

МИНИСТЕРСТВО ОБРАЗОВАНИЯ КИРОВСКОЙ ОБЛАСТИ
Кировское областное государственное профессиональное образовательное
бюджетное учреждение
"Слободской колледж педагогики и социальных отношений"

КУРСОВОЙ ПРОЕКТ

по ПМ 01 «Разработка программных модулей» на тему:
**РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ДЛЯ АВТОМАТИЗАЦИИ
ПРОДАЖИ БИЛЕТОВ В КИНОТЕАТРЕ**

Выполнил: Платунов Павел
Андреевич

Специальность 09.02.07
Информационные системы и
программирование

Группа 21П-1
Форма обучения: очная

Руководитель: Калинин
Арсений Олегович

Дата защиты курсового проекта:

Председатель ПЦК:

Оценка за защиту курсового проекта:

Слободской
2024

ОГЛАВЛЕНИЕ

Введение	3
1. Анализ предметной области.....	5
2. Разработка технического задания.....	12
3. Описание алгоритмов и схема функционирования программного модуля	16
4. Тестирование программного модуля	22
Заключение	25
Список литературы	26
Приложения	28

ВВЕДЕНИЕ

В настоящее время достаточно популярным местом отдыха являются кинотеатры. Каждый день множество людей посещают различные кинотеатры для просмотра как недавно, так и давно вышедших фильмов, мультфильмов и других кинокартин. Из-за большого количества клиентов могут возникать очереди при продаже билетов на фильмы. Все это зависит от способности кассира быстро и эффективно работать, а также от наличия современных систем автоматизации.

Раньше продажа билетов производилась только в физическом виде, что занимало хоть и мало времени на заполнение одного билета, но при наплыве клиентов могло уходить очень много времени на обслуживание всей толпы. Тоже самое можно сказать и про сбор и анализ данных при работе кинотеатра, когда нужно было записывать количество проданных билетов и их цену на разные фильмы для того, чтобы в конце рабочего дня подвести итоги продаж и решить является ли показываемый фильм коммерчески выгодным. Сейчас же мы имеем возможность автоматизировать данные бизнес-процессы для повышения эффективности работы кассира и администрации кинотеатра.

Среди основных преимуществ такой автоматизации можно отметить:

- Сокращение времени обслуживания клиентов за счет автоматизации регистрации на сеанс по выбранному фильму и автоматизации печати билетов на выбранный клиентом фильм;
- Увеличение количества обрабатываемых клиентов, которое происходит за счет повышения эффективности работы кассира и сокращения времени обслуживания клиента;
- Повышение удобства клиента за счет уменьшения очередей из людей, которые хотят посмотреть фильм;
- Снижение затрат на содержание персонала за счет уменьшения количества кассиров, ведь при введении систем автоматизации эффективность

работы кассиров увеличиться, а время, затрачиваемое на обслуживание одного клиента уменьшиться;

- Сбор и аналитика данных также будут гораздо быстрее, эффективней, точней, а что самое важное удобней, ведь с введением автоматизации теперь можно отслеживать проданные билеты как на фильмы, так и на отдельный сеанс данного фильма для составления вывода о успешности показываемой кинокартины.

Цель курсового проекта – разработка программного обеспечения для автоматизации продажи билетов на фильмы в кинотеатре.

Задачи исследования:

- Описать предметную область.
- Разработать технического задание на создание программного продукта.
- Описать архитектуру программы.
- Описать алгоритмы и функционирование программы.
- Провести тестирование и опытную эксплуатацию.
- Разработать руководство оператора

Объект исследования – процесс продажи и учета проданных билетов в кинотеатре.

Предмет исследования – разработка программного модуля для автоматизации продажи билетов в кинотеатре.

Методы исследования: системный анализ и функциональное моделирование.

Информационную систему исследования составили официальные нормативно-правовые источники, данные об использовании современных информационных систем. Структура работы состоит из введения, трех глав, заключения, списка используемой литературы и приложений.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Кинотеатр — общественное здание или его часть с оборудованием для публичной демонстрации кинофильмов. И для его успешного функционирования необходимо выполнять множество различных функций таких как добавление новых фильмов, составление сеансов по фильмам, продажа билетов, подготовка и показ кинолент, и анализ данных для определения успешности фильма. Успешное функционирование системы зависит от четкой организации работы персонала и эффективного взаимодействия между его различными ролями.

Разрабатываемый программный модуль должен обладать возможностями выбора фильма и конкретного сеанса, на который посетитель хочет сходить, также обеспечить возможность выбора места в зале и печать билета после бронирования места. Еще программный модуль должно обеспечить базовый набор функциональности для обеспечения работы модуля без работы пользователя с базой данных на прямую через сторонние системы управления базой данных такие как SQL Server Management Studio, а также иметь возможность собирать и анализировать данные, полученные с продажи билетов для определения успешности фильма. Дополнительно стоит добавить возможность настройки конфигурации зала для большей универсальности программного модуля.

Для выполнения всех этих немало важных функций необходим обученный персонал:

- Администратор — управляет системой, работает с базой данных, настраивает конфигурацию зала, управляет персоналом кинотеатра.
- Директор — управляет персоналом, отслеживает продажи билетов на фильмы.
- Букер (менеджер по кинопрокату) — добавляет новые фильмы, составляет и корректирует расписание сеансов, отслеживает продажи билетов на фильмы.

- Кассир – работает с расчетно-кассовым оборудованием, продает билеты на фильмы.

Для более четкого понятия рабочих задач персонала была составлена диаграмма вариантов использования (Рисунок 1).

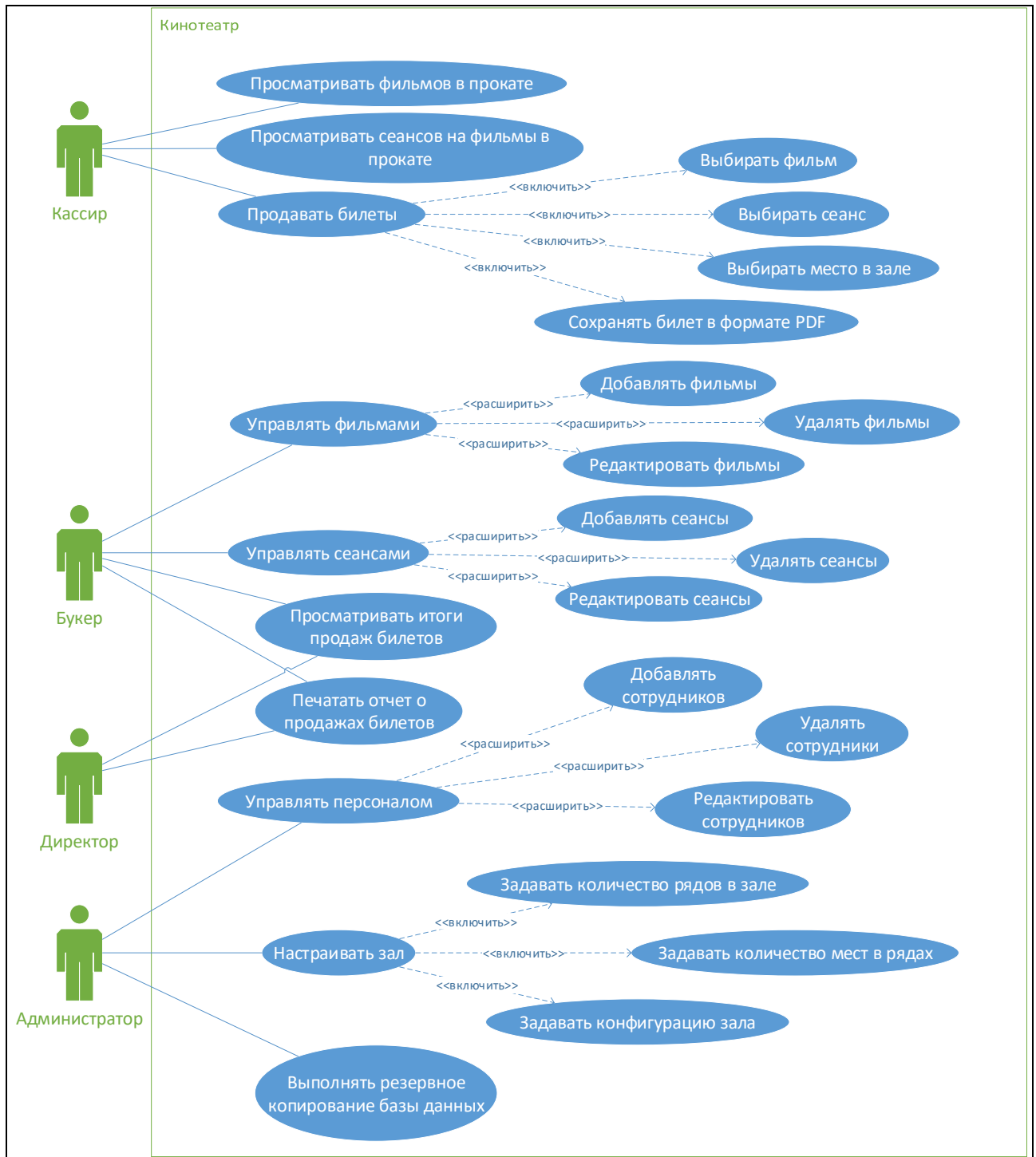


Рисунок 1 - диаграмма вариантов использования

По выделенным объектам и атрибутам была составлена ER – диаграмма показывающая структура создаваемой базы данных (Рисунок 2).

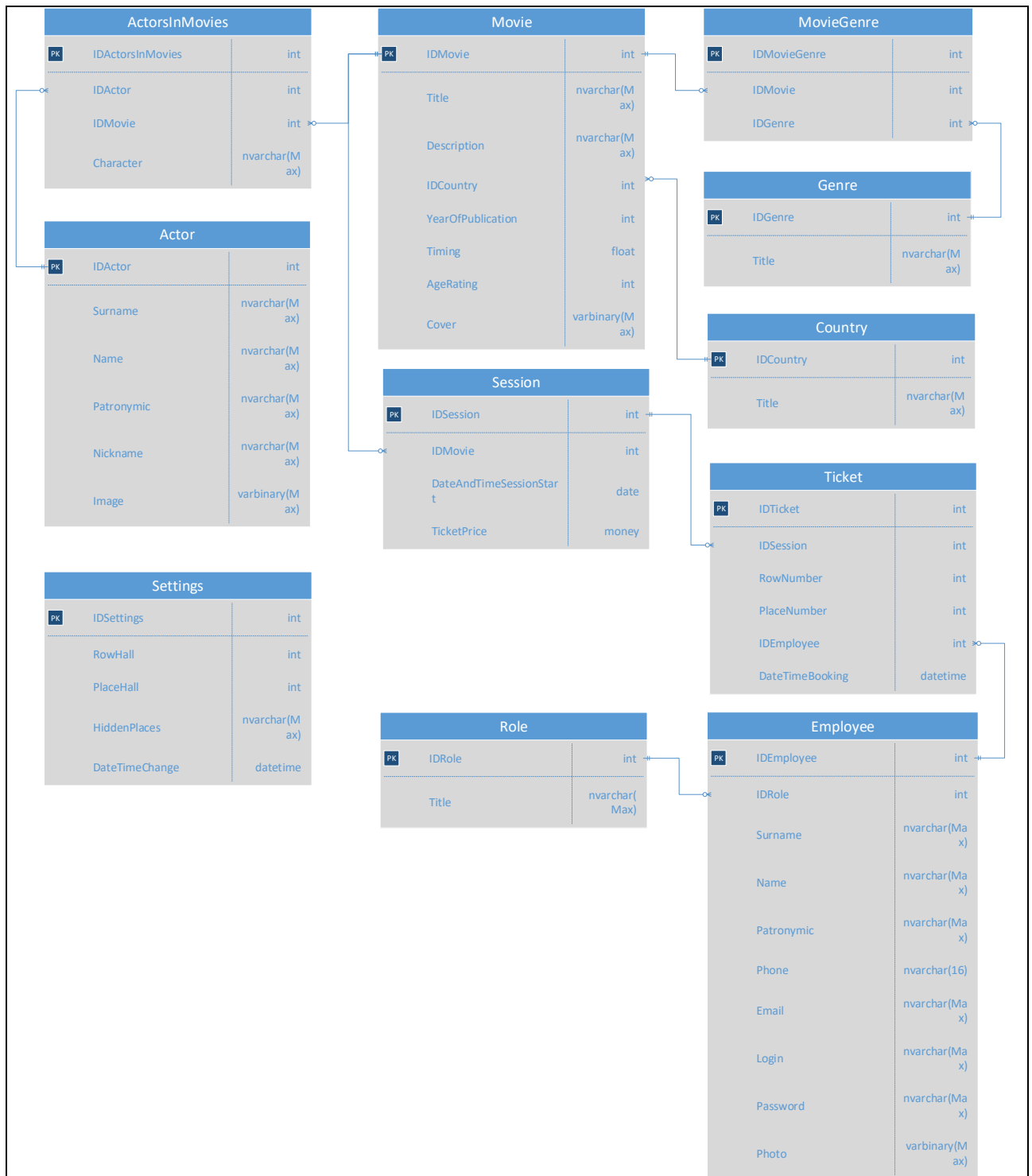


Рисунок 2 - ER диаграмма

С помощью построенной ER диаграммы, а также выделенным объектам и атрибутам была создана база данных (Рисунок 3).

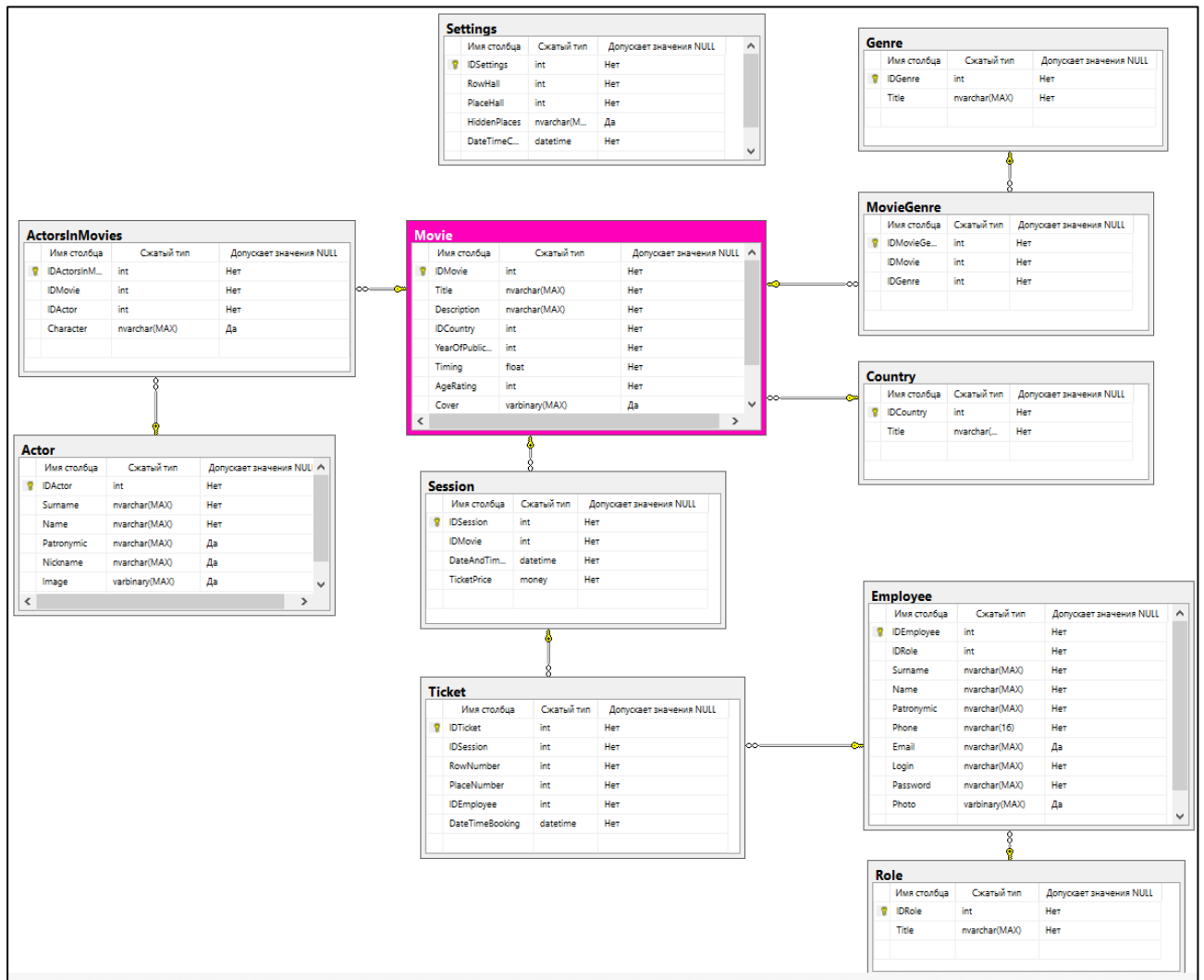


Рисунок 3 - Диаграмма базы данных

Программы аналоги:

InTickets - это платформа для продажи билетов онлайн. Она предоставляет инструменты для создания мероприятий, управления билетами, продажи, обработки платежей и анализа данных. InTickets ориентирована на организаторов событий всех размеров, от небольших концертов до крупных фестивалей (Рисунок 4).

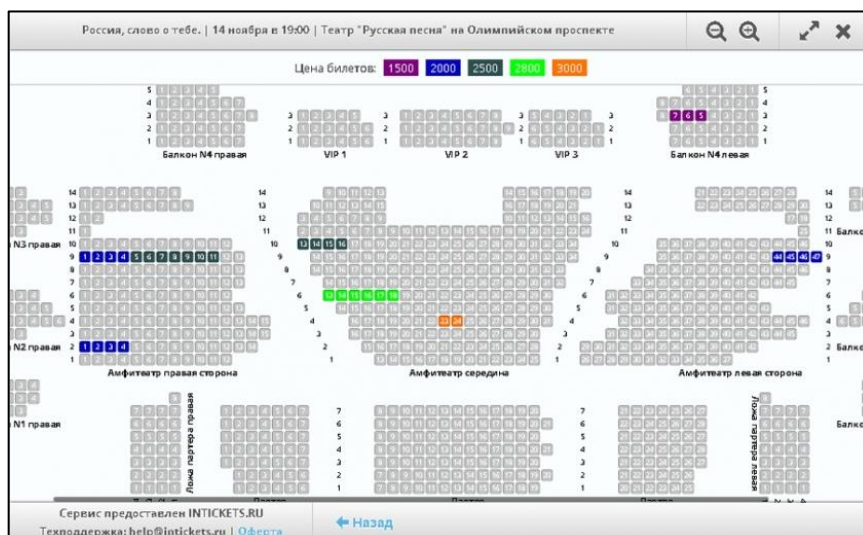


Рисунок 4 – InTickets

Преимущества:

- Есть мобильное приложение
- Гибкая настройка приложения позволяет адаптировать систему под специфику кинотеатра.
- Интеграция с различными системами такими как 1С, Bitrix.
- Стабильная работа
- Быстрая и качественная техническая поддержка

Недостатки:

- Высокая цена покупки с предварительной договоренностью с поставщиком
- Сложный в понимание интерфейс
- Ограниченное количество функций в бесплатной версии

TicketTool.net - это онлайн-сервис для создания и продажи билетов на события. Он предлагает простой и интуитивный интерфейс для создания мероприятий, установки цен на билеты, настройки дизайна и интеграции с различными платежными системами. TicketTool.net подходит для организаторов небольших мероприятий, таких как концерты, спектакли, семинары и т. д. (Рисунок 5).

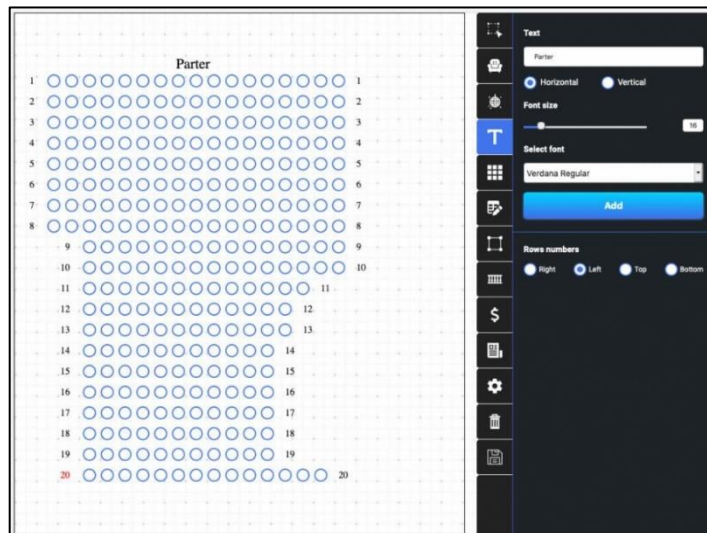


Рисунок 5 - TicketTool.net

Преимущества:

- Низкая стоимость программы 2,5% с продаж
- Простой интерфейс
- Удобный функционал позволяет легко продавать билеты, управлять

расписанием сеансов и отслеживать продажи.

- Мобильная версия
- Бесплатная пробная демо версия

Недостатки:

- Ограниченные возможности настройки
- Меньше интеграций чем у InTicket
- Техническая поддержка своим качеством может варьироваться в

зависимости от купленного плана

Ticket Tech - это компания, которая разрабатывает и предлагает программное обеспечение для управления билетами. Их программное обеспечение позволяет организаторам событий продавать билеты онлайн, на месте и через мобильные приложения. Ticket Tech фокусируется на обеспечении комплексного решения для управления билетами, которое включает в себя функции, такие как управление местами, управление запасами, обработка платежей и отчетность (Рисунок 6).

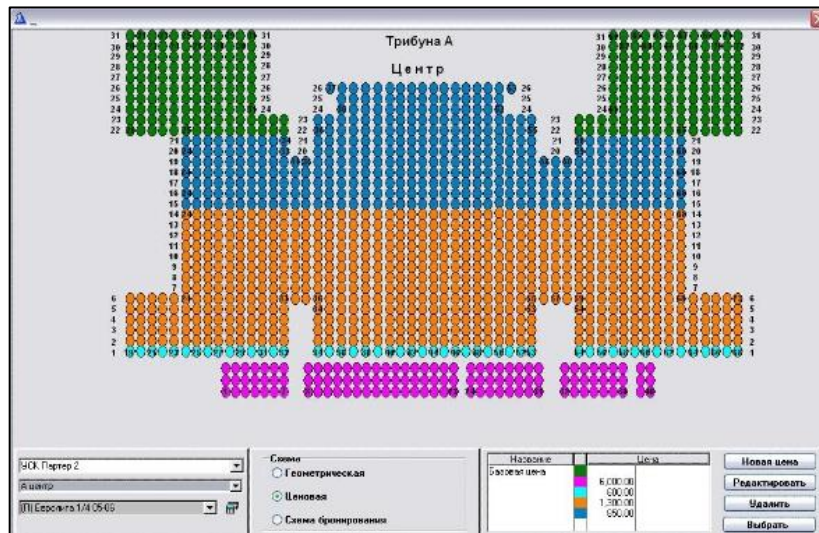


Рисунок 6 - Ticket Tech

Преимущества:

- Предлагает передовые решения, например, автоматизированный маркетинг и анализ данных

- Высокая скорость обработки данных
- Позволяет легко продвигать кинотеатр в социальных сетях
- Обеспечивает высокую степень защиты информации
- Многоязычная поддержка

Недостатки:

- Может быть очень дорогой до 12% с продаж
- Сложность настройки
- Ограниченное количество интеграций
- Отсутствие бесплатной пробной версии

2. РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ

Наименование программы – «Cinema». Программа предназначена для автоматизации работы кинотеатра ГОСТ 19.101-77 [1], ГОСТ 19.201-78 [2].

Разработка программы ведется на основании учебного плана и перечня тем утвержденных на заседании предметно цикловой комиссии информатики и программирования.

Функциональным назначением программы является автоматизация продажи билетов в кинотеатре.

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

- Управление сотрудниками;
- Управление фильмами;
- Управление сеансами;
- Настройка конфигурации зала;
- Расчет количества проданных билетов на фильмы;
- Расчет количества проданных билетов на сеансы выбранного фильма;
- Сохранение отчета по количеству проданных билетов на фильмы в файл формата pdf;
- Сохранение отчета по количеству проданных билетов на сеансы выбранного фильма в файл формата pdf;
- Продажа билетов на выбранный сеанс фильма;
- Открытие pdf файла проданного билета для его печати или сохранения;

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением заказчиком совокупности организационно-технических мероприятий, перечень которых приведен ниже:

- организация бесперебойного питания технических средств;

- использование лицензионного программного обеспечения;
- отсутствие вредоносного программного обеспечения, наличие антивирусной программы;
- соблюдение правил и требований по эксплуатации технических средств.

Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать 5 минут при условии соблюдения условий эксплуатации технических и программных средств.

Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.

Отказы программы возможны вследствие некорректных действий оператора (пользователя) при взаимодействии с операционной системой. Во избежание возникновения отказов программы по указанной выше причине следует обеспечить работу пользователя без предоставления ему административных привилегий.

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

В состав технических средств должен входить IBM-совместимый персональный компьютер (ПЭВМ), включающий себя:

- процессор с тактовой частотой, 1 ГГц, не менее;
- оперативную память объемом 512 Мб, не менее;
- жесткий диск со свободным местом 500 Мб, не менее;
- монитор, с разрешением экрана 1366*768, не менее;
- оптический привод;
- компьютерная мышь;
- клавиатура;

Исходные коды программы должны быть реализованы на языке C#. В качестве интегрированной среды разработки программы должна быть использована среда программирования Microsoft Visual Studio 2022 и Microsoft SQL Server 2014 Management Studio.

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционной системы Windows 7/8/10/11.

Программное обеспечение поставляется в виде изделия на CD диске либо USB накопителя.

Упаковка программного изделия должна осуществляться в упаковочную тару предприятия-изготовителя компакт диска или USB накопителя.

Требования к транспортировке и хранению должны соответствовать условиям эксплуатации носителей, на которых находится программный продукт.

Программа должна обеспечивать взаимодействие с пользователем посредством графического пользовательского интерфейса.

Предварительный состав программной документации включает в себя следующие документы:

- техническое задание;
- руководство оператора.

Разработка должна быть проведена в следующие стадии и этапы:

1. Анализ требований:

На стадии анализ требований формулируются цели и задачи проекта. Создается основа для дальнейшего проектирования

2. Проектирование:

На стадии проектирование должны быть выполнены перечисленные ниже этапы работ:

- разработка программной документации;

На этапе разработка программной документации должна быть выполнена разработка технического задания.

При разработке технического задания должны быть выполнены перечисленные работы: постановка задачи, определение и уточнение требований к техническим средствам, определение требований к программе, определение стадий, этапов и сроков разработки программы и документации на нее, выбор языков программирования.

- разработка алгоритма программы;

На этапе разработки алгоритма программы должен быть разработан алгоритм работы программы.

- кодирование;

На стадии кодирования происходит реализация алгоритмов в среде программирования.

- тестирование и отладка.

На стадии тестирования и отладки происходит проверка алгоритмов, реализованных в программе на работоспособность в различных ситуациях. Исправление выявленных ошибок, повторное тестирование.

Приемо-сдаточные испытания должны проводиться при использовании технических средств. Приемка программы заключается в проверке работоспособности программы путем ввода реальных или демонстрационных данных.

Во время приемки работы разработчик предоставляет программу и документацию, которая к ней прилагается. Проводятся испытания программы, при успешных испытаниях программа вводится в эксплуатацию. При ошибках, недопустимых для успешной работы программного продукта – отправляется на доработку.

Было описано техническое задание, содержащее в себе информацию о программном продукте, его функциях, эксплуатации и требования, которые должны учитываться при создании программы и документации к ней.

3. ОПИСАНИЕ АЛГОРИТМОВ И СХЕМА ФУНКЦИОНИРОВАНИЯ ПРОГРАММНОГО МОДУЛЯ

Наименование программы – «Сinema». Программа предназначена для автоматизации продажи билетов на сеансы фильмов.

Функциональным назначением программы является автоматизация процесса продажи билетов.

Перед продажей билета на сеанс фильма кассир должен выбрать место в зале, после выбора места в зале информация о нем отобразиться на информационной панели “Бронирование билета” (Рисунок 7).

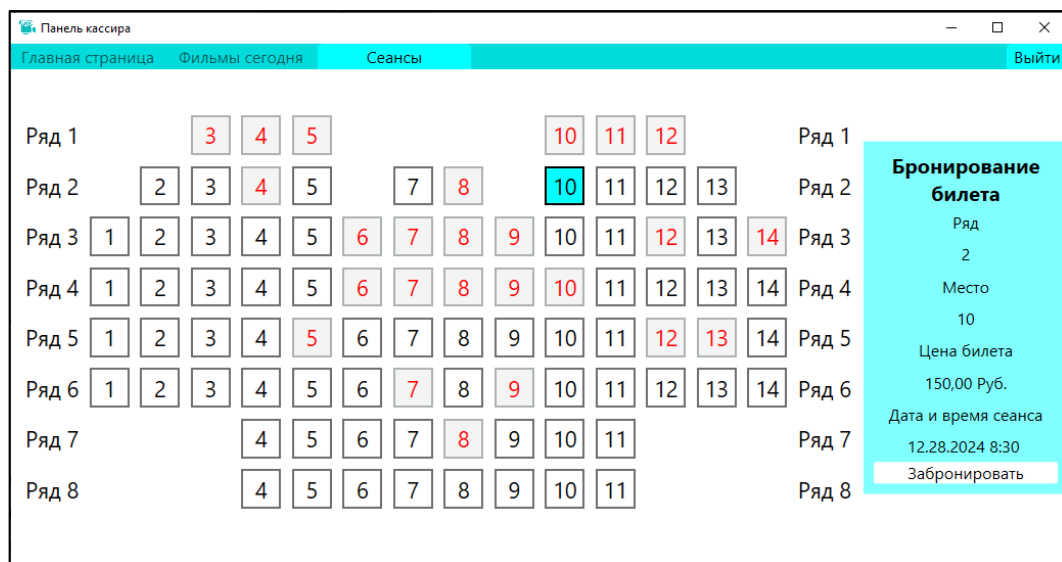


Рисунок 7 – Окно продажи билета на выбранный сеанс

Для отображения выбранного места в зале используется метод «HallButton_Click» данный метод определяет нажатую кнопку меняет её стилистику и выводит данные о выбранном месте на информационную панель (Рисунок 8,9).

```
private void HallButton_Click(object sender, RoutedEventArgs e)
{
    try
    {
        Button clickedButton = sender as Button;
        if (clickedButton != null)
        {
```

Рисунок 8 – Код метода HallButton_Click


```

if (selectedRowPlaceButton != null)
{
    selectedRowPlaceButton.Background = Brushes.White;
    selectedRowPlaceButton.Foreground = Brushes.Black;
    selectedRowPlaceButton.BorderBrush = (Brush)(new BrushConverter().ConvertFrom("#FF7070"));
}

Match match = Regex.Match(clickedButton.Name, @"Row(\d+)Place(\d+)");

if (match.Success)
{
    selectedRowNumber = int.Parse(match.Groups[1].Value) + 1;
    selectedPlaceNumber = int.Parse(match.Groups[2].Value);
    clickedButton.Background = Brushes.Aqua;
    clickedButton.Foreground = Brushes.Black;
    clickedButton.BorderBrush = Brushes.Black;
    selectedRowPlaceButton = clickedButton;
}

SelectedPlaceChange();
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
}
}

```

Рисунок 9 – Продолжение метода HallButton_Click

Поле выбора места в зале кассир нажимает на кнопку “Забронировать” и подтверждает свой выбор (Рисунок 10)

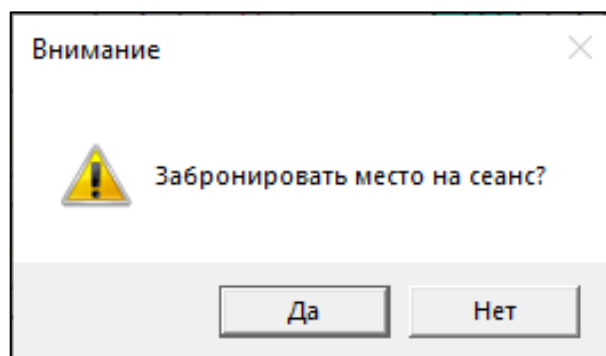


Рисунок 10 – Окно подтверждения бронирования

В результате перед кассиром откроется документ формата PDF с готовым к печати или сохранению билетом (Рисунок 11).

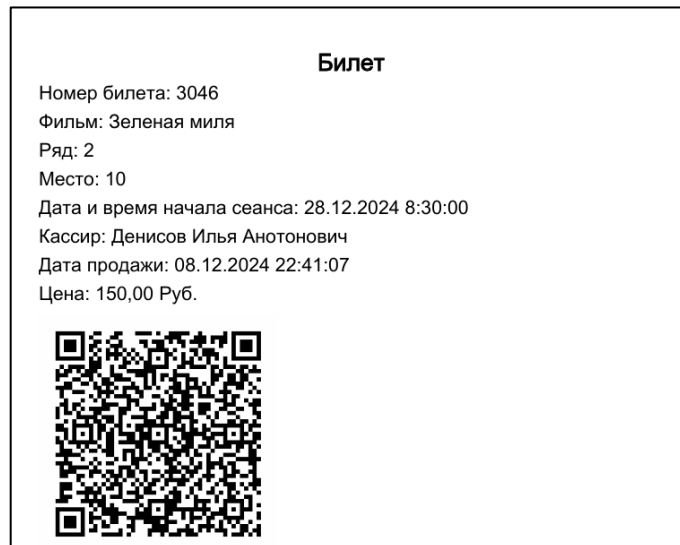


Рисунок 11 – Билет на выбранный сеанс фильма

Алгоритм работы модуля продажи билетов приведен ниже (Рисунок 12).

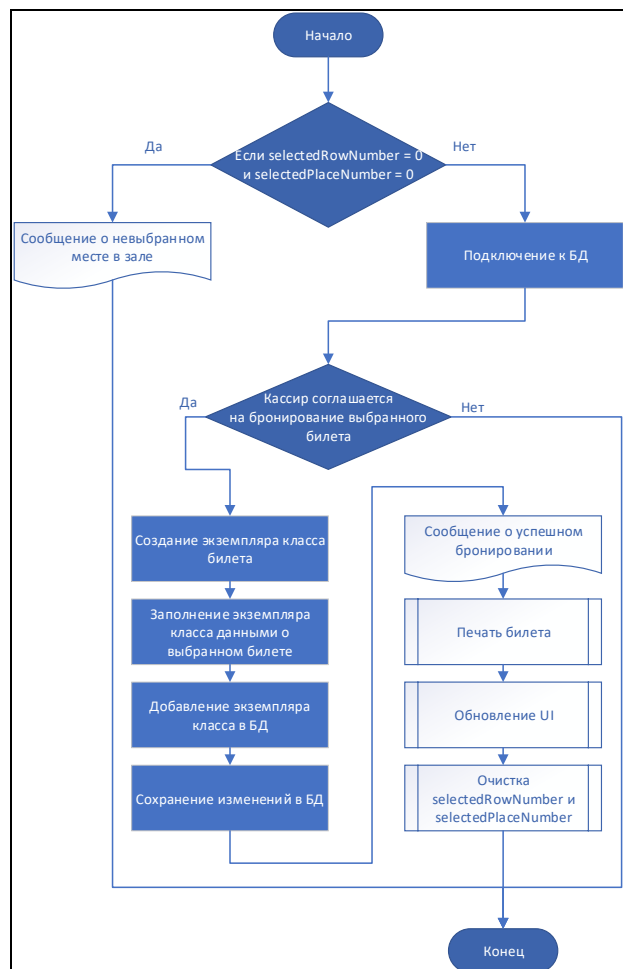


Рисунок 12 - Блок схема функции по продаже билетов на выбранный сеанс фильма

Для сохранения информации о забронированном месте на выбранный сеанс кино используется метод «BookingTicket_Click», данный метод проверяет выбрал ли кассир место в зале, если нет то выводит соответствующее сообщение, в противном случае выводит сообщение пользователю для подтверждения бронирования билета в случае согласия кассира метод создает экземпляр класса Ticket в который записывает выбранное место и ряд в зале, сотрудника осуществляющего данную операцию, а также дату и время бронирования билета (Рисунок 13,14).

```
private void BookingTicket_Click(object sender, RoutedEventArgs e)
{
    try
    {
        if (selectedRowNumber == 0 && selectedPlaceNumber == 0)
        {
            MessageBox.Show("Место в зале не выбранно", "Внимание", MessageBoxButton.OK,
            MessageBoxImage.Warning);
            return;
        }

        using (var dataBase = new CinemaEntities())
        {
            if (MessageBox.Show("Забронировать место на сеанс?", "Внимание", MessageBoxButton.YesNo,
            MessageBoxImage.Warning) == MessageBoxResult.Yes)
            {
                var newTicket = new Ticket();

                newTicket.IDSession = TransmittedData.idSelectedCashierSession;
                newTicket.RowNumber = selectedRowNumber;
                newTicket.PlaceNumber = selectedPlaceNumber;
                newTicket.IDEmployee = TransmittedData.idEmployee;
                newTicket.DateTimeBooking = DateTime.Now;

                dataBase.Ticket.Add(newTicket);
                dataBase.SaveChanges();

                MessageBox.Show("Место на сеанс зарезервировано", "Готово", MessageBoxButton.OK,
                MessageBoxImage.Information);

                Task.Run(() => PrintTicket(newTicket.IDTicket));

                selectedRowPlaceButton.Background = (Brush)(new BrushConverter().ConvertFrom("#FFDDDDDD"));
                selectedRowPlaceButton.BorderBrush = (Brush)(new BrushConverter().ConvertFrom("#FF707070"));
                selectedRowPlaceButton.Foreground = Brushes.Red;
                selectedRowPlaceButton.IsEnabled = false;

                ClearPlaceChange();
            }
        }
    }
    catch (Exception ex)
```

Рисунок 13 – Код метода BookingTicket_Click

```

    {
        MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

```

Рисунок 14 – Продолжение метода BookingTicket_Click

После сохранения информации в базе данных запускается метод «PrintTicket» (Рисунок 15,16). Данный метод предназначен для вывода билета в PDF формате кассиру. Данный метод использует библиотеку «ITextSharp» для формирования PDF файла и библиотеку «QR Coder» для формирования QR кода.

```

private async Task PrintTicket(int selectedTicket)
{
    try
    {
        using (var dataBase = new CinemaEntities())
        {
            var ticketData = (from ticket in dataBase.Ticket
                             join
                             session in dataBase.Session on ticket.IDSession equals session.IDSession into sessionGroup
                             from session in sessionGroup.DefaultIfEmpty()
                             join
                             movie in dataBase.Movie on session.IDMovie equals movie.IDMovie into movieGroup
                             from movie in movieGroup.DefaultIfEmpty()
                             join
                             employee in dataBase.Employee on ticket.IDEmployee equals employee.IDEmployee into
employeeGroup
                             from employee in employeeGroup.DefaultIfEmpty()
                             where (ticket.IDTicket == selectedTicket)
                             select new
                             {
                                 ticket.IDTicket,
                                 ticket.RowNumber,
                                 ticket.PlaceNumber,
                                 ticket.DateTimeBooking,
                                 movie = movie.Title,
                                 session.DateAndTimeSession,
                                 employee.Surname,
                                 employee.Name,
                                 employee.Patronymic,
                                 session.TicketPrice
                             }).FirstOrDefault();

            string tempFilePath = Path.GetTempFileName() + ".pdf";

            string qrCodeText =
$"{{ticketData.IDTicket}}|{{ticketData.movie}}|{{ticketData.DateAndTimeSession}}|{{ticketData.Surname}}
{{ticketData.Name}} |{{ticketData.Patronymic}}|{{Math.Round(ticketData.TicketPrice,2)}}";
            QRCodeGenerator qrGenerator = new QRCodeGenerator();
            QRCodeData qrCodeData = qrGenerator.CreateQRCode(qrCodeText, QRCodeGenerator.ECCLLevel.Q);
            QRCode qrCode = new QRCode(qrCodeData);
            using (System.Drawing.Bitmap qrCodeBitmap = qrCode.GetGraphic(20, System.Drawing.Color.Black,
System.Drawing.Color.White, null))
            {

```

Рисунок 15 – Код метода PrintTicket

```

        using (MemoryStream ms = new MemoryStream())
        {
            qrCodeBitmap.Save(ms, System.Drawing.Imaging.ImageFormat.Png); // Сохраняем в PNG
            byte[] qrCodeBytes = ms.ToArray();

            Document doc = new Document();
            PdfWriter writer = PdfWriter.GetInstance(doc, new FileStream($"{tempFilePath}", FileMode.Create));

            BaseFont baseFont = BaseFont.CreateFont("C:\\Windows\\Fonts\\arial.ttf", BaseFont.IDENTITY_H,
            BaseFont.NOT_EMBEDDED);
            Font font = new Font(baseFont, 16);
            BaseFont baseFontHead = BaseFont.CreateFont("C:\\Windows\\Fonts\\arial.ttf",
            BaseFont.IDENTITY_H, BaseFont.NOT_EMBEDDED);
            Font fontHead = new Font(baseFont, 20, Font.BOLD);

            doc.Open();
            Paragraph mainParagraph = new Paragraph("Билет", fontHead);
            mainParagraph.Alignment = Element.ALIGN_CENTER;
            Paragraph paragraph = new Paragraph("Номер билета: " + ticketData.IDTicket, font);
            Paragraph paragraph1 = new Paragraph("Фильм: " + ticketData.movie, font);
            Paragraph paragraph2 = new Paragraph("Ряд: " + ticketData.RowNumber, font);
            Paragraph paragraph3 = new Paragraph("Место: " + ticketData.PlaceNumber, font);
            Paragraph paragraph4 = new Paragraph("Дата и время начала сеанса: " +
            ticketData.DateAndTimeSession, font);
            Paragraph paragraph5 = new Paragraph("Кассир: " + ticketData.Surname + " " + ticketData.Name + " "
            + ticketData.Patronymic, font);
            Paragraph paragraph6 = new Paragraph("Дата продажи: " + ticketData.DateTimeBooking, font);
            Paragraph paragraph7 = new Paragraph("Цена: " +
            ticketData.TicketPrice.ToString().Remove(ticketData.TicketPrice.ToString().Length - 2, 2) + " Руб.", font);

            doc.Add(mainParagraph);
            doc.Add(paragraph);
            doc.Add(paragraph1);
            doc.Add(paragraph2);
            doc.Add(paragraph3);
            doc.Add(paragraph4);
            doc.Add(paragraph5);
            doc.Add(paragraph6);
            doc.Add(paragraph7);
            iTextSharp.text.Image image = iTextSharp.text.Image.GetInstance(qrCodeBytes);
            image.ScaleToFit(200, 200);
            doc.Add(image);
            doc.Close();
        }
    }
    Process.Start(tempFilePath);
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

```

Рисунок 16 – Продолжение метода PrintTicket

Полный код программы можно посмотреть в Приложении 1.

4. ТЕСТИРОВАНИЕ ПРОГРАММНОГО МОДУЛЯ

Для проведения тестирования программы мною было произведено базовое тестирование во время разработки программы. При тестировании был выявлен ряд ошибок, которые возникли в ходе выполнения программы.

- Попытка бронирования билета без выбора места в зале
Без выбора места в зале нажатие на кнопку “Забронировать”

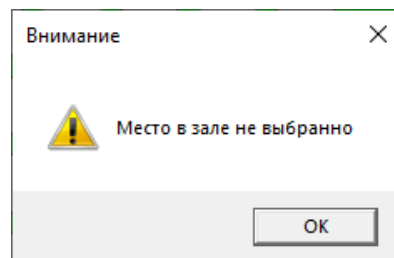


Рисунок 17 - Ошибка бронирования места

Ожидаемый результат: Сообщение о невозможности забронировать билет так как не выбрано место в зале.

Фактический результат: Сообщение о невозможности забронировать билет так как не выбрано место в зале. (Рисунок 17)

Решение проблемы: для вывода сообщения о невыбранном месте в зале реализована проверка значений выбранного ряда и места в зале.

- Попытка бронирования билета на занятое место
Попытка выбора места в зале, на который уже забронирован билет

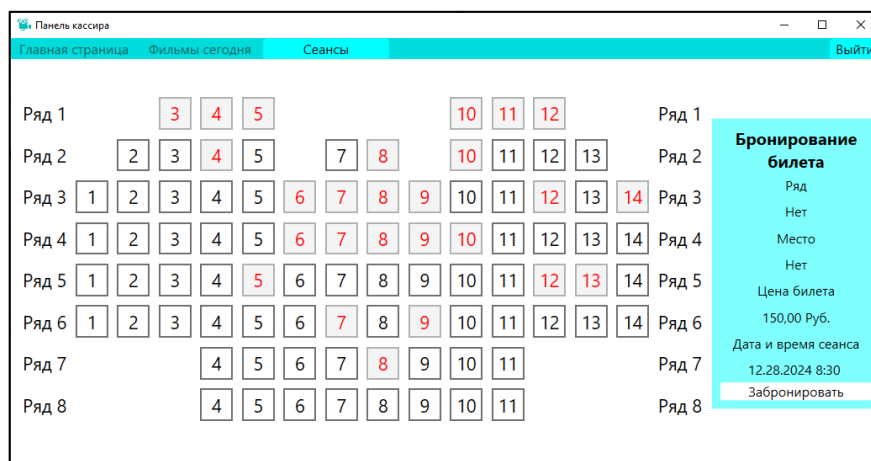


Рисунок 18 - Невозможность бронирования билета на занятое место

Ожидаемый результат: Невозможность выбора ряда и места в зале, на который уже куплен билет.

Фактический результат: Невозможность выбора ряда и места в зале, на который уже куплен билет. (Рисунок 18)

Решение проблемы: для решения данной проблемы места, на которые были забронированные билеты выделяются красным цветом, а также отключается взаимодействие с ними.

- Попытка забронировать билет повторно после бронирования

После бронирования билета на свободное место нажать на кнопку бронирования 2-й раз.

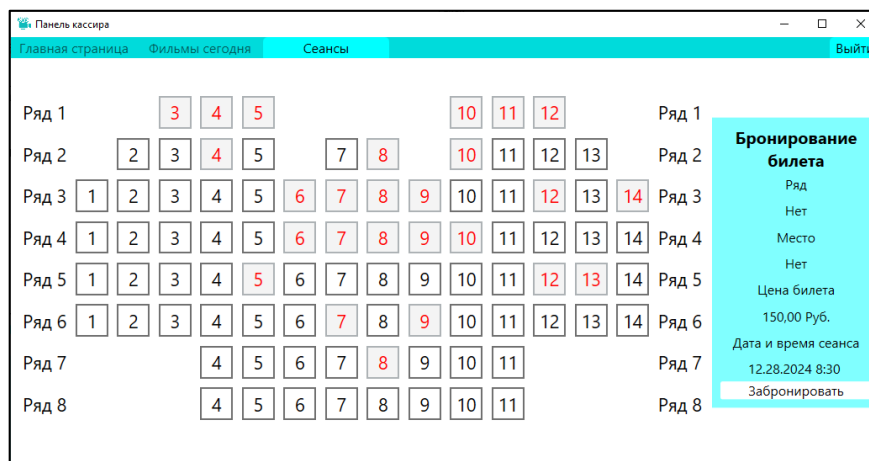


Рисунок 19 - Блокировка бронирования билета повторно

Ожидаемый результат: Значения ряда и места для бронирования на них билета сбрасываются, а в пользовательском интерфейсе место меняет цвет на красный, а также отключается взаимодействие с ним.

Фактический результат: Значения ряда и места для бронирования на них билета сбрасываются, а в пользовательском интерфейсе место меняет цвет на красный, а также отключается взаимодействие с ним. (Рисунок 19)

Решение проблемы: Сброс значений ряда и места и блокировка места в интерфейсе пользователя.

- Изменение размера интерфейса в зависимости от размера окна

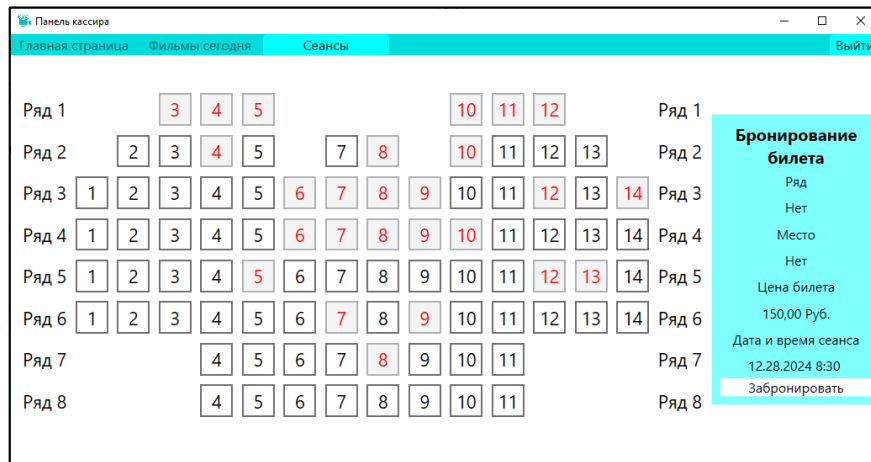


Рисунок 20 - Приложение при разрешении окна 800x1000

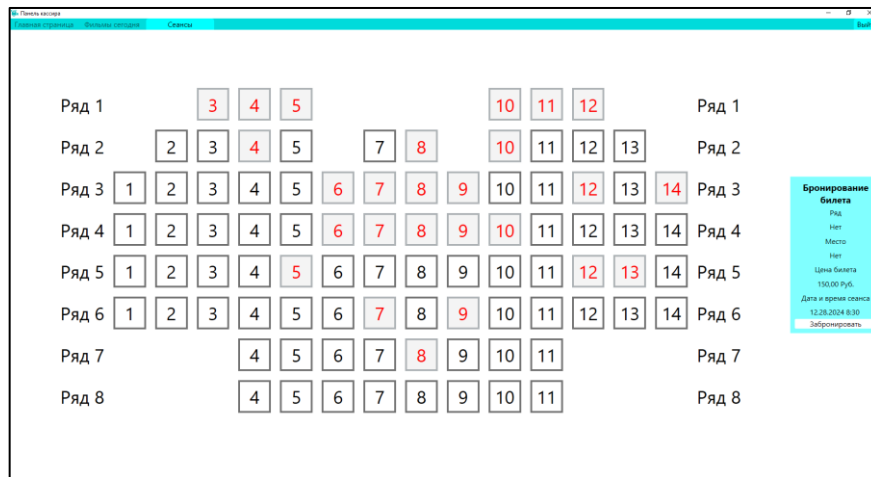


Рисунок 21 - Приложение при разрешении окна 1920x1080

Ожидаемый результат: Интерфейс автоматически изменяет размер по размер окна.

Фактический результат: Интерфейс автоматически изменяет размер по размер окна. (Рис. 20, 21)

Решение проблемы: Использование события `SizeChange` у окна для получения его размеров и исходя из них подгонки элементов при помощи `RenderTransform`.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсового проекта был создан программный модуль для автоматизации продажи билетов в кинотеатре. Данный программный модуль позволяет кинотеатрам сократить времени обслуживания клиентов, увеличить пропускную способность кинотеатра, повысить удобство посетителей и снизить затраты на персонал. Также программный модуль позволяет собирать и анализировать данные коммерческой успешности фильма для планирования дальнейшей работы.

Преимущества программного модуля по сравнению с аналогами состоит в простом и интуитивно понятном интерфейсе для любого сотрудника кинотеатра, а также в возможности сбора и анализа данных для определения коммерчески успешного фильма.

В перспективах развития программного модуля можно отметить расширение функциональности таких как интеграция с системами внешних платежей (различные банковские карты, электронные кошельки, мобильные платежи), персонализация интерфейса для каждого пользователя программного модуля, использование данного модуля для реализации онлайн бронирования билета на сайте кинотеатра, встраивание программного модуля в состав десктопного приложения для большего охвата рабочих задач кинотеатра и улучшение производительности.

Таким образом, разработанное приложение представляет собой эффективное решение для оптимизации работы кинотеатра и повышения уровня сервиса для клиентов.

СПИСОК ЛИТЕРАТУРЫ

1. ГОСТ 19.101-77. Единая система программной документации. Виды программ и программных документов, введ. 01.01.1978. – г. Москва: Изд-во стандартов, 1980. – 4 с.
2. ГОСТ 19.201-78. Единая система программной документации. Техническое задание. Требования к содержанию и оформлению, введ. 01.01.1980. – М.: Изд-во стандартов, 1988. – 3 с.
3. Алекс Дэвис Асинхронное программирование в C# 5.0. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2013. – 120 с.: ил.
4. Введение в ADO.NET, 2015 [Электронный ресурс] URL: <https://metanit.com/sharp/adonet/1.1.php?ysclid=m2xsepz8p6557122232> (дата обращения 15.10.2024)
5. Джон П. Смит Entity Framework Core в действии / пер. с англ. Д. А. Беликова. – М.: ДМК Пресс, 2022. – 690 с.: ил.
6. Обзор Entity Framework Core — EF Core | Microsoft Learn, 2023 [Электронный ресурс] URL: <https://learn.microsoft.com/ru-ru/ef/core/> (дата обращения: 25.09.2024)
7. Работа с LINQ – C# | Microsoft Learn, 2023 [Электронный ресурс] URL: <https://learn.microsoft.com/ru-ru/dotnet/csharp/tutorials/working-with-linq> (дата обращения: 2.10.2024)
8. Трунин В. Путь программиста T-SQL. Теория и практика – М.: Info-comp.ru, 2020 – 204 с.
9. ADO.NET | Microsoft Learn, 2023 [Электронный ресурс] URL: <https://learn.microsoft.com/ru-ru/dotnet/framework/data/adonet/> (дата обращения 10.10.2024)
10. C# и .NET | LINQ, 2022 [Электронный ресурс] URL: <https://metanit.com/sharp/tutorial/15.1.php?ysclid=m2xsgnx83x118354100> (дата обращения 3.10.2024)

11. C# и .NET | Асинхронные методы, async и await, 2022 [Электронный ресурс] URL: <https://metanit.com/sharp/tutorial/13.3.php?ysclid=m2xsjn72a4147828932> (дата обращения 16.11.2024)
12. Metanit C# | Запись таблиц в PDF, 2012 [Электронный ресурс] URL: <https://metanit.com/sharp/articles/25.php?ysclid=m2xqyr4e12641032739> (дата обращения: 25.09.2024)
13. MS SQL Server в Entity Framework Core и C#, провайдер Microsoft.EntityFrameworkCore.SqlServer, метод UseSqlServer, 2021 [Электронный ресурс] URL: <https://metanit.com/sharp/efcore/7.1.php?ysclid=m2xs59czu1292960590> (дата обращения 10.10.2024)
14. Windows Presentation Foundation - WPF .NET Framework | Microsoft Learn, 2023 [Электронный ресурс] URL: <https://learn.microsoft.com/ru-ru/dotnet/desktop/wpf/?view=netframeworkdesktop-4.8> (дата обращения 27.09.2024)
15. WPF и C# | Полное руководство, 2023 [Электронный ресурс] URL: <https://metanit.com/sharp/wpf> (дата обращения: 26.09.2024)

ПРИЛОЖЕНИЯ

КОД ПРОГРАММЫ

BackUpControls.xaml

```

<Page x:Class="Cinema.Pages.BackUpControls"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:local="clr-namespace:Cinema.Pages"
mc:Ignorable="d"
d:DesignHeight="450" d:DesignWidth="800"
Title="BackUpControls">

    <Grid Style="{DynamicResource MainColorStyle}">
        <Grid Margin="10">
            <Grid.ColumnDefinitions>
                <ColumnDefinition/>
                <ColumnDefinition/>
            </Grid.ColumnDefinitions>

            <StackPanel Grid.Column="0" HorizontalAlignment="Center">
                <TextBlock Margin="0,0,0,10" Text="Резервное
копирование" HorizontalAlignment="Center"
Style="{DynamicResource TitleTextBlockStyle}"/>
                <WrapPanel Margin="0,0,0,10">
                    <TextBox Margin="0,0,10,0" x:Name="BackUpPath"
Width="250" IsReadOnly="True" Style="{DynamicResource
MainTextBoxStyle}"/>
                    <Button x:Name="OpenSavePath" Content="Обзор"
Width="120" Style="{DynamicResource MainButtonStyle}"
Click="OpenSavePath_Click"/>
                </WrapPanel>
                <Button x:Name="SaveBackUp" Content="Сохранить"
Width="180" Style="{DynamicResource MainButtonStyle}"
Click="SaveBackUp_Click"/>
            </StackPanel>

            <StackPanel Grid.Column="1" HorizontalAlignment="Center">
                <TextBlock Margin="0,0,0,10" Text="Восстановление"
HorizontalAlignment="Center" Style="{DynamicResource
TitleTextBlockStyle}"/>
                <WrapPanel Margin="0,0,0,10">
                    <TextBox Margin="0,0,0,10" x:Name="RestorePath"
IsReadOnly="True" Width="250" Style="{DynamicResource
MainTextBoxStyle}"/>
                    <Button x:Name="OpenRestorePath" Content="Обзор"
Width="120" Style="{DynamicResource MainButtonStyle}"
Click="OpenRestorePath_Click"/>
                </WrapPanel>
            </StackPanel>
        </Grid>
    </Grid>

```

```

        <Button x:Name="RestoreDataBase"
Content="Восстановить" Width="180" Style="{DynamicResource
MainButtonStyle}" Click="RestoreDataBase_Click"/>
    </StackPanel>
</Grid>
</Grid>
</Page>

```

BackUpControls.xaml.cs

```

using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace Cinema.Pages
{
    /// <summary>
    /// Логика взаимодействия для BackUpControls.xaml
    /// </summary>
    public partial class BackUpControls : Page
    {
        public BackUpControls()
        {
            InitializeComponent();
        }

        public string saveBackUpPath;
        public string restorePath;

        private void OpenSavePath_Click(object sender, RoutedEventArgs
e)
        {
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.Filter = "BackUp (*.BAK)|*.BAK";

            if (saveFileDialog.ShowDialog() == true)
            {
                saveBackUpPath = saveFileDialog.FileName;
            }
        }
    }
}

```

```

        BackUpPath.Text = saveBackUpPath;
    }
}

private void SaveBackUp_Click(object sender, RoutedEventArgs e)
{
    if(saveBackUpPath != null)
    {
        using (var dataBase = new CinemaEntities())
        {
            var backUp =
dataBase.Database.SqlQuery<CinemaEntities>($"BACKUP
DATABASE [Cinema] TO DISK = '{saveBackUpPath}'").ToList();
            MessageBox.Show("Резервное копирование
выполнено", "Готово", MessageBoxButton.OK,
MessageBoxImage.Information);
        }
    }
}

private void OpenRestorePath_Click(object sender,
RoutedEventArgs e)
{
    OpenFileDialog openFileDialog = new OpenFileDialog();
    openFileDialog.Filter = "BackUp (*.BAK)|*.BAK";

    if (openFileDialog.ShowDialog() == true)
    {
        restorePath = openFileDialog.FileName;
        RestorePath.Text = restorePath;
    }
}

private void RestoreDataBase_Click(object sender,
RoutedEventArgs e)
{
    if (restorePath != null)
    {
        using (var dataBase = new CinemaEntities())
        {
            var backUp =
dataBase.Database.SqlQuery<CinemaEntities>($"USE master; ALTER
DATABASE [Cinema] SET SINGLE_USER WITH ROLLBACK
IMMEDIATE; RESTORE DATABASE [Cinema] FROM DISK =
'{restorePath}' WITH REPLACE; ALTER DATABASE [Cinema] SET
MULTI_USER;").ToList();
            MessageBox.Show("Восстановление выполнено",
"Готово", MessageBoxButton.OK, MessageBoxImage.Information);
        }
    }
}

```

```

    }

```

CurrentMovieCashierControls.xaml

```

<Page x:Class="Cinema.Pages.CurrentMovieCashierControls"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:local="clr-namespace:Cinema.Pages"
mc:Ignorable="d"
d:DesignHeight="450" d:DesignWidth="800"
Title="MovieCashierControls" SizeChanged="Page_SizeChanged"
Loaded="Page_Loaded">

    <Grid Style="{DynamicResource MainColorStyle}">
        <StackPanel>
            <Grid Style="{DynamicResource SecondColorStyle}">
                <WrapPanel Margin="10,0,10,0"
VerticalAlignment="Center">
                    <TextBlock Text="Поиск:" Margin="0,0,10,0"
Style="{DynamicResource MainTextBlockStyle}" />
                    <TextBox x:Name="FindData" Width="200"
Style="{DynamicResource MainTextBoxStyle}"
TextChanged="FindData_TextChanged" />
                    <Rectangle Margin="10,0,10,0" Style="{DynamicResource
SecondColorSeparatorsRectangleStyle}" />
                    <TextBlock Text="Найдено: " Style="{DynamicResource
MainTextBlockStyle}" />
                    <TextBlock x:Name="FindCounterData" Text="0/0"
Style="{DynamicResource MainTextBlockStyle}" />
                </WrapPanel>
            </Grid>

            <ListView x:Name="MovieCashierList" Height="400"
d:ItemsSource="{d:SampleData ItemCount=2}"
ScrollViewer.CanContentScroll="False"
MouseDown="MovieCashierList_MouseDoubleClick">
                <ListView.ItemContainerStyle>
                    <Style TargetType="ListViewItem">
                        <Setter Property="HorizontalContentAlignment"
Value="Stretch" />
                    </Style>
                </ListView.ItemContainerStyle>
                <ListView.View>
                    <GridView>
                        <GridViewColumn x:Name="MovieCodeGrid">
                            <Label Content="Код фильма"
Style="{DynamicResource MainLabelStyle}" />
                        </GridViewColumn>
                        <GridViewColumn.CellTemplate>
                            <DataTemplate>

```

```

        <Label Content="{Binding movieCashierID}"
HorizontalAlignment="Center" Style="{DynamicResource
MainLableStyle}"/>
        </DataTemplate>
    </GridViewColumn.CellTemplate>
</GridViewColumn>
    <GridViewColumn x:Name="ImageGrid">
        <Label Content="Постер" Style="{DynamicResource
MainLableStyle}"/>
        <GridViewColumn.CellTemplate>
            <DataTemplate>
                <Image Source="{Binding movieCashierCover}"
MaxHeight="400" HorizontalAlignment="Center"/>
            </DataTemplate>
        </GridViewColumn.CellTemplate>
    </GridViewColumn>
    <GridViewColumn x:Name="MovieInfoGrid">
        <Label Content="Информация о фильме"
Style="{DynamicResource MainLableStyle}"/>
        <GridViewColumn.CellTemplate>
            <DataTemplate>
                <StackPanel>
                    <Label Content="{Binding
movieCashierTitle}" Style="{DynamicResource MainLableStyle}"/>
                    <WrapPanel>
                        <Label Content="Жанр(ы):"
Style="{DynamicResource MainLableStyle}"/>
                        <Label Content="{Binding
movieCashierGenre}" Style="{DynamicResource MainLableStyle}"/>
                    </WrapPanel>
                    <WrapPanel>
                        <Label Content="Год публикации:"
Style="{DynamicResource MainLableStyle}"/>
                        <Label Content="{Binding
movieCashierYearOfPublication}" Style="{DynamicResource
MainLableStyle}"/>
                    </WrapPanel>
                    <WrapPanel>
                        <Label Content="Хронометраж:"
Style="{DynamicResource MainLableStyle}"/>
                        <Label Content="{Binding
movieCashierTiming}" Style="{DynamicResource MainLableStyle}"/>
                    </WrapPanel>
                    <WrapPanel>
                        <Label Content="Возрастной рейтинг:"
Style="{DynamicResource MainLableStyle}"/>
                        <Label Content="{Binding
movieCashierAgeRating}" Style="{DynamicResource
MainLableStyle}"/>
                    </WrapPanel>
                    <WrapPanel>
                        <Label Content="Страна:"
Style="{DynamicResource MainLableStyle}"/>

```

```

        <Label Content="{Binding
movieCashierCountry}" Style="{DynamicResource
MainLableStyle}"/>
    </WrapPanel>
</WrapPanel>
    <Label Content="Актёры:"
Style="{DynamicResource MainLableStyle}"/>
    <TextBlock Text="{Binding
movieCashierActors}" Margin="5,0,0,0" VerticalAlignment="Center"
TextWrapping="Wrap" Style="{DynamicResource
MainTextBlockStyle}"/>
    </WrapPanel>
</StackPanel>
</DataTemplate>
</GridViewColumn.CellTemplate>
</GridViewColumn>
    <GridViewColumn x:Name="DescriptionGrid">
        <Label Content="Описание"
Style="{DynamicResource MainLableStyle}"/>
        <GridViewColumn.CellTemplate>
            <DataTemplate>
                <TextBlock x:Name="DescriptionText"
Text="{Binding movieCashierDescription}" TextWrapping="Wrap"
Style="{DynamicResource MainTextBlockStyle}"/>
            </DataTemplate>
        </GridViewColumn.CellTemplate>
    </GridViewColumn>
</GridView>
</ListView.View>
</ListView>
</StackPanel>
</Grid>
</Page>

```

CurrentMovieCashierControls.xaml.cs

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using static Cinema.Authorization;

namespace Cinema.Pages

```

```

{
    /// <summary>
    /// Логика взаимодействия для CurrentMovieCashierControls.xaml
    /// </summary>
    public partial class CurrentMovieCashierControls : Page
    {
        public CurrentMovieCashierControls()
        {
            InitializeComponent();
        }

        public class MovieCashierData
        {
            public int movieCashierID { get; set; }
            public BitmapImage movieCashierCover { get; set; }
            public string movieCashierTitle { get; set; }
            public string movieCashierGenre { get; set; }
            public int movieCashierYearOfPublication { get; set; }
            public double movieCashierTiming { get; set; }
            public string movieCashierAgeRating { get; set; }
            public string movieCashierCountry { get; set; }
            public string movieCashierDescription { get; set; }
            public string movieCashierActors { get; set; }
        }

        List<MovieCashierData> moviesCashierDataList = new
        List<MovieCashierData>();

        private void Page_Loaded(object sender, RoutedEventArgs e)
        {
            try
            {
                MovieCashierList.ItemsSource = moviesCashierDataList;

                LoadData(null);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Ошибка",
                MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }

        private void LoadData(string findLine)
        {
            try
            {
                using (var dataBase = new CinemaEntities())
                {
                    moviesCashierDataList.Clear();

                    DateTime dateTimeToday = DateTime.Now;

                    DateTime dateTimeTomorrow =
                    DateTime.Now.AddDays(1).Date;

                    var movieData = (from movie in dataBase.Movie
                                    join
                                    movieGenre in dataBase.MovieGenre on
                                    movie.IDMovie equals movieGenre.IDMovie into movieGenreGroup
                                    from movieGenre in
                                    movieGenreGroup.DefaultIfEmpty()
                                    join
                                    genre in dataBase.Genre on
                                    movieGenre.IDGenre equals genre.IDGenre into genreGroup
                                    from genre in genreGroup.DefaultIfEmpty()
                                    join
                                    country in dataBase.Country on
                                    movie.IDCountry equals country.IDCountry into countryGroup
                                    from country in countryGroup.DefaultIfEmpty()
                                    join
                                    actorsInMovies in dataBase.ActorsInMovies on
                                    movie.IDMovie equals actorsInMovies.IDMovie into
                                    actorsInMoviesGroup
                                    from actorsInMovie in
                                    actorsInMoviesGroup.DefaultIfEmpty()
                                    join
                                    actors in dataBase.Actor on
                                    actorsInMovie.IDActor equals actors.IDActor into actorsGroup
                                    from actors in actorsGroup.DefaultIfEmpty()
                                    join
                                    session in dataBase.Session on movie.IDMovie
                                    equals session.IDMovie into sessionGroup
                                    from session in sessionGroup.DefaultIfEmpty()
                                    where ((findLine == null ||
                                    movie.Title.Contains(findLine) || genre.Title.Contains(findLine) ||
                                    movie.YearOfPublication.ToString().Contains(findLine) ||
                                    movie.Description.Contains(findLine) ||
                                    country.Title.Contains(findLine) || actors.Surname.Contains(findLine) ||
                                    actors.Name.Contains(findLine) || actors.Patronymic.Contains(findLine)
                                    || actors.Nickname.Contains(findLine)) &&
                                    (session.DateAndTimeSession >= dateTimeToday &&
                                    session.DateAndTimeSession <= dateTimeTomorrow))
                                    select new
                                    {
                                        movie.IDMovie,
                                        movie.Cover,
                                        movieTitle = movie.Title,
                                        movieGenre = genre.Title,
                                        movie.YearOfPublication,
                                        movie.Timing,
                                        movie.AgeRating,
                                        movie.Description,
                                        movieCountry = country.Title,
                                        movieActor = actors.Surname + " " +
                                        actors.Name + " " + actors.Patronymic + " " + actors.Nickname

```



```

    }
    ).ToList();

    var groupedMovieCashierData = movieData.GroupBy(m
=> m.IDMovie)
    .Select(g => new
    {
        g.Key,
        genres = g.Select(m => m.movieGenre).Distinct(),
        actors = g.Select(m => m.movieActor).Distinct()
    }).ToList();

    var result = groupedMovieCashierData.Select(g =>
    {
        var movie = movieData.FirstOrDefault(m => m.IDMovie
== g.Key);
        return new
        {
            movie.IDMovie,
            movie.Cover,
            movie.movieTitle,
            movieGenres = g.genres,
            movie.YearOfPublication,
            movie.Timing,
            movie.AgeRating,
            movie.Description,
            movie.movieCountry,
            movieActors = g.actors
        };
    }).ToList();

    foreach (var movieLine in result)
    {
        MovieCashierData movieDataClass = new
MovieCashierData();

        movieDataClass.movieCashierID = movieLine.IDMovie;

        if (movieLine.Cover != null)
        {
            BitmapImage cover = new BitmapImage();

            cover.BeginInit();
            cover.StreamSource = new
MemoryStream(movieLine.Cover);
            cover.EndInit();

            movieDataClass.movieCashierCover = cover;
        }
        else
        {
            movieDataClass.movieCashierCover = new
BitmapImage(new Uri("pack://application:,,,/Resource/NoImage.png",
UriKind.Absolute));
        }

        movieDataClass.movieCashierTitle =
movieLine.movieTitle;

        string movieGenreTemp = "";
        int movieGenresCounterTemp = 1;
        foreach (var genere in movieLine.movieGenres)
        {
            if (movieGenresCounterTemp !=
movieLine.movieGenres.Count())
                movieGenreTemp += genere + ", ";
            else
                movieGenreTemp += genere;

            movieGenresCounterTemp++;
        }
        movieDataClass.movieCashierGenre =
movieGenreTemp;

        movieDataClass.movieCashierYearOfPublication =
movieLine.YearOfPublication;
        movieDataClass.movieCashierTiming =
movieLine.Timing;
        movieDataClass.movieCashierAgeRating =
movieLine.AgeRating + "+";
        movieDataClass.movieCashierDescription =
movieLine.Description;
        movieDataClass.movieCashierCountry =
movieLine.movieCountry;

        string movieActorTemp = "";
        int movieActorCounterTemp = 1;
        foreach (var actor in movieLine.movieActors)
        {
            if (movieActorCounterTemp !=
movieLine.movieActors.Count())
                movieActorTemp += actor + ", ";
            else
                movieActorTemp += actor;

            movieActorCounterTemp++;
        }
        movieDataClass.movieCashierActors =
movieActorTemp;
    }
}

```

```

        moviesCashierDataList.Add(movieDataClass);
    }

    var fullMovieCashierData = (from movie in
        dataBase.Movie
                                join
                                session in dataBase.Session on
        movie.IDMovie equals session.IDMovie into sessionGroup
                                from session in
        sessionGroup.DefaultIfEmpty()
                                where (session.DateAndTimeSession >=
        dateTimeToday && session.DateAndTimeSession <=
        dateTimeTomorrow)
                                group movie by movie.Title into
        groupedMovies
                                select new
                                {
                                    MovieTitle = groupedMovies.Key,
                                    TotalSessions =
        groupedMovies.Count()
                                }).ToList();

    FindCounterData.Text = result.Count() + "/" +
    fullMovieCashierData.Count();

    MovieCashierList.Items.Refresh();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void Page_SizeChanged(object sender,
    SizeChangedEventArgs e)
{
    try
    {
        MovieCodeGrid.Width = 0;
        ImageGrid.Width = ActualWidth - ActualWidth * 0.8;
        MovieInfoGrid.Width = ActualWidth - ActualWidth * 0.7;
        DescriptionGrid.Width = ActualWidth - ActualWidth * 0.5;
        MovieCashierList.Height = ActualHeight - 35;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

```

```

private void FindData_TextChanged(object sender,
    TextChangedEventArgs e)
{
    LoadData(FindData.Text);
}

private void MovieCashierList_MouseDoubleClick(object sender,
    MouseButtonEventArgs e)
{
    try
    {
        var selectedCashierMovie = MovieCashierList.SelectedItem
        as MovieCashierData;
        if (selectedCashierMovie != null)
        {
            TransmittedData.idSelectedCashierMovie =
            selectedCashierMovie.movieCashierID;

            CashierPanel cashierPanel = Window.GetWindow(this) as
            CashierPanel;
            cashierPanel.SessionPageOpen();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
}

```

EmployeeControls.xaml

```

<Page x:Class="Cinema.EmployeeControls"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
    compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:Cinema"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="EmployeeControls" Loaded="Page_Loaded"
    SizeChanged="Page_SizeChanged">

    <Grid Style="{DynamicResource MainColorStyle}">
        <StackPanel>
            <Grid Style="{DynamicResource SecondColorStyle}">
                <WrapPanel Margin="10,0,0,0" VerticalAlignment="Center">
                    <Button x:Name="AddEmployee" Margin="0,0,5,0"
                        Content="Добавить" HorizontalAlignment="Left" Width="120"
                        Style="{DynamicResource TabControlButtonStyle}"
                        Click="AddEmployee_Click"/>
                </WrapPanel>
            </Grid>
        </StackPanel>
    </Grid>

```

```

        <Button x:Name="RemoveEmployee" Content="Удалить"
HorizontalAlignment="Left" Width="120" Style="{DynamicResource
TabControlButtonStyle}" Click="RemoveEmployee_Click"/>
        <Rectangle Margin="10,0,10,0" Style="{DynamicResource
SecondColorSeparatorsRectangleStyle}"/>
        <TextBlock Text="Поиск:" Margin="0,0,10,0"
Style="{DynamicResource MainTextBlockStyle}"/>
        <TextBox x:Name="FindData" Margin="0,0,10,0"
Width="200" Style="{DynamicResource MainTextBoxStyle}"
TextChanged="FindData_TextChanged"/>
        <Rectangle Margin="10,0,10,0" Style="{DynamicResource
SecondColorSeparatorsRectangleStyle}"/>
        <TextBlock Text="Найдено: " Style="{DynamicResource
MainTextBlockStyle}"/>
        <TextBlock x:Name="FindCounterData" Text="0/0"
Style="{DynamicResource MainTextBlockStyle}"/>
    </WrapPanel>
</Grid>

<ListView x:Name="EmployeeList" Height="400"
d:ItemsSource="{d:SampleData ItemCount=2}"
ScrollViewer.CanContentScroll="False"
MouseDoubleClick="EmployeeList_MouseDoubleClick">
    <ListView.ItemContainerStyle>
        <Style TargetType="ListViewItem">
            <Setter Property="HorizontalContentAlignment"
Value="Stretch" />
        </Style>
    </ListView.ItemContainerStyle>
    <ListView.View>
        <GridView>
            <GridViewColumn x:Name="EmployeeCodeGrid">
                <Label Content="Код сотрудника"
Style="{DynamicResource MainLableStyle}"/>
                <GridViewColumn.CellTemplate>
                    <DataTemplate>
                        <Label Content="{Binding employeeID}"
HorizontalAlignment="Center" Style="{DynamicResource
MainLableStyle}"/>
                    </DataTemplate>
                </GridViewColumn.CellTemplate>
            </GridViewColumn>
            <GridViewColumn x:Name="ImageGrid">
                <Label Content="Фотография"
Style="{DynamicResource MainLableStyle}"/>
                <GridViewColumn.CellTemplate>
                    <DataTemplate>
                        <Image Source="{Binding employeePhoto}"
MaxHeight="400" HorizontalAlignment="Center"/>
                    </DataTemplate>
                </GridViewColumn.CellTemplate>
            </GridViewColumn>
            <GridViewColumn x:Name="EmployeeInfoGrid">

```

```

                <Label Content="Информация о сотруднике"
Style="{DynamicResource MainLableStyle}"/>
                <GridViewColumn.CellTemplate>
                    <DataTemplate>
                        <StackPanel>
                            <WrapPanel>
                                <Label Content="{Binding
employeeSurname}" Style="{DynamicResource MainLableStyle}"/>
                                <Label Content="{Binding
employeeName}" Style="{DynamicResource MainLableStyle}"/>
                                <Label Content="{Binding
employeePatronymic}" Style="{DynamicResource MainLableStyle}"/>
                            </WrapPanel>
                            <WrapPanel>
                                <Label Content="Телефон:"
Style="{DynamicResource MainLableStyle}"/>
                                <Label Content="{Binding
employeePhone}" Style="{DynamicResource MainLableStyle}"/>
                            </WrapPanel>
                            <WrapPanel>
                                <Label Content="Почта:"
Style="{DynamicResource MainLableStyle}"/>
                                <Label Content="{Binding
employeeEmail}" Style="{DynamicResource MainLableStyle}"/>
                            </WrapPanel>
                            <WrapPanel>
                                <Label Content="Должность:"
Style="{DynamicResource MainLableStyle}"/>
                                <Label Content="{Binding employeeRole}"
Style="{DynamicResource MainLableStyle}"/>
                            </WrapPanel>
                        </StackPanel>
                    </DataTemplate>
                </GridViewColumn.CellTemplate>
            </GridViewColumn>
            <GridViewColumn x:Name="AuthorizationGrid">
                <Label Content="Данные авторизации"
Style="{DynamicResource MainLableStyle}"/>
                <GridViewColumn.CellTemplate>
                    <DataTemplate>
                        <StackPanel>
                            <WrapPanel>
                                <Label Content="Логин:"
Style="{DynamicResource MainLableStyle}"/>
                                <Label Content="{Binding
employeeLogin}" Style="{DynamicResource MainLableStyle}"/>
                            </WrapPanel>
                            <WrapPanel>
                                <Label Content="Пароль:"
Style="{DynamicResource MainLableStyle}"/>
                                <Label Content="{Binding
employeePassword}" Style="{DynamicResource MainLableStyle}"/>
                            </WrapPanel>
                        </StackPanel>
                    </DataTemplate>
                </GridViewColumn.CellTemplate>
            </GridViewColumn>
        </GridView>
    </ListView.View>
</ListView>

```

```

        </StackPanel>
    </DataTemplate>
</GridViewColumn.CellTemplate>
</GridViewColumn>
</GridView>
</ListView.View>
</ListView>
</StackPanel>
</Grid>
</Page>

```

EmployeeControls.xaml.cs

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media.Imaging;
using static Cinema.Authorization;

namespace Cinema
{
    /// <summary>
    /// Логика взаимодействия для EmployeeControls.xaml
    /// </summary>
    public partial class EmployeeControls : Page
    {
        public EmployeeControls()
        {
            InitializeComponent();
        }

        public class EmployeeData
        {
            public int employeeID { get; set; }
            public BitmapImage employeePhoto { get; set; }
            public string employeeSurname { get; set; }
            public string employeeName { get; set; }
            public string employeePatronymic { get; set; }
            public string employeePhone { get; set; }
            public string employeeEmail { get; set; }
            public string employeeLogin { get; set; }
            public string employeePassword { get; set; }
            public string employeeRole { get; set; }
        }

        List<EmployeeData> employeeDataList = new
        List<EmployeeData>();

        private void Page_Loaded(object sender, RoutedEventArgs e)
        {

```

```

            try
            {
                EmployeeList.ItemsSource = employeeDataList;

                LoadData(null);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Ошибка",
                MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }

        private void LoadData(string findLine)
        {
            try
            {
                using (var dataBase = new CinemaEntities())
                {
                    employeeDataList.Clear();

                    var employeeData = (from employee in dataBase.Employee
                                        join
                                        role in dataBase.Role on employee.IDRole
                                        equals role.IDRole into roleGroup
                                        from role in roleGroup.DefaultIfEmpty()
                                        where (findLine == null ||
                                        employee.Surname.Contains(findLine) ||
                                        employee.Name.Contains(findLine) ||
                                        employee.Patronymic.Contains(findLine) ||
                                        employee.Phone.Contains(findLine) ||
                                        employee.Email.Contains(findLine) ||
                                        employee.Login.Contains(findLine) ||
                                        employee.Password.Contains(findLine))
                                        select new
                                        {
                                            employee.IDEmployee,
                                            employee.Photo,
                                            employee.Surname,
                                            employee.Name,
                                            employee.Patronymic,
                                            employee.Phone,
                                            employee.Email,
                                            employee.Login,
                                            employee.Password,
                                            employeeRole = role.Title,
                                        }).ToList();

                    foreach (var employeeLine in employeeData)
                    {
                        EmployeeData employeeDataClass = new
                        EmployeeData();

```

```

        employeeDataClass.employeeID =
employeeLine.IDEmployee;

        if (employeeLine.Photo != null)
        {
            BitmapImage bitmapImage = new BitmapImage();
            bitmapImage.BeginInit();
            bitmapImage.StreamSource = new
MemoryStream(employeeLine.Photo);
            bitmapImage.EndInit();

            employeeDataClass.employeePhoto = bitmapImage;
        }
        else
        {
            employeeDataClass.employeePhoto = new
BitmapImage(new Uri("pack://application:,,,/Resource/NoImage.png",
UriKind.Absolute));
        }

        employeeDataClass.employeeSurname =
employeeLine.Surname;
        employeeDataClass.employeeName =
employeeLine.Name;
        employeeDataClass.employeePatronymic =
employeeLine.Patronymic;
        employeeDataClass.employeePhone =
employeeLine.Phone;
        employeeDataClass.employeeEmail =
employeeLine.Email;
        employeeDataClass.employeeLogin =
employeeLine.Login;
        employeeDataClass.employeePassword =
employeeLine.Password;
        employeeDataClass.employeeRole =
employeeLine.employeeRole;

        employeeDataList.Add(employeeDataClass);
    }

    var fullEmployeeData = dataBase.Employee.ToList();
    FindCounterData.Text = employeeData.Count() + "/" +
fullEmployeeData.Count();

    EmployeeList.Items.Refresh();
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}
}

private void Page_SizeChanged(object sender,
SizeChangedEventArgs e)
{
    try
    {
        EmployeeCodeGrid.Width = 0;
        ImageGrid.Width = ActualWidth - ActualWidth * 0.8;
        EmployeeInfoGrid.Width = ActualWidth - ActualWidth * 0.6;
        AuthorizationGrid.Width = ActualWidth - ActualWidth * 0.6;
        EmployeeList.Height = ActualHeight - 35;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void FindData_TextChanged(object sender,
TextChangedEventArgs e)
{
    LoadData(FindData.Text);
}

private void EmployeeList_MouseDoubleClick(object sender,
MouseButtonEventArgs e)
{
    try
    {
        var selectedEmployee = EmployeeList.SelectedItem as
EmployeeData;
        if (selectedEmployee != null)
        {
            TransmittedData.idSelectedEmployee =
selectedEmployee.employeeID;

            AddOrEditEmployee addOrEditEmployee = new
AddOrEditEmployee();
            addOrEditEmployee.ShowDialog();

            LoadData(FindData.Text);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void AddEmployee_Click(object sender, RoutedEventArgs
e)

```

```

{
    try
    {
        TransmittedData.idSelectedEmployee = -1;

        AddOrEditEmployee addOrEditEmployee = new
AddOrEditEmployee();
        addOrEditEmployee.ShowDialog();

        LoadData(FindData.Text);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void RemoveEmployee_Click(object sender,
RoutedEventArgs e)
{
    try
    {
        var selectedEmployee = EmployeeList.SelectedItem as
EmployeeData;

        if (selectedEmployee.employeeID ==
TransmittedData.idEmployee)
        {
            MessageBox.Show("Вы не можете удалить свою же
учетную запись", "Внимание", MessageBoxButton.OK,
MessageBoxImage.Warning);
            return;
        }

        if (selectedEmployee.employeeRole == "Администратор")
        {
            MessageBox.Show("Вы не можете удалить учетную
запись администратора", "Внимание", MessageBoxButton.OK,
MessageBoxImage.Warning);
            return;
        }

        if (selectedEmployee != null)
        {
            if (MessageBox.Show("Вы действительно хотите удалить
данного сотрудника?", "Внимание", MessageBoxButton.YesNo,
MessageBoxImage.Warning) == MessageBoxResult.Yes)
            {
                using (var dataBase = new CinemaEntities())
                {
                    var removeEmployee = dataBase.Employee.Where(w
=> w.IDEmployee == selectedEmployee.employeeID).FirstOrDefault();

```

```

                    var removeTickets = dataBase.Ticket.Where(w =>
w.IDEmployee == selectedEmployee.employeeID).ToList();

                    if (removeTickets.Count > 0)
                    {
                        if (MessageBox.Show("Удаляемый сотрудник
имеет проданные билеты. Вы действительно хотите его удалить?",
"Внимание", MessageBoxButton.YesNo, MessageBoxImage.Warning)
== MessageBoxResult.No)
                            return;
                    }

                    dataBase.Ticket.RemoveRange(removeTickets);
                    dataBase.Employee.Remove(removeEmployee);

                    dataBase.SaveChanges();
                    MessageBox.Show("Данные были удалены",
"Готово", MessageBoxButton.OK, MessageBoxImage.Information);
                }

                LoadData(FindData.Text);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }
}

```

MovieAnalysis.xaml

```

<Page x:Class="Cinema.MovieAnalysis"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:local="clr-namespace:Cinema"
mc:Ignorable="d"
d:DesignHeight="450" d:DesignWidth="800"
Title="TicketAnalysis" SizeChanged="Page_SizeChanged"
Loaded="Page_Loaded">

    <Grid Style="{DynamicResource MainColorStyle}">
        <StackPanel>
            <Grid Style="{DynamicResource SecondColorStyle}">
                <WrapPanel Margin="10,0,0,0" VerticalAlignment="Center">
                    <Button x:Name="GoBackPage" Content="Назад"
Width="80" Style="{DynamicResource TabControlButtonStyle}"
Click="GoBackPage_Click"/>

```

```

        <Rectangle Margin="10,0,10,0" Fill="Black"
Style="{DynamicResource SecondColorSeparatorsRectangleStyle}"/>
        <CheckBox x:Name="FindDateChecker" Margin="0,4,5,0"
Style="{DynamicResource MainCheckBoxStyle}"
Click="FindDateChecker_Click"/>
        <TextBlock Text="C:" Margin="0,0,10,0"
Style="{DynamicResource MainTextBlockStyle}"/>
        <DatePicker x:Name="BeginDate" Margin="0,0,10,0"
Style="{DynamicResource SecondDatePickerStyle}"
SelectedDateChanged="FindDate_SelectedDateChanged"/>
        <TextBlock Text="по" Margin="0,0,10,0"
Style="{DynamicResource MainTextBlockStyle}"/>
        <DatePicker x:Name="EndDate"
Style="{DynamicResource SecondDatePickerStyle}"
SelectedDateChanged="FindDate_SelectedDateChanged"/>
        <Rectangle Margin="10,0,10,0" Style="{DynamicResource
SecondColorSeparatorsRectangleStyle}"/>
        <TextBlock Text="Найдено: " Style="{DynamicResource
MainTextBlockStyle}"/>
        <TextBlock x:Name="FindCounterData" Text="0/0"
Style="{DynamicResource MainTextBlockStyle}"/>
        </WrapPanel>
        <WrapPanel Margin="0,0,10,0"
HorizontalAlignment="Right" VerticalAlignment="Center">
            <Button x:Name="Print" Content="Печать отчета"
Width="120" Style="{DynamicResource TabControlButtonStyle}"
Click="Print_Click"/>
        </WrapPanel>
    </Grid>

    <ListView x:Name="SessionList" Height="400"
d:ItemsSource="{d:SampleData ItemCount=2}"
ScrollViewer.CanContentScroll="False">
        <ListView.ItemContainerStyle>
            <Style TargetType="ListViewItem">
                <Setter Property="HorizontalContentAlignment"
Value="Stretch" />
            </Style>
        </ListView.ItemContainerStyle>
        <ListView.View>
            <GridView>
                <GridViewColumn x:Name="SessionCodeGrid">
                    <Label Content="Код сессии"
Style="{DynamicResource MainLableStyle}"/>
                    <GridViewColumn.CellTemplate>
                        <DataTemplate>
                            <Label Content="{Binding idSession}"
HorizontalAlignment="Center" Style="{DynamicResource
MainLableStyle}"/>
                        </DataTemplate>
                    </GridViewColumn.CellTemplate>
                </GridViewColumn>
                <GridViewColumn x:Name="SessionInformationGrid">

```

```

                    <Label Content="Информация"
Style="{DynamicResource MainLableStyle}"/>
                    <GridViewColumn.CellTemplate>
                        <DataTemplate>
                            <StackPanel>
                                <WrapPanel>
                                    <Label Content="Дата сеанса:"
Style="{DynamicResource MainLableStyle}"/>
                                    <Label Content="{Binding dateSession}"
Style="{DynamicResource MainLableStyle}"/>
                                </WrapPanel>
                                <WrapPanel>
                                    <Label Content="Время сеанса:"
Style="{DynamicResource MainLableStyle}"/>
                                    <Label Content="{Binding timeSession}"
Style="{DynamicResource MainLableStyle}"/>
                                </WrapPanel>
                            </StackPanel>
                        </DataTemplate>
                    </GridViewColumn.CellTemplate>
                </GridViewColumn>
            <GridViewColumn x:Name="AnalyticsGrid">
                <Label Content="Аналитика продаж"
Style="{DynamicResource MainLableStyle}"/>
                <GridViewColumn.CellTemplate>
                    <DataTemplate>
                        <StackPanel>
                            <WrapPanel>
                                <Label Content="Количество проданных
билетов:" Style="{DynamicResource MainLableStyle}"/>
                                <Label Content="{Binding
ticketCountSession}" Style="{DynamicResource MainLableStyle}"/>
                            </WrapPanel>
                            <WrapPanel>
                                <Label Content="Цена билета:"
Style="{DynamicResource MainLableStyle}"/>
                                <Label Content="{Binding ticketPrice}"
Style="{DynamicResource MainLableStyle}"/>
                            </WrapPanel>
                            <WrapPanel>
                                <Label Content="Суммарные сборы:"
Style="{DynamicResource MainLableStyle}"/>
                                <Label Content="{Binding
ticketSummarySession}" Style="{DynamicResource
MainLableStyle}"/>
                            </WrapPanel>
                        </StackPanel>
                    </DataTemplate>
                </GridViewColumn.CellTemplate>
            </GridViewColumn>
        </GridView>
    </ListView.View>
</ListView>

```

```

        </StackPanel>
    </Grid>
</Page>

```

MovieAnalysis.xaml.cs

```

using iTextSharp.text;
using iTextSharp.text.pdf;
using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using static Cinema.Authorization;

namespace Cinema
{
    /// <summary>
    /// Логика взаимодействия для MovieAnalysis.xaml
    /// </summary>
    public partial class MovieAnalysis : Page
    {
        public MovieAnalysis()
        {
            InitializeComponent();

            public class SessionDataAnalysis
            {
                public int idSession { get; set; }
                public string dateSession { get; set; }
                public string timeSession { get; set; }
                public string ticketCountSession { get; set; }
                public string ticketPrice { get; set; }
                public string ticketSummarySession { get; set; }
            }

            List<SessionDataAnalysis> sessionDataAnalysesList = new
            List<SessionDataAnalysis>();

            private void Page_Loaded(object sender, RoutedEventArgs e)
            {
                try
                {
                    SessionList.ItemsSource = sessionDataAnalysesList;

                    LoadData(BeginDate, EndDate);
                }
                catch (Exception ex)
                {

```

```

                    MessageBox.Show(ex.Message, "Ошибка",
                    MessageBoxButton.OK, MessageBoxImage.Error);
                }
            }

            private void LoadData(DatePicker beginDate, DatePicker endDate)
            {
                try
                {
                    using (var dataBase = new CinemaEntities())
                    {
                        sessionDataAnalysesList.Clear();

                        var sessionData = (from session in dataBase.Session
                                        join
                                        ticket in dataBase.Ticket on session.IDSession
                                        equals ticket.IDSession into ticketGroup
                                        from ticket in ticketGroup.DefaultIfEmpty()
                                        where ((session.IDMovie ==
                                        TransmittedData.idSelectedMovieAnalysis) &&
                                        ((beginDate.SelectedDate == null || session.DateAndTimeSession >=
                                        beginDate.SelectedDate.Value) && (endDate.SelectedDate == null ||
                                        session.DateAndTimeSession <= endDate.SelectedDate.Value)))
                                        select new
                                        {
                                            session.IDSession,
                                            session.DateAndTimeSession,
                                            TicketID = ticket == null ? 0 :
                                            ticket.IDTicket,
                                            session.TicketPrice,
                                        }).ToList();

                        var groupedSessionData = sessionData.GroupBy(m =>
                        m.IDSession)
                        .Select(g => new
                        {
                            g.Key,
                            TicketCount = g.Count(),
                            TotalCost = g.Sum(t => t.TicketPrice)
                        }).ToList();

                        var result = groupedSessionData.Select(g =>
                        {
                            var session = sessionData.FirstOrDefault(m =>
                            m.IDSession == g.Key);
                            return new
                            {
                                session.IDSession,
                                session.DateAndTimeSession,
                                session.TicketPrice,
                                g.TicketCount,
                                g.TotalCost
                            };

```



```

    }).ToList();

    foreach (var sessionLine in result)
    {
        SessionDataAnalysis sessionDataAnalysisClass = new
SessionDataAnalysis();

        sessionDataAnalysisClass.idSession =
sessionLine.IDSession;

        sessionDataAnalysisClass.dateSession =
sessionLine.DateAndTimeSession.ToString().Split(' ')[0];

        sessionDataAnalysisClass.timeSession =
sessionLine.DateAndTimeSession.ToString().Split(' ')[1];

        sessionDataAnalysisClass.ticketPrice =
sessionLine.TicketPrice.ToString().Remove(sessionLine.TicketPrice.To
String().Length - 2, 2) + " py6."; ;

        if (sessionLine.TicketCount == 1)
        {
            var ticketCount = dataBase.Ticket.Where(w =>
w.IDSession == sessionLine.IDSession).Count();

            sessionDataAnalysisClass.ticketCountSession =
ticketCount.ToString();

            sessionDataAnalysisClass.ticketSummarySession =
(sessionLine.TicketPrice *
ticketCount).ToString().Remove((sessionLine.TicketPrice *
ticketCount).ToString().Length - 2, 2) + " py6.";
        }
        else
        {
            sessionDataAnalysisClass.ticketCountSession =
sessionLine.TicketCount.ToString();

            sessionDataAnalysisClass.ticketSummarySession =
sessionLine.TotalCost.ToString().Remove(sessionLine.TotalCost.ToStri
ng().Length - 2, 2) + " py6.";
        }

        sessionDataAnalysesList.Add(sessionDataAnalysisClass);
    }

    var fullSessionData = dataBase.Session.Where(w =>
w.IDMovie == TransmittedData.idSelectedMovieAnalysis).ToList();

    FindCounterData.Text = result.Count() + "/" +
fullSessionData.Count();

    SessionList.Items.Refresh();
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}
}

private void Page_SizeChanged(object sender,
SizeChangedEventArgs e)
{
    try
    {
        SessionCodeGrid.Width = 0;
        SessionInformationGrid.Width = ActualWidth - ActualWidth
* 0.5;
        AnalyticsGrid.Width = ActualWidth - ActualWidth * 0.5;
        SessionList.Height = ActualHeight - 35;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void GoBackPage_Click(object sender, RoutedEventArgs
e)
{
    try
    {
        Window activeWindow = Window.GetWindow(this);

        if (activeWindow.Title == "Панель директора")
        {
            DirectorsPanel directorsPanel = Window.GetWindow(this)
as DirectorsPanel;

            directorsPanel.TicketAnalysisOpen();
        }
        else
        if (activeWindow.Title == "Панель менеджера")
        {
            BookerPanel bookerPanel = Window.GetWindow(this) as
BookerPanel;

            bookerPanel.TicketAnalysisOpen();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void FindDateChecker_Click(object sender,
RoutedEventArgs e)
{
    try

```

```

{
    if (FindDateChecker.IsChecked == false)
    {
        BeginDate.IsEnabled = false;
        BeginDate.SelectedDate = null;
        EndDate.IsEnabled = false;
        EndDate.SelectedDate = null;
    }
    else
    {
        BeginDate.IsEnabled = true;
        BeginDate.SelectedDate = DateTime.Now;
        EndDate.IsEnabled = true;
        EndDate.SelectedDate = DateTime.Now;
    }

    LoadData(BeginDate, EndDate);
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
    MessageBoxButton.OK, MessageBoxImage.Error);
}

private void FindDate_SelectedDateChanged(object sender,
SelectionChangedEventArgs e)
{
    LoadData(BeginDate, EndDate);
}

private void Print_Click(object sender, RoutedEventArgs e)
{
    try
    {
        using (var dataBase = new CinemaEntities())
        {
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.Filter = "PDF files (*.pdf)|*.pdf";

            var sessionData = (from session in dataBase.Session
                                join
                                movie in dataBase.Movie on session.IDMovie
                                equals movie.IDMovie into movieGroup
                                from movie in movieGroup.DefaultIfEmpty()
                                where ((session.IDMovie ==
TransmittedData.idSelectedMovieAnalysis) &&
((BeginDate.SelectedDate == null || session.DateAndTimeSession >=
BeginDate.SelectedDate.Value) && (EndDate.SelectedDate == null ||
session.DateAndTimeSession <= EndDate.SelectedDate.Value)))
                                select new
                                {
                                    session.IDSession,
                                    movie.Title,
                                    movie.YearOfPublication,
                                    session.DateAndTimeSession,
                                }).FirstOrDefault();

            if (saveFileDialog.ShowDialog() == true)
            {
                Document doc = new Document();
                PdfWriter writer = PdfWriter.GetInstance(doc, new
                FileStream($"{saveFileDialog.FileName}", FileMode.Create));
                doc.SetPageSize(PageSize.A4.Rotate());

                BaseFont baseFont =
                BaseFont.CreateFont("C:\\Windows\\Fonts\\arial.ttf",
                BaseFont.IDENTITY_H, BaseFont.NOT_EMBEDDED);
                Font font = new Font(baseFont, 12);

                BaseFont baseFontHead =
                BaseFont.CreateFont("C:\\Windows\\Fonts\\arial.ttf",
                BaseFont.IDENTITY_H, BaseFont.NOT_EMBEDDED);
                Font fontHead = new Font(baseFont, 20, Font.BOLD);

                doc.Open();

                Paragraph mainParagraph = new Paragraph($"Отчет
продаж по фильму {sessionData.Title + " " +
sessionData.YearOfPublication}", fontHead);
                mainParagraph.Alignment = Element.ALIGN_CENTER;
                doc.Add(mainParagraph);

                PdfPTable table = new PdfPTable(4);
                table.SpacingBefore = 10;
                table.WidthPercentage = 100;

                table.AddCell(new Paragraph("Код сеанса", font));
                table.AddCell(new Paragraph("Дата и время
проведения сеанса", font));
                table.AddCell(new Paragraph("Количество проданных
билетов", font));
                table.AddCell(new Paragraph("Выручка с проданных
билетов", font));

                foreach (var line in sessionDataAnalysesList)
                {
                    table.AddCell(new
                    Paragraph(line.idSession.ToString(), font));
                    table.AddCell(new Paragraph(line.dateSession + " " +
                    line.timeSession, font));
                    table.AddCell(new Paragraph(line.ticketCountSession,
                    font));
                    table.AddCell(new
                    Paragraph(line.ticketSummarySession.ToString(), font));
                }
            }
        }
    }
}

```

```

    }
    doc.Add(table);

    double ticketPriceSumm = 0;
    foreach (var line in sessionDataAnalysesList)
    {
        ticketPriceSumm +=
Convert.ToDouble(line.ticketSummarySession.Replace(" руб.", ""));
    }

    Paragraph paragraph = new Paragraph("Итого: " +
ticketPriceSumm.ToString() + " Руб.", font);
    doc.Add(paragraph);

    doc.Close();
    MessageBox.Show("Файл сохранен", "Готово",
MessageBoxButton.OK, MessageBoxImage.Information);

    Process.Start(saveFileDialog.FileName);
    }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
    }
}

```

MovieCashierControls.xaml

```

<Page x:Class="Cinema.MovieCashierControls"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:Cinema"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="MovieCashierControls" SizeChanged="Page_SizeChanged"
    Loaded="Page_Loaded">

    <Grid Style="{DynamicResource MainColorStyle}">
        <StackPanel>
            <Grid Style="{DynamicResource SecondColorStyle}">
                <WrapPanel Margin="10,0,10,0"
VerticalAlignment="Center">
                    <TextBlock Text="Поиск:" Margin="0,0,10,0"
Style="{DynamicResource MainTextBlockStyle}" />

```

```

                    <TextBox x:Name="FindData" Width="200"
Style="{DynamicResource MainTextBoxStyle}"
TextChanged="FindData_TextChanged" />
                    <Rectangle Margin="10,0,10,0" Style="{DynamicResource
SecondColorSeparatorsRectangleStyle}" />
                    <TextBlock Text="Найдено: " Style="{DynamicResource
MainTextBlockStyle}" />
                    <TextBlock x:Name="FindCounterData" Text="0/0"
Style="{DynamicResource MainTextBlockStyle}" />
                </WrapPanel>
            </Grid>

            <ListView x:Name="MovieCashierList" Height="400"
d:ItemsSource="{d:SampleData ItemCount=2}"
ScrollViewer.CanContentScroll="False"
MouseDoubleClick="MovieCashierList_MouseDoubleClick">
                <ListView.ItemContainerStyle>
                    <Style TargetType="ListViewItem">
                        <Setter Property="HorizontalContentAlignment"
Value="Stretch" />
                    </Style>
                </ListView.ItemContainerStyle>
                <ListView.View>
                    <GridView>
                        <GridViewColumn x:Name="MovieCodeGrid">
                            <Label Content="Код фильма"
Style="{DynamicResource MainLableStyle}" />
                            <GridViewColumn.CellTemplate>
                                <DataTemplate>
                                    <Label Content="{Binding movieCashierID}"
HorizontalAlignment="Center" Style="{DynamicResource
MainLableStyle}" />
                                </DataTemplate>
                            </GridViewColumn.CellTemplate>
                        </GridViewColumn>
                        <GridViewColumn x:Name="ImageGrid">
                            <Label Content="Постер" Style="{DynamicResource
MainLableStyle}" />
                            <GridViewColumn.CellTemplate>
                                <DataTemplate>
                                    <Image Source="{Binding movieCashierCover}"
MaxHeight="400" HorizontalAlignment="Center" />
                                </DataTemplate>
                            </GridViewColumn.CellTemplate>
                        </GridViewColumn>
                        <GridViewColumn x:Name="MovieInfoGrid">
                            <Label Content="Информация о фильме"
Style="{DynamicResource MainLableStyle}" />
                            <GridViewColumn.CellTemplate>
                                <DataTemplate>
                                    <StackPanel>
                                        <Label Content="{Binding
movieCashierTitle}" Style="{DynamicResource MainLableStyle}" />

```

```

        <WrapPanel>
            <Label Content="Жанр(ы):"
Style="{DynamicResource MainLableStyle}" />
            <Label Content="{Binding
movieCashierGenre}" Style="{DynamicResource MainLableStyle}" />
        </WrapPanel>
        <WrapPanel>
            <Label Content="Год публикации:"
Style="{DynamicResource MainLableStyle}" />
            <Label Content="{Binding
movieCashierYearOfPublication}" Style="{DynamicResource
MainLableStyle}" />
        </WrapPanel>
        <WrapPanel>
            <Label Content="Хронометраж:"
Style="{DynamicResource MainLableStyle}" />
            <Label Content="{Binding
movieCashierTiming}" Style="{DynamicResource MainLableStyle}" />
        </WrapPanel>
        <WrapPanel>
            <Label Content="Возрастной рейтинг:"
Style="{DynamicResource MainLableStyle}" />
            <Label Content="{Binding
movieCashierAgeRating}" Style="{DynamicResource
MainLableStyle}" />
        </WrapPanel>
        <WrapPanel>
            <Label Content="Страна:"
Style="{DynamicResource MainLableStyle}" />
            <Label Content="{Binding
movieCashierCountry}" Style="{DynamicResource
MainLableStyle}" />
        </WrapPanel>
        <WrapPanel>
            <Label Content="Актеры:"
Style="{DynamicResource MainLableStyle}" />
            <TextBlock Text="{Binding
movieCashierActors}" Margin="5,0,0,0" VerticalAlignment="Center"
TextWrapping="Wrap" Style="{DynamicResource
MainTextBlockStyle}" />
        </WrapPanel>
    </StackPanel>
</DataTemplate>
</GridViewColumn.CellTemplate>
</GridViewColumn>
<GridViewColumn x:Name="DescriptionGrid">
    <Label Content="Описание"
Style="{DynamicResource MainLableStyle}" />
    <GridViewColumn.CellTemplate>
        <DataTemplate>
            <TextBlock x:Name="DescriptionText"
Text="{Binding movieCashierDescription}" TextWrapping="Wrap"
Style="{DynamicResource MainTextBlockStyle}" />

```

```

        </DataTemplate>
    </GridViewColumn.CellTemplate>
</GridViewColumn>
</GridView>
</ListView.View>
</ListView>
</StackPanel>
</Grid>
</Page>

```

MovieCashierControls.xaml.cs

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media.Imaging;
using static Cinema.Authorization;

namespace Cinema
{
    /// <summary>
    /// Логика взаимодействия для MovieCashierControls.xaml
    /// </summary>
    public partial class MovieCashierControls : Page
    {
        public MovieCashierControls()
        {
            InitializeComponent();
        }

        public class MovieCashierData
        {
            public int movieCashierID { get; set; }
            public BitmapImage movieCashierCover { get; set; }
            public string movieCashierTitle { get; set; }
            public string movieCashierGenre { get; set; }
            public int movieCashierYearOfPublication { get; set; }
            public double movieCashierTiming { get; set; }
            public string movieCashierAgeRating { get; set; }
            public string movieCashierCountry { get; set; }
            public string movieCashierDescription { get; set; }
            public string movieCashierActors { get; set; }
        }

        List<MovieCashierData> moviesCashierDataList = new
List<MovieCashierData>();

        private void Page_Loaded(object sender, RoutedEventArgs e)
        {
            try

```

```

{
    MovieCashierList.ItemsSource = moviesCashierDataList;

    LoadData(null);
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
}

private void LoadData(string findLine)
{
    try
    {
        using (var dataBase = new CinemaEntities())
        {
            moviesCashierDataList.Clear();

            var movieData = (from movie in dataBase.Movie
                            join
                                movieGenre in dataBase.MovieGenre on
                                movie.IDMovie equals movieGenre.IDMovie into movieGenreGroup
                                from movieGenre in
                                    movieGenreGroup.DefaultIfEmpty()
                                join
                                    genre in dataBase.Genre on
                                    movieGenre.IDGenre equals genre.IDGenre into genreGroup
                                    from genre in genreGroup.DefaultIfEmpty()
                                join
                                    country in dataBase.Country on
                                    movie.IDCountry equals country.IDCountry into countryGroup
                                    from country in countryGroup.DefaultIfEmpty()
                                join
                                    actorsInMovies in dataBase.actorsInMovies on
                                    movie.IDMovie equals actorsInMovies.IDMovie into
                                    actorsInMoviesGroup
                                    from actorsInMovie in
                                        actorsInMoviesGroup.DefaultIfEmpty()
                                join
                                    actors in dataBase.Actor on
                                    actorsInMovie.IDActor equals actors.IDActor into actorsGroup
                                    from actors in actorsGroup.DefaultIfEmpty()
                                join
                                    session in dataBase.Session on movie.IDMovie
                                    equals session.IDMovie into sessionGroup
                                    from session in sessionGroup.DefaultIfEmpty()
                                    where ((findLine == null ||
                                        movie.Title.Contains(findLine) || genre.Title.Contains(findLine) ||
                                        movie.YearOfPublication.ToString().Contains(findLine) ||
                                        movie.Description.Contains(findLine) ||
                                        country.Title.Contains(findLine) || actors.Surname.Contains(findLine) ||
                                        actors.Name.Contains(findLine) || actors.Patronymic.Contains(findLine)
                                        || actors.Nickname.Contains(findLine)) &&
                                        (session.DateAndTimeSession >= DateTime.Now))
                                select new
                                {
                                    movie.IDMovie,
                                    movie.Cover,
                                    movieTitle = movie.Title,
                                    movieGenre = genre.Title,
                                    movie.YearOfPublication,
                                    movie.Timing,
                                    movie.AgeRating,
                                    movie.Description,
                                    movieCountry = country.Title,
                                    movieActor = actors.Surname + " " +
                                        actors.Name + " " + actors.Patronymic + " " + actors.Nickname
                                }
                                ).ToList();

            var groupedMovieCashierData = movieData.GroupBy(m
=> m.IDMovie)
                .Select(g => new
                {
                    g.Key,
                    genres = g.Select(m => m.movieGenre).Distinct(),
                    actors = g.Select(m => m.movieActor).Distinct()
                }).ToList();

            var result = groupedMovieCashierData.Select(g =>
            {
                var movie = movieData.FirstOrDefault(m => m.IDMovie
                    == g.Key);

                return new
                {
                    movie.IDMovie,
                    movie.Cover,
                    movie.movieTitle,
                    movieGenres = g.genres,
                    movie.YearOfPublication,
                    movie.Timing,
                    movie.AgeRating,
                    movie.Description,
                    movie.movieCountry,
                    movieActors = g.actors
                };
            }).ToList();

            foreach (var movieLine in result)
            {
                MovieCashierData movieDataClass = new
                MovieCashierData();

                movieDataClass.movieCashierID = movieLine.IDMovie;

```

```

        if (movieLine.Cover != null)
        {
            BitmapImage cover = new BitmapImage();

            cover.BeginInit();
            cover.StreamSource = new
MemoryStream(movieLine.Cover);
            cover.EndInit();

            movieDataClass.movieCashierCover = cover;
        }
        else
        {
            movieDataClass.movieCashierCover = new
BitmapImage(new Uri("pack://application:,,,/Resource/NoImage.png",
UriKind.Absolute));
        }

        movieDataClass.movieCashierTitle =
movieLine.movieTitle;

        string movieGenreTemp = "";
        int movieGenresCounterTemp = 1;
        foreach (var genre in movieLine.movieGenres)
        {
            if (movieGenresCounterTemp !=
movieLine.movieGenres.Count())
                movieGenreTemp += genre + ", ";
            else
                movieGenreTemp += genre;

            movieGenresCounterTemp++;
        }
        movieDataClass.movieCashierGenre =
movieGenreTemp;

        movieDataClass.movieCashierYearOfPublication =
movieLine.YearOfPublication;
        movieDataClass.movieCashierTiming =
movieLine.Timing;
        movieDataClass.movieCashierAgeRating =
movieLine.AgeRating + "+";
        movieDataClass.movieCashierDescription =
movieLine.Description;
        movieDataClass.movieCashierCountry =
movieLine.movieCountry;

        string movieActorTemp = "";
        int movieActorCounterTemp = 1;
        foreach (var actor in movieLine.movieActors)
        {

```

```

            if (movieActorCounterTemp !=
movieLine.movieActors.Count())
            {
                movieActorTemp += actor + ", ";
            }
            else
            {
                movieActorTemp += actor;
            }

            movieActorCounterTemp++;
        }
        movieDataClass.movieCashierActors =
movieActorTemp;

        moviesCashierDataList.Add(movieDataClass);
    }

    var fullMovieCashierData = (from movie in
dataBase.Movie
                                join
                                session in dataBase.Session on
movie.IDMovie equals session.IDMovie into sessionGroup
                                from session in
sessionGroup.DefaultIfEmpty()
                                where (session.DateAndTimeSession >=
DateTime.Now)
                                group movie by movie.Title into
groupedMovies
                                select new
                                {
                                    MovieTitle = groupedMovies.Key,
                                    TotalSessions =
groupedMovies.Count()
                                }).ToList();

    FindCounterData.Text = result.Count() + "/" +
fullMovieCashierData.Count();

    MovieCashierList.Items.Refresh();
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}
}

private void Page_SizeChanged(object sender,
SizeChangedEventArgs e)
{
    try

```

```

{
    MovieCodeGrid.Width = 0;
    ImageGrid.Width = ActualWidth - ActualWidth * 0.8;
    MovieInfoGrid.Width = ActualWidth - ActualWidth * 0.7;
    DescriptionGrid.Width = ActualWidth - ActualWidth * 0.5;
    MovieCashierList.Height = ActualHeight - 35;
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
    MessageBoxButton.OK, MessageBoxImage.Error);
}

private void FindData_TextChanged(object sender,
TextChangedEventArgs e)
{
    LoadData(FindData.Text);
}

private void MovieCashierList_MouseDoubleClick(object sender,
MouseButtonEventArgs e)
{
    try
    {
        var selectedCashierMovie = MovieCashierList.SelectedItem
as MovieCashierData;
        if (selectedCashierMovie != null)
        {
            TransmittedData.idSelectedCashierMovie =
selectedCashierMovie.movieCashierID;

            CashierPanel cashierPanel = Window.GetWindow(this) as
CashierPanel;
            cashierPanel.SessionPageOpen();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
    MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
}

```

MovieControls.xaml

```

<Page x:Class="Cinema.MovieControls"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

```

```

xmlns:local="clr-namespace:Cinema"
mc:Ignorable="d"
d:DesignHeight="450" d:DesignWidth="800"
Title="MovieControls" SizeChanged="Page_SizeChanged"
Loaded="Page_Loaded">

<Grid Style="{DynamicResource MainColorStyle}">
    <StackPanel>
        <Grid Style="{DynamicResource SecondColorStyle}">
            <WrapPanel Margin="10,0,0,0" VerticalAlignment="Center">
                <Button x:Name="AddMovie" Margin="0,0,5,0"
Content="Добавить" HorizontalAlignment="Left" Width="120"
Style="{DynamicResource TabControlButtonStyle}"
Click="AddMovie_Click"/>
                <Button x:Name="RemoveMovie" Content="Удалить"
HorizontalAlignment="Left" Width="120" Style="{DynamicResource
TabControlButtonStyle}" Click="RemoveMovie_Click"/>
                <Rectangle Margin="10,0,10,0" Style="{DynamicResource
SecondColorSeparatorsRectangleStyle}" />
                <TextBlock Text="Поиск:" Margin="0,0,10,0"
Style="{DynamicResource MainTextBlockStyle}" />
                <TextBox x:Name="FindData" Width="200"
Style="{DynamicResource MainTextBoxStyle}"
TextChanged="FindData_TextChanged"/>
                <Rectangle Margin="10,0,10,0" Style="{DynamicResource
SecondColorSeparatorsRectangleStyle}" />
                <TextBlock Text="Найдено: " Style="{DynamicResource
MainTextBlockStyle}" />
                <TextBlock x:Name="FindCounterData" Text="0/0"
Style="{DynamicResource MainTextBlockStyle}" />
            </WrapPanel>
        </Grid>

        <ListView x:Name="MovieList" Height="400"
d:ItemsSource="{d:SampleData ItemCount=2}"
ScrollViewer.CanContentScroll="False"
MouseDoubleClick="MovieList_MouseDoubleClick">
            <ListView.ItemContainerStyle>
                <Style TargetType="ListViewItem">
                    <Setter Property="HorizontalContentAlignment"
Value="Stretch" />
                </Style>
            </ListView.ItemContainerStyle>
            <ListView.View>
                <GridView>
                    <GridViewColumn x:Name="MovieCodeGrid">
                        <Label Content="Код фильма"
Style="{DynamicResource MainLableStyle}" />
                    </GridViewColumn>
                    <GridViewColumn.CellTemplate>
                        <DataTemplate>
                            <Label Content="{Binding movieID}"
HorizontalAlignment="Center" Style="{DynamicResource
MainLableStyle}" />
                        </DataTemplate>
                    </GridViewColumn>
                </GridView>
            </ListView.View>
        </ListView>
    </StackPanel>
</Grid>

```

```

        </DataTemplate>
    </GridViewColumn.CellTemplate>
</GridViewColumn>
<GridViewColumn x:Name="ImageGrid">
    <Label Content="Постер" Style="{DynamicResource
MainLableStyle}"/>
    <GridViewColumn.CellTemplate>
        <DataTemplate>
            <Image Source="{Binding movieCover}"
MaxHeight="400" HorizontalAlignment="Center"/>
        </DataTemplate>
    </GridViewColumn.CellTemplate>
</GridViewColumn>
<GridViewColumn x:Name="MovieInfoGrid">
    <Label Content="Информация о фильме"
Style="{DynamicResource MainLableStyle}"/>
    <GridViewColumn.CellTemplate>
        <DataTemplate>
            <StackPanel>
                <Label Content="{Binding movieTitle}"
Style="{DynamicResource MainLableStyle}"/>
                <WrapPanel>
                    <Label Content="Жанр(ы):"
Style="{DynamicResource MainLableStyle}"/>
                    <Label Content="{Binding movieGenre}"
Style="{DynamicResource MainLableStyle}"/>
                </WrapPanel>
                <WrapPanel>
                    <Label Content="Год публикации:"
Style="{DynamicResource MainLableStyle}"/>
                    <Label Content="{Binding
movieYearOfPublication}" Style="{DynamicResource
MainLableStyle}"/>
                </WrapPanel>
                <WrapPanel>
                    <Label Content="Хронометраж:"
Style="{DynamicResource MainLableStyle}"/>
                    <Label Content="{Binding movieTiming}"
Style="{DynamicResource MainLableStyle}"/>
                </WrapPanel>
                <WrapPanel>
                    <Label Content="Возрастной рейтинг:"
Style="{DynamicResource MainLableStyle}"/>
                    <Label Content="{Binding
movieAgeRating}" Style="{DynamicResource MainLableStyle}"/>
                </WrapPanel>
                <WrapPanel>
                    <Label Content="Страна:"
Style="{DynamicResource MainLableStyle}"/>
                    <Label Content="{Binding movieCountry}"
Style="{DynamicResource MainLableStyle}"/>
                </WrapPanel>
            </StackPanel>
        </DataTemplate>
    </GridViewColumn.CellTemplate>
</GridViewColumn>

```

```

        <Label Content="Актеры:"
Style="{DynamicResource MainLableStyle}"/>
        <TextBlock Text="{Binding movieActors}"
Margin="5,0,0,0" VerticalAlignment="Center" TextWrapping="Wrap"
Style="{DynamicResource MainTextBlockStyle}"/>
    </WrapPanel>
</StackPanel>
</DataTemplate>
</GridViewColumn.CellTemplate>
</GridViewColumn>
<GridViewColumn x:Name="DescriptionGrid">
    <Label Content="Описание"
Style="{DynamicResource MainLableStyle}"/>
    <GridViewColumn.CellTemplate>
        <DataTemplate>
            <TextBlock Text="{Binding movieDescription}"
TextWrapping="Wrap" Style="{DynamicResource
MainTextBlockStyle}"/>
        </DataTemplate>
    </GridViewColumn.CellTemplate>
</GridViewColumn>
</GridView>
</ListView.View>
</ListView>
</StackPanel>
</Grid>
</Page>

```

MovieControls.xaml.cs

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media.Imaging;
using static Cinema.Authorization;

namespace Cinema
{
    /// <summary>
    /// Логика взаимодействия для MovieControls.xaml
    /// </summary>
    public partial class MovieControls : Page
    {
        public MovieControls()
        {
            InitializeComponent();
        }

        public class MovieData
        {

```



```

public int movieID { get; set; }
public BitmapImage movieCover { get; set; }
public string movieTitle { get; set; }
public string movieGenre { get; set; }
public int movieYearOfPublication { get; set; }
public double movieTiming { get; set; }
public string movieAgeRating { get; set; }
public string movieCountry { get; set; }
public string movieDescription { get; set; }
public string movieActors { get; set; }
}

List<MovieData> moviesDataList = new List<MovieData>();

private void Page_Loaded(object sender, RoutedEventArgs e)
{
    try
    {
        MovieList.ItemsSource = moviesDataList;

        LoadData(null);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void LoadData(string findLine)
{
    try
    {
        using (var dataBase = new CinemaEntities())
        {
            moviesDataList.Clear();

            var movieData = (from movie in dataBase.Movie
                            join
                                movieGenre in dataBase.MovieGenre on
                                    movie.IDMovie equals movieGenre.IDMovie into movieGenreGroup
                                from movieGenre in
                                    movieGenreGroup.DefaultIfEmpty()
                                join
                                    genre in dataBase.Genre on
                                        movieGenre.IDGenre equals genre.IDGenre into genreGroup
                                from genre in genreGroup.DefaultIfEmpty()
                                join
                                    country in dataBase.Country on
                                        movie.IDCountry equals country.IDCountry into countryGroup
                                from country in countryGroup.DefaultIfEmpty()
                                join
                                    actorsInMovies in dataBase.ActorsInMovies on
                                        movie.IDMovie equals actorsInMovies.IDMovie into
                                            actorsInMoviesGroup
                                from actorsInMovie in
                                    actorsInMoviesGroup.DefaultIfEmpty()
                                join
                                    actors in dataBase.Actor on
                                        actorsInMovie.IDActor equals actors.IDActor into actorsGroup
                                from actors in actorsGroup.DefaultIfEmpty()
                                where (findLine == null ||
                                    movie.Title.Contains(findLine) || genre.Title.Contains(findLine) ||
                                    movie.YearOfPublication.ToString().Contains(findLine) ||
                                    movie.Description.Contains(findLine) ||
                                    country.Title.Contains(findLine) || actors.Surname.Contains(findLine) ||
                                    actors.Name.Contains(findLine) || actors.Patronymic.Contains(findLine)
                                    || actors.Nickname.Contains(findLine))
                                select new
                                {
                                    movie.IDMovie,
                                    movie.Cover,
                                    movieTitle = movie.Title,
                                    movieGenre = genre.Title,
                                    movie.YearOfPublication,
                                    movie.Timing,
                                    movie.AgeRating,
                                    movie.Description,
                                    movieCountry = country.Title,
                                    movieActor = actors.Surname + " " +
                                        actors.Name + " " + actors.Patronymic + " " + actors.Nickname
                                }
                                ).ToList();

            var groupedMovieData = movieData.GroupBy(m =>
            m.IDMovie)
            .Select(g => new
            {
                g.Key,
                genres = g.Select(m => m.movieGenre).Distinct(),
                actors = g.Select(m => m.movieActor).Distinct()
            }).ToList();

            var result = groupedMovieData.Select(g =>
            {
                var movie = movieData.FirstOrDefault(m => m.IDMovie
                == g.Key);

                return new
                {
                    movie.IDMovie,
                    movie.Cover,
                    movie.movieTitle,
                    movieGenres = g.genres,
                    movie.YearOfPublication,
                    movie.Timing,

```

```

        movie.AgeRating,
        movie.Description,
        movie.movieCountry,
        movieActors = g.actors
    };
}).ToList();

foreach (var movieLine in result)
{
    MovieData movieDataClass = new MovieData();

    movieDataClass.movieID = movieLine.IDMovie;

    if (movieLine.Cover != null)
    {
        BitmapImage cover = new BitmapImage();

        cover.BeginInit();
        cover.StreamSource = new
MemoryStream(movieLine.Cover);
        cover.EndInit();

        movieDataClass.movieCover = cover;
    }
    else
    {
        movieDataClass.movieCover = new BitmapImage(new
Uri("pack://application:,,,/Resource/NoImage.png", UriKind.Absolute));
    }

    movieDataClass.movieTitle = movieLine.movieTitle;

    string movieGenreTemp = "";
    int movieGenresCounterTemp = 1;
    foreach (var genre in movieLine.movieGenres)
    {
        if (movieGenresCounterTemp !=
movieLine.movieGenres.Count())
            movieGenreTemp += genre + ", ";
        else
            movieGenreTemp += genre;

        movieGenresCounterTemp++;
    }
    movieDataClass.movieGenre = movieGenreTemp;

    movieDataClass.movieYearOfPublication =
movieLine.YearOfPublication;
    movieDataClass.movieTiming = movieLine.Timing;
    movieDataClass.movieAgeRating =
movieLine.AgeRating + "+";
    movieDataClass.movieDescription =
movieLine.Description;

    movieDataClass.movieCountry =
movieLine.movieCountry;

    string movieActorTemp = "";
    int movieActorCounterTemp = 1;
    foreach (var actor in movieLine.movieActors)
    {
        if (movieActorCounterTemp !=
movieLine.movieActors.Count())
        {
            movieActorTemp += actor + ", ";
        }
        else
        {
            movieActorTemp += actor;
        }

        movieActorCounterTemp++;
    }
    movieDataClass.movieActors = movieActorTemp;

    moviesDataList.Add(movieDataClass);
}

var fullMovieData = dataBase.Movie.ToList();
FindCounterData.Text = result.Count() + "/" +
fullMovieData.Count();

MovieList.Items.Refresh();
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}

private void Page_SizeChanged(object sender,
SizeChangedEventArgs e)
{
    try
    {
        MovieCodeGrid.Width = 0;
        ImageGrid.Width = ActualWidth - ActualWidth * 0.8;
        MovieInfoGrid.Width = ActualWidth - ActualWidth * 0.7;
        DescriptionGrid.Width = ActualWidth - ActualWidth * 0.5;
        MovieList.Height = ActualHeight - 35;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

```

```

    }
}

private void MovieList_MouseDoubleClick(object sender,
MouseButtonEventArgs e)
{
    try
    {
        var selectedMovie = MovieList.SelectedItem as MovieData;
        if (selectedMovie != null)
        {
            TransmittedData.idSelectedMovie =
selectedMovie.movieID;

            AddOrEditMovie addOrEditMovie = new
AddOrEditMovie();
            addOrEditMovie.ShowDialog();

            LoadData(FindData.Text);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void FindData_TextChanged(object sender,
TextChangedEventArgs e)
{
    LoadData(FindData.Text);
}

private void AddMovie_Click(object sender, RoutedEventArgs e)
{
    try
    {
        TransmittedData.idSelectedMovie = -1;

        AddOrEditMovie addOrEditMovie = new AddOrEditMovie();
        addOrEditMovie.ShowDialog();

        LoadData(FindData.Text);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void RemoveMovie_Click(object sender, RoutedEventArgs
e)
{
    try
    {
        var selectedMovie = MovieList.SelectedItem as MovieData;
        if (selectedMovie != null)
        {
            if (MessageBox.Show("Вы действительно хотите удалить
данный фильм?", "Внимание", MessageBoxButton.YesNo,
MessageBoxImage.Warning) == MessageBoxResult.Yes)
            {
                using (var dataBase = new CinemaEntities())
                {
                    var removeMovie = dataBase.Movie.Where(w =>
w.IDMovie == selectedMovie.movieID).FirstOrDefault();

                    var removeMovieGenre =
dataBase.MovieGenre.Where(w => w.IDMovie ==
selectedMovie.movieID).ToList();

                    var removeActorsInMovies =
dataBase.ActorsInMovies.Where(w => w.IDMovie ==
selectedMovie.movieID).ToList();

                    var removeTicket = (from session in dataBase.Session
join
ticket in dataBase.Ticket on
session.IDSession equals ticket.IDSession
where (session.IDMovie ==
selectedMovie.movieID)
select ticket).ToList();

                    var removeSession = dataBase.Session.Where(w =>
w.IDMovie == selectedMovie.movieID).ToList();

                    if (removeSession.Count > 0)
                    {
                        if (MessageBox.Show("Удаляемый фильм
присутствует в сессии. Вы действительно хотите его удалить?",
"Внимание", MessageBoxButton.YesNo, MessageBoxImage.Warning)
== MessageBoxResult.No)
                        {
                            return;
                        }

                        dataBase.Ticket.RemoveRange(removeTicket);
                        dataBase.Session.RemoveRange(removeSession);

                        dataBase.ActorsInMovies.RemoveRange(removeActorsInMovies);

                        dataBase.MovieGenre.RemoveRange(removeMovieGenre);
                        dataBase.Movie.Remove(removeMovie);

                        dataBase.SaveChanges();
                        MessageBox.Show("Данные были удалены",
"Готово", MessageBoxButton.OK, MessageBoxImage.Information);
                    }
                }
            }
        }
    }
}

```

```

        LoadData(FindData.Text);
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
    MessageBoxButton.OK, MessageBoxImage.Error);
}
}
}
}

```

PlacesCashierControls.xaml

```

<Page x:Class="Cinema.PlacesCashierControls"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:local="clr-namespace:Cinema"
mc:Ignorable="d"
d:DesignHeight="550" d:DesignWidth="1050"
Title="PlacesCashierControls" Background="White"
Loaded="Page_Loaded" SizeChanged="Page_SizeChanged">

    <Grid Style="{DynamicResource MainColorStyle}">
        <StackPanel x:Name="Hall" Margin="0,0,198,0"
VerticalAlignment="Center" HorizontalAlignment="Center"
SizeChanged="Hall_SizeChanged"/>
        <Grid VerticalAlignment="Center"
HorizontalAlignment="Right" Margin="10,0,0,0" Width="200">
            <Rectangle Style="{DynamicResource
SecondColorRectanglePlaceStyle}" Opacity="0.5"/>
            <StackPanel>
                <TextBlock Text="Бронирование билета"
Margin="0,10,0,0" TextWrapping="Wrap" Style="{DynamicResource
TitleTextBlockStyle}"/>
                <StackPanel>
                    <Label Content="Ряд" HorizontalAlignment="Center"
Style="{DynamicResource MainLableStyle}"/>
                    <Label x:Name="SelectedRow" Content="Her"
HorizontalAlignment="Center" Style="{DynamicResource
MainLableStyle}"/>
                </StackPanel>
                <StackPanel>
                    <Label Content="Место" HorizontalAlignment="Center"
Style="{DynamicResource MainLableStyle}"/>
                    <Label x:Name="SelectedPlace" Content="Her"
HorizontalAlignment="Center" Style="{DynamicResource
MainLableStyle}"/>
                </StackPanel>
            </Grid>
        </Grid>
    </Page>

```

```

        <StackPanel>
            <Label Content="Цена билета"
HorizontalAlignment="Center" Style="{DynamicResource
MainLableStyle}"/>
            <Label x:Name="TicketPrice" Content=""
HorizontalAlignment="Center" Style="{DynamicResource
MainLableStyle}"/>
        </StackPanel>
    </StackPanel>
    <StackPanel>
        <Label Content="Дата и время сеанса"
HorizontalAlignment="Center" Style="{DynamicResource
MainLableStyle}"/>
        <Label x:Name="DateTimeSession" Content=""
HorizontalAlignment="Center" Style="{DynamicResource
MainLableStyle}"/>
    </StackPanel>
    <Button x:Name="BookingTicket" Margin="10,0,10,10"
Content="Забронировать" Style="{DynamicResource
TabControlButtonStyle}" Click="BookingTicket_Click"/>
</StackPanel>
</Grid>
</Grid>
</Page>

```

PlacesCashierControls.xaml.cs

```

using iTextSharp.text;
using iTextSharp.text.pdf;
using QRCode;
using System;
using
System.Data.Entity.Core.Common.CommandTrees.ExpressionBuilder;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Xml.Linq;
using static Cinema.Authorization;

namespace Cinema
{
    /// <summary>
    /// Логика взаимодействия для PlacesCashierControls.xaml
    /// </summary>
    public partial class PlacesCashierControls : Page
    {
        public PlacesCashierControls()
        {
            InitializeComponent();
        }
    }
}

```

```

public int selectedRowNumber;
public int selectedPlaceNumber;
public Button selectedRowPlaceButton;

private void Page_Loaded(object sender, RoutedEventArgs e)
{
    LoadData();
}

private void LoadData()
{
    try
    {
        using (var dataBase = new CinemaEntities())
        {
            var ticketData = dataBase.Ticket.Where(w => w.IDSession
== TransmittedData.idSelectedCashierSession).ToList();

            var settingsData = dataBase.Settings.OrderByDescending(o
=> o.DateTimeChange).FirstOrDefault();

            if (settingsData != null)
            {
                Hall.Children.Clear();

                for (int row = 0; row < settingsData.RowHall; row++)
                {
                    WrapPanel wrapPanel = new WrapPanel();

                    TextBlock textBlockBegin = new TextBlock();
                    textBlockBegin.Text = $"Пяд {row + 1}";
                    textBlockBegin.FontSize = 10;
                    textBlockBegin.Margin = new Thickness(0, 0, 5, 0);
                    textBlockBegin.VerticalAlignment =
VerticalAlignment.Center;

                    TextBlock textBlockEnd = new TextBlock();
                    textBlockEnd.Text = $"Пяд {row + 1}";
                    textBlockEnd.FontSize = 10;
                    textBlockEnd.VerticalAlignment =
VerticalAlignment.Center;

                    wrapPanel.Children.Add(textBlockBegin);
                    for (int place = 1; place <= settingsData.PlaceHall;
place++)
                    {
                        Button button = new Button();
                        button.Content = place;
                        button.Name = $"Row{row}Place{place}";
                        button.Width = 18;
                        button.Height = 18;
                        button.FontSize = 10;

                        button.Background = Brushes.White;
                        button.Margin = new Thickness(0, 0, 5, 0);
                        button.Click += HallButton_Click;

                        if (settingsData.HiddenPlaces != null)
                        {
                            string[] hidePlaceTemp =
settingsData.HiddenPlaces.Split('|');
                            foreach (var hidePlaceLine in hidePlaceTemp)
                            {
                                Match match = Regex.Match(hidePlaceLine,
@"Row(\d+)Place(\d+)");

                                if (match.Success)
                                {
                                    if (row + 1 ==
int.Parse(match.Groups[1].Value) + 1 && place ==
int.Parse(match.Groups[2].Value))
                                    {
                                        button.Click -= HallButton_Click;
                                        button.Opacity = 0;
                                        button.IsEnabled = false;
                                    }
                                }
                            }
                        }

                        foreach (var ticketLine in ticketData)
                        {
                            if (row + 1 == ticketLine.RowNumber && place
== ticketLine.PlaceNumber)
                            {
                                button.Foreground = Brushes.Red;
                                button.IsEnabled = false;
                            }
                        }

                        wrapPanel.Children.Add(button);
                    }
                    wrapPanel.Children.Add(textBlockEnd);

                    StackPanel stackPanel = new StackPanel();
                    stackPanel.Margin = new Thickness(0, 0, 0, 5);
                    stackPanel.HorizontalAlignment =
HorizontalAlignment.Center;
                    stackPanel.Children.Add(wrapPanel);

                    Hall.Children.Add(stackPanel);
                }
            }
            else
            {
                Label lable = new Label();

```

```

        lable.FontSize = 26;
        lable.Foreground = Brushes.Red;
        lable.Content = "Зал не размечен
администратором.\rОбратитесь с системному администратору.";
        lable.HorizontalAlignment =
HorizontalAlignment.Center;
        lable.HorizontalContentAlignment =
HorizontalAlignment.Center;

        Hall.Children.Add(lable);
    }

    var sessionData = dataBase.Session.Where(w =>
w.IDSession ==
TransmittedData.idSelectedCashierSession).FirstOrDefault();
    TicketPrice.Content =
sessionData.TicketPrice.ToString().Remove(sessionData.TicketPrice.To
String().Length - 2, 2) + " руб.";
    DateTimeSession.Content =
sessionData.DateAndTimeSession;
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}

private void Page_SizeChanged(object sender,
SizeChangedEventArgs e)
{
    ResizeWindows();
}

private void Hall_SizeChanged(object sender,
SizeChangedEventArgs e)
{
    ResizeWindows();
}

private void ResizeWindows()
{
    try
    {
        double maxWidth = this.ActualWidth - 10;
        double maxHeight = this.ActualHeight - 10;

        double scale = this.ActualWidth / (Hall.ActualWidth * 1.28);
        double centerX = Hall.ActualWidth / 2;
        double centerY = Hall.ActualHeight / 2;

        double newWidth = Hall.ActualWidth * scale;

        double newHeight = Hall.ActualHeight * scale;

        if (newWidth > maxWidth)
        {
            scale = maxWidth / Hall.ActualWidth;
        }
        if (newHeight > maxHeight)
        {
            scale = Math.Min(scale, maxHeight / Hall.ActualHeight);
        }

        ScaleTransform scaleTransform = new ScaleTransform(scale,
scale, centerX, centerY);
        Hall.RenderTransform = scaleTransform;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }

    private void HallButton_Click(object sender, RoutedEventArgs e)
    {
        try
        {
            Button clickedButton = sender as Button;

            if (clickedButton != null)
            {
                if (selectedRowPlaceButton != null)
                {
                    selectedRowPlaceButton.Background = Brushes.White;
                    selectedRowPlaceButton.Foreground = Brushes.Black;
                    selectedRowPlaceButton.BorderBrush = (Brush)(new
BrushConverter().ConvertFrom("#FF707070"));
                }

                Match match = Regex.Match(clickedButton.Name,
@"Row(\d+)Place(\d+)");

                if (match.Success)
                {
                    selectedRowNumber = int.Parse(match.Groups[1].Value)
+ 1;
                    selectedPlaceNumber =
int.Parse(match.Groups[2].Value);

                    clickedButton.Background = Brushes.Aqua;
                    clickedButton.Foreground = Brushes.Black;
                    clickedButton.BorderBrush = Brushes.Black;
                    selectedRowPlaceButton = clickedButton;
                }
            }
        }
    }
}

```

```

        SelectedPlaceChange();
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
}

private void SelectedPlaceChange()
{
    try
    {
        if (selectedRowNumber != 0)
        {
            SelectedRow.Content = selectedRowNumber;
        }
        else
        {
            SelectedRow.Content = "Her";
        }

        if (selectedPlaceNumber != 0)
        {
            SelectedPlace.Content = selectedPlaceNumber;
        }
        else
        {
            SelectedPlace.Content = "Her";
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void ClearPlaceChange()
{
    try
    {
        selectedRowNumber = 0;
        selectedPlaceNumber = 0;
        selectedRowPlaceButton = null;
        SelectedPlaceChange();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

```

```

    }
}

private async Task PrintTicket(int selectedTicket)
{
    try
    {
        using (var dataBase = new CinemaEntities())
        {
            var ticketData = (from ticket in dataBase.Ticket
                join
                    session in dataBase.Session on ticket.IDSession
                        equals session.IDSession into sessionGroup
                    from session in sessionGroup.DefaultIfEmpty()
                join
                    movie in dataBase.Movie on session.IDMovie
                        equals movie.IDMovie into movieGroup
                    from movie in movieGroup.DefaultIfEmpty()
                join
                    employee in dataBase.Employee on
                        ticket.IDEmployee equals employee.IDEmployee into employeeGroup
                    from employee in
                        employeeGroup.DefaultIfEmpty()
                where (ticket.IDTicket == selectedTicket)
                select new
                {
                    ticket.IDTicket,
                    ticket.RowNumber,
                    ticket.PlaceNumber,
                    ticket.DateTimeBooking,
                    movie = movie.Title,
                    session.DateAndTimeSession,
                    employee.Surname,
                    employee.Name,
                    employee.Patronymic,
                    session.TicketPrice
                }).FirstOrDefault();

            string tempFilePath = Path.GetTempFileName() + ".pdf";

            string qrCodeText =
                $"{ticketData.IDTicket}{{ticketData.movie}}{{ticketData.DateAndTimeS
                    ession}}{{ticketData.Surname} {ticketData.Name}
                    {ticketData.Patronymic}}{{Math.Round(ticketData.TicketPrice,2)}}";

            QRCodeGenerator qrGenerator = new QRCodeGenerator();
            QRCodeData qrCodeData =
                qrGenerator.CreateQrCode(qrCodeText,
                    QRCodeGenerator.ECCLLevel.Q);

            QRCode qrCode = new QRCode(qrCodeData);
            using (System.Drawing.Bitmap qrCodeBitmap =
                qrCode.GetGraphic(20, System.Drawing.Color.Black,
                    System.Drawing.Color.White, null))
            {

```

```

using (MemoryStream ms = new MemoryStream())
{
    qrCodeBitmap.Save(ms,
System.Drawing.Imaging.ImageFormat.Png); // Сохраняем в PNG
    byte[] qrCodeBytes = ms.ToArray();

    Document doc = new Document();
    PdfWriter writer = PdfWriter.GetInstance(doc, new
FileStream($"{tempFilePath}", FileMode.Create));

    BaseFont baseFont =
BaseFont.CreateFont("C:\\Windows\\Fonts\\arial.ttf",
BaseFont.IDENTITY_H, BaseFont.NOT_EMBEDDED);
    Font font = new Font(baseFont, 16);

    BaseFont baseFontHead =
BaseFont.CreateFont("C:\\Windows\\Fonts\\arial.ttf",
BaseFont.IDENTITY_H, BaseFont.NOT_EMBEDDED);
    Font fontHead = new Font(baseFont, 20, Font.BOLD);

    doc.Open();

    Paragraph mainParagraph = new Paragraph("Билет",
fontHead);
    mainParagraph.Alignment =
Element.ALIGN_CENTER;

    Paragraph paragraph = new Paragraph("Номер
билета: " + ticketData.IDTicket, font);
    Paragraph paragraph1 = new Paragraph("Фильм: " +
ticketData.movie, font);
    Paragraph paragraph2 = new Paragraph("Ряд: " +
ticketData.RowNumber, font);
    Paragraph paragraph3 = new Paragraph("Место: " +
ticketData.PlaceNumber, font);
    Paragraph paragraph4 = new Paragraph("Дата и время
начала сеанса: " + ticketData.DateAndTimeSession, font);
    Paragraph paragraph5 = new Paragraph("Кассир: " +
ticketData.Surname + " " + ticketData.Name + " " +
ticketData.Patronymic, font);
    Paragraph paragraph6 = new Paragraph("Дата
продажи: " + ticketData.DateTimeBooking, font);
    Paragraph paragraph7 = new Paragraph("Цена: " +
ticketData.TicketPrice.ToString().Remove(ticketData.TicketPrice.ToStri
ng().Length - 2, 2) + " Руб.", font);

    doc.Add(mainParagraph);
    doc.Add(paragraph);
    doc.Add(paragraph1);
    doc.Add(paragraph2);
    doc.Add(paragraph3);
    doc.Add(paragraph4);
    doc.Add(paragraph5);
    doc.Add(paragraph6);

    doc.Add(paragraph7);

    iTextSharp.text.Image image =
iTextSharp.text.Image.GetInstance(qrCodeBytes);
    image.ScaleToFit(200, 200);
    doc.Add(image);

    doc.Close();
}

Process.Start(tempFilePath);
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}

private void BookingTicket_Click(object sender, RoutedEventArgs
e)
{
    try
    {
        if (selectedRowNumber == 0 && selectedPlaceNumber == 0)
        {
            MessageBox.Show("Место в зале не выбранно",
"Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);
            return;
        }

        using (var dataBase = new CinemaEntities())
        {
            if (MessageBox.Show("Забронировать место на сеанс?",
"Внимание", MessageBoxButton.YesNo, MessageBoxImage.Warning)
== MessageBoxResult.Yes)
            {
                var newTicket = new Ticket();

                newTicket.IDSession =
TransmittedData.idSelectedCashierSession;
                newTicket.RowNumber = selectedRowNumber;
                newTicket.PlaceNumber = selectedPlaceNumber;
                newTicket.IDEmployee = TransmittedData.idEmployee;
                newTicket.DateTimeBooking = DateTime.Now;

                dataBase.Ticket.Add(newTicket);
                dataBase.SaveChanges();

                MessageBox.Show("Место на сеанс зарезервировано",
"Готово", MessageBoxButton.OK, MessageBoxImage.Information);
            }
        }
    }
}

```



```

Task.Run(() => PrintTicket(new Ticket.IDTicket));

        selectedRowPlaceButton.Background = (Brush)(new
BrushConverter().ConvertFrom("#FFDDDDDD"));
        selectedRowPlaceButton.BorderBrush = (Brush)(new
BrushConverter().ConvertFrom("#FF707070"));
        selectedRowPlaceButton.Foreground = Brushes.Red;
        selectedRowPlaceButton.IsEnabled = false;

        ClearPlaceChange();
    }
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}
}
}
}
}

```

SessionCashierControls.xaml

```

<Page x:Class="Cinema.SessionCashierControls"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:local="clr-namespace:Cinema"
mc:Ignorable="d"
d:DesignHeight="450" d:DesignWidth="800"
Title="SessionControls" SizeChanged="Page_SizeChanged"
Loaded="Page_Loaded">

    <Grid Style="{DynamicResource MainColorStyle}">
        <StackPanel>
            <Grid Style="{DynamicResource SecondColorStyle}">
                <WrapPanel Margin="10,0,10,0"
VerticalAlignment="Center">
                    <TextBlock Text="Поиск:" Margin="0,0,10,0"
Style="{DynamicResource MainTextBlockStyle}" />
                    <TextBox x:Name="FindData" Width="200"
Style="{DynamicResource MainTextBoxStyle}"
TextChanged="FindData_TextChanged" />
                    <Rectangle Margin="10,0,10,0" Style="{DynamicResource
SecondColorSeparatorsRectangleStyle}" />
                    <CheckBox x:Name="FindDateChecker" Margin="0,4,5,0"
Style="{DynamicResource MainCheckBoxStyle}"
Click="FindDateChecker_Click" />
                    <TextBlock Text="Дата сеанса:" Margin="0,0,10,0"
Style="{DynamicResource MainTextBlockStyle}" />

```

```

<DatePicker x:Name="FindSessionData"
Style="{DynamicResource SecondDatePickerStyle}"
SelectedDateChanged="FindSessionData_SelectedDateChanged" />
                <Rectangle Margin="10,0,10,0" Style="{DynamicResource
SecondColorSeparatorsRectangleStyle}" />
                <TextBlock Text="Найдено: " Style="{DynamicResource
MainTextBlockStyle}" />
                <TextBlock x:Name="FindCounterData" Text="0/0"
Style="{DynamicResource MainTextBlockStyle}" />
            </WrapPanel>
        </Grid>

        <ListView x:Name="SessionCashierList" Height="400"
d:ItemsSource="{d:SampleData ItemCount=2}"
ScrollViewer.CanContentScroll="False"
MouseDoubleClick="SessionCashierList_MouseDoubleClick">
            <ListView.ItemContainerStyle>
                <Style TargetType="ListViewItem">
                    <Setter Property="HorizontalContentAlignment"
Value="Stretch" />
                </Style>
            </ListView.ItemContainerStyle>
            <ListView.View>
                <GridView>
                    <GridViewColumn x:Name="SessionCodeGrid">
                        <Label Content="Код сессии"
Style="{DynamicResource MainLableStyle}" />
                        <GridViewColumn.CellTemplate>
                            <DataTemplate>
                                <Label Content="{Binding sessionCashierID}"
HorizontalAlignment="Center" Style="{DynamicResource
MainLableStyle}" />
                            </DataTemplate>
                        </GridViewColumn.CellTemplate>
                    </GridViewColumn>
                    <GridViewColumn x:Name="SessionCoverGrid">
                        <Label Content="Постер" Style="{DynamicResource
MainLableStyle}" />
                        <GridViewColumn.CellTemplate>
                            <DataTemplate>
                                <Image Source="{Binding
sessionCashierMovieCover}" MaxHeight="400"
HorizontalAlignment="Center" />
                            </DataTemplate>
                        </GridViewColumn.CellTemplate>
                    </GridViewColumn>
                    <GridViewColumn x:Name="SessionInfoGrid">
                        <Label Content="Информация о фильме"
Style="{DynamicResource MainLableStyle}" />
                        <GridViewColumn.CellTemplate>
                            <DataTemplate>
                                <StackPanel>

```

```

        <Label Content="{Binding
sessionCashierMovieTitle}" Style="{DynamicResource
MainLableStyle}"/>
        <WrapPanel>
            <Label Content="Жанр(ы):"
Style="{DynamicResource MainLableStyle}"/>
            <Label Content="{Binding
sessionCashierMovieGenre}" Style="{DynamicResource
MainLableStyle}"/>
        </WrapPanel>
        <WrapPanel>
            <Label Content="Год публикации:"
Style="{DynamicResource MainLableStyle}"/>
            <Label Content="{Binding
sessionCashierMovieYearOfPublication}" Style="{DynamicResource
MainLableStyle}"/>
        </WrapPanel>
        <WrapPanel>
            <Label Content="Хронометраж:"
Style="{DynamicResource MainLableStyle}"/>
            <Label Content="{Binding
sessionCashierMovieYearOfPublication}" Style="{DynamicResource
MainLableStyle}"/>
        </WrapPanel>
        <WrapPanel>
            <Label Content="Возрастной рейтинг:"
Style="{DynamicResource MainLableStyle}"/>
            <Label Content="{Binding
sessionCashierMovieAgeRating}" Style="{DynamicResource
MainLableStyle}"/>
        </WrapPanel>
        <WrapPanel>
            <Label Content="Страна:"
Style="{DynamicResource MainLableStyle}"/>
            <Label Content="{Binding
sessionCashierMovieCountry}" Style="{DynamicResource
MainLableStyle}"/>
        </WrapPanel>
        <WrapPanel>
            <Label Content="Актеры:"
Style="{DynamicResource MainLableStyle}"/>
            <TextBlock Text="{Binding
sessionCashierActors}" Margin="5,0,0,0" VerticalAlignment="Center"
TextWrapping="Wrap" Style="{DynamicResource
MainTextBlockStyle}"/>
        </WrapPanel>
    </StackPanel>
</DataTemplate>
</GridViewColumn.CellTemplate>
</GridViewColumn>
<GridViewColumn x:Name="SessionDateTimeGrid">
    <Label Content="Информация о сеансе"
Style="{DynamicResource MainLableStyle}"/>

```

```

<GridViewColumn.CellTemplate>
    <DataTemplate>
        <StackPanel>
            <WrapPanel>
                <Label Content="Дата начала сеанса:"
Style="{DynamicResource MainLableStyle}"/>
                <Label Content="{Binding
sessionCashierDateSessionStart}" Style="{DynamicResource
MainLableStyle}"/>
            </WrapPanel>
            <WrapPanel>
                <Label Content="Время начала сеанса:"
Style="{DynamicResource MainLableStyle}"/>
                <Label Content="{Binding
sessionCashierTimeSessionStart}" Style="{DynamicResource
MainLableStyle}"/>
            </WrapPanel>
            <WrapPanel>
                <Label Content="Места в зале:"
Style="{DynamicResource MainLableStyle}"/>
                <Label Content="{Binding
sessionSeatsInHall}" Style="{DynamicResource MainLableStyle}"/>
            </WrapPanel>
        </StackPanel>
    </DataTemplate>
</GridViewColumn.CellTemplate>
</GridViewColumn>
<GridViewColumn x:Name="SessionPriceGrid">
    <Label Content="Цена билета"
Style="{DynamicResource MainLableStyle}"/>
    <GridViewColumn.CellTemplate>
        <DataTemplate>
            <Label Content="{Binding
sessionCashierTicketPrice}" Style="{DynamicResource
MainLableStyle}"/>
        </DataTemplate>
    </GridViewColumn.CellTemplate>
</GridViewColumn>
</GridView>
</ListView.View>
</ListView>
</StackPanel>
</Grid>
</Page>

```

SessionCashierControls.xaml.cs

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;

```

```

using System.Windows.Media.Imaging;
using static Cinema.Authorization;

namespace Cinema
{
    /// <summary>
    /// Логика взаимодействия для SessionCashierControls.xaml
    /// </summary>
    public partial class SessionCashierControls : Page
    {
        public SessionCashierControls()
        {
            InitializeComponent();
        }

        public class SessionCashierData
        {
            public int sessionCashierID { get; set; }
            public BitmapImage sessionCashierMovieCover { get; set; }
            public string sessionCashierMovieTitle { get; set; }
            public string sessionCashierMovieGenre { get; set; }
            public int sessionCashierMovieYearOfPublication { get; set; }
            public double sessionCashierMovieTiming { get; set; }
            public string sessionCashierMovieAgeRating { get; set; }
            public string sessionCashierMovieCountry { get; set; }
            public string sessionCashierActors { get; set; }
            public string sessionCashierDateSessionStart { get; set; }
            public string sessionCashierTimeSessionStart { get; set; }
            public string sessionCashierTicketPrice { get; set; }
            public string sessionSeatsInHall { get; set; }
        }

        List<SessionCashierData> sessionCashierDataList = new
        List<SessionCashierData>();

        private void Page_Loaded(object sender, RoutedEventArgs e)
        {
            try
            {
                SessionCashierList.ItemsSource = sessionCashierDataList;

                LoadData(null, FindSessionData);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Ошибка",
                MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }

        private void LoadData(string findLine, DatePicker findDate)
        {
            try
            {
                using (var dataBase = new CinemaEntities())
                {
                    sessionCashierDataList.Clear();

                    DatePicker endDate = new DatePicker();
                    endDate.SelectedDate = null;

                    if (findDate.SelectedDate != null)
                    {
                        {
                            endDate.SelectedDate =
                            findDate.SelectedDate.Value.AddDays(1);
                        }
                        else
                        {
                            {
                                endDate.SelectedDate = null;
                            }
                        }
                    }

                    var sessionCashierData = (from session in dataBase.Session
                                            join
                                            movie in dataBase.Movie on
                                            session.IDMovie equals movie.IDMovie into movieGroup
                                            from movie in
                                            movieGroup.DefaultIfEmpty()
                                            join
                                            movieGenre in dataBase.MovieGenre on
                                            movie.IDMovie equals movieGenre.IDMovie into movieGenreGroup
                                            from movieGenre in
                                            movieGenreGroup.DefaultIfEmpty()
                                            join
                                            actorsInMovies in
                                            dataBase.ActorsInMovies on movie.IDMovie equals
                                            actorsInMovies.IDMovie into actorsInMoviesGroup
                                            from actorsInMovies in
                                            actorsInMoviesGroup.DefaultIfEmpty()
                                            join
                                            actors in dataBase.Actor on
                                            actorsInMovies.IDActor equals actors.IDActor into actorsGroup
                                            from actors in
                                            actorsGroup.DefaultIfEmpty()
                                            join
                                            genre in dataBase.Genre on
                                            movieGenre.IDGenre equals genre.IDGenre into genreGroup
                                            from genre in genreGroup.DefaultIfEmpty()
                                            join
                                            country in dataBase.Country on
                                            movie.IDCountry equals country.IDCountry into countryGroup
                                            from country in
                                            countryGroup.DefaultIfEmpty()
                                            join
                                            ticket in dataBase.Ticket on
                                            session.IDSession equals ticket.IDSession into ticketGroup
                                            from ticket in ticketGroup.DefaultIfEmpty()

```

```

        where ((findLine == null ||
movie.Title.Contains(findLine) || genre.Title.Contains(findLine) ||
movie.YearOfPublication.ToString().Contains(findLine) ||
movie.Description.Contains(findLine) ||
country.Title.Contains(findLine)) && ((findDate.SelectedDate == null
&& endDate.SelectedDate == null) || (session.DateAndTimeSession >=
findDate.SelectedDate.Value && session.DateAndTimeSession <=
endDate.SelectedDate.Value)) && (session.DateAndTimeSession >=
DateTime.Now) && (session.IDMovie ==
TransmittedData.idSelectedCashierMovie))
        select new
        {
            session.IDSession,
            movie.Cover,
            movieTitle = movie.Title,
            movieGenre = genre.Title,
            movie.YearOfPublication,
            movie.Timing,
            movie.AgeRating,
            movieCountry = country.Title,
            movieActors = actors.Surname + " " +
actors.Name + " " + actors.Patronymic + " " + actors.Nickname,
            session.DateAndTimeSession,
            TicketID = ticket == null ? 0 :
ticket.IDTicket,
            session.TicketPrice
        }).ToList();

var groupedSessionCashierData =
sessionCashierData.GroupBy(m => m.IDSession)
        .Select(g => new
        {
            g.Key,
            genres = g.Select(m => m.movieGenre).Distinct(),
            actors = g.Select(m => m.movieActors).Distinct(),
            TicketCount = g.Count()
        }).ToList();

var result = groupedSessionCashierData.Select(g =>
{
    var session = sessionCashierData.FirstOrDefault(m =>
m.IDSession == g.Key);
    return new
    {
        session.IDSession,
        session.Cover,
        session.movieTitle,
        session.MovieGenres = g.genres,
        session.YearOfPublication,
        session.Timing,
        session.AgeRating,
        session.movieCountry,
        session.DateAndTimeSession,

```

```

        session.TicketPrice,
        movieActors = g.actors,
        g.TicketCount,
    });
}).ToList();

foreach (var sessionLine in result)
{
    SessionCashierData sessionDataClass = new
SessionCashierData();

    sessionDataClass.sessionCashierID =
sessionLine.IDSession;

    if (sessionLine.Cover != null)
    {
        BitmapImage cover = new BitmapImage();

        cover.BeginInit();
        cover.StreamSource = new
MemoryStream(sessionLine.Cover);
        cover.EndInit();

        sessionDataClass.sessionCashierMovieCover = cover;
    }
    else
    {
        sessionDataClass.sessionCashierMovieCover = new
BitmapImage(new Uri("pack://application:;./Resource/NoImage.png",
UriKind.Absolute));
    }

    sessionDataClass.sessionCashierMovieTitle =
sessionLine.movieTitle;

    string sessionMovieGenreTemp = "";
    int movieGenresCounterTemp = 1;
    foreach (var genere in sessionLine.sessionMovieGenres)
    {
        if (movieGenresCounterTemp !=
sessionLine.sessionMovieGenres.Count())
            sessionMovieGenreTemp += genere + ", ";
        else
            sessionMovieGenreTemp += genere;

        movieGenresCounterTemp++;
    }
    sessionDataClass.sessionCashierMovieGenre =
sessionMovieGenreTemp;

    string movieActorTemp = "";
    int movieActorCounterTemp = 1;
    foreach (var actor in sessionLine.movieActors)

```

```

        {
            if (movieActorCounterTemp !=
sessionLine.movieActors.Count())
            {
                movieActorTemp += actor + ", ";
            }
            else
            {
                movieActorTemp += actor;
            }

            movieActorCounterTemp++;
        }
        sessionDataClass.sessionCashierActors =
movieActorTemp;

        sessionDataClass.sessionCashierMovieYearOfPublication =
sessionLine.YearOfPublication;
        sessionDataClass.sessionCashierMovieTiming =
sessionLine.Timing;
        sessionDataClass.sessionCashierMovieAgeRating =
sessionLine.AgeRating + "+";
        sessionDataClass.sessionCashierMovieCountry =
sessionLine.movieCountry;
        sessionDataClass.sessionCashierDateSessionStart =
sessionLine.DateAndTimeSession.ToString().Split(' ')[0];
        sessionDataClass.sessionCashierTimeSessionStart =
sessionLine.DateAndTimeSession.ToString().Split(' ')[1];
        sessionDataClass.sessionCashierTicketPrice =
sessionLine.TicketPrice.ToString().Remove(sessionLine.TicketPrice.To
String().Length - 2, 2) + " руб.";

        var seatsHallData =
dataBase.Settings.OrderByDescending(o =>
o.IDSettings).FirstOrDefault();
        if (seatsHallData != null)
        {
            int hidePlaceCount =
seatsHallData.HiddenPlaces.Split('|').Count();

            int purchasedTickets = sessionLine.TicketCount /
((movieGenresCounterTemp - 1) * (movieActorCounterTemp - 1));
            int totalTickets = (seatsHallData.RowHall *
seatsHallData.PlaceHall) - hidePlaceCount;

            if (purchasedTickets == 1)
            {
                var ticketCount = dataBase.Ticket.Where(w =>
w.IDSession == sessionLine.IDSession).Count();
                sessionDataClass.sessionSeatsInHall = ticketCount
+ "/" + totalTickets;
            }

            else
            {
                sessionDataClass.sessionSeatsInHall =
purchasedTickets + "/" + totalTickets;
            }
        }
        else
        {
            sessionDataClass.sessionSeatsInHall = "Зал не
размечен";
        }

        sessionCashierDataList.Add(sessionDataClass);
    }

    var fullSessionCashierData = dataBase.Session.Where(w
=> w.DateAndTimeSession >= DateTime.Now && w.IDMovie ==
TransmittedData.idSelectedCashierMovie).ToList();
    FindCounterData.Text = result.Count() + "/" +
fullSessionCashierData.Count();
    SessionCashierList.Items.Refresh();
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}

private void Page_SizeChanged(object sender,
SizeChangedEventArgs e)
{
    try
    {
        SessionCodeGrid.Width = 0;
        SessionCoverGrid.Width = ActualWidth - ActualWidth * 0.8;
        SessionInfoGrid.Width = ActualWidth - ActualWidth * 0.7;
        SessionDateTimeGrid.Width = ActualWidth - ActualWidth *
0.7;
        SessionPriceGrid.Width = ActualWidth - ActualWidth * 0.8;
        SessionCashierList.Height = ActualHeight - 35;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void FindData_TextChanged(object sender,
TextChangedEventArgs e)
{
    try

```

```

    {
        if (FindDateChecker.IsChecked == false)
        {
            FindSessionData.SelectedDate = null;
        }
        LoadData(FindData.Text, FindSessionData);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void FindDateChecker_Click(object sender,
RoutedEventArgs e)
{
    try
    {
        if (FindDateChecker.IsChecked == false)
        {
            FindSessionData.IsEnabled = false;
            FindSessionData.SelectedDate = null;
        }
        else
        {
            FindSessionData.IsEnabled = true;
            FindSessionData.SelectedDate = DateTime.Now.Date;
        }
        LoadData(FindData.Text, FindSessionData);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void FindSessionData_SelectedDateChanged(object sender,
SelectionChangedEventArgs e)
{
    try
    {
        if (FindDateChecker.IsChecked == false)
        {
            FindSessionData.SelectedDate = null;
        }
        LoadData(FindData.Text, FindSessionData);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

```

```

    }
}

private void SessionCashierList_MouseDoubleClick(object sender,
MouseButtonEventArgs e)
{
    try
    {
        var selectedCashierSession = SessionCashierList.SelectedItem
as SessionCashierData;
        if (selectedCashierSession != null)
        {
            TransmittedData.idSelectedCashierSession =
selectedCashierSession.sessionCashierID;

            CashierPanel cashierPanel = Window.GetWindow(this) as
CashierPanel;
            cashierPanel.PlacesPageOpen();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Ошибка",
            MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }
}

```

SessionControls.xaml

```

<Page x:Class="Cinema.SessionControls"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:local="clr-namespace:Cinema"
mc:Ignorable="d"
d:DesignHeight="450" d:DesignWidth="800"
Title="SessionControls" SizeChanged="Page_SizeChanged"
Loaded="Page_Loaded">

    <Grid Style="{DynamicResource MainColorStyle}">
        <StackPanel>
            <Grid Style="{DynamicResource SecondColorStyle}">
                <WrapPanel Margin="10,0,10,0"
VerticalAlignment="Center">
                    <Button x:Name="AddSession" Margin="0,0,5,0"
Content="Добавить" HorizontalAlignment="Center" Width="120"
Style="{DynamicResource TabControlButtonStyle}"
Click="AddSession_Click"/>

```

```

        <Button x:Name="RemoveSession" Content="Удалить"
HorizontalAlignment="Left" Width="120" Style="{DynamicResource
TabControlButtonStyle}" Click="RemoveSession_Click"/>
        <Rectangle Margin="10,0,10,0" Style="{DynamicResource
SecondColorSeparatorsRectangleStyle}"/>
        <TextBlock Text="Поиск:" Margin="0,0,10,0"
Style="{DynamicResource MainTextBlockStyle}"/>
        <TextBox x:Name="FindData" Width="200"
Style="{DynamicResource MainTextBoxStyle}"
TextChanged="FindData_TextChanged"/>
        <Rectangle Margin="10,0,10,0" Style="{DynamicResource
SecondColorSeparatorsRectangleStyle}"/>
        <CheckBox x:Name="FindDateChecker" Margin="0,4,5,0"
Style="{DynamicResource MainCheckBoxStyle}"
Click="FindDateChecker_Click"/>
        <TextBlock Text="Дата сеанса:" Margin="0,0,10,0"
Style="{DynamicResource MainTextBlockStyle}"/>
        <DatePicker x:Name="FindSessionData"
Style="{DynamicResource SecondDatePickerStyle}"
SelectedDateChanged="FindSessionData_SelectedDateChanged"/>
        <Rectangle Margin="10,0,10,0" Style="{DynamicResource
SecondColorSeparatorsRectangleStyle}"/>
        <TextBlock Text="Найдено: " Style="{DynamicResource
MainTextBlockStyle}"/>
        <TextBlock x:Name="FindCounterData" Text="0/0"
Style="{DynamicResource MainTextBlockStyle}"/>
    </WrapPanel>
</Grid>

```

```

    <ListView x:Name="SessionList" Height="400"
d:ItemsSource="{d:SampleData ItemCount=2}"
ScrollViewer.CanContentScroll="False"
MouseDoubleClick="SessionList_MouseDoubleClick">
        <ListView.ItemContainerStyle>
            <Style TargetType="ListViewItem">
                <Setter Property="HorizontalContentAlignment"
Value="Stretch" />
            </Style>
        </ListView.ItemContainerStyle>
        <ListView.View>
            <GridView>
                <GridViewColumn x:Name="SessionCodeGrid">
                    <Label Content="Код сессии"
Style="{DynamicResource MainLableStyle}"/>
                    <GridViewColumn.CellTemplate>
                        <DataTemplate>
                            <Label Content="{Binding sessionID}"
HorizontalAlignment="Center" Style="{DynamicResource
MainLableStyle}"/>
                        </DataTemplate>
                    </GridViewColumn.CellTemplate>
                </GridViewColumn>
                <GridViewColumn x:Name="SessionCoverGrid">

```

```

                    <Label Content="Постер" Style="{DynamicResource
MainLableStyle}"/>
                    <GridViewColumn.CellTemplate>
                        <DataTemplate>
                            <Image Source="{Binding sessionMovieCover}"
MaxHeight="400" HorizontalAlignment="Center"/>
                        </DataTemplate>
                    </GridViewColumn.CellTemplate>
                </GridViewColumn>
                <GridViewColumn x:Name="SessionInfoGrid">
                    <Label Content="Информация о фильме"
Style="{DynamicResource MainLableStyle}"/>
                    <GridViewColumn.CellTemplate>
                        <DataTemplate>
                            <StackPanel>
                                <Label Content="{Binding
sessionMovieTitle}" Style="{DynamicResource MainLableStyle}"/>
                                <WrapPanel>
                                    <Label Content="Жанр(ы):"
Style="{DynamicResource MainLableStyle}"/>
                                    <Label Content="{Binding
sessionMovieGenre}" Style="{DynamicResource MainLableStyle}"/>
                                </WrapPanel>
                                <WrapPanel>
                                    <Label Content="Год публикации:"
Style="{DynamicResource MainLableStyle}"/>
                                    <Label Content="{Binding
sessionMovieYearOfPublication}" Style="{DynamicResource
MainLableStyle}"/>
                                </WrapPanel>
                                <WrapPanel>
                                    <Label Content="Хронометраж:"
Style="{DynamicResource MainLableStyle}"/>
                                    <Label Content="{Binding
sessionMovieTiming}" Style="{DynamicResource MainLableStyle}"/>
                                </WrapPanel>
                                <WrapPanel>
                                    <Label Content="Возрастной рейтинг:"
Style="{DynamicResource MainLableStyle}"/>
                                    <Label Content="{Binding
sessionMovieAgeRating}" Style="{DynamicResource
MainLableStyle}"/>
                                </WrapPanel>
                                <WrapPanel>
                                    <Label Content="Страна:"
Style="{DynamicResource MainLableStyle}"/>
                                    <Label Content="{Binding
sessionMovieCountry}" Style="{DynamicResource
MainLableStyle}"/>
                                </WrapPanel>
                                <Label Content="Актеры:"
Style="{DynamicResource MainLableStyle}"/>

```

```

        <TextBlock Text="{Binding
sessionActors}" Margin="5,0,0,0" VerticalAlignment="Center"
TextWrapping="Wrap" Style="{DynamicResource
MainTextBlockStyle}"/>
    </WrapPanel>
</StackPanel>
</DataTemplate>
</GridViewColumn.CellTemplate>
</GridViewColumn>
<GridViewColumn
x:Name="SessionInformationSeasonGrid">
    <Label Content="Информация"
Style="{DynamicResource MainLableStyle}"/>
    <GridViewColumn.CellTemplate>
        <DataTemplate>
            <StackPanel>
                <WrapPanel>
                    <Label Content="Дата начала сеанса:"
Style="{DynamicResource MainLableStyle}"/>
                    <Label Content="{Binding
sessionDateSessionStart}" Style="{DynamicResource
MainLableStyle}"/>
                </WrapPanel>
                <WrapPanel>
                    <Label Content="Время начала сеанса:"
Style="{DynamicResource MainLableStyle}"/>
                    <Label Content="{Binding
sessionTimeSessionStart}" Style="{DynamicResource
MainLableStyle}"/>
                </WrapPanel>
            </StackPanel>
        </DataTemplate>
    </GridViewColumn.CellTemplate>
</GridViewColumn>
<GridViewColumn x:Name="SessionPriceGrid">
    <Label Content="Цена билета"
Style="{DynamicResource MainLableStyle}"/>
    <GridViewColumn.CellTemplate>
        <DataTemplate>
            <Label Content="{Binding sessionTicketPrice}"
Style="{DynamicResource MainLableStyle}"/>
        </DataTemplate>
    </GridViewColumn.CellTemplate>
</GridViewColumn>
</GridView>
</ListView.View>
</ListView>
</StackPanel>
</Grid>
</Page>

```

SessionControls.xaml.cs

```
using System;
```

```

using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media.Imaging;
using static Cinema.Authorization;

```

```
namespace Cinema
```

```

{
    /// <summary>
    /// Логика взаимодействия для SessionControls.xaml
    /// </summary>
    public partial class SessionControls : Page
    {
        public SessionControls()
        {
            InitializeComponent();
        }

        public class SessionData
        {
            public int sessionID { get; set; }
            public BitmapImage sessionMovieCover { get; set; }
            public string sessionMovieTitle { get; set; }
            public string sessionMovieGenre { get; set; }
            public int sessionMovieYearOfPublication { get; set; }
            public double sessionMovieTiming { get; set; }
            public string sessionMovieAgeRating { get; set; }
            public string sessionMovieCountry { get; set; }
            public string sessionActors { get; set; }
            public string sessionDateSessionStart { get; set; }
            public string sessionTimeSessionStart { get; set; }
            public string sessionTicketPrice { get; set; }
        }
    }

```

```
List<SessionData> sessionDataList = new List<SessionData>();
```

```

private void Page_Loaded(object sender, RoutedEventArgs e)
{
    try
    {
        SessionList.ItemsSource = sessionDataList;

        LoadData(null, FindSessionData);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

```



```

private void LoadData(string findLine, DatePicker findDate)
{
    try
    {
        using (var dataBase = new CinemaEntities())
        {
            sessionDataList.Clear();

            DatePicker endDate = new DatePicker();
            endDate.SelectedDate = null;

            if (findDate.SelectedDate != null)
            {
                endDate.SelectedDate =
findDate.SelectedDate.Value.AddDays(1);
            }
            else
            {
                endDate.SelectedDate = null;
            }

            var sessionData = (from session in dataBase.Session
                                join
                                movie in dataBase.Movie on session.IDMovie
equals movie.IDMovie into movieGroup
                                from movie in movieGroup.DefaultIfEmpty()
                                join
                                actorsInMovies in dataBase.actorsInMovies on
movie.IDMovie equals actorsInMovies.IDMovie into
actorsInMoviesGroup
                                from actorsInMovie in
actorsInMoviesGroup.DefaultIfEmpty()
                                join
                                actors in dataBase.Actor on
actorsInMovie.IDActor equals actors.IDActor into actorsGroup
                                from actors in actorsGroup.DefaultIfEmpty()
                                join
                                movieGenre in dataBase.MovieGenre on
movie.IDMovie equals movieGenre.IDMovie into movieGenreGroup
                                from movieGenre in
movieGenreGroup.DefaultIfEmpty()
                                join
                                genre in dataBase.Genre on
movieGenre.IDGenre equals genre.IDGenre into genreGroup
                                from genre in genreGroup.DefaultIfEmpty()
                                join
                                country in dataBase.Country on
movie.IDCountry equals country.IDCountry into countryGroup
                                from country in countryGroup.DefaultIfEmpty()
                                where ((findLine == null ||
movie.Title.Contains(findLine) || genre.Title.Contains(findLine) ||
movie.YearOfPublication.ToString().Contains(findLine) ||

```

```

movie.Description.Contains(findLine) ||
country.Title.Contains(findLine)) && ((findDate.SelectedDate == null
&& endDate.SelectedDate == null) || (session.DateAndTimeSession >=
findDate.SelectedDate.Value && session.DateAndTimeSession <=
endDate.SelectedDate.Value)))

            select new
            {
                session.IDSession,
                movie.Cover,
                movieTitle = movie.Title,
                movieGenre = genre.Title,
                movieActors = actors.Surname + " " +
actors.Name + " " + actors.Patronymic + " " + actors.Nickname,
                movie.YearOfPublication,
                movie.Timing,
                movie.AgeRating,
                movieCountry = country.Title,
                session.DateAndTimeSession,
                session.TicketPrice
            }).ToList();

            var groupedSessionData = sessionData.GroupBy(m =>
m.IDSession)
                .Select(g => new
                {
                    g.Key,
                    genres = g.Select(m => m.movieGenre).Distinct(),
                    actors = g.Select(m => m.movieActors).Distinct(),
                }).ToList();

            var result = groupedSessionData.Select(g =>
            {
                var session = sessionData.FirstOrDefault(m =>
m.IDSession == g.Key);
                return new
                {
                    session.IDSession,
                    session.Cover,
                    session.movieTitle,
                    sessionMovieGenres = g.genres,
                    session.YearOfPublication,
                    session.Timing,
                    session.AgeRating,
                    session.movieCountry,
                    session.DateAndTimeSession,
                    session.TicketPrice,
                    movieActors = g.actors,
                };
            }).ToList();

            foreach (var sessionLine in result)
            {
                SessionData sessionDataClass = new SessionData();

```

```

        sessionDataClass.sessionID = sessionLine.IDSession;

        if (sessionLine.Cover != null)
        {
            BitmapImage cover = new BitmapImage();

            cover.BeginInit();
            cover.StreamSource = new
MemoryStream(sessionLine.Cover);
            cover.EndInit();

            sessionDataClass.sessionMovieCover = cover;
        }
        else
        {
            sessionDataClass.sessionMovieCover = new
BitmapImage(new Uri("pack://application:,,,/Resource/NoImage.png",
UriKind.Absolute));
        }

        sessionDataClass.sessionMovieTitle =
sessionLine.movieTitle;

        string sessionMovieGenreTemp = "";
        int movieGenresCounterTemp = 1;
        foreach (var genere in sessionLine.sessionMovieGenres)
        {
            if (movieGenresCounterTemp !=
sessionLine.sessionMovieGenres.Count())
                sessionMovieGenreTemp += genere + ", ";
            else
                sessionMovieGenreTemp += genere;

            movieGenresCounterTemp++;
        }

        sessionDataClass.sessionMovieGenre =
sessionMovieGenreTemp;

        string movieActorTemp = "";
        int movieActorCounterTemp = 1;
        foreach (var actor in sessionLine.movieActors)
        {
            if (movieActorCounterTemp !=
sessionLine.movieActors.Count())
            {
                movieActorTemp += actor + ", ";
            }
            else
            {
                movieActorTemp += actor;
            }
        }

        movieActorCounterTemp++;
    }
    sessionDataClass.sessionActors = movieActorTemp;

    sessionDataClass.sessionMovieYearOfPublication =
sessionLine.YearOfPublication;
    sessionDataClass.sessionMovieTiming =
sessionLine.Timing;
    sessionDataClass.sessionMovieAgeRating =
sessionLine.AgeRating + "+";
    sessionDataClass.sessionMovieCountry =
sessionLine.movieCountry;
    sessionDataClass.sessionDateSessionStart =
sessionLine.DateAndTimeSession.ToString().Split(' ')[0];
    sessionDataClass.sessionTimeSessionStart =
sessionLine.DateAndTimeSession.ToString().Split(' ')[1];
    sessionDataClass.sessionTicketPrice =
sessionLine.TicketPrice.ToString().Remove(sessionLine.TicketPrice.To
String().Length - 2, 2) + " руб.";

    sessionDataList.Add(sessionDataClass);
}

var fullSessionData = dataBase.Session.ToList();
FindCounterData.Text = result.Count() + "/" +
fullSessionData.Count();

SessionList.Items.Refresh();
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}
}

private void Page_SizeChanged(object sender,
SizeChangedEventArgs e)
{
    try
    {
        SessionCodeGrid.Width = 0;
        SessionCoverGrid.Width = ActualWidth - ActualWidth * 0.8;
        SessionInfoGrid.Width = ActualWidth - ActualWidth * 0.6;
        SessionInformationSeasonGrid.Width = ActualWidth -
ActualWidth * 0.8;
        SessionPriceGrid.Width = ActualWidth - ActualWidth * 0.8;
        SessionList.Height = ActualHeight - 35;
    }
    catch (Exception ex)
    {

```

```

        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void FindData_TextChanged(object sender,
TextChangedEventArgs e)
{
    try
    {
        if (FindDateChecker.IsChecked == false)
        {
            FindSessionData.SelectedDate = null;
        }
        LoadData(FindData.Text, FindSessionData);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void FindDateChecker_Click(object sender,
RoutedEventArgs e)
{
    try
    {
        if (FindDateChecker.IsChecked == false)
        {
            FindSessionData.IsEnabled = false;
            FindSessionData.SelectedDate = null;
        }
        else
        {
            FindSessionData.IsEnabled = true;
            FindSessionData.SelectedDate = DateTime.Now.Date;
        }
        LoadData(FindData.Text, FindSessionData);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void FindSessionData_SelectedDateChanged(object sender,
SelectionChangedEventArgs e)
{
    try
    {
        if (FindDateChecker.IsChecked == false)

```

```

    {
        FindSessionData.SelectedDate = null;
    }
    LoadData(FindData.Text, FindSessionData);
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
    MessageBoxButton.OK, MessageBoxImage.Error);
}

private void SessionList_MouseDoubleClick(object sender,
MouseButtonEventArgs e)
{
    try
    {
        var selectedSession = SessionList.SelectedItem as
        SessionData;
        if (selectedSession != null)
        {
            TransmittedData.idSelectedSession =
            selectedSession.sessionID;

            AddOrEditSession addOrEditSession = new
            AddOrEditSession();
            addOrEditSession.ShowDialog();

            if (FindDateChecker.IsChecked == false)
            {
                FindSessionData.SelectedDate = null;
            }
            LoadData(FindData.Text, FindSessionData);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void AddSession_Click(object sender, RoutedEventArgs e)
{
    try
    {
        TransmittedData.idSelectedSession = -1;

        AddOrEditSession addOrEditSession = new
        AddOrEditSession();
        addOrEditSession.ShowDialog();

        if (FindDateChecker.IsChecked == false)

```

```

        {
            FindSessionData.SelectedDate = null;
        }
        LoadData(FindData.Text, FindSessionData);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void RemoveSession_Click(object sender,
RoutedEventArgs e)
{
    try
    {
        var selectedSession = SessionList.SelectedItem as
SessionData;
        if (selectedSession != null)
        {
            if (MessageBox.Show("Вы действительно хотите удалить
данный сеанс?", "Внимание", MessageBoxButton.YesNo,
MessageBoxImage.Warning) == MessageBoxResult.Yes)
            {
                using (var dataBase = new CinemaEntities())
                {
                    var removeSession = dataBase.Session.Where(w =>
w.IDSession == selectedSession.sessionID).FirstOrDefault();
                    var removeTicket = dataBase.Ticket.Where(w =>
w.IDSession == selectedSession.sessionID).ToList();

                    if (removeTicket.Count > 0)
                    {
                        if (MessageBox.Show("Удаляемый сеанс
присутствует в билетах. Вы действительно хотите его удалить?",
"Внимание", MessageBoxButton.YesNo, MessageBoxImage.Warning)
== MessageBoxResult.No)
                            return;
                    }

                    dataBase.Ticket.RemoveRange(removeTicket);
                    dataBase.Session.Remove(removeSession);

                    dataBase.SaveChanges();
                    MessageBox.Show("Данные были удалены",
"Готово", MessageBoxButton.OK, MessageBoxImage.Information);
                }

                if (FindDateChecker.IsChecked == false)
                {
                    FindSessionData.SelectedDate = null;
                }
            }
        }
    }
}

```

```

        LoadData(FindData.Text, FindSessionData);
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
    MessageBoxButton.OK, MessageBoxImage.Error);
}
}
}

```

SettingsControls.xaml

```

<Page x:Class="Cinema.Pages.SettingsControls"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:local="clr-namespace:Cinema.Pages"
mc:Ignorable="d"
d:DesignHeight="450" d:DesignWidth="800"
Title="SettingsControls" SizeChanged="Page_SizeChanged">

    <Grid Style="{DynamicResource MainColorStyle}">
        <StackPanel x:Name="Hall" Margin="0,0,198,0"
VerticalAlignment="Center" HorizontalAlignment="Center"
SizeChanged="Hall_SizeChanged">
            <Grid VerticalAlignment="Center" HorizontalAlignment="Right"
Width="200">
                <Rectangle Style="{DynamicResource
SecondColorRectanglePlaceStyle}" />
                <StackPanel>
                    <TextBlock Margin="0,10,0,0" Text="Настройка зала"
Style="{DynamicResource TitleTextBlockStyle}" />
                    <Grid Margin="0,0,0,10" HorizontalAlignment="Center">
                        <Label Content="Ряды" Style="{DynamicResource
MainLableStyle}" />
                        <ComboBox x:Name="SelectedRowHall"
Margin="70,0,0,0" HorizontalAlignment="Left" Width="100"
Style="{DynamicResource MainComboBoxStyle}"
ItemContainerStyle="{DynamicResource
MainSelectedComboBoxStyle}"
SelectionChanged="SelectedRowHall_SelectionChanged" />
                    </Grid>
                    <Grid Margin="0,0,0,10" HorizontalAlignment="Center">
                        <Label Content="Местра" FontSize="16" />
                        <ComboBox x:Name="SelectedPlaceHall"
Margin="70,0,0,0" HorizontalAlignment="Left" Width="100"
Style="{DynamicResource MainComboBoxStyle}"
ItemContainerStyle="{DynamicResource

```

```

MainSelectedComboBoxStyle}"
SelectionChanged="SelectedPlaceHall_SelectionChanged"/>
</Grid>
<Button x:Name="SaveHallSettings" Content="Сохранить"
Margin="10,0,10,10" Style="{DynamicResource
TabControlButtonStyle}" Click="SaveHallSettings_Click"/>
</StackPanel>
</Grid>
</Grid>
</Page>

```

SettingsControls.xaml.cs

```

using System;
using System.Collections.Generic;
using
System.Data.Entity.Core.Common.CommandTrees.ExpressionBuilder;
using System.Linq;
using System.Text.RegularExpressions;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;

namespace Cinema.Pages
{
    /// <summary>
    /// Логика взаимодействия для SettingsControls.xaml
    /// </summary>
    public partial class SettingsControls : Page
    {
        public SettingsControls()
        {
            InitializeComponent();
            LoadData();
        }

        public int selectedRowHall;
        public int selectedPlaceHall;
        public List<string> hidePlace = new List<string>();

        private void LoadData()
        {
            try
            {
                List<int> numericRowList = new List<int>();
                List<int> numericPlaceList = new List<int>();

                for (int i = 1; i < 21; i++)
                {
                    numericRowList.Add(i);
                }

                for (int i = 1; i < 31; i++)
                {

```

```

                    numericPlaceList.Add(i);
                }

                SelectedRowHall.ItemsSource = numericRowList;
                SelectedPlaceHall.ItemsSource = numericPlaceList;

                using (var dataBase = new CinemaEntities())
                {
                    var loadData = dataBase.Settings.OrderByDescending(o =>
                        o.DateTimeChange).FirstOrDefault();

                    if (loadData != null)
                    {
                        SelectedRowHall.SelectedItem = loadData.RowHall;
                        SelectedPlaceHall.SelectedItem = loadData.PlaceHall;
                        if (loadData.HiddenPlaces != null)
                        {
                            CreateHall(loadData.HiddenPlaces.Split('|'));
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Ошибка",
                    MessageBoxButton.OK, MessageBoxImage.Error);
            }

            private void Page_SizeChanged(object sender,
                SizeChangedEventArgs e)
            {
                ResizeWindows();
            }

            private void Hall_SizeChanged(object sender,
                SizeChangedEventArgs e)
            {
                ResizeWindows();
            }

            private void ResizeWindows()
            {
                try
                {
                    double maxWidth = this.ActualWidth - 10;
                    double maxHeight = this.ActualHeight - 10;

                    double scale = this.ActualWidth / (Hall.ActualWidth * 1.28);
                    double centerX = Hall.ActualWidth / 2;
                    double centerY = Hall.ActualHeight / 2;

                    double newWidth = Hall.ActualWidth * scale;

```

```

double newHeight = Hall.ActualHeight * scale;

if (newWidth > maxWidth)
{
    scale = maxWidth / Hall.ActualWidth;
}
if (newHeight > maxHeight)
{
    scale = Math.Min(scale, maxHeight / Hall.ActualHeight);
}

ScaleTransform scaleTransform = new ScaleTransform(scale,
scale, centerX, centerY);
Hall.RenderTransform = scaleTransform;
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
    MessageBoxButton.OK, MessageBoxImage.Error);
}

private void SelectedRowHall_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{
    try
    {
        selectedRowHall = (int)SelectedRowHall.SelectedItem;
        CreateHall(null);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void SelectedPlaceHall_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{
    try
    {
        selectedPlaceHall = (int)SelectedPlaceHall.SelectedItem;
        CreateHall(null);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void CreateHall(string[] hidePlaceLoadData)
{
    try
    {
        if (selectedRowHall != 0 && selectedPlaceHall != 0)
        {
            Hall.Children.Clear();
            hidePlace.Clear();

            for (int row = 0; row < selectedRowHall; row++)
            {
                WrapPanel wrapPanel = new WrapPanel();

                TextBlock textBlockBegin = new TextBlock();
                textBlockBegin.Text = $"Ряд {row + 1}";
                textBlockBegin.FontSize = 10;
                textBlockBegin.Margin = new Thickness(0, 0, 5, 0);
                textBlockBegin.VerticalAlignment =
                VerticalAlignment.Center;

                TextBlock textBlockEnd = new TextBlock();
                textBlockEnd.Text = $"Ряд {row + 1}";
                textBlockEnd.FontSize = 10;
                textBlockEnd.VerticalAlignment =
                VerticalAlignment.Center;

                wrapPanel.Children.Add(textBlockBegin);
                for (int place = 1; place <= selectedPlaceHall; place++)
                {
                    Button button = new Button();
                    button.Content = place;
                    button.Name = $"Row{row}Place{place}";
                    button.Click += HidePlace_Click;
                    button.Width = 18;
                    button.Height = 18;
                    button.FontSize = 10;
                    button.Background = Brushes.White;
                    button.Margin = new Thickness(0, 0, 5, 0);

                    if (hidePlaceLoadData != null)
                    {
                        foreach (var hidePlaceLine in hidePlaceLoadData)
                        {
                            Match match = Regex.Match(hidePlaceLine,
                            @"Row(\d+)Place(\d+)");

                            if (match.Success)
                            {
                                if (row + 1 ==
                                int.Parse(match.Groups[1].Value) + 1 && place ==
                                int.Parse(match.Groups[2].Value))
                                {
                                    button.Tag = button.Name;
                                    button.Content = "";
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

        button.Opacity = 0.2;
        hidePlace.Add(button.Tag.ToString());
    }
}
}

wrapPanel.Children.Add(button);
}
wrapPanel.Children.Add(textBlockEnd);

StackPanel stackPanel = new StackPanel();
stackPanel.Margin = new Thickness(0, 0, 0, 5);
stackPanel.HorizontalAlignment =
HorizontalAlignment.Center;
stackPanel.Children.Add(wrapPanel);

Hall.Children.Add(stackPanel);
}
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
    MessageBoxButton.OK, MessageBoxImage.Error);
}

private void HidePlace_Click(object sender, RoutedEventArgs e)
{
    try
    {
        Button button = (Button)sender;

        if (button.Content.ToString() != "")
        {
            button.Tag = button.Name;
            button.Content = "";
            button.Opacity = 0.2;
            hidePlace.Add(button.Tag.ToString());
        }
        else
        {
            button.Name = button.Tag.ToString();
            button.Content = int.Parse(Regex.Match(button.Name,
@"Row(\d+)Place(\d+)").Groups[2].Value);
            button.Tag = null;
            button.Opacity = 1;
            hidePlace.Remove(hidePlace.FirstOrDefault(w => w ==
button.Name.ToString()));
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void SaveHallSettings_Click(object sender,
RoutedEventArgs e)
{
    try
    {
        using (var dataBase = new CinemaEntities())
        {
            var newHall = new Settings();

            newHall.RowHall = (int)SelectedRowHall.SelectedItem;
            newHall.PlaceHall = (int)SelectedPlaceHall.SelectedItem;
            newHall.DateTimeChange = DateTime.Now;

            string hidePlaceTemp = null;
            foreach (var hidePlaceLine in hidePlace)
            {
                if (hidePlaceLine != hidePlace.LastOrDefault())
                {
                    hidePlaceTemp += hidePlaceLine + "|";
                }
                else
                {
                    hidePlaceTemp += hidePlaceLine;
                }
            }

            newHall.HiddenPlaces = hidePlaceTemp;

            dataBase.Settings.Add(newHall);
            dataBase.SaveChanges();

            MessageBox.Show("Настройка зала была сохранена",
            "Готово", MessageBoxButton.OK, MessageBoxImage.Information);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
}

TicketAnalysis.xaml
<Page x:Class="Cinema.TicketAnalysis"

```

```

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:Cinema"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="TicketAnalysis" SizeChanged="Page_SizeChanged"
    Loaded="Page_Loaded">

    <Grid Style="{DynamicResource MainColorStyle}">
        <StackPanel>
            <Grid Style="{DynamicResource SecondColorStyle}">
                <WrapPanel VerticalAlignment="Center" Margin="10,0,0,0">
                    <TextBlock Text="Поиск:" Margin="0,0,10,0"
Style="{DynamicResource MainTextBlockStyle}" />
                    <TextBox x:Name="FindData" Width="200"
Style="{DynamicResource MainTextBoxStyle}"
TextChanged="FindData_TextChanged" />
                    <Rectangle Margin="10,0,10,0" Style="{DynamicResource
SecondColorSeparatorsRectangleStyle}" />
                    <CheckBox x:Name="FindDateChecker" Margin="0,4,5,0"
Style="{DynamicResource MainCheckBoxStyle}"
Click="FindDateChecker_Click" />
                    <TextBlock Text="C:" Margin="0,0,10,0"
Style="{DynamicResource MainTextBlockStyle}" />
                    <DatePicker x:Name="BeginDate" Margin="0,0,10,0"
Style="{DynamicResource SecondDatePickerStyle}"
SelectedDateChanged="BeginDate_SelectedDateChanged" />
                    <TextBlock Text="по" Margin="0,0,10,0"
Style="{DynamicResource MainTextBlockStyle}" />
                    <DatePicker x:Name="EndDate"
Style="{DynamicResource SecondDatePickerStyle}"
SelectedDateChanged="EndDate_SelectedDateChanged" />
                    <Rectangle Margin="10,0,10,0" Style="{DynamicResource
SecondColorSeparatorsRectangleStyle}" />
                    <TextBlock Text="Найдено: " Style="{DynamicResource
MainTextBlockStyle}" />
                    <TextBlock x:Name="FindCounterData" Text="0/0"
Style="{DynamicResource MainTextBlockStyle}" />
                </WrapPanel>
                <WrapPanel HorizontalAlignment="Right"
VerticalAlignment="Center" Margin="0,0,10,0">
                    <Button x:Name="Print" Content="Печать отчета"
Width="120" Style="{DynamicResource TabControlButtonStyle}"
Click="Print_Click" />
                </WrapPanel>
            </Grid>

            <ListView x:Name="TicketList" Height="400"
d:ItemsSource="{d:SampleData ItemCount=2}"

```

```

ScrollViewer.CanContentScroll="False"
MouseDownClick="TicketList_MouseDoubleClick">
        <ListView.ItemContainerStyle>
            <Style TargetType="ListViewItem">
                <Setter Property="HorizontalContentAlignment"
Value="Stretch" />
            </Style>
        </ListView.ItemContainerStyle>
    </ListView.View>
    <GridView>
        <GridViewColumn x:Name="MovieCodeGrid">
            <Label Content="Код фильма"
Style="{DynamicResource MainLableStyle}" />
            <GridViewColumn.CellTemplate>
                <DataTemplate>
                    <Label Content="{Binding ticketMovieID}"
HorizontalAlignment="Center" Style="{DynamicResource
MainLableStyle}" />
                </DataTemplate>
            </GridViewColumn.CellTemplate>
        </GridViewColumn>
        <GridViewColumn x:Name="ImageGrid">
            <Label Content="Постер" Style="{DynamicResource
MainLableStyle}" />
            <GridViewColumn.CellTemplate>
                <DataTemplate>
                    <Image Source="{Binding ticketMovieCover}"
MaxHeight="400" HorizontalAlignment="Center" />
                </DataTemplate>
            </GridViewColumn.CellTemplate>
        </GridViewColumn>
        <GridViewColumn x:Name="MovieInfoGrid">
            <Label Content="Информация о фильме"
Style="{DynamicResource MainLableStyle}" />
            <GridViewColumn.CellTemplate>
                <DataTemplate>
                    <StackPanel>
                        <Label Content="{Binding ticketMovieTitle}"
Style="{DynamicResource MainLableStyle}" />
                        <WrapPanel>
                            <Label Content="Жанр(ы):"
Style="{DynamicResource MainLableStyle}" />
                            <Label Content="{Binding
ticketMovieGenre}" Style="{DynamicResource MainLableStyle}" />
                        </WrapPanel>
                        <WrapPanel>
                            <Label Content="Год публикации:"
Style="{DynamicResource MainLableStyle}" />
                            <Label Content="{Binding
ticketMovieYearOfPublication}" Style="{DynamicResource
MainLableStyle}" />
                        </WrapPanel>
                    </StackPanel>
                </DataTemplate>
            </GridViewColumn>
        </GridView>
    </ListView>

```



```

        <Label Content="Хронометраж:"
Style="{DynamicResource MainLableStyle}"/>
        <Label Content="{Binding
ticketMovieTiming}" Style="{DynamicResource MainLableStyle}"/>
    </WrapPanel>
    <WrapPanel>
        <Label Content="Возрастной рейтинг:"
Style="{DynamicResource MainLableStyle}"/>
        <Label Content="{Binding
ticketMovieAgeRating}" Style="{DynamicResource
MainLableStyle}"/>
    </WrapPanel>
    <WrapPanel>
        <Label Content="Страна:"
Style="{DynamicResource MainLableStyle}"/>
        <Label Content="{Binding
ticketMovieCountry}" Style="{DynamicResource MainLableStyle}"/>
    </WrapPanel>
    <WrapPanel>
        <Label Content="Актеры:"
Style="{DynamicResource MainLableStyle}"/>
        <TextBlock Text="{Binding
ticketMovieActors}" Margin="5,0,0,0" VerticalAlignment="Center"
TextWrapping="Wrap" Style="{DynamicResource
MainTextBlockStyle}"/>
    </WrapPanel>
</StackPanel>
</DataTemplate>
</GridViewColumn.CellTemplate>
</GridViewColumn>
<GridViewColumn x:Name="AnalyticsGrid">
    <Label Content="Аналитика продаж"
Style="{DynamicResource MainLableStyle}"/>
    <GridViewColumn.CellTemplate>
        <DataTemplate>
            <StackPanel>
                <WrapPanel>
                    <Label Content="Количество проданных
билетов:" Style="{DynamicResource MainLableStyle}"/>
                    <Label Content="{Binding ticketCount}"
Style="{DynamicResource MainLableStyle}"/>
                </WrapPanel>
                <WrapPanel>
                    <Label Content="Суммарные сборы:"
Style="{DynamicResource MainLableStyle}"/>
                    <Label Content="{Binding
ticketSummaryCost}" Style="{DynamicResource MainLableStyle}"/>
                </WrapPanel>
            </StackPanel>
        </DataTemplate>
    </GridViewColumn.CellTemplate>
</GridViewColumn>
</GridView>

```

```

    </ListView.View>
</ListView>
</StackPanel>
</Grid>
</Page>

```

TicketAnalysis.xaml.cs

```

using iTextSharp.text;
using iTextSharp.text.pdf;
using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media.Imaging;
using static Cinema.Authorization;

namespace Cinema
{
    /// <summary>
    /// Логика взаимодействия для TicketAnalysis.xaml
    /// </summary>
    public partial class TicketAnalysis : Page
    {
        public TicketAnalysis()
        {
            InitializeComponent();
        }

        public class TicketData
        {
            public int ticketMovieID { get; set; }
            public BitmapImage ticketMovieCover { get; set; }
            public string ticketMovieTitle { get; set; }
            public string ticketMovieGenre { get; set; }
            public int ticketMovieYearOfPublication { get; set; }
            public double ticketMovieTiming { get; set; }
            public string ticketMovieAgeRating { get; set; }
            public string ticketMovieCountry { get; set; }
            public string ticketMovieActors { get; set; }
            public string ticketCount { get; set; }
            public string ticketSummaryCost { get; set; }
        }

        List<TicketData> ticketDataList = new List<TicketData>();

        private void Page_Loaded(object sender, RoutedEventArgs e)
        {
            try

```

```

{
    TicketList.ItemsSource = ticketDataList;

    LoadData(null, BeginDate, EndDate);
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
}

private void LoadData(string findLine, DatePicker beginDate,
    DatePicker endDate)
{
    try
    {
        using (var dataBase = new CinemaEntities())
        {
            ticketDataList.Clear();

            var ticketData = (from ticket in dataBase.Ticket
                join
                    session in dataBase.Session on ticket.IDSession
equals session.IDSession into SessionGroup
                from session in SessionGroup.DefaultIfEmpty()
                join
                    movie in dataBase.Movie on session.IDMovie
equals movie.IDMovie into movieGroup
                from movie in movieGroup.DefaultIfEmpty()
                join
                    actorsInMovies in dataBase.ActorsInMovies on
movie.IDMovie equals actorsInMovies.IDMovie into
actorsInMoviesGroup
                from actorsInMovie in
actorsInMoviesGroup.DefaultIfEmpty()
                join
                    actors in dataBase.Actor on
actorsInMovie.IDActor equals actors.IDActor into actorsGroup
                from actors in actorsGroup.DefaultIfEmpty()
                join
                    movieGenre in dataBase.MovieGenre on
movie.IDMovie equals movieGenre.IDMovie into movieGenreGroup
                from movieGenre in
movieGenreGroup.DefaultIfEmpty()
                join
                    genre in dataBase.Genre on
movieGenre.IDGenre equals genre.IDGenre into genreGroup
                from genre in genreGroup.DefaultIfEmpty()
                join
                    country in dataBase.Country on
movie.IDCountry equals country.IDCountry into countryGroup
                from country in countryGroup.DefaultIfEmpty()

```

```

                where ((findLine == null ||
                    movie.Title.Contains(findLine) || genre.Title.Contains(findLine) ||
                    movie.YearOfPublication.ToString().Contains(findLine) ||
                    movie.Description.Contains(findLine) ||
                    country.Title.Contains(findLine)) && ((beginDate.SelectedDate == null
                        || session.DateAndTimeSession > beginDate.SelectedDate.Value) &&
                    (endDate.SelectedDate == null || session.DateAndTimeSession <
                        endDate.SelectedDate.Value))))
                select new
                {
                    movie.IDMovie,
                    movie.Cover,
                    movieTitle = movie.Title,
                    movieGenre = genre.Title,
                    movie.YearOfPublication,
                    movie.Timing,
                    movie.AgeRating,
                    movieCountry = country.Title,
                    movieActors = actors.Surname + " " +
                        actors.Name + " " + actors.Patronymic + " " + actors.Nickname,
                    session.TicketPrice
                }
            ).ToList();

            var groupedMovieData = ticketData.GroupBy(m =>
                m.IDMovie)
                .Select(g => new
                {
                    g.Key,
                    genres = g.Select(m => m.movieGenre).Distinct(),
                    actors = g.Select(m => m.movieActors).Distinct(),
                    TicketCount = g.Count(),
                    TotalCost = g.Sum(t => t.TicketPrice)
                }).ToList();

            var result = groupedMovieData.Select(g =>
            {
                var movie = ticketData.FirstOrDefault(m => m.IDMovie
                    == g.Key);

                return new
                {
                    movie.IDMovie,
                    movie.Cover,
                    movie.movieTitle,
                    movieGenres = g.genres,
                    movie.YearOfPublication,
                    movie.Timing,
                    movie.AgeRating,
                    movie.movieCountry,
                    movieActors = g.actors,
                    g.TicketCount,
                    g.TotalCost,
                    movie.TicketPrice,

```

```

    };
    }).ToList();

    foreach (var ticketLine in result)
    {
        TicketData ticketDataClass = new TicketData();

        ticketDataClass.ticketMovieID = ticketLine.IDMovie;

        if (ticketLine.Cover != null)
        {
            BitmapImage cover = new BitmapImage();

            cover.BeginInit();
            cover.StreamSource = new
MemoryStream(ticketLine.Cover);
            cover.EndInit();

            ticketDataClass.ticketMovieCover = cover;
        }
        else
        {
            ticketDataClass.ticketMovieCover = new
BitmapImage(new Uri("pack://application:,,,/Resource/NoImage.png",
UriKind.Absolute));
        }

        ticketDataClass.ticketMovieTitle =
ticketLine.movieTitle;

        string movieGenreTemp = "";
        int movieGenresCounterTemp = 1;
        foreach (var genre in ticketLine.movieGenres)
        {
            if (movieGenresCounterTemp !=
ticketLine.movieGenres.Count())
                movieGenreTemp += genre + ", ";
            else
                movieGenreTemp += genre;

            movieGenresCounterTemp++;
        }
        ticketDataClass.ticketMovieGenre = movieGenreTemp;

        string movieActorTemp = "";
        int movieActorCounterTemp = 1;
        foreach (var actor in ticketLine.movieActors)
        {
            if (movieActorCounterTemp !=
ticketLine.movieActors.Count())
            {
                movieActorTemp += actor + ", ";
            }
        }

        else
        {
            movieActorTemp += actor;
        }

        movieActorCounterTemp++;
    }
    ticketDataClass.ticketMovieActors = movieActorTemp;

    ticketDataClass.ticketMovieYearOfPublication =
ticketLine.YearOfPublication;
    ticketDataClass.ticketMovieTiming = ticketLine.Timing;
    ticketDataClass.ticketMovieAgeRating =
ticketLine.AgeRating + "+";
    ticketDataClass.ticketMovieCountry =
ticketLine.movieCountry;

    ticketDataClass.ticketCount = (ticketLine.TicketCount /
((movieGenresCounterTemp - 1) * (movieActorCounterTemp -
1))).ToString();
    ticketDataClass.ticketSummaryCost =
(ticketLine.TotalCost / ((movieGenresCounterTemp - 1) *
(movieActorCounterTemp -
1))).ToString().Remove((ticketLine.TotalCost /
((movieGenresCounterTemp - 1) * (movieActorCounterTemp -
1))).ToString().Length - 2, 2) + " py6.";

    ticketDataList.Add(ticketDataClass);
}

var fullMovieData = (from ticket in dataBase.Ticket
join
    session in dataBase.Session on
ticket.IDSession equals session.IDSession into sessionGroup
from session in
    sessionGroup.DefaultIfEmpty()
join
    movie in dataBase.Movie on session.IDMovie
equals movie.IDMovie into movieGroup
from movie in movieGroup.DefaultIfEmpty()
group ticket by movie into movieTickets
where movieTickets.Count() > 1
select new
{
    Movie = movieTickets.Key,
    TicketCount = movieTickets.Count()
}).ToList();

FindCounterData.Text = result.Count() + "/" +
fullMovieData.Count();

TicketList.Items.Refresh();
}

```

```

    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void Page_SizeChanged(object sender,
SizeChangedEventArgs e)
{
    try
    {
        MovieCodeGrid.Width = 0;
        ImageGrid.Width = ActualWidth - ActualWidth * 0.8;
        MovieInfoGrid.Width = ActualWidth - ActualWidth * 0.6;
        AnalyticsGrid.Width = ActualWidth - ActualWidth * 0.6;
        TicketList.Height = ActualHeight - 35;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void BeginDate_SelectedDateChanged(object sender,
SelectionChangedEventArgs e)
{
    LoadData(FindData.Text, BeginDate, EndDate);
}

private void EndDate_SelectedDateChanged(object sender,
SelectionChangedEventArgs e)
{
    LoadData(FindData.Text, BeginDate, EndDate);
}

private void FindDateChecker_Click(object sender,
RoutedEventArgs e)
{
    try
    {
        if (FindDateChecker.IsChecked == false)
        {
            BeginDate.IsEnabled = false;
            BeginDate.SelectedDate = null;
            EndDate.IsEnabled = false;
            EndDate.SelectedDate = null;
        }
        else
        {
            BeginDate.IsEnabled = true;
            BeginDate.SelectedDate = DateTime.Now;
            EndDate.IsEnabled = true;
            EndDate.SelectedDate = DateTime.Now;
        }

        LoadData(FindData.Text, BeginDate, EndDate);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void FindData_TextChanged(object sender,
TextChangedEventArgs e)
{
    LoadData(FindData.Text, BeginDate, EndDate);
}

private void TicketList_MouseDoubleClick(object sender,
MouseButtonEventArgs e)
{
    try
    {
        var selectedSession = TicketList.SelectedItem as TicketData;
        if (selectedSession != null)
        {
            TransmittedData.idSelectedMovieAnalysis =
            selectedSession.ticketMovieID;

            Window activeWindow = Window.GetWindow(this);

            if (activeWindow.Title == "Панель директора")
            {
                DirectorsPanel directorsPanel =
                Window.GetWindow(this) as DirectorsPanel;
                directorsPanel.MovieAnalysisOpen();
            }
            else
            if (activeWindow.Title == "Панель менеджера")
            {
                BookerPanel bookerPanel = Window.GetWindow(this)
                as BookerPanel;
                bookerPanel.MovieAnalysisOpen();
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

```

```

    }

    private void Print_Click(object sender, RoutedEventArgs e)
    {
        try
        {
            using (var dataBase = new CinemaEntities())
            {
                string findLine = FindData.Text;

                SaveFileDialog saveFileDialog = new SaveFileDialog();
                saveFileDialog.Filter = "PDF files (*.pdf)|*.pdf";

                if (saveFileDialog.ShowDialog() == true)
                {
                    Document doc = new Document();
                    PdfWriter writer = PdfWriter.GetInstance(doc, new
FileStream($"{saveFileDialog.FileName}", FileMode.Create));
                    doc.SetPageSize(PageSize.A4.Rotate());

                    BaseFont baseFont =
BaseFont.CreateFont("C:\\Windows\\Fonts\\arial.ttf",
BaseFont.IDENTITY_H, BaseFont.NOT_EMBEDDED);
                    Font font = new Font(baseFont, 12);

                    BaseFont baseFontHead =
BaseFont.CreateFont("C:\\Windows\\Fonts\\arial.ttf",
BaseFont.IDENTITY_H, BaseFont.NOT_EMBEDDED);
                    Font fontHead = new Font(baseFont, 20, Font.BOLD);

                    doc.Open();

                    Paragraph mainParagraph = new Paragraph("Отчет
продаж", fontHead);
                    mainParagraph.Alignment = Element.ALIGN_CENTER;
                    doc.Add(mainParagraph);

                    PdfPTable table = new PdfPTable(10);
                    table.SpacingBefore = 10;
                    table.WidthPercentage = 100;

                    table.AddCell(new Paragraph("Код фильма", font));
                    table.AddCell(new Paragraph("Название фильма",
font));
                    table.AddCell(new Paragraph("Жанры", font));
                    table.AddCell(new Paragraph("Год публикации", font));
                    table.AddCell(new Paragraph("Длительность фильма",
font));
                    table.AddCell(new Paragraph("Возрастной рейтинг",
font));
                    table.AddCell(new Paragraph("Страна", font));
                    table.AddCell(new Paragraph("Актеры", font));

                    table.AddCell(new Paragraph("Количество проданных
билетов", font));
                    table.AddCell(new Paragraph("Выручка с проданных
билетов", font));

                    foreach (var line in ticketDataList)
                    {
                        table.AddCell(new
Paragraph(line.ticketMovieID.ToString(), font));
                        table.AddCell(new Paragraph(line.ticketMovieTitle,
font));
                        table.AddCell(new Paragraph(line.ticketMovieGenre,
font));
                        table.AddCell(new
Paragraph(line.ticketMovieYearOfPublication.ToString(), font));
                        table.AddCell(new
Paragraph(line.ticketMovieTiming.ToString(), font));
                        table.AddCell(new
Paragraph(line.ticketMovieAgeRating, font));
                        table.AddCell(new
Paragraph(line.ticketMovieCountry, font));
                        table.AddCell(new Paragraph(line.ticketMovieActors,
font));
                        table.AddCell(new Paragraph(line.ticketCount, font));
                        table.AddCell(new Paragraph(line.ticketSummaryCost,
font));
                    }
                    doc.Add(table);

                    double ticketPriceSumm = 0;
                    foreach (var line in ticketDataList)
                    {
                        ticketPriceSumm +=
Convert.ToDouble(line.ticketSummaryCost.Replace(" руб.", ""));
                    }

                    Paragraph paragraph = new Paragraph("Итого: " +
ticketPriceSumm.ToString() + " руб.", font);
                    doc.Add(paragraph);

                    doc.Close();
                    MessageBox.Show("Файл сохранен", "Готово",
MessageBoxButton.OK, MessageBoxImage.Information);

                    Process.Start(saveFileDialog.FileName);
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }

```

```

    }
}
}

```

AddOrEditActor.xaml

```

<Window x:Class="Cinema.AddOrEditActor"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:Cinema"
    mc:Ignorable="d"
    Title="Актер" Height="370" Width="500"
    Loaded="Window_Loaded" ResizeMode="NoResize"
    WindowStartupLocation="CenterScreen">
    <Grid Style="{DynamicResource MainColorStyle}">
        <StackPanel>
            <Grid>
                <Rectangle Style="{DynamicResource
SecondColorRectangleStyle}"/>
                <TextBlock Text="АКТЕР" Style="{DynamicResource
TitleTextBlockStyle}"/>
            </Grid>
            <Grid HorizontalAlignment="Center" Margin="10">
                <StackPanel>
                    <Grid Margin="0,0,0,10">
                        <Label Content="Фамилия:" Style="{DynamicResource
MainLableStyle}"/>
                        <TextBox x:Name="ActorSurname"
HorizontalAlignment="Left" Margin="100,0,0,0" Width="350"
Style="{DynamicResource MainTextBoxStyle}"/>
                    </Grid>
                    <Grid Margin="0,0,0,10">
                        <Label Content="Имя:" Style="{DynamicResource
MainLableStyle}"/>
                        <TextBox x:Name="ActorName"
HorizontalAlignment="Left" Margin="100,0,0,0" Width="350"
Style="{DynamicResource MainTextBoxStyle}"/>
                    </Grid>
                    <Grid Margin="0,0,0,10">
                        <Label Content="Отчество:" Style="{DynamicResource
MainLableStyle}"/>
                        <TextBox x:Name="ActorPatronymic"
HorizontalAlignment="Left" Margin="100,0,0,0" Width="350"
Style="{DynamicResource MainTextBoxStyle}"/>
                    </Grid>
                    <Grid Margin="0,0,0,10">
                        <Label Content="Прозвище:"
Style="{DynamicResource MainLableStyle}"/>
                        <TextBox x:Name="ActorNickname"
HorizontalAlignment="Left" Margin="100,0,0,0" Width="350"
Style="{DynamicResource MainTextBoxStyle}"/>

```

```

</Grid>
<Grid Margin="0,0,0,10">
    <StackPanel>
        <Grid>
            <Label Content="Фотография:"
Style="{DynamicResource MainLableStyle}"/>
            <Image x:Name="ActorPhoto"
HorizontalAlignment="Left" Margin="170,0,0,5" Width="200"
Height="200" Visibility="Collapsed"/>
        </Grid>
        <Button x:Name="LoadPhoto"
HorizontalAlignment="Left" Margin="170,0,0,0" Content="Загрузить"
Width="200" Style="{DynamicResource MainButtonStyle}"
Click="LoadPhoto_Click"/>
    </StackPanel>
</Grid>
</StackPanel>
</Grid>
<Button x:Name="Save" Margin="0,0,0,0"
Content="Сохранить" Width="450" Style="{DynamicResource
MainButtonStyle}" Click="Save_Click"/>
</StackPanel>
</Grid>
</Window>

```

AddOrEditActor.xaml.cs

```

using Microsoft.Win32;
using System;
using System.IO;
using System.Linq;
using System.Windows;
using System.Windows.Media.Imaging;
using static Cinema.Authorization;

namespace Cinema
{
    /// <summary>
    /// Логика взаимодействия для AddOrEditActor.xaml
    /// </summary>
    public partial class AddOrEditActor : Window
    {
        public AddOrEditActor()
        {
            InitializeComponent();
        }

        public string actorPhotoPath;

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            ActorPhoto.Visibility = Visibility.Collapsed;
            LoadData();
        }
    }
}

```

```

private void LoadData()
{
    try
    {
        using (var dataBase = new CinemaEntities())
        {
            if (TransmittedData.idSelectedActor != -1)
            {
                var actorData = dataBase.Actor.Where(w => w.IDActor
== TransmittedData.idSelectedActor).FirstOrDefault();

                ActorSurname.Text = actorData.Surname;
                ActorName.Text = actorData.Name;
                ActorPatronymic.Text = actorData.Patronymic;
                ActorNickname.Text = actorData.Nickname;

                if (actorData.Image != null)
                {
                    BitmapImage bitmapImage = new BitmapImage();
                    bitmapImage.BeginInit();
                    bitmapImage.StreamSource = new
MemoryStream(actorData.Image);
                    bitmapImage.EndInit();

                    this.Height = 535;
                    ActorPhoto.Visibility = Visibility.Visible;
                    ActorPhoto.Source = bitmapImage;
                }
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void LoadPhoto_Click(object sender, RoutedEventArgs e)
{
    try
    {
        OpenFileDialog openFileDialog = new OpenFileDialog();
        openFileDialog.Filter = "PNG (*.png)*.png|JPG
(*.jpg)*.jpg";

        if (openFileDialog.ShowDialog() == true)
        {
            actorPhotoPath = openFileDialog.FileName;

            this.Height = 535;
            ActorPhoto.Visibility = Visibility.Visible;

```

```

        ActorPhoto.Source = new BitmapImage(new
Uri(openFileDialog.FileName));
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void Save_Click(object sender, RoutedEventArgs e)
{
    try
    {
        if (ActorSurname.Text == "" && ActorName.Text == "")
        {
            MessageBox.Show("Данные не были заполнены",
"Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);
            return;
        }

        using (var dataBase = new CinemaEntities())
        {
            if (TransmittedData.idSelectedActor != -1)
            {
                var actorData = dataBase.Actor.Where(w => w.IDActor
== TransmittedData.idSelectedActor).FirstOrDefault();

                actorData.Surname = ActorSurname.Text;
                actorData.Name = ActorName.Text;
                actorData.Patronymic = ActorPatronymic.Text;
                actorData.Nickname = ActorNickname.Text;

                if (actorPhotoPath != null)
                {
                    actorData.Image = File.ReadAllBytes(actorPhotoPath);
                }

                dataBase.SaveChanges();
            }
            else
            {
                var newActorData = new Actor();

                newActorData.Surname = ActorSurname.Text;
                newActorData.Name = ActorName.Text;
                newActorData.Patronymic = ActorPatronymic.Text;
                newActorData.Nickname = ActorNickname.Text;

                if (actorPhotoPath != null)
                {

```

```

        newActorData.Image =
File.ReadAllBytes(actorPhotoPath);
    }

    dataBase.Actor.Add(newActorData);
    dataBase.SaveChanges();
}

this.Close();
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
    MessageBoxButton.OK, MessageBoxImage.Error);
}
}
}
}

```

AddOrEditEmployee.xaml

```

<Window x:Class="Cinema.AddOrEditEmployee"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    xmlns:local="clr-namespace:Cinema"
    mc:Ignorable="d"
    Title="Сотрудник" Height="530" Width="500"
    Loaded="Window_Loaded" ResizeMode="NoResize"
    WindowStartupLocation="CenterScreen">

    <Grid Style="{DynamicResource MainColorStyle}">
        <StackPanel>
            <Grid>
                <Rectangle Style="{DynamicResource
SecondColorRectangleStyle}" />
                <TextBlock Text="СОТРУДНИК"
Style="{DynamicResource TitleTextBlockStyle}" />
            </Grid>
            <Grid HorizontalAlignment="Center" Margin="10">
                <StackPanel>
                    <Grid Margin="0,0,0,10">
                        <Label Content="Фамилия:" Style="{DynamicResource
MainLableStyle}" />
                        <TextBox x:Name="EmployeeSurname"
HorizontalAlignment="Left" Margin="100,0,0,0" Width="350"
Style="{DynamicResource MainTextBoxStyle}" />
                    </Grid>
                    <Grid Margin="0,0,0,10">
                        <Label Content="Имя:" Style="{DynamicResource
MainLableStyle}" />

```

```

                        <TextBox x:Name="EmployeeName"
HorizontalAlignment="Left" Margin="100,0,0,0" Width="350"
Style="{DynamicResource MainTextBoxStyle}" />
                    </Grid>
                    <Grid Margin="0,0,0,10">
                        <Label Content="Отчество:" Style="{DynamicResource
MainLableStyle}" />
                        <TextBox x:Name="EmployeePatronymic"
HorizontalAlignment="Left" Margin="100,0,0,0" Width="350"
Style="{DynamicResource MainTextBoxStyle}" />
                    </Grid>
                    <Grid Margin="0,0,0,10">
                        <Label Content="Телефон:" Style="{DynamicResource
MainLableStyle}" />
                        <TextBox x:Name="EmployeePhone"
HorizontalAlignment="Left" Margin="100,0,0,0" Width="350"
Style="{DynamicResource MainTextBoxStyle}" />
                    </Grid>
                    <Grid Margin="0,0,0,10">
                        <Label Content="Эл. Почта:"
Style="{DynamicResource MainLableStyle}" />
                        <TextBox x:Name="EmployeeEmail"
HorizontalAlignment="Left" Margin="100,0,0,0" Width="350"
Style="{DynamicResource MainTextBoxStyle}" />
                    </Grid>
                    <Grid Margin="0,0,0,10">
                        <Label Content="Должность:"
Style="{DynamicResource MainLableStyle}" />
                        <ComboBox x:Name="EmployeeRole"
Margin="100,0,0,0" Width="350" Style="{DynamicResource
MainComboBoxStyle}" ItemContainerStyle="{DynamicResource
MainSelectedComboBoxStyle}" />
                    </Grid>
                    <Grid Margin="0,0,0,10">
                        <Label Content="Логин:" Style="{DynamicResource
MainLableStyle}" />
                        <TextBox x:Name="EmployeeLogin"
HorizontalAlignment="Left" Margin="100,0,0,0" Width="350"
Style="{DynamicResource MainTextBoxStyle}" />
                    </Grid>
                    <Grid Margin="0,0,0,10">
                        <Label Content="Пароль:" Style="{DynamicResource
MainLableStyle}" />
                        <TextBox x:Name="EmployeePassword"
HorizontalAlignment="Left" Margin="100,0,0,0" Width="350"
Style="{DynamicResource MainTextBoxStyle}" />
                    </Grid>
                    <Grid Margin="0,0,0,10">
                        <StackPanel>
                            <Grid>
                                <Label Content="Фотография:"
Style="{DynamicResource MainLableStyle}" />

```



```

        <Image x:Name="EmployeePhoto"
HorizontalAlignment="Left" Margin="170,0,0,5" Width="200"
Height="200" Visibility="Collapsed"/>
    </Grid>
    <Button x:Name="LoadPhoto"
HorizontalAlignment="Left" Margin="170,0,0,0" Content="Загрузить"
Width="200" Style="{DynamicResource MainButtonStyle}"
Click="LoadPhoto_Click"/>
    </StackPanel>
</Grid>
</StackPanel>
</Grid>
<Button x:Name="Save" Margin="0,0,0,0"
Content="Сохранить" Width="450" Style="{DynamicResource
MainButtonStyle}" Click="Save_Click"/>
</StackPanel>
</Grid>
</Window>

```

AddOrEditEmployee.xaml.cs

```

using Microsoft.Win32;
using System;
using System.IO;
using System.Linq;
using System.Text.RegularExpressions;
using System.Windows;
using System.Windows.Media.Imaging;
using static Cinema.Authorization;

namespace Cinema
{
    /// <summary>
    /// Логика взаимодействия для AddOrEditEmployee.xaml
    /// </summary>
    public partial class AddOrEditEmployee : Window
    {
        public AddOrEditEmployee()
        {
            InitializeComponent();

            public string employeePhotoPath;

            private void Window_Loaded(object sender, RoutedEventArgs e)
            {
                LoadData();
            }

            private void LoadData()
            {
                try
                {
                    using (var dataBase = new CinemaEntities())

```

```

{
    var roleData = dataBase.Role.Select(s => s.Title).ToList();
    EmployeeRole.ItemsSource = roleData;

    if (TransmittedData.idSelectedEmployee != -1)
    {
        var employeeData = (from employee in
dataBase.Employee
                        join
                        role in dataBase.Role on employee.IDRole
equals role.IDRole into roleGroup
                        from role in roleGroup.DefaultIfEmpty()
                        where (employee.IDEmployee ==
TransmittedData.idSelectedEmployee)
                        select new
                        {
                            employee.IDEmployee,
                            employee.Photo,
                            employee.Surname,
                            employee.Name,
                            employee.Patronymic,
                            employee.Phone,
                            employee.Email,
                            employee.Login,
                            employee.Password,
                            employeeRole = role.Title,
                        }).FirstOrDefault();

        EmployeeSurname.Text = employeeData.Surname;
        EmployeeName.Text = employeeData.Name;
        EmployeePatronymic.Text = employeeData.Patronymic;
        EmployeePhone.Text = employeeData.Phone;
        EmployeeEmail.Text = employeeData.Email;
        EmployeeRole.SelectedItem =
employeeData.employeeRole;
        EmployeