Министерство цифрового развития, связи и массовых коммуникаций РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

КУРСОВАЯ РАБОТА

Вариант 29
Кинотеатр

Выполнил студент группы ИП-111 Тетяков И.С.
Проверил доцент кафедры ПМиК к.т.н Мерзлякова Е.Ю.

Новосибирск 2023

Содержание

Введение 3

ЧАСТЬ 1 4

- Анализ задач и пользователей 4
- Выбор репрезентативных задач 5
- Заимствования 7
- Черновое описание дизайна 8

ЧАСТЬ 2 11

- Описание программы 11
- Скриншоты работы программы 13

ЧАСТЬ 3 16

- CWT-анализ 16
- GOMS-анализ 19

Соответствие требованиям курсового проекта 22

Используемые источники 23

Листинг 24

Введение

Целью курсового проекта является закрепление навыков, полученных в ходе изучения дисциплины «Визуальное программирование и человеко-машинное взаимодействие» и разработка приложения «Фильмотека». Для достижения поставленной цели необходимо выполнение следующих задач:

- Провести первые 4 этапа проблемно-центрированного дизайна программного продукта.
- Провести CWT-анализ разработанного интерфейса.
- Провести GOMS-анализ разработанного интерфейса.
- По результатам CWT и GOMS анализов доработать интерфейс программы и выполнить создание макета или прототипа.

Проект представляет из себя систему администрирования кинотеатра, ориентированную на интуитивное использование. Администратор имеет возможность управлять списком фильмов, вносить изменения и удалять записи. Также ему предоставляется функционал бронирования мест в зале для определенного фильма и времени.

Интерфейс приложения разработан с учетом простоты освоения как для опытных пользователей, так и для новичков. Дружелюбные уведомления поддерживают пользователя на каждом этапе, предоставляя информацию о действиях и оповещая об ошибках.

Важно отметить, что пользовательская активность сохраняется автоматически, обеспечивая сохранность данных даже при перезапуске приложения. Это придает уверенность пользователям в их действиях и создает удобные условия для работы с приложением.

ЧАСТЬ 1

Анализ задач и пользователей

Цель:

Найти двух человек, которые могут быть заинтересованы в решении предложенной задачи. Дайте их краткое описание (возраст, образование, профессия, навыки в выбранной сфере, навыки владения компьютером). Люди должны быть заинтересованы в решении предложенной задачи.

Решение:

Программа каталога просмотренных фильмов может заинтересовать любого: взрослого человека, ребенка, пару, который просматривает фильмы и хотел бы их как-то каталогизировать.

В качестве заинтересованных лиц были выбраны 2 следующих человека:

- Мама: возраст – 49 лет, образование – три высших, профессия – директор школы, навыки в выбранной сфере – иногда смотрит фильмы, навыки владения компьютером – средние.
- Брат: возраст – 27 лет, образование – среднее специальное, профессия – не работает официально, занимается перепродажей автомобилей. Смотрит большее количество фильмов, сериалов, аниме, мультфильмов, навыки владения компьютером – высокие.

Выбор репрезентативных задач

Цель:

Определение ключевых функциональных задач и второстепенных действий для обеспечения эффективного использования разрабатываемой программы.

Решение:

Репрезентативные задачи:

1. Просмотр записей:
   - Разработка интуитивного интерфейса для удобного просмотра подробной информации о просмотренных фильмах.
2. Добавление записей:
   - Реализация удобного интерфейса, позволяющего пользователям легко добавлять новые записи о фильмах в систему.
3. Редактирование записей:
   - Предоставление пользователю удобных средств для редактирования существующих данных о фильмах.
4. Удаление записей:
   - Создание интуитивного интерфейса для безопасного и простого удаления записей из фильмотеки.
5. Хранение данных:
   - Разработка системы сохранения данных в базе данных, обеспечивая их сохранность после завершения работы с приложением.
6. Бронирование мест:
   - Разработка интерфейса бронирования мест в зале на определенный фильм, дату и время.

Второстепенные задачи:

1. Выставление даты просмотра:
   - Добавление функционала для установки и отображения даты просмотра фильма.
2. Добавление комментария:
   - Реализация возможности добавления комментариев пользователями о просмотренных фильмах.
3. Выход из приложения:
   - Обеспечение простого и безопасного завершения работы с приложением.

Такое четкое выделение репрезентативных и второстепенных задач поможет фокусироваться на ключевых аспектах функционала и обеспечит эффективное использование программы.

Заимствования

Задача:

Найти приложения или сайты, с которых можно заимствовать какие-либо решения интерфейса, приведите ссылку на источник. Для обогащения пользовательского интерфейса разрабатываемого приложения, рассматривается возможность вдохновения и заимствования элементов дизайна из существующих приложений и веб-сайтов. Выберите и напишите, что именно Вы будете заимствовать из данных приложений и зачем.

Решение:

1) **КиноПоиск**

С данного сервиса была заимствована идея организации просмотренных фильмов в виде списка, а также дата просмотра и оценка.

Также с данного ресурса взято несколько описаний фильмов, которые будут заранее находиться в базе данных при первом запуске приложения.

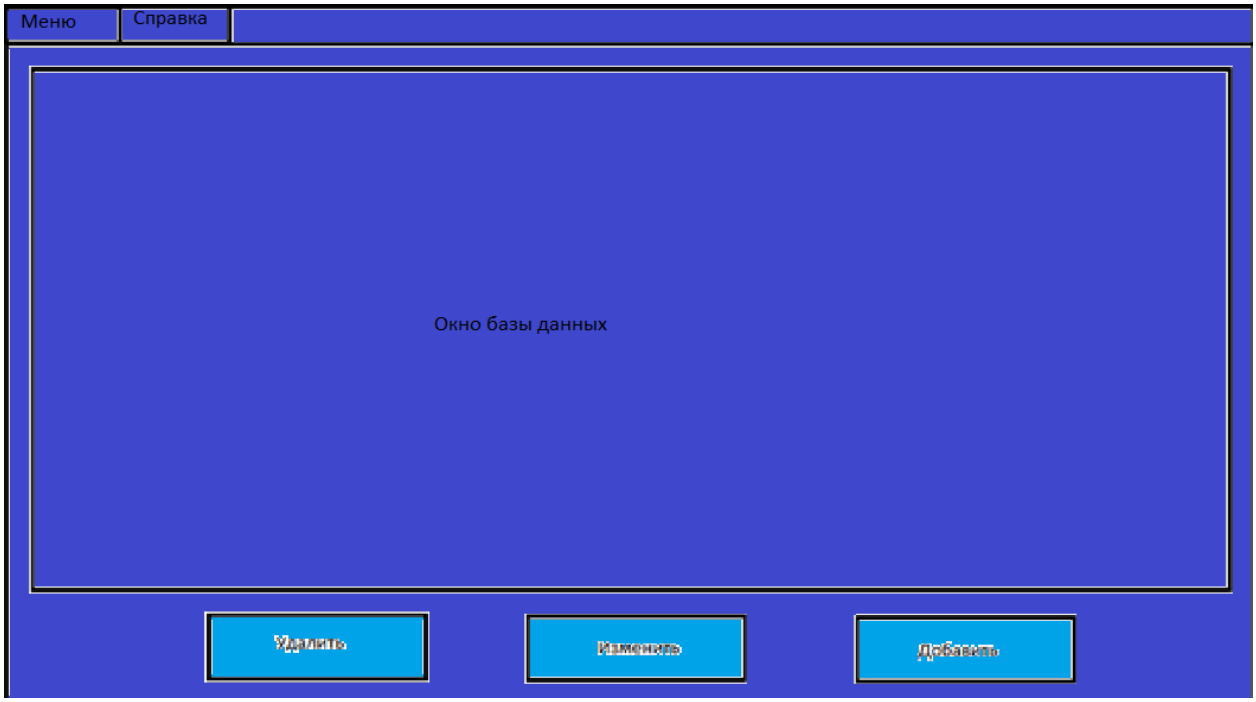
№	фильм	дата и время	моя оценка
1	Зеленая книга (2018) Green Book 8.345 (412 830) 130 мин.	09.08.2019, 16:02	8
2	Первому игроку приготовиться (2018) Ready Player One 7.388 (307 610) 140 мин.	09.08.2019, 16:01	10
3	Дэдпул 2 (2018) Deadpool 2 7.357 (326 500) 119 мин.	26.07.2019, 11:43	10
4	Человек-муравей и Оса (2018) Ant-Man and the Wasp 6.888 (181 786) 118 мин.	26.07.2019, 11:42	10
5	Излом времени (2018) A Wrinkle in Time 4.610 (31 339) 109 мин.	26.07.2019, 11:42	6
6	Тихоокеанский рубеж 2 (2018) Pacific Rim: Uprising 5.729 (93 702) 110 мин.	26.07.2019, 11:42	8
7	Тихоокеанский рубеж (2013) Pacific Rim 6.958 (185 786) 131 мин.	26.07.2019, 11:42	8
8	Сверхъестественное (сериал, 2005 – 2020) Supernatural 8.227 (296 873) 43 мин.	06.04.2019, 14:59	8
9	Чернобыль: Зона отчуждения (сериал, 2014 – 2017) 7.845 (114 744) 48 мин.	06.04.2019, 14:58	4
10	Шерлок (сериал, 2010 – 2017) Sherlock 8.871 (387 078) 90 мин.	06.04.2019, 14:58	10
11	Очень странные дела (сериал, 2016 – ...) Stranger Things 8.420 (245 007) 51 мин.	06.04.2019, 14:58	10
12	Черное зеркало (сериал, 2011 – 2019) Black Mirror 8.513 (190 447) 43 мин.	06.04.2019, 14:58	10

Черновое описание дизайна

Задача:

Описать черновой вариант дизайна словами и графически (иллюстрации с пояснениями). Черновой вариант должен отражать все внешние элементы интерфейса и их назначение.

Решение:

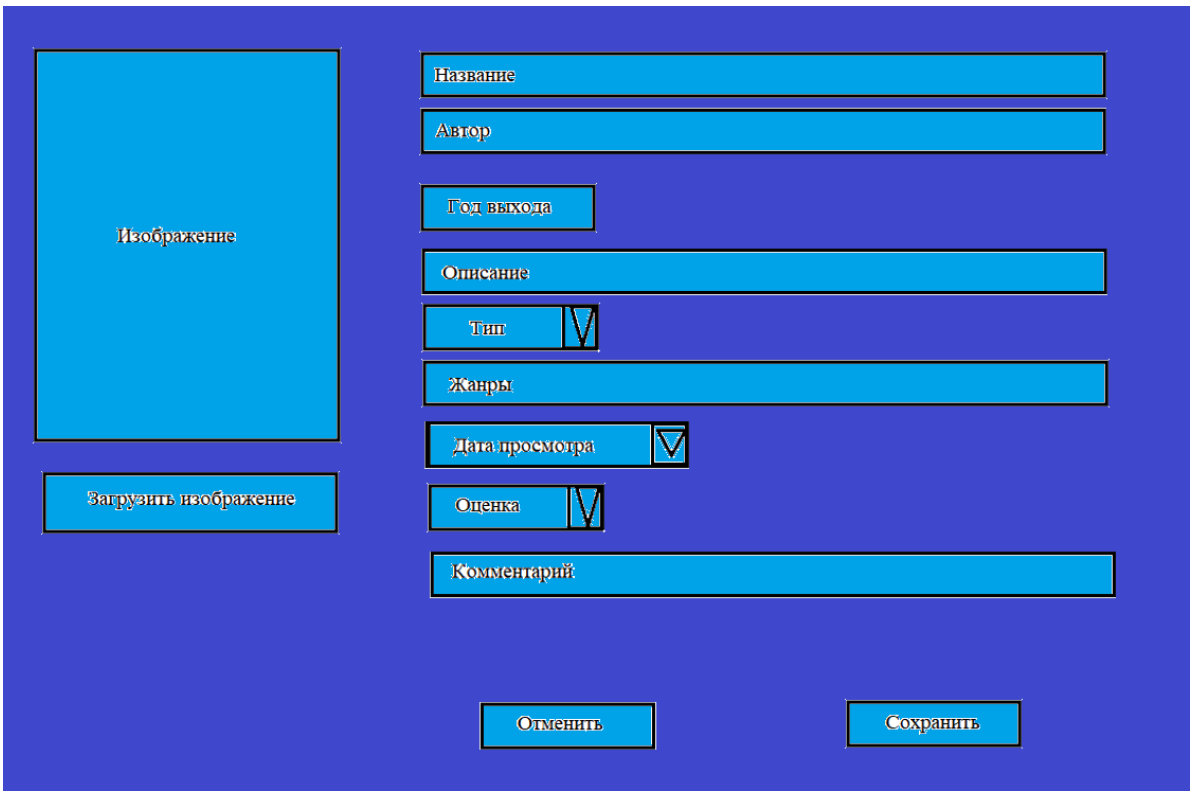


(рис. 1) Главное окно приложения

В данном приложении будет производиться отображение базы данных. При нажатии на кнопку «Изменить» должно открыться окно изменения записи. При нажатии на кнопку «Удалить» запись должна удаляться из базы. При нажатии на кнопку «Добавить» должно открываться окно создания новой записи.

Меню: содержит добавление, изменение, удаление, а также выход из программы.

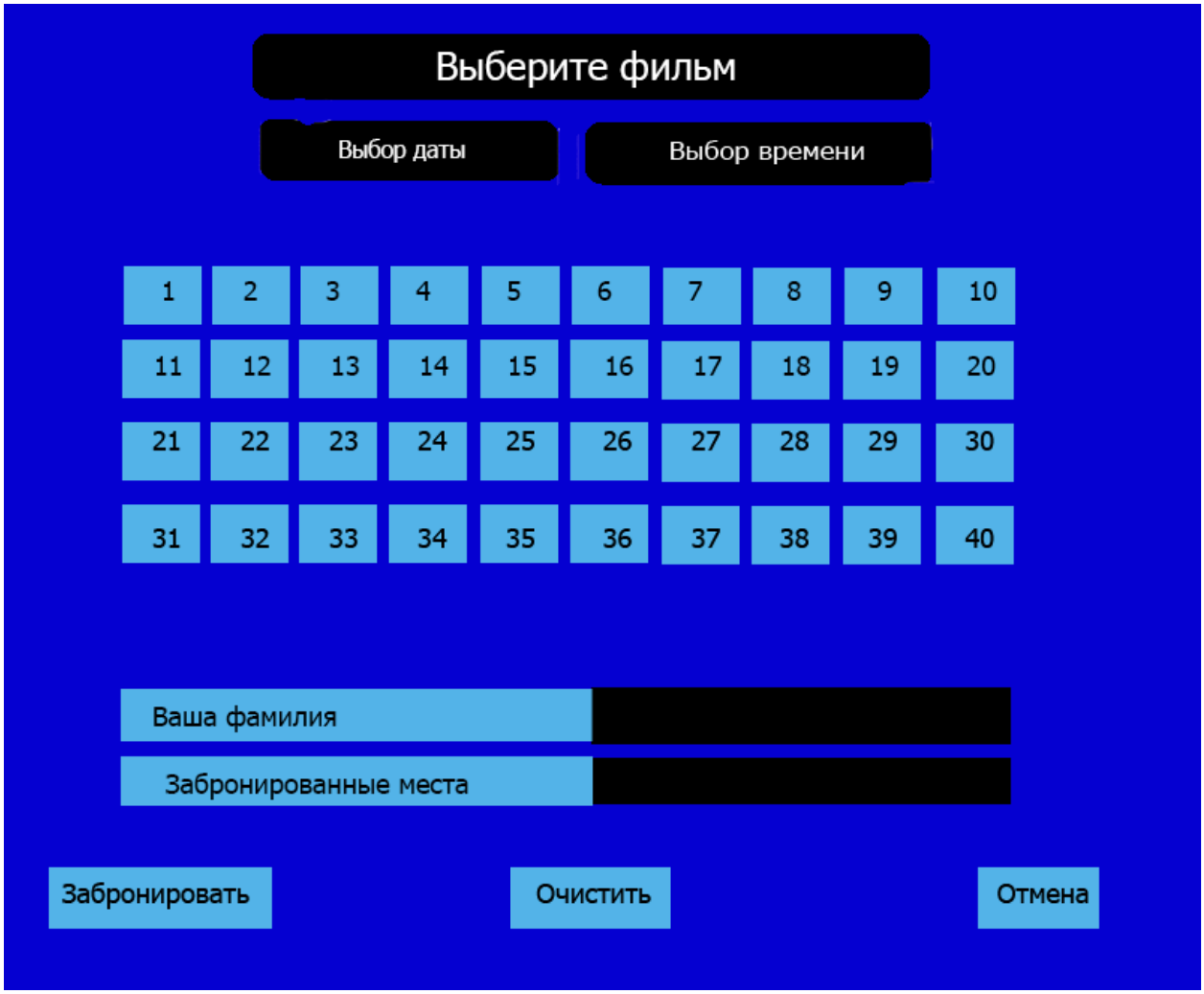
Справка: содержит инструкцию пользования программой и сведения о программе



(рис. 2) Окно редактирования\создания рецепта

При создании нового рецепта пользователь должен обязательно загрузить изображение и заполнить все поля, кроме комментария. При нажатии на кнопку «Тип» должно появляться меню с доступными вариантами, одну из которых пользователь обязательно должен выбрать. При нажатии на кнопку «Оценка» должно появляться меню с доступными вариантами. При нажатии на кнопку «Дата просмотра» должен появляться календарь для выбора даты. При нажатии на кнопку «Загрузить изображение» должно открываться окно выбора изображения. При нажатии на кнопку сохранить при условии, что заполнены все поля, рецепт должен добавляться в базу данных. Если одно из полей не заполнено, должно появиться уведомление, которое не даст сохранить рецепт. При нажатии на кнопку «Отменить» окно создания рецепта закроется без сохранения написанных в нём данных.

Окно редактирования практически не отличается от окна создания, единственным отличием является то, что все данные заранее заполнены и кнопка «Сохранить» переименована в «Изменить».



(рис. 3) Окно бронирования мест

Окно бронирования мест позволяет выбрать фильм из списка, в зависимости от фильма выбрать доступную дату, в зависимости от доступной даты для определенного фильма выбрать время. Нажимая на места от 1 до 40, они будут показывать другим цветом. Далее вписывается фамилия, а забронированные места автоматически заполняются в зависимости от выбранных цифр выше. Нажимая кнопку «Забронировать», появляется уведомление, а данные сохраняются в базе данных. Нажимая кнопку «Очистить», все стирается, на кнопку «Отмена» происходит возврат на главное окно.

ЧАСТЬ 2

Описание программы

Программа "Фильмотека", разработанная на платформе Qt 5.12.0, предоставляет пользователю обширный функционал:

1. Взаимодействие с фильмами:
   1.1 Просмотр подробной информации о фильмах.
   1.2 Добавление новых фильмов с удобным интерфейсом.
   1.3 Удаление выбранных записей из коллекции.
   1.4 Редактирование данных о фильмах для актуализации информации.
   1.5 Бронирование мест в зале кинотеатра
2. Хранение фильмов в базе данных:
3. Навигатор помощи:
   3.1 Подробное описание функционала приложения в удобном навигаторе.
   3.2 Обеспечение пользователей информацией о каждой функции для легкого освоения приложения.
4. Всплывающие уведомления-подсказки:
   4.1 Предоставление интуитивных и информативных всплывающих уведомлений для улучшения опыта пользователя.
   4.2 Поддержка обратной связи через наглядные подсказки для удобного взаимодействия.

Программа структурирована с использованием шести ключевых классов, каждый из которых отвечает за определенный функционал:

1. **Главный класс (MainWindow):**

- Основной контроллер, обеспечивающий взаимодействие и координацию работы остальных классов.

- Управление главным окном приложения и основными действиями пользователя.

2. **Класс создания записи (AddRecord):**

- Реализация окна для добавления новых записей в фильмотеку.

- Предоставление пользователю интерфейса для ввода данных о фильме.

3. **Класс редактирования записи (EditRecord):**

- Разработка окна редактирования существующих записей в фильмотеке.

- Предоставление возможности изменения информации о фильме.

4. **Класс навигатора помощи (HelpInformation):**

- Создание окна с навигатором для ознакомления пользователя с функционалом приложения.

- Интеграция с QTextBrowser для отображения HTML-страниц с подробным описанием.

5. **Класс обработки базы данных (DataBase):**

- Взаимодействие с базой данных SQLite.

6. Класс бронирования мест (Tickets):

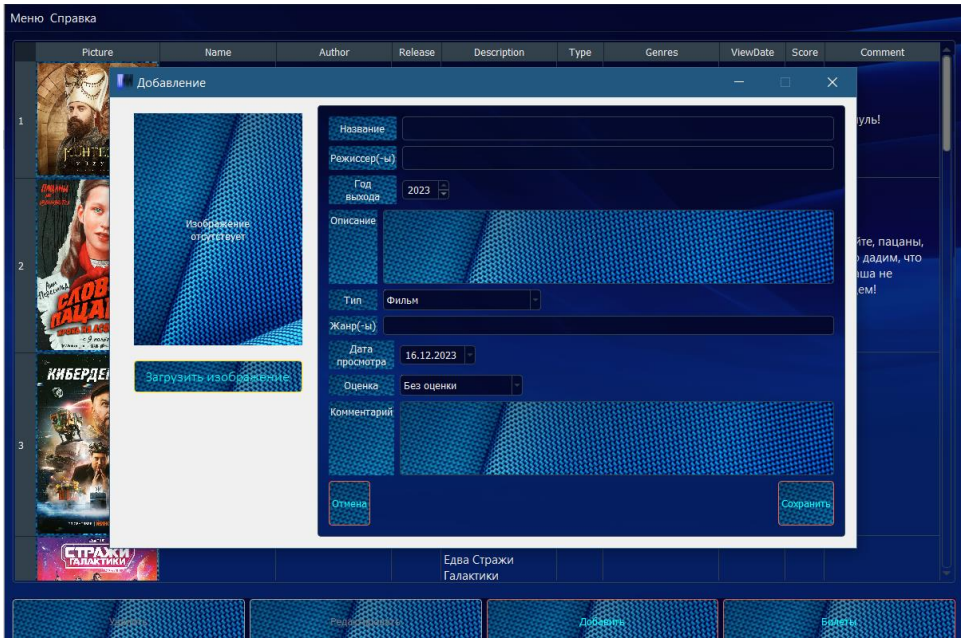
- Реализация окна бронирования мест (1-40) на каждый фильм с определенной датой и временем.

База данных реализована на языке SQLite с использованием библиотеки QSql для эффективного хранения и обработки данных. Навигатор помощи предоставляет подробные инструкции в удобном формате HTML через QTextBrowser. Все элементы программы спроектированы для обеспечения легкости использования и понимания пользователем.

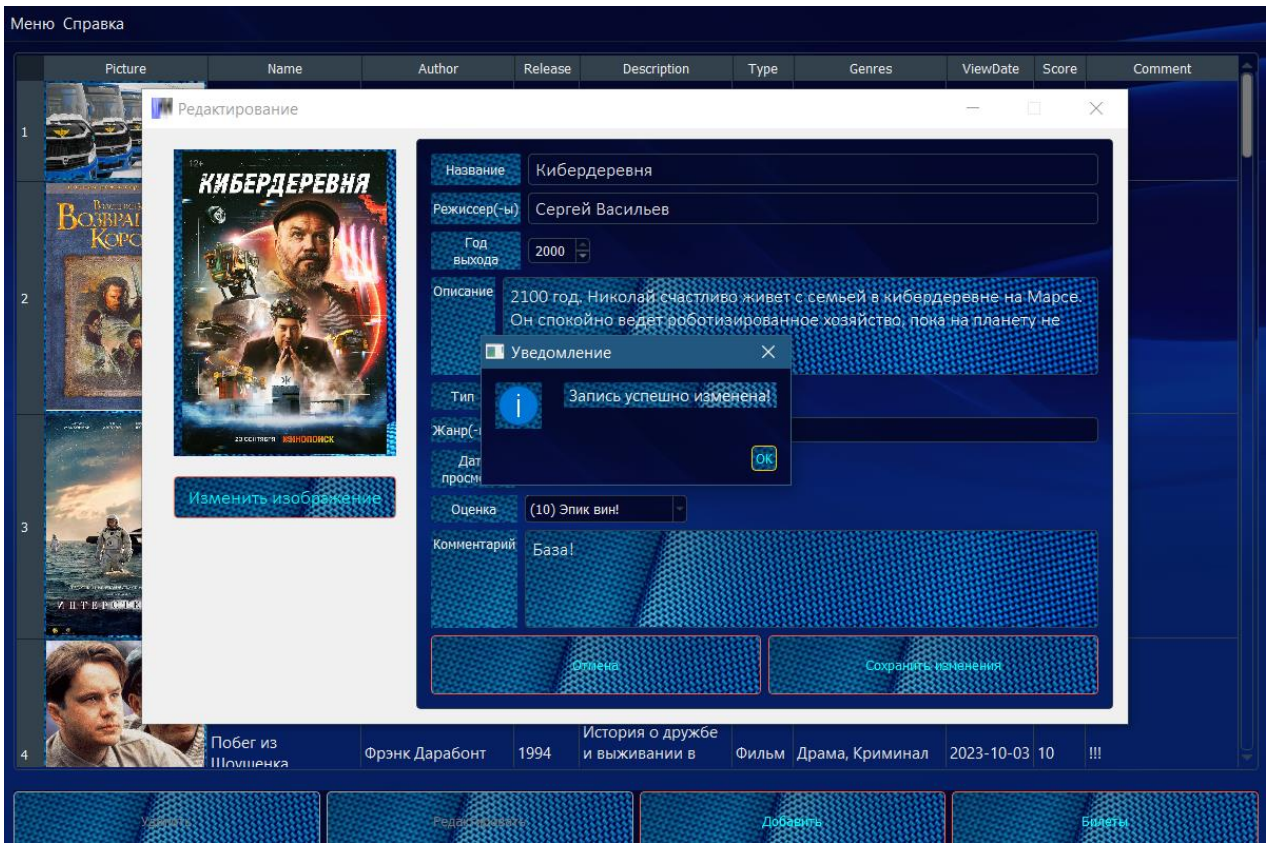
Скриншоты работы программы

(рис. 1) Окно заставки

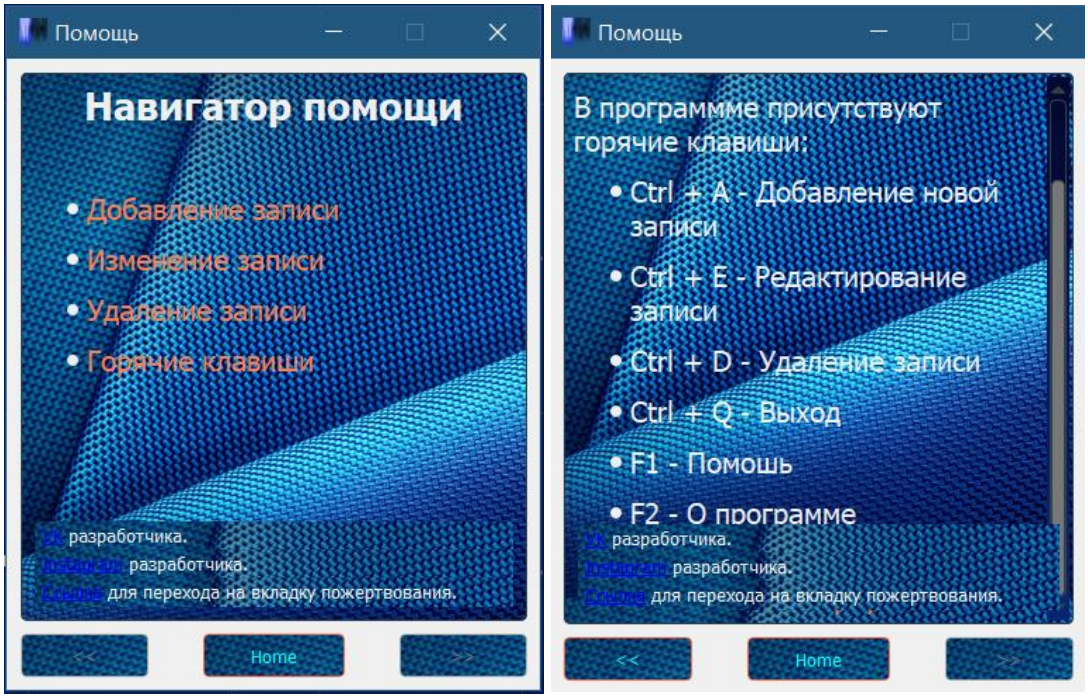
(рис. 2) Основное окно



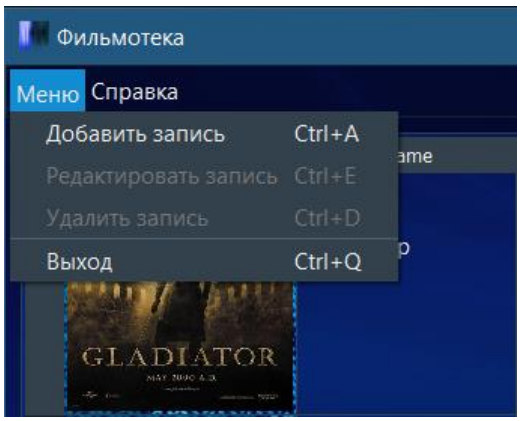
(рис. 3) Добавление записи



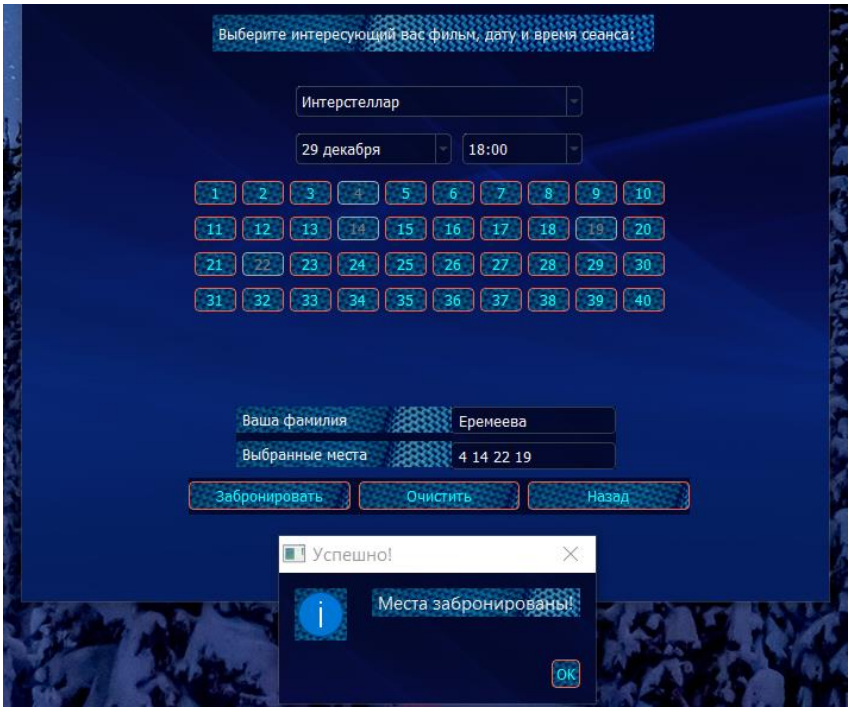
(рис. 4) Редактирование записи



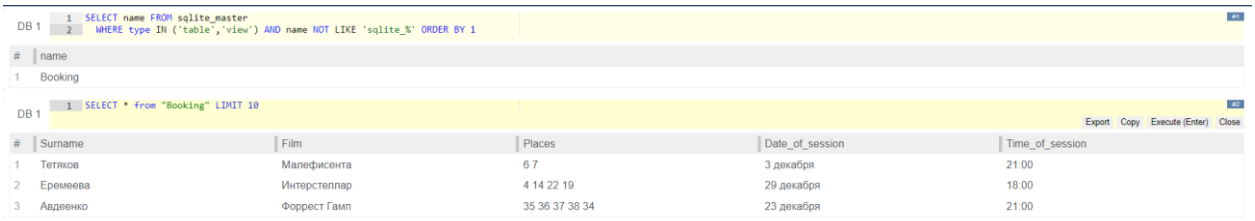
(рис. 5) Навигатор помощи



(рис. 6) Реализация Меню и Справки



(рис. 7) Реализация бронирования мест



DB 1

```
SELECT name FROM sqlite_master
  WHERE type IN ('table','view') AND name NOT LIKE 'sqlite_%' ORDER BY 1
```

#	name
1	Booking

DB 1

```
SELECT * from "Booking" LIMIT 10
```

Export Copy Execute (Enter) Close

#	Surname	Film	Places	Date_of_session	Time_of_session
1	Тетяков	Малефисента	6 7	3 декабря	21:00
2	Еремеева	Интерстеллар	4 14 22 19	29 декабря	18:00
3	Авдеенко	Форрест Гамп	35 36 37 38 34	23 декабря	21:00

(рис. 8) Сохранение информации о бронировании в базе данных

ЧАСТЬ 3

CWT-анализ

Задача №1:

Создание записи, но отмена ее добавления.

CWT-анализ первой задачи:

История действий пользователя, впервые столкнувшегося с данной задачей:

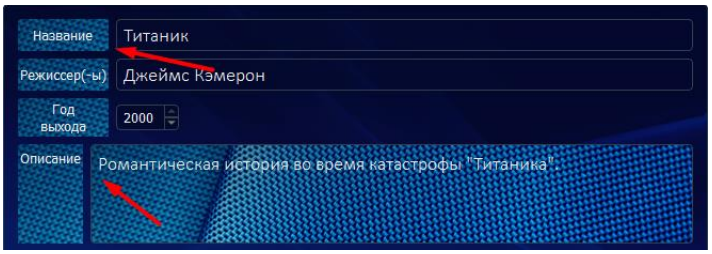
1. Запуск приложения

   Пользователь двойным кликом запускает приложение и попадает в основное окно программы
2. Нажатие кнопки «Добавить»

   Пользователь нажимает на кнопку «Добавить», создается диалоговое окно формы добавления записи

3. Заполнение полей

   Пользователь видит перед собой форму добавления, он заполняет поля «Название» и «Описание»



4. Нажатие кнопки «Отмена»;

   Пользователь нажимает на кнопку «Отмена», диалоговое окно формы добавления закрывается

Найденные проблемы:

1. Отсутствует какое-либо уведомление пользователя о том, что данные будут стерты;

Возможные способы решения:

1. Добавление диалогового окно с информацией о том, что при нажатии «Отмена», введенные данные будут стерты;
2. Не удалять данные при закрытии несохраненной формы;

Задача №2:

Удаление записи.

CWT-анализ первой задачи:

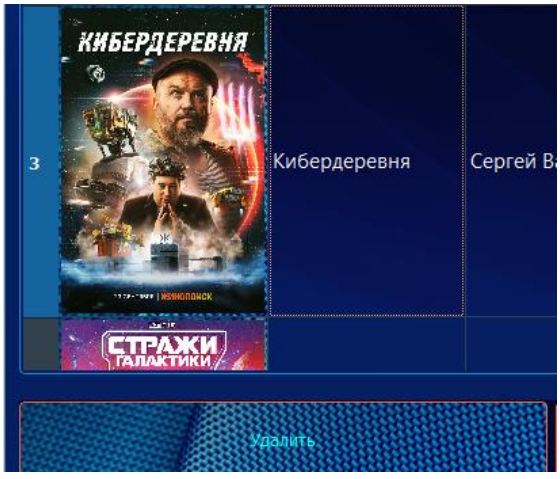
История действий пользователя, впервые столкнувшегося с данной задачей:

1. Запуск приложения

   Пользователь двойным кликом запускает приложение и попадает в основное окно программы

2. Выделение записи

   Пользователь в таблице с записями кликает на строку №1, строка выделяется, кнопка «Удалить» становится активной



3. Нажатие кнопки «Удалить»

   Пользователь нажимает на кнопку «Удалить», происходит вывод диалогового информационного окна об успешном удалении записи

Найденные проблемы:

1. Возможно случайное удаление записи;

Возможные способы решения:

1. Добавление диалогового окна с подтверждением удаления записи;

GOMS-анализ

Описание анализа

Практически все интерфейсные взаимодействия можно описать следующими операциями:

K – нажатие клавиши;

B – клик кнопкой мыши;

P – наведение указателя мыши;

R – ожидание ответной реакции компьютера;

H – перенос руки с клавиатуры на мышь или наоборот

D – проведение с помощью мыши прямой линии (например, выделение или прокрутка текста);

M – мыслительная подготовка (к осуществлению одной из перечисленных операций).

Разные пользователи выполняют указанные операции за разное время. Однако, GOMS исследует работу опытного пользователя. Многочисленные исследования выявили средние значения времени операций, выполняемых опытными пользователями.

- K 0.2 с
- B 0.2 с
- P 1.1 с
- H 0.4 с
- M 1.35 с

Цель №1: изменение записи с последующим ее сохранением

Для выполнения цели сформулируем подцели:

1. Выбрать интересующую запись из таблицы
2. Нажатие кнопки «Редактировать»
3. Установить указатель на поле «Название»
4. Удаление последнего символа в поле «Название»
5. Сохранение изменений

Теперь опишем методы для каждой подцели и распишем каждый метод с точностью до операции:

1. Выбрать интересующую запись из таблицы

   MPB (1.35 + 1.1 + 0.2 = 2.65)

2. Нажатие кнопки «Редактировать»

   MPB (1.35 + 1.1 + 0.2 = 2.65)

3. Установить указатель на поле «Название»

   MPB (1.35 + 1.1 + 0.2 = 2.65)

4. Удаление последнего символа в поле «Название»

   HK (0.4 + 0.2 = 0.6)

5. Сохранение изменений

   MHPB (1.35 + 0.4 + 1.1 + 0.2 = 3.05)

Итог: **(3 * 2.65 + 0.6 +3.05) = 11,6 с**

В качестве возможных решений: добавить горячую клавишу для выполнения пункта «Сохранение изменений», в качестве комбинации сохранения можно использовать общепринятую «Ctrl + S».

В данном случае удалось бы добиться следующего результата:

1. Выбрать интересующую запись из таблицы

   MPB (1.35 + 1.1 + 0.2 = 2.65)

2. Нажатие кнопки «Редактировать»
   MPB (1.1 + 0.2 = 2.65)
3. Установить указатель на поле «Название»

   MPB (1.35 + 1.1 + 0.2 = 2.65)

4. Удаление последнего символа в поле «Название»

   HK (0.4 + 0.2 = 0.6)

5. Сохранение изменений

   MKK (1.35 + 0.2 + 0.2 = 1.75)

Итог: **(2.65 + 1.3 + 0.2 + 2.65 + 0.6 +1.75) = 10.3 с**

*(Аналогично для формы добавления)

Цель №2: выход из приложения

Для выполнения цели сформулируем подцели:

1. Навестись на кнопку «Меню»
2. Закрыть программу выбором из контекстного меню пункта «Выход»

Теперь опишем методы для каждой подцели и распишем каждый метод с точностью до операции:

1. Навестись на кнопку «Меню»
   MPB (1.35 + 1.1 + 0.2 = 2.65)
2. Закрыть программу выбором из контекстного меню пункта «Выход»
   PB (1.1 + 0.2 = 1.3)

Итог: **(2.65 + 1.3) = 3.95 с**

Время также удалось бы сократилось путем добавления общепринятой горячей клавиши «Ctrl + Q», которая бы отвечала за моментальное закрытие программы после нажатия.

Тогда было бы:

1. Закрытие программы
   MKK (1.35 + 0.2 + 0.2 = 1.75)

Итог: **1.75 с**

Соответствие требованиям курсового проекта

Требования:	Выполнено:	Комментарий:
реализация основных задач пользователя по выбранной теме	+	Все основные задачи по теме Фильмотеки реализованы!
работа с файлами	+	В проекте пользователь работает с файлами, пример: изображения
использование SQLite	+	База данных реализована на языке SQLite, взаимодействующим с функционалом библиотеки QSql.
проверки на ввод некорректных значений в программе	+	Имеются. К примеру, уведомления о том, что поля остались пустыми, недопущение введения букв там, где требуются только цифры и т.д.
CWT, GOMS	+	Подробно описываются в отчете
использование таймера	+	При загрузке приложения открывается заставка с иммитацией загрузки от 0 до 100 (прописывается циклом)
информационная наполненность	+	Добавлена подробная информация по 15+ фильмам и сериалам.
сохранение результатов предыдущей работы пользователя	+	Вся информация успешно сохраняется.
использование менеджеров компоновки на всех формах	+	Интерфейс был проработан и скомпонован с учетом всех условий, рекомендаций и эстетических соображений.
использование QSettings	+	QSettings используется для чтения значения из настроек при создании главного окна, как пример.
реализация навигатора справки	+	Реализован, имеет гиперссылки, по которым открывается HTML документ.
Использование отдельного файла css для всего проекта	+	style.qss позволяет прописать стили для каждого элемента в объекте, также допуская объявление его настроек при каких-то действиях (нажатие, ожидание и т.д.)

Используемые источники

1. SQLite Viewer– использовался для просмотра базы данных в процессе разработки проекта;
   URL: https://inloop.github.io/sqlite-viewer/
2. Qt Documentation – официальная документация по разработке в среде Qt;
   URL: https://doc.qt.io/

Листинг

Главная страница интерфейса:

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QLabel>
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    // Устанавливаем фиксированный размер окна
    this->setMaximumSize(this->width(), this->height());
    this->setMinimumSize(this->width(), this->height());
    // Инициализация объекта базы данных и подключение к ней
    db = new DataBase();
    db->connectToDataBase();
    // Создание модели данных и настройка таблицы
    model = new QSqlTableModel;
    model->setTable(TABLE);
    model->setHeaderData(0, Qt::Horizontal, "id");
    ui->tableView->setModel(model);
    ui->tableView->setColumnHidden(0, true);
    ui->tableView->setSelectionBehavior(QAbstractItemView::SelectRows);
    ui->tableView->setSelectionMode(QAbstractItemView::SingleSelection);
    // Обновление таблицы и настройка формы редактирования и добавления записей
    updateTable();
    editForm = new EditRecord();
    editForm->setWindowModality(Qt::ApplicationModal);
    addForm = new AddRecord();
    addForm->setWindowModality(Qt::ApplicationModal);
    // Связывание сигналов и слотов для передачи данных между формами
    connect(addForm, SIGNAL(sendToWidget(QByteArray, QString, QString, int, QString, QString, QString, QDate, qint8,
QString)),
            this, SLOT(addRecordDataBase(QByteArray, QString, QString, int, QString, QString, QString, QDate, qint8, QString)));
    connect(this, SIGNAL(sendForEdit(int, QByteArray, QString, QString, int, QString, QString, QString, QDate, qint8,
QString)),
            editForm, SLOT(catchInfo(int, QByteArray, QString, QString, int, QString, QString, QString, QDate, qint8, QString)));
    connect(editForm, SIGNAL(sendToWidgetUpdate(int, QByteArray, QString, QString, int, QString, QString, QString, QDate,
qint8, QString)),
            this, SLOT(editRecordDataBase(int, QByteArray, QString, QString, int, QString, QString, QString, QDate, qint8,
QString)));
    // Связывание событий с кнопками и элементами меню
    connect(ui->menu1_add_record, SIGNAL(triggered()), this, SLOT(slotAdd()));
    connect(ui->addButton, SIGNAL(clicked()), this, SLOT(slotAdd()));
    connect(ui->menu1_edit_record, SIGNAL(triggered()), this, SLOT(slotEdit()));
    connect(ui->editButton, SIGNAL(clicked()), this, SLOT(slotEdit()));
    connect(ui->menu1_del_record, SIGNAL(triggered()), this, SLOT(slotDel()));
    connect(ui->delButton, SIGNAL(clicked()), this, SLOT(slotDel()));
    connect(ui->menu1_exit, SIGNAL(triggered()), qApp, SLOT(quit()));
    connect(ui->menu2_help, SIGNAL(triggered()), this, SLOT(slotInfo()));
    connect(ui->menu2_about, SIGNAL(triggered()), this, SLOT(slotAbout()));
}
MainWindow::~MainWindow()
{
    delete ui;
}
// Обновление таблицы
void MainWindow::updateTable()
{
    currRow = -1;
    currID = -1;
    model->select();
```

```
    model->setSort(8, Qt::AscendingOrder);
    // Выключение кнопок редактирования и удаления, если нет записей
    if (currRow == -1) {
        ui->menu1_edit_record->setEnabled(false);
        ui->menu1_del_record->setEnabled(false);
        ui->editButton->setEnabled(false);
        ui->delButton->setEnabled(false);
    } else {
        ui->menu1_edit_record->setEnabled(true);
        ui->menu1_del_record->setEnabled(true);
        ui->editButton->setEnabled(true);
        ui->delButton->setEnabled(true);
    }
    // Скрытие таблицы и отображение сообщения, если нет записей
    if (model->rowCount() == 0) {
        ui->label->show();
        ui->tableView->hide();
        return;
    }
    ui->label->hide();
    ui->tableView->show();
    QPixmap pic = QPixmap();
    // Загрузка изображений для каждой строки и установка их в ячейки таблицы
    for (int i = 0; i < model->rowCount(); i++) {
        pic.loadFromData(model->data(model->index(i, 1)).toByteArray());
        QLabel *imageLabel = new QLabel();
        imageLabel->setPixmap(pic.scaled(150, 400, Qt::KeepAspectRatio));
        ui->tableView->setIndexWidget(model->index(i, 1), imageLabel);
    }
    // Настройка размеров колонок
    ui->tableView->horizontalHeader()->setSectionResizeMode(1, QHeaderView::ResizeToContents);
    ui->tableView->horizontalHeader()->setSectionResizeMode(2, QHeaderView::Stretch);
    ui->tableView->horizontalHeader()->setSectionResizeMode(3, QHeaderView::Stretch);
    ui->tableView->horizontalHeader()->setSectionResizeMode(4, QHeaderView::ResizeToContents);
    ui->tableView->horizontalHeader()->setSectionResizeMode(5, QHeaderView::Stretch);
    ui->tableView->horizontalHeader()->setSectionResizeMode(6, QHeaderView::ResizeToContents);
    ui->tableView->horizontalHeader()->setSectionResizeMode(7, QHeaderView::Stretch);
    ui->tableView->horizontalHeader()->setSectionResizeMode(8, QHeaderView::ResizeToContents);
    ui->tableView->horizontalHeader()->setSectionResizeMode(9, QHeaderView::ResizeToContents);
    ui->tableView->horizontalHeader()->setSectionResizeMode(10, QHeaderView::Stretch);

    ui->tableView->verticalHeader()->setSectionResizeMode(QHeaderView::ResizeToContents);
    ui->tableView->setEditTriggers(QAbstractItemView::NoEditTriggers);
}
// Слот для открытия формы добавления записи
void MainWindow::slotAdd()
{
    addForm->show();
}
// Слот для добавления записи в базу данных
void MainWindow::addRecordDataBase(const QByteArray &pic, const QString &name, const QString &author, const int
&releaseYear, const QString &description, const QString &type, const QString &genres, const QDate &viewDate, const qint8
&score, const QString &comment)
{
    db->insertIntoTable(pic, name, author, releaseYear, description, type, genres, viewDate, score, comment);
    updateTable();
    currRow = -1;
    currID = -1;
}
// Слот для открытия формы редактирования записи
void MainWindow::slotEdit()
{
    if (currRow != -1) {
        QString name = model->data(model->index(currRow, 2)).toString();
```

```
        QString author = model->data(model->index(currRow, 3)).toString();
        int releaseYear = model->data(model->index(currRow, 4)).toDate().year();
        QString description = model->data(model->index(currRow, 5)).toString();
        QString type = model->data(model->index(currRow, 6)).toString();
        QString genres = model->data(model->index(currRow, 7)).toString();
        QDate viewDate = model->data(model->index(currRow, 8)).toDate();
        qint8 score = static_cast<qint8>(model->data(model->index(currRow, 9)).toInt());
        QString comment = model->data(model->index(currRow, 10)).toString();
        QByteArray inByteArray = model->data(model->index(currRow, 1)).toByteArray();
        emit sendForEdit(currID, inByteArray, name, author, releaseYear, description, type, genres, viewDate, score, comment);
        editForm->show();
    }
}
// Слот для редактирования записи в базе данных
void MainWindow::editRecordDataBase(const int &id, const QByteArray &pic, const QString &name, const QString &author,
const int &releaseYear, const QString &description, const QString &type, const QString &genres, const QDate &viewDate,
const qint8 &score, const QString &comment)
{
    currRow = -1;
    currID = -1;
    db->editInTable(id, pic, name, author, releaseYear, description, type, genres, viewDate, score, comment);
    updateTable();
}
// Слот для удаления записи из базы данных
void MainWindow::slotDel()
{
    if (currRow != -1) {
        db->deleteFromDatabase(currID);
        updateTable();
        QMessageBox::information(nullptr, "Уведомление", "Запись успешно удалена");
        currRow = -1;
        currID = -1;
    }
}
// Слот для открытия формы справки
void MainWindow::slotInfo()
{
    HelpInformation* form = new HelpInformation();
    form->setWindowModality(Qt::ApplicationModal);
    form->show();
}
// Слот для открытия окна "О программе"
void MainWindow::slotAbout()
{
    QMessageBox::about(this, "О программе", "Версия: 1.3 Alpha\n\nРазработчик: Тетяков Илья, ИП-111\n\n          © 2023-
2024 уч.год, СибГУТИ");
}
// Слот для обработки события клика по строке таблицы
void MainWindow::on_tableView_clicked(const QModelIndex &index)
{
    ui->menu1_edit_record->setEnabled(true);
    ui->menu1_del_record->setEnabled(true);
    ui->editButton->setEnabled(true);
    ui->delButton->setEnabled(true);
    currID = model->data(model->index(index.row(), 0)).toInt();
    currRow = index.row();
}
```

Исходный код проекта со всеми материалами опубликован в GitHub по ссылке: https://github.com/Diablox777/QT-Filmoteka-CursWork/tree/main