Наименование столбца | Пояснение |

| 1 | 2 |
| --- | --- |
| content\_id | Идентификатор записи. Первичный ключ таблицы |
| name | Название |
| studio | Студия |
| genre | Жанр |
| description | Описание контента |
| content\_partner\_id | Идентификатор записи. Внешний ключ таблицы |
| url\_video | Поле для хранения ссылки на видео |
| image | Поле для хранения обложки фильма |
| rating | Средняя оценка контента |

Таблица 4. Пояснение к таблице ContentPartner

|  |  |
| --- | --- |
| Наименование столбца | Пояснение |

| 1 | 2 |
| --- | --- |
| content\_partner\_id | Идентификатор записи. Первичный ключ таблицы |
| login | Логин |
| hash\_password | Хэш-пароль |
| firstname | Имя контент-партнера |
| lastname | Фамилия контент-партнера |
| patronymic | Отчество контент-партнера |
| birthday\_date | Дата рождение контент-партнера |
| registration\_date | Дата создания записи |
| role | Роль пользователя |

Таблица 5. Пояснение к таблице Genre

|  |  |
| --- | --- |
| Наименование столбца | Пояснение |

| 1 | 2 |
| --- | --- |
| name | Название жанра |

Таблица 6. Пояснение к таблице Person

|  |  |
| --- | --- |
| Наименование столбца | Пояснение |

| 1 | 2 |
| --- | --- |
| person\_id | Идентификатор записи. Первичный ключ таблицы |
| firstname | Имя персоны |
| lastname | Фамилия персоны |
| patronymic | Отчество персоны |
| birthday\_date | Дата рождения персоны |

Таблица 7. Пояснение к таблице PersonRole

|  |  |
| --- | --- |
| Наименование столбца | Пояснение |

| 1 | 2 |
| --- | --- |
| person\_id | Идентификатор записи. Внешний ключ таблицы |
| content\_id | Идентификатор записи. Внешний ключ таблицы |
| role\_id | Идентификатор записи. Внешний ключ таблицы |

Таблица 8. Пояснение к таблице Rate

|  |  |
| --- | --- |
| Наименование столбца | Пояснение |

| 1 | 2 |
| --- | --- |
| rate\_id | Идентификатор записи. Первичный ключ таблицы |
| user\_id | Идентификатор записи. Внешний ключ таблицы |
| content\_id | Идентификатор записи. Внешний ключ таблицы |
| rate\_value | Значение оценки |
| rating\_date | Дата создания записи |

Таблица 9. Пояснение к таблице RegistrationCode

|  |  |
| --- | --- |
| Наименование столбца | Пояснение |

| 1 | 2 |
| --- | --- |
| role | Номер заказа. Первичный ключ |
| value | Значение кода регистрации |

Таблица 10. Пояснение к таблице Role

|  |  |
| --- | --- |
| Наименование столбца | Пояснение |

| 1 | 2 |
| --- | --- |
| role\_id | Идентификатор записи. Первичный ключ таблицы |
| description | Описание роли |

Таблица 11. Пояснение к таблице Studio

|  |  |
| --- | --- |
| Наименование столбца | Пояснение |

| 1 | 2 |
| --- | --- |
| name | Название студии |

Таблица 12. Пояснение к таблице User

|  |  |
| --- | --- |
| Наименование столбца | Пояснение |

| 1 | 2 |
| --- | --- |
| user\_id | Идентификатор записи. Первичный ключ таблицы |
| login | Логин |
| hash\_password | Хэш-пароль |
| registration\_date | Дата создания записи |
| role | Роль пользователя |

Таблица 13. Пояснение к таблице View

|  |  |
| --- | --- |
| Наименование столбца | Пояснение |

| 1 | 2 |
| --- | --- |
| view\_id | Идентификатор записи. Первичный ключ таблицы |
| user\_id | Идентификатор записи. Внешний ключ таблицы |
| content\_id | Идентификатор записи. Внешний ключ таблицы |
| view\_date | Дата создания записи |

**3. Разработка и реализация АИС**

**3.1 Структура базы данных**

В базе данных всего 13 таблиц, описание которых приводилось в предыдущем пункте. Также в базе данных были созданы триггер rate\_update\_trigger и триггерная функция update\_average\_rating. Триггер срабатывает после добавления записи в таблицу Rate или после её редактирования. Он запускает функцию, которая высчитывает среднее значение рейтинга на фильме.

Код создания триггера rate\_update\_trigger

CREATE TRIGGER rate\_update\_trigger

AFTER INSERT OR UPDATE ON public."Rate"

FOR EACH ROW

EXECUTE FUNCTION update\_average\_rating();

Код создания триггерной функции update\_average\_rating

CREATE OR REPLACE FUNCTION public.update\_average\_rating()

RETURNS trigger

LANGUAGE 'plpgsql'

COST 100

VOLATILE NOT LEAKPROOF

AS $BODY$

DECLARE

avg\_rating double precision;

BEGIN

-- Вычисляем средний рейтинг для конкретного контента

SELECT AVG(rate\_value)

INTO avg\_rating

FROM public."Rate"

WHERE content\_id = NEW.content\_id;

-- Обновляем поле рейтинга в таблице Content

UPDATE public."Content"

SET rating = avg\_rating

WHERE content\_id = NEW.content\_id;

RETURN NEW;

END;

$BODY$;

ALTER FUNCTION public.update\_average\_rating()

OWNER TO postgres;

**3.2 Взаимодействие с базой данных**

Для взаимодействия с базой данных из программы происходит с помощью команд библиотеки Npgsql.

Соединение с базой данных происходит следующим образом. Устанавливается строка соединения:

string connectionString = "Host = localhost; Username = postgres; Password = 1234; Database = newDB"

Далее выполняется команда:

var connection = new NpgsqlConnection(connectionString)

Для взаимодействия с базой данных используются простые запросы. Для получения данных из таблицы используется SELECT запрос, для обновления таблицы – UPDATE запрос, для вставки новых значений в таблицу – INSERT запрос и для удаления записи – DELETE запрос.

**3.3 Структура программы**

В программе содержится 13 форм и 14 разработанных классов. Сначала рассмотрим структуру и предназначение форм.

Форма Main содержит 5 кнопок и текстовое поле, два комбобокса а также таблица с фильмами(рисунок 2).

Таблица при нажатии пересылает на форму с фильмом соответственно с нажатой строкой

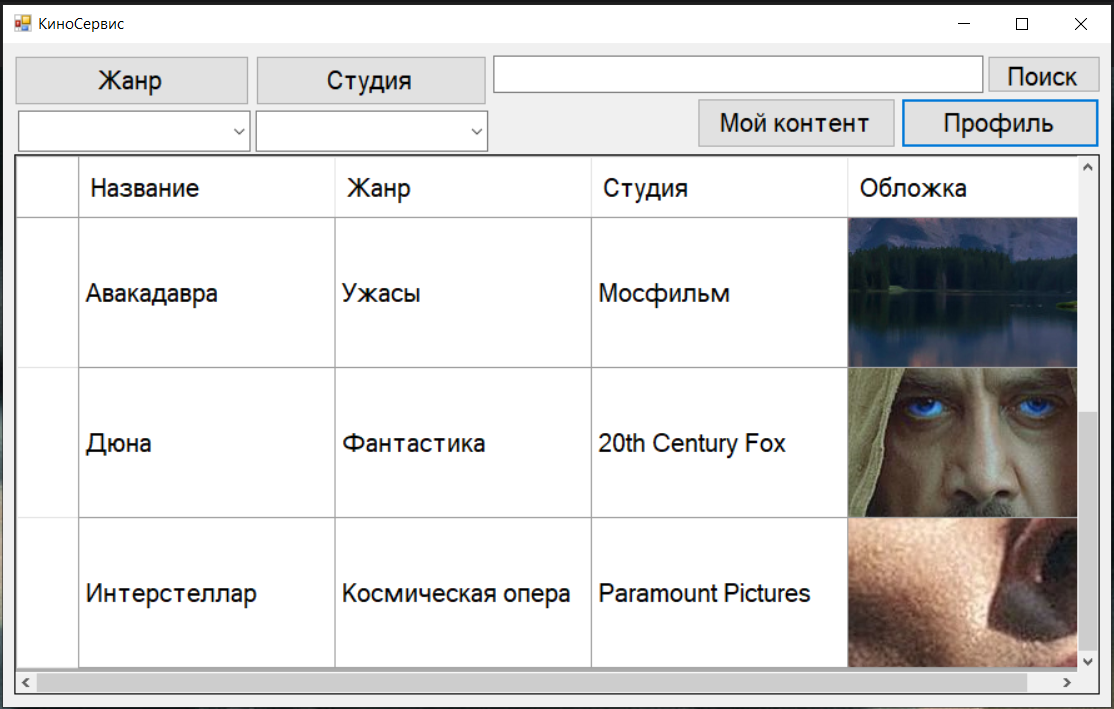


Рисунок 2 – Форма Main

Форма Readers\_form содержит 2 текстовых поля, элемент DataGridView и 5 кнопок (рисунок 3).

* текстовые поля информируют пользователя о том, что представлено в таблице и что нужно сделать;
* элемент DataGridView выводит информацию о читателях;
* кнопки «Редактировать» и «Добавить» открывают форму Reader\_form;
* кнопка «Удалить» удаляет выбранную запись как на элементе, так и в таблице в базе данных;
* кнопка «Бронь» открывает форму Bookings\_form;
* кнопка «Заказ» открывает форму Orders\_form.

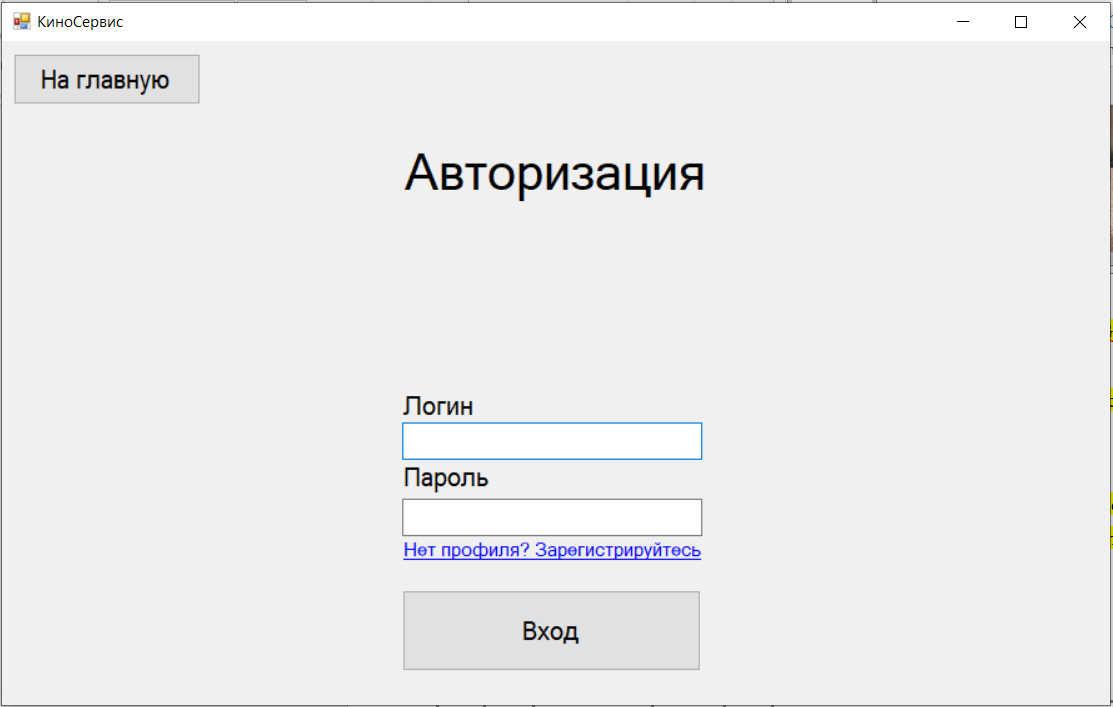


Рисунок 3 – Форма Login

Форма Reader\_form содержит 2 текстовых полей и 1 кнопки (рисунок 4).

* половина текстовых полей являются информирующими читателя, что нужно писать в другую половину текстовых полей;
* кнопка «Сохранить изменения» сохраняет информацию из текстовых полей в базе данных;
* кнопка «Пароль» генерирует логин читателя и открывает форму PasswordInput\_form.

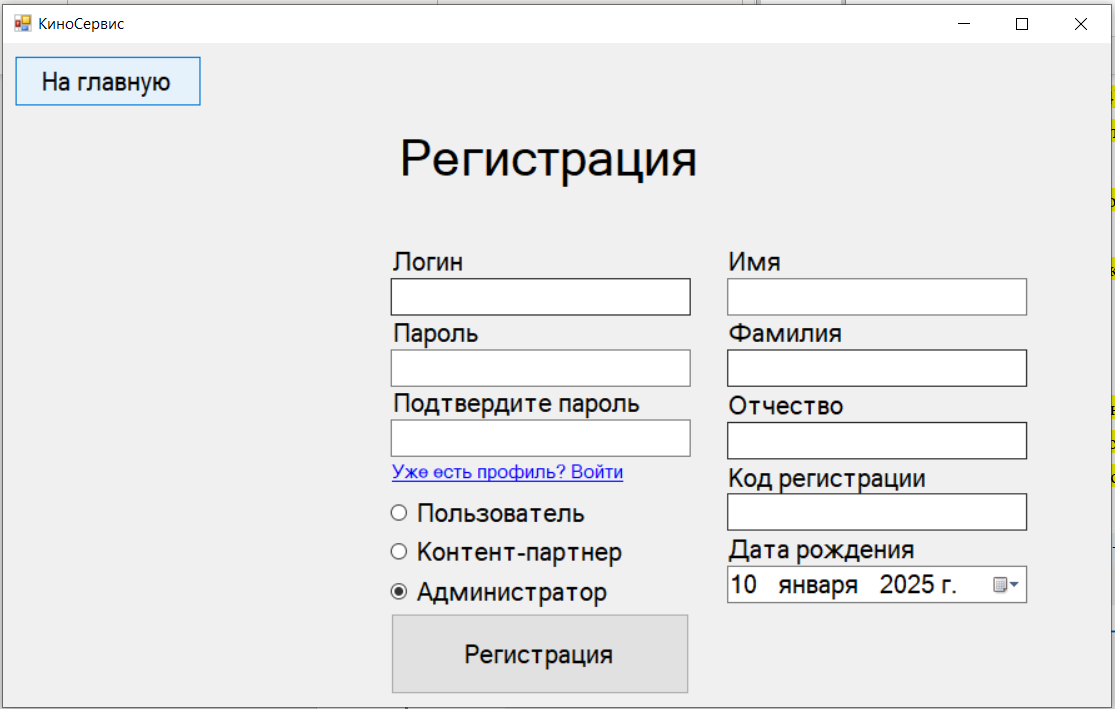


Рисунок 4 – Форма Registration

Форма PasswordInput\_form содержит 2 текстовых поля и 1 кнопку (рисунок 5).

* первое текстовое поле описывает второе, в которое вводится пароль;
* кнопка «Сохранить изменения» сохраняет информацию из текстового поля в базе данных.

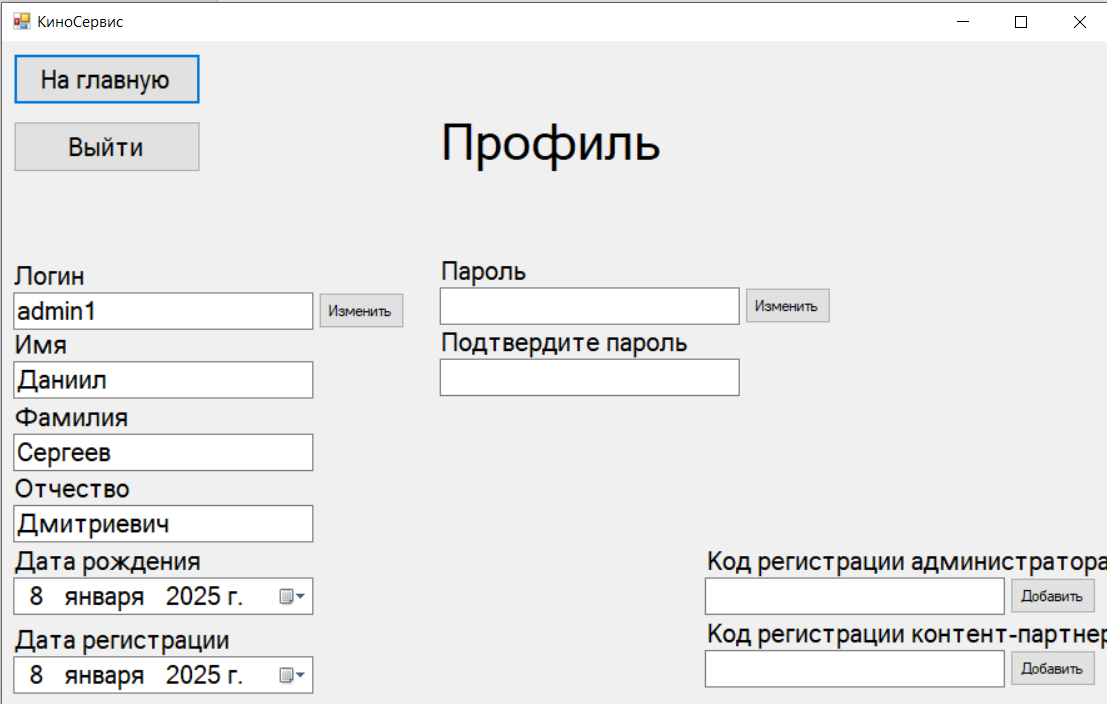


Рисунок 5 – Форма Profile

Форма Bookings\_form содержит 2 текстовых поля, элемент DataGridView и 4 кнопки (рисунок 6)

* текстовые поля информируют пользователя о том, что представлено в таблице и что нужно сделать;
* элемент DataGridView выводит информацию о бронировании;
* кнопки «Редактировать» и «Добавить» открывают форму Booking\_form;
* кнопка «Удалить» удаляет выбранную запись как на элементе, так и в таблице в базе данных;
* кнопка «Читатель» открывает форму Readers\_form.

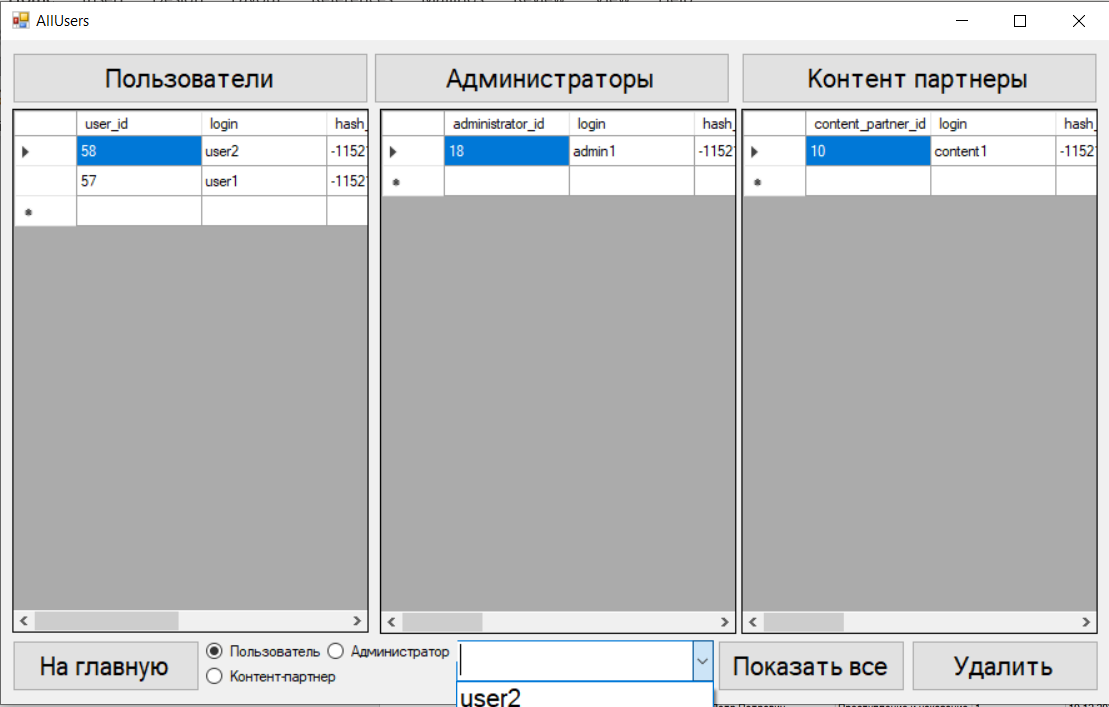


Рисунок 6 – Форма AllUsers

Форма Booking\_form содержит 12 текстовых полей, 2 элемента AllUsers и 3 кнопки (рисунок 7).

* 7 текстовых полей являются информационными, остальные текстовые поля используются для внесения данных;
* элементы ChechBox предназначены для изменения полей с логическим значением;
* кнопка «Сохранить изменения» сохраняет информацию из текстовых полей в базе данных;
* кнопка «Книги» открывает форму Books\_form;
* кнопка «Читатели» открывает форму Readers\_form.

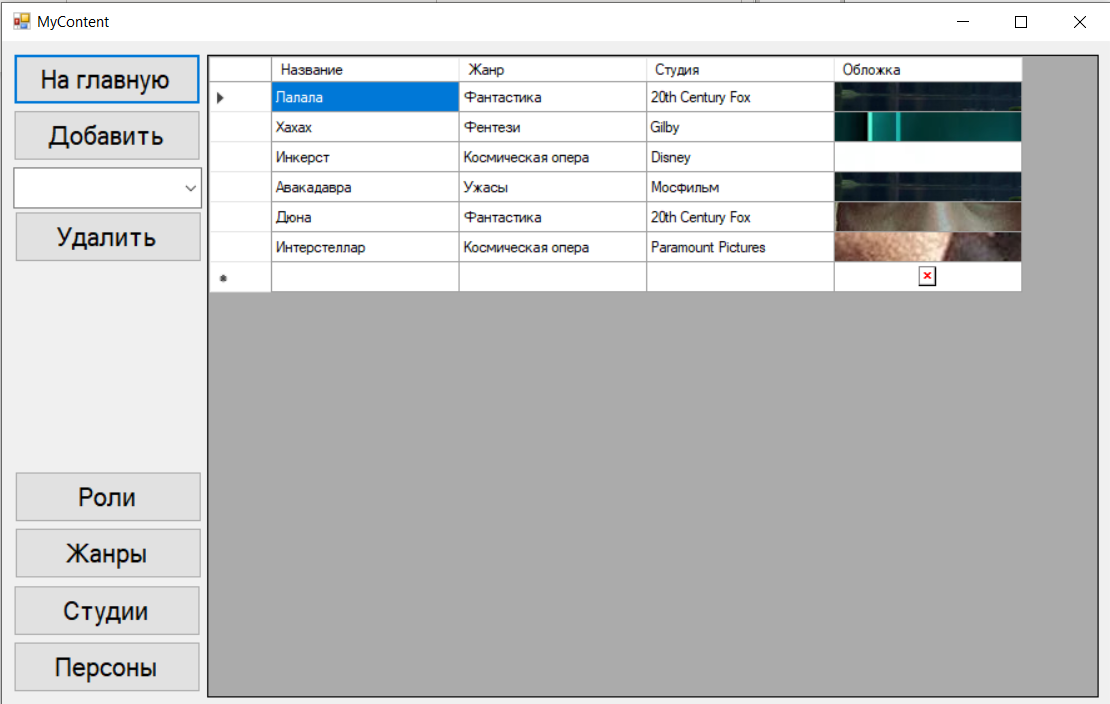


Рисунок 7 – Форма MyContent

Форма MyContent содержит 2 текстовых поля, элемент DataGridView и 4 кнопки (рисунок 8).

* текстовые поля информируют пользователя о том, что представлено в таблице и что нужно сделать;
* элемент DataGridView выводит информацию о книгах;
* кнопки «Редактировать» и «Добавить» открывают форму Book\_form;
* кнопка «Удалить» удаляет выбранную запись как на элементе, так и в таблице в базе данных;
* кнопка «Авторы» открывает форму Authors\_form.

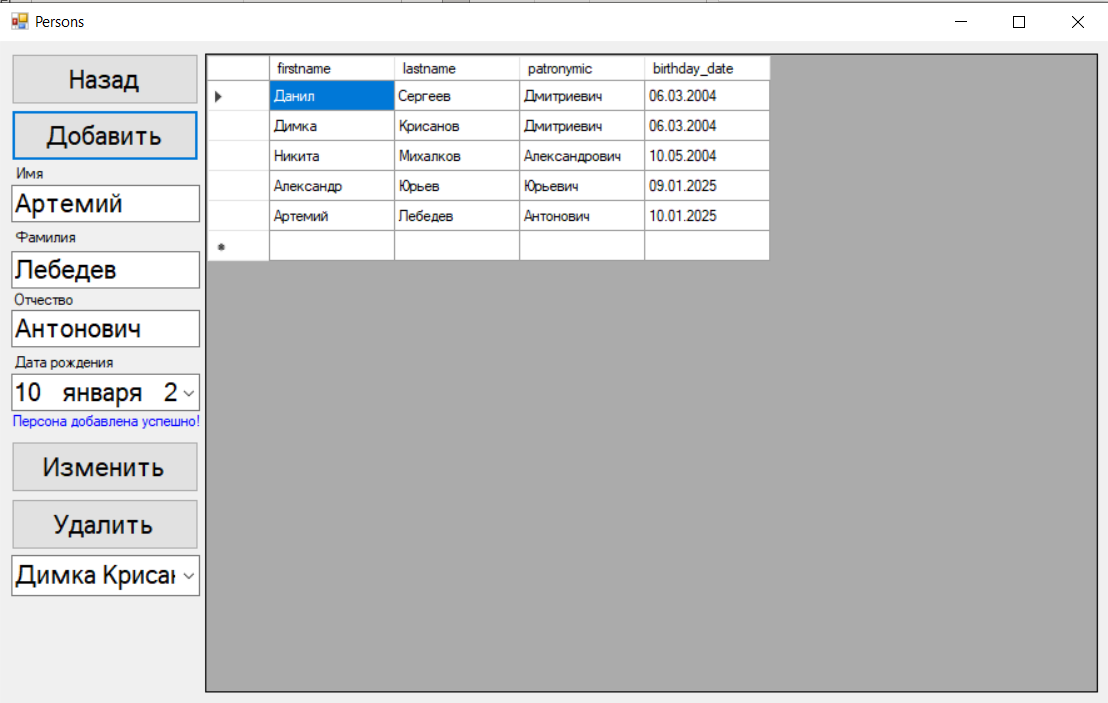


Рисунок 8 – Форма Persons

Форма Book\_form содержит 18 текстовых полей, 1 элемент Persons и 1 кнопку (рисунок 9).

* 9 текстовых полей являются информационными, остальные текстовые поля используются для внесения данных;
* элемент PictureBox выводит или изменяет картинку, хранящуюся в базе данных;
* кнопка «Сохранить изменения» сохраняет информацию из текстовых полей в базе данных.

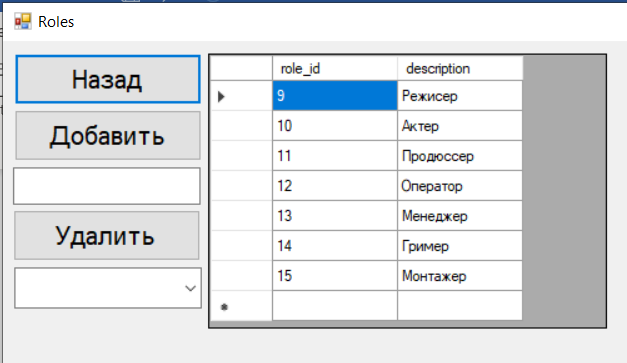


Рисунок 9 – Форма Roles

Форма Roles содержит 2 текстовых поля, элемент DataGridView и 4 кнопки (рисунок 10).

* текстовые поля информируют пользователя о том, что представлено в таблице и что нужно сделать;
* элемент DataGridView выводит информацию об авторах;
* кнопки «Редактировать» и «Добавить» открывают форму Author\_form;
* кнопка «Удалить» удаляет выбранную запись как на элементе, так и в таблице в базе данных;
* кнопка «Книги» открывает форму Books\_form.

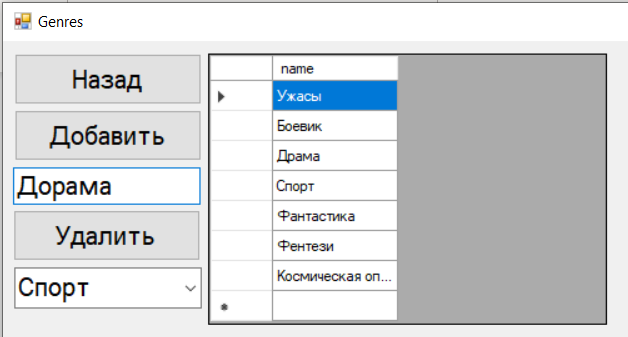


Рисунок 10 – Форма Genres

Форма Genres содержит 6 текстовых полей и 1 кнопку (рисунок 11).

* 3 текстовых поля являются информационными, остальные текстовые поля используются для внесения данных;
* кнопка «Сохранить изменения» сохраняет информацию из текстовых полей в базе данных.

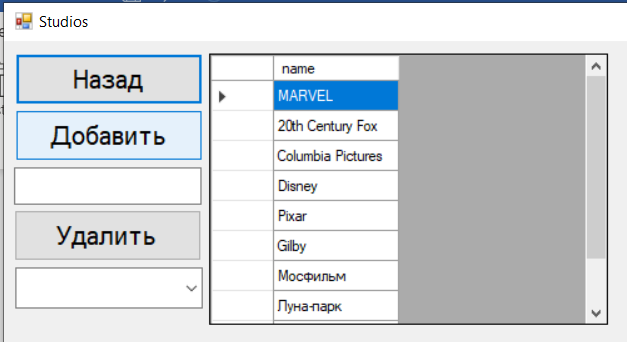


Рисунок 11 – Форма Studios

Форма Studios содержит 2 текстовых поля, элемент DataGridView и 4 кнопки (рисунок 12).

* текстовые поля информируют пользователя о том, что представлено в таблице и что нужно сделать;
* элемент DataGridView выводит информацию о заказах;
* кнопки «Редактировать» и «Добавить» открывают форму Order\_form;
* кнопка «Удалить» удаляет выбранную запись как на элементе, так и в таблице в базе данных;
* кнопка «Читатель» открывает форму Readers\_form.



Рисунок 12 – Форма DetailsMyContent

Форма DetailsMyContent содержит 14 текстовых полей и 2 кнопки (рисунок 13).

* 7 текстовых полей являются информационными, остальные текстовые поля используются для внесения данных;
* кнопка «Сохранить изменения» сохраняет информацию из текстовых полей в базе данных;
* кнопка «Читатели» открывает форму Readers\_form.

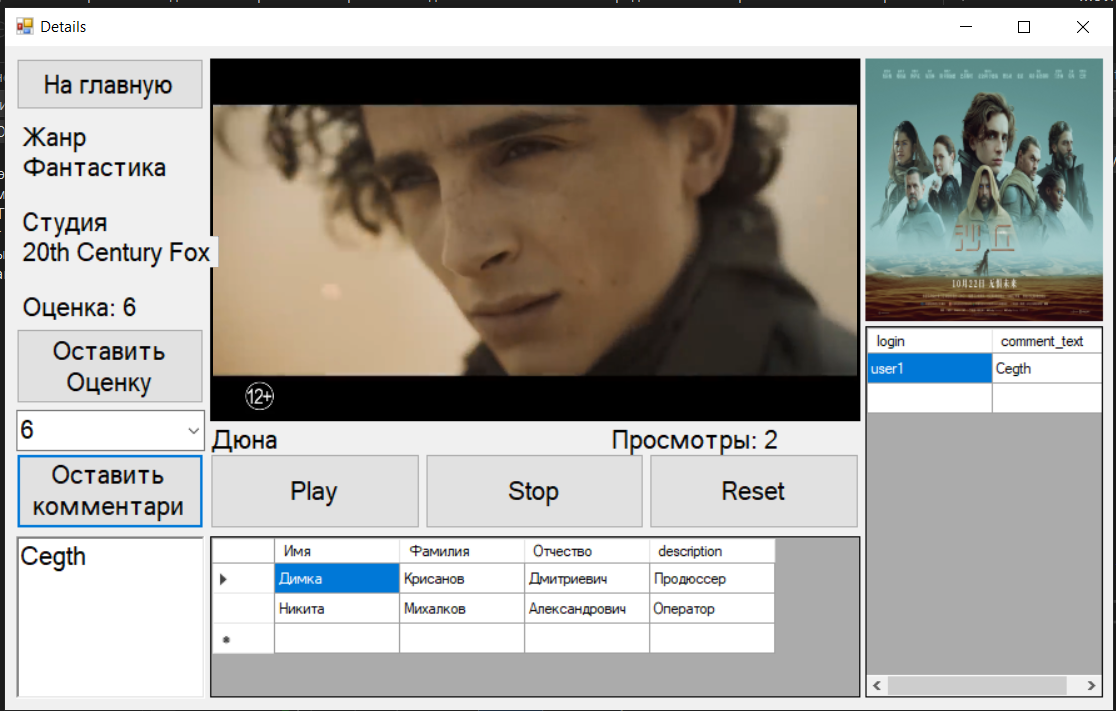


Рисунок 13 – Форма Details

Класс Administarator

private int administrator\_id;

Хранит уникальный идентификатор администратора.

private string login;

Хранит логин администратора (уникальное имя пользователя для входа в систему).

private string hash\_password;

Хранит хэш пароля администратора.

private string firstname;

Хранит имя администратора.

private string lastname;

Хранит фамилию администратора.

private string patronymic;

Хранит отчество администратора.

private DateTime birthday\_date;

Хранит дату рождения администратора.

private DateTime registration\_date;

Хранит дату регистрации администратора.

Конструктор

public Administrator()

Инициализирует новый экземпляр класса Administrator.

Методы

public void Registration(string login, string password, string firstname, string lastname, string patronimyc, DateTime birthday\_date)

Параметры:

string login: Логин для нового администратора.

string password: Пароль для нового администратора.

string firstname: Имя администратора.

string lastname: Фамилия администратора.

string patronimyc: Отчество администратора.

DateTime birthday\_date: Дата рождения администратора.

Описание: Методы выполняет регистрацию нового администратора в системе. Хранит логин, хэш пароля, имя, фамилию, отчество, дату рождения и дату регистрации, после чего добавляет данные в базу данных через SQL-запрос.

public bool CheckPassword(string login\_, string password\_)

Параметры:

string login\_: Логин администратора для проверки.

string password\_: Пароль администратора для проверки.

Описание: Метод проверяет соответствие логина и пароля заданным в системе. Возвращает true, если логин и пароль верные, и false в противном случае.

public bool Entry(string login, string password)

Параметры:

string login: Логин администратора для входа.

string password: Пароль администратора для входа.

Описание: Метод проверяет корректность логина и пароля, а затем, если они верные, выполняет вход в систему, обновляя состояние пользователя.

Класс User

private int user\_id;

Хранит уникальный идентификатор пользователя.

private string login;

Хранит логин пользователя (уникальное имя для входа в систему).

private string hash\_password;

Хранит хэш пароля пользователя.

private DateTime registration\_date;

Хранит дату регистрации пользователя в системе.

Методы

public void Registration(string login, string password)

Параметры:

string login: Логин для нового пользователя.

string password: Пароль для нового пользователя.

Описание: Метод выполняет регистрацию нового пользователя в системе. Сохраняет логин, хэш пароля и дату регистрации после чего добавляет данные в базу данных через SQL-запрос.

public bool CheckPassword(string login\_, string password\_)

Параметры:

string login\_: Логин пользователя для проверки.

string password\_: Пароль пользователя для проверки.

Описание: Метод проверяет соответствие логина и пароля заданным в системе. Возвращает true, если логин и пароль верные, и false в противном случае.

public bool Entry(string login, string password)

Параметры:

string login: Логин пользователя для входа.

string password: Пароль пользователя для входа.

Описание: Метод проверяет корректность логина и пароля, а затем, если они верные, выполняет вход в систему и обновляет состояние пользователя.

public void Exit()

Описание: Метод выполняет выход текущего пользователя из системы, обновляя состояние пользователя.

private int content\_id;

Хранит уникальный идентификатор контента.

private string name;

Хранит название контента.

private string studio;

Хранит название студии, создавшей контент.

private string genre;

Хранит жанр контента.

private string description;

Хранит описание контента.

private int rating;

Хранит рейтинг контента (может быть добавлен позже, если потребуется).

private int content\_partner\_id;

Хранит идентификатор партнёра контента, если применимо.

Методы

public void CreateContent(string name, string studio, string genre, string description, int content\_partner\_id, string url\_video)

Параметры:

string name: Название контента.

string studio: Название студии.

string genre: Жанр контента.

string description: Описание контента.

int content\_partner\_id: Идентификатор партнёра контента (может быть null или иметь значение).

string url\_video: URL видео для контента.

Описание: Метод создает новый контент, добавляя его в базу данных. Если content\_partner\_id равен 99999, то поле content\_partner\_id устанавливается в NULL; в противном случае оно сохраняется.

public void UpdateContent(string name, string studio, string genre, string description, int content\_partner\_id, string url\_video, int content\_id)

Параметры:

string name: Новое название контента.

string studio: Новое название студии.

string genre: Новый жанр контента.

string description: Новое описание контента.

int content\_partner\_id: Новый идентификатор партнёра контента.

string url\_video: Новый URL видео для контента.

int content\_id: Идентификатор контента, который нужно обновить.

Описание: Метод обновляет существующий контент в базе данных по заданному content\_id.

Класс ConnectionDB

public void InsertDB(string query)

Параметры:

string query: SQL-запрос для выполнения вставки данных в базу.

Описание: Метод устанавливает соединение с базой данных и выполняет SQL-запрос для вставки данных. Если возникает ошибка, сообщение об ошибке отображается через MessageBox.

public DataTable SelectDB(string query)

Параметры:

string query: SQL-запрос для выборки данных из базы.

Описание: Метод устанавливает соединение с базой данных, выполняет SQL-запрос на выборку данных и возвращает результаты в виде объекта DataTable. Если происходит ошибка, сообщение об ошибке отображается, и возвращается null.

public void InsertImage(byte[] byteArray, int id)

Параметры:

byte[] byteArray: Массив байтов, представляющий изображение, которое нужно сохранить в базе данных.

int id: Идентификатор контента, к которому относится изображение.

Описание: Метод устанавливает соединение с базой данных и выполняет обновление строки в таблице Content, устанавливая поле image равным переданному массиву байтов. Метод использует параметризованный запрос для предотвращения SQL инъекций.

Примечания

Класс ConnectionDB предоставляет базовые методы для взаимодействия с PostgreSQL-базой данных, используя библиотеку Npgsql. Обработку ошибок реализовано через блоки try-catch, и тестирование на наличие ошибок выполнено на уровне выполнения SQL-запросов.

Класс ContentPartner

private int content\_partner\_id;

Хранит уникальный идентификатор партнера контента.

private string login;

Хранит логин партнера контента.

private string hash\_password;

Хранит хэшированный пароль для проверки подлинности.

private string firstname;

Хранит имя партнера контента.

private string lastname;

Хранит фамилию партнера контента.

private string patronymic;

Хранит отчество партнера контента.

private DateTime birthday\_date;

Хранит дату рождения партнера контента.

private DateTime registration\_date;

Хранит дату регистрации партнера контента.

Методы

public void Registration(string login, string password, string firstname, string lastname, string patronimyc, DateTime birthday\_date)

Параметры:

string login: Логин для регистрации.

string password: Пароль для создания учетной записи.

string firstname: Имя партнера.

string lastname: Фамилия партнера.

string patronimyc: Отчество партнера.

DateTime birthday\_date: Дата рождения партнера.

Описание: Метод выполняет процесс регистрации нового партнера контента. Хэширует пароль и создает запись в базе данных ContentPartner, включая информацию о логине, имени, фамилии, отчестве и датах рождения и регистрации.

public bool CheckPassword(string login\_, string password\_)

Параметры:

string login\_: Логин для проверки.

string password\_: Пароль для проверки.

Описание: Метод проверяет, соответствует ли введенный логин и пароль (после хэширования) записи в базе данных. Возвращает true, если пара логин-пароль найдена, и false в противном случае.

public bool Entry(string login, string password)

Параметры:

string login: Логин для входа.

string password: Пароль для входа.

Описание: Метод выполняет процесс входа партнера контента. Сначала проверяет правильность логина и пароля с помощью метода CheckPassword. Если проверка успешна, извлекает идентификатор партнера контента и устанавливает пользовательское состояние для входа. Возвращает true, если вход выполнен успешно, и false в противном случае.

Примечания

Класс ContentPartner предоставляет функциональность для управления учетными записями партнеров контента, включая регистрацию и аутентификацию. Методы взаимодействуют с базой данных через класс ConnectionDB, обеспечивая выполнение операций вставки и выборки. Хэширование пароля реализовано простым вызовом GetHashCode, однако для большей безопасности рекомендуется использовать более надежные алгоритмы хэширования, такие как SHA256 или BCrypt

**3.4 Инструкция пользователя**

Данный онлайн-киносервис предназначен для управления данными о пользователях и фильмах. В нем можно просматривать, изменять, добавлять и удалять информацию из каталога фильмов и заказов.

При входе на платформу появляется главное меню, в котором представлено 5 кнопок:

− Для перехода в раздел с информацией о пользователях нажмите на кнопку «Пользователи»;

− Для доступа к информации о заказах нажмите на кнопку «Заказы»;

− Для просмотра каталога фильмов нажмите на кнопку «Фильмы»;

− Для получения информации об авторах (режиссерах) нажмите на кнопку «Авторы».

В разделе «Пользователи» представлена таблица с информацией о пользователях. Внизу окна находятся 5 кнопок:

− Чтобы редактировать информацию о пользователе, нажмите на нужную строку, затем нажмите кнопку «Редактировать», чтобы открыть окно с данными о пользователе;

− Чтобы добавить нового пользователя, нажмите кнопку «Добавить», и откроется форма с пустыми полями;

− Чтобы удалить пользователя, выберите строку, затем нажмите кнопку «Удалить». Появится подтверждение удаления, где можно нажать «Да» для подтверждения или «Нет» для отмены. Если выбрано «Нет», появится уведомление об отмене удаления;

− Чтобы увидеть заказы пользователя, выберите строку и нажмите кнопку «Заказ», появится окно с его заказами.

В окне «Пользователь» есть текстовые поля для редактирования информации. Внизу окна располагаются 2 кнопки:

− Чтобы изменить необходимую информацию, выберите соответствующее текстовое поле;

− Чтобы сохранить изменения, нажмите кнопку «Сохранить изменения». После этого окно закроется;

− Кнопка «Пароль» позволяет получить доступ к логину пользователя и сгенерировать случайный пароль. После нажатия на кнопку откроется окно «Пароль».

В окне «Пароль» пользователь вводит новый пароль, после чего может нажать на кнопку «Сохранить изменения» для подтверждения.

В разделе «Фильмы» представлена информация о фильмах в таблице. Внизу окна находятся 4 кнопки:

− Для редактирования информации о фильме выберите строку и нажмите кнопку «Редактировать», откроется окно с данными о фильме;

− Для добавления нового фильма нажмите кнопку «Добавить», появится форма с пустыми полями;

− Чтобы удалить фильм, выберите строку, нажмите кнопку «Удалить» и подтвердите действие;

− Чтобы узнать информацию об авторах фильма, выберите строку и нажмите кнопку «Авторы».

В окне «Фильм» есть текстовые поля для редактирования информации. Также присутствует рамка для отображения обложки фильма:

− Чтобы изменить информацию о фильме, выберите необходимое текстовое поле;

− В рамке «Обложка» отображается изображение, которое можно заменить, нажав на нее и выбрав файл;

− Чтобы сохранить изменения, нажмите кнопку «Сохранить изменения».

В разделе «Авторы» представлена информация об авторах в виде таблицы. Внизу окна располагаются 4 кнопки:

− Для редактирования информации об авторе выберите строку, затем нажмите кнопку «Редактировать»;

− Чтобы добавить нового автора, нажмите кнопку «Добавить», откроется окно с пустыми полями;

− Чтобы удалить автора, выберите строку и нажмите кнопку «Удалить», подтвердите действие, если требуется;

− Чтобы посмотреть фильмы автора, выберите строку и нажмите кнопку «Фильмы».

В окне «Автор» есть текстовые поля для редактирования данных об авторе, а также одна кнопка для сохранения изменений.

В разделе «Заказы» представлена информация о заказах в виде таблицы. Внизу окна находятся 4 кнопки:

− Для редактирования заказа выберите строку и нажмите кнопку «Редактировать» для открытия окна с информацией о заказе;

− Для создания нового заказа нажмите кнопку «Добавить»;

− Чтобы удалить заказ, выберите строку и подтвердите действие;

− Чтобы увидеть информацию о пользователе, сделавшем заказ, выберите строку и нажмите кнопку «Пользователь».

В окне «Заказ» содержатся текстовые поля для редактирования данных о заказе. Внизу окна расположены 2 кнопки:

− Чтобы изменить информацию о заказе, выберите нужное текстовое поле;

− Нажмите кнопку «Сохранить изменения» для подтверждения изменений;

− Кнопка «Пользователи» позволяет получить доступ к информации о пользователях.

**4. Тестирование АИС**

Попробуем при авторизации пользователя не ввести одно из полей входа. Программа будет выводить нам ошибку о неправильных введенных данных, как показано на рисунке 14. Этот фрагмент проверяет текстовое поле и выдаёт сообщение, что о несоответствии данных в заполнении поля.

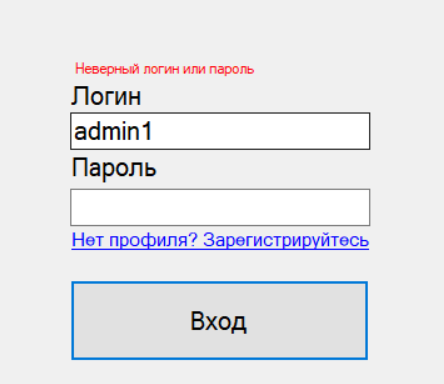


Рисунок 14.

Теперь во вкладке Persons попробуем удалить не выбранный элемент, в таком случае должно появиться сообщение, о том, что необходимо сначала выбрать персону на рисунке 15.

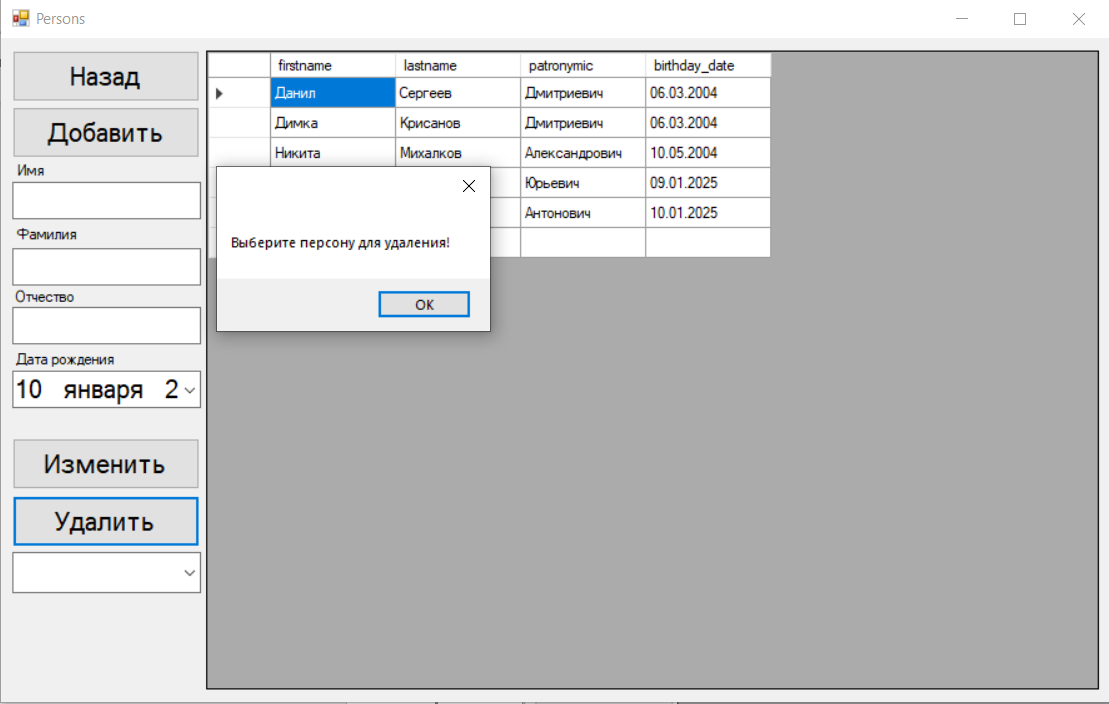


Рисунок 15.

Далее пробуем удалить невыбранную запись из таблицы контента, опять же выскакивает обрабатываемое сообщение о необходимости данных на рисунке 16.

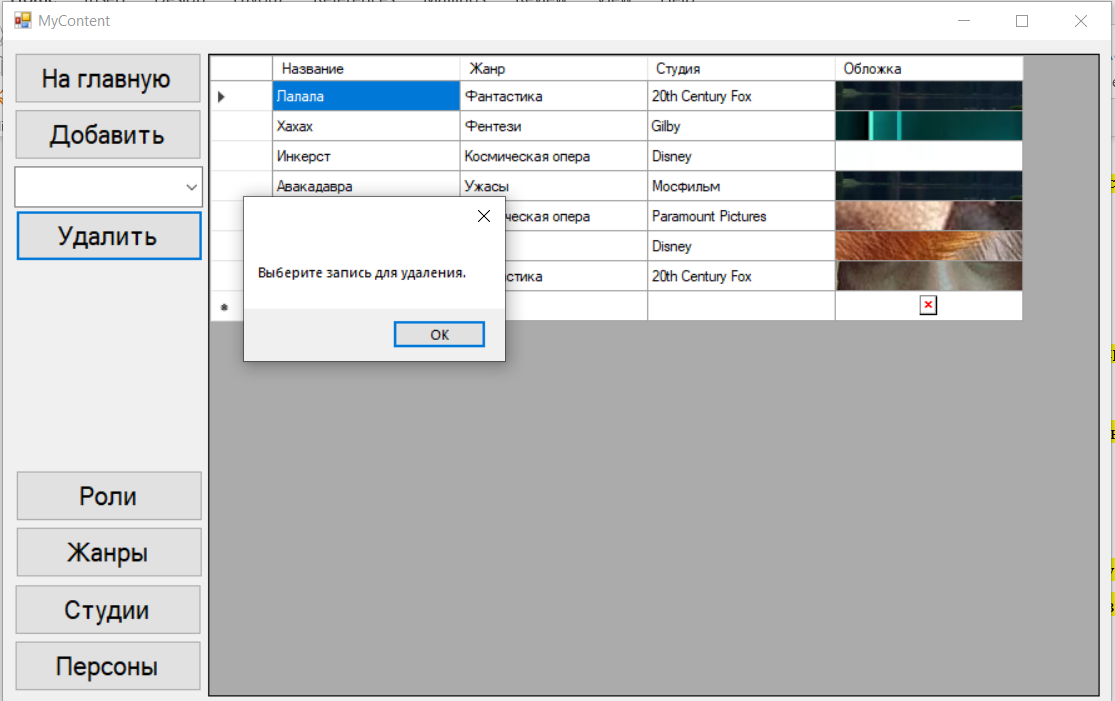


Рисунок 16.

Попытка загрузить видео не заполняя все данные, при нажатии на кнопкку проверяется наличие всех данных, включая ссылку на видео и наличие изображения фильма.

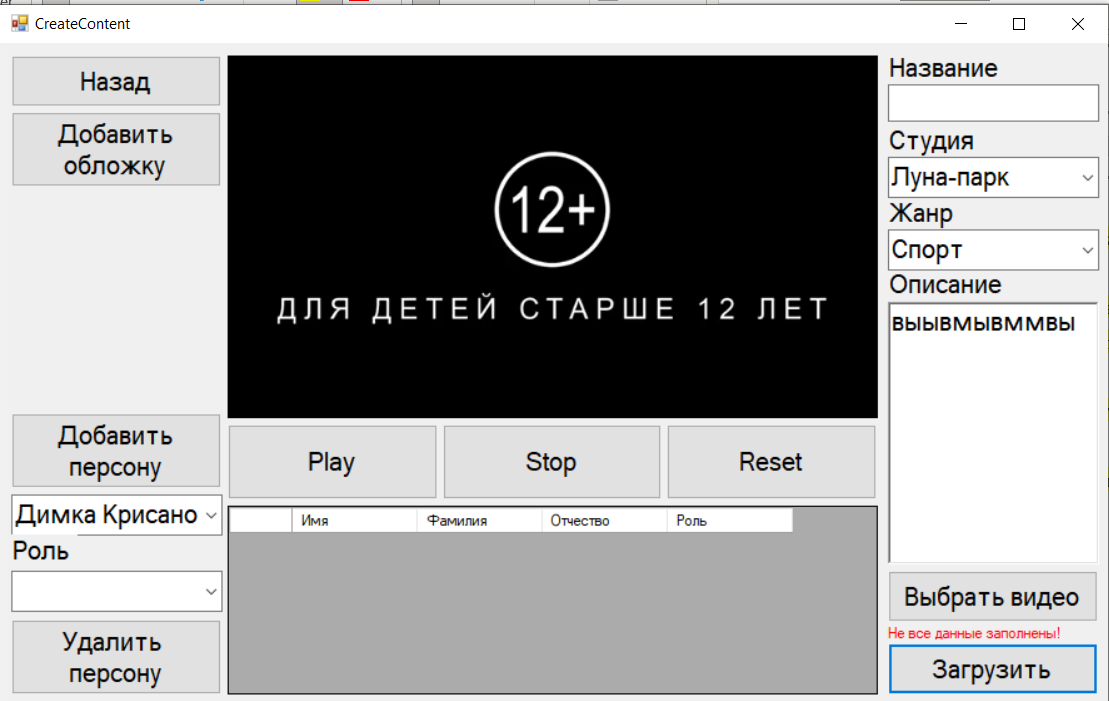


Рисунок 17.

**Заключение**

В ходе выполнения данной работы была разработана автоматизированная информационная система для хранения и просмотра фильмов, которая включает в себя приложение на C# с использованием Windows Forms и интеграцию с базой данных PostgreSQL.

Целью разработки было создание удобного и функционального инструмента для управления кинематографическими ресурсами, что значительно упрощает работу администраторов и улучшает качество обслуживания пользователей.

Созданное приложение обеспечивает основные функции, такие как учет фильмов, регистрацию пользователей, управление заказами и возвратом фильмов, а также возможность создания запросов на добавление новых фильмов в коллекцию. Пользовательский интерфейс был разработан с учетом удобства и доступности, что позволяет легко ориентироваться в системе.

Использование PostgreSQL в качестве базы данных обеспечило надежное хранение и обработку информации, что является критически важным для сохранения данных пользователей и ресурсов. В результате реализации проекта были достигнуты все поставленные задачи, и система готова к дальнейшему использованию и доработке.

**Список литературы**

1. Васюткина, И. А. Разработка приложений на C# с использованием СУБД PostgreSQL: учебное пособие : [Текст]/ И. А. Васюткина, Г. В. Трошина, М. И. Бычков, С. А. Менжулин . – Новосибирск: Новосибирский государственный технический университет, 2015. – 143 с.
2. Наместников, А.М. Базы данных. Практический курс. В 2ч. Ч. 1. Объектно-реляционные базы данных на примере PostgreSQL 9.5: учебное пособие / А.М. Наместников, А.А. Филиппов. – Ульяновск: УлГТУ, 2017. – 113c.
3. Самохвалов, Э. Н. Введение в проектирование и разработку приложений на языке программирования С#: учебное пособие / Э. Н. Самохвалов, Г. И. Ревунков, Ю. Е. Гапанюк. – М.: Издательство МГТУ им. Н. Э. Баумана, 2018. – 244 с.
4. О персональных данных: Федеральный закон от 27 июля 2006 года № 152-ФЗ // Собрание законодательства Российской Федерации. – 2006. - № 31. – С. 3451-3457.
5. Чарнолуский, П. А. Системы управления базами данных: Учебник / П. А. Чарнолуский. — 2-е изд., перераб. и доп. — М.: Горячая линия — Телеком, 2017. — 416 с.

**Приложения**

**Приложение А**

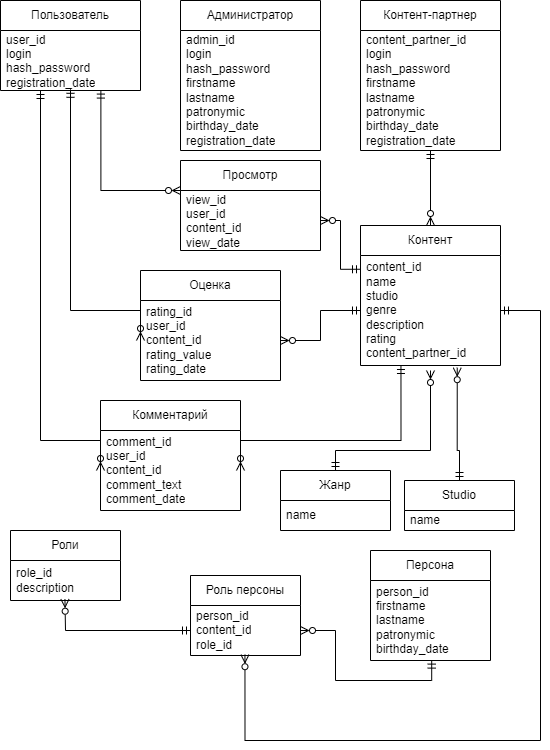


Рисунок А.1 – Логическая модель базы данных.

**Приложение Б**

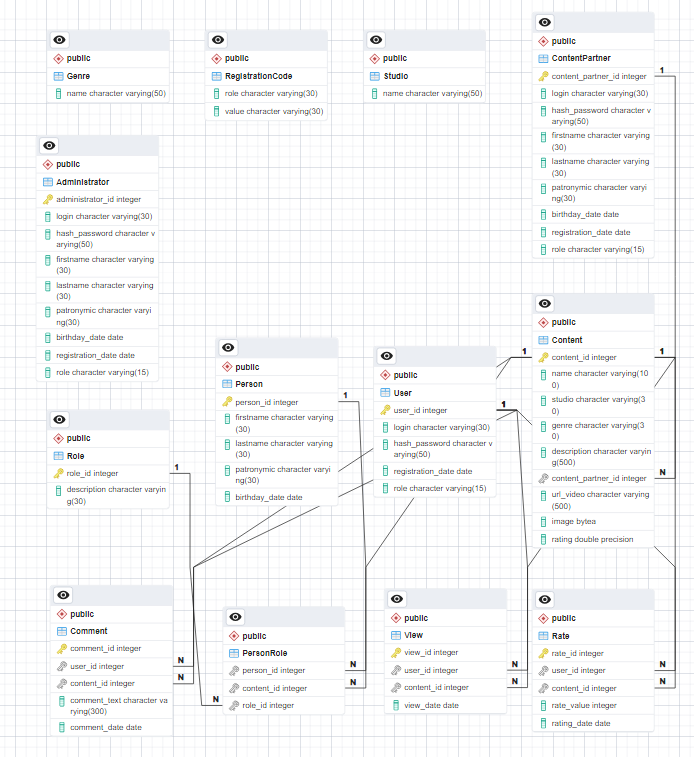


Рисунок Б.1 – Физическая модель базы данных

**Приложение В**

Ссылка на репозиторий с кодом:

<https://github.com/DiRasher04/Movie-service-app>

В рамках данной работы была создана информационная система хранения и поиска фильмов и модели данных. Система включает в себя базу данных, в которой содержится информация о фильмах, их режиссерах, актерах, пользователях, записях о бронировании и возврате фильмов, а также о запросах на добавление новых фильмов в коллекцию. Также было разработано приложение для администраторов, в котором можно удобно работать с данными в таблицах базы данных через графический интерфейс.

Работа состоит из вводной части, анализа технического задания, разработки моделей данных, разработки и реализации информационной системы, тестирования системы, заключения, списка литературы и приложений.

As part of this work, an information system for storage and search of movies and data models were created. The system includes a database that contains information about movies, their directors, actors, users, records of bookings and returns, as well as requests for adding new movies to the collection. An application for administrators was also developed, which allows for convenient interaction with the data in the database tables through a graphical interface.

The work consists of an introductory section, an analysis of the technical requirements, development of data models, development and implementation of the information system, system testing, conclusion, a list of references, and appendices.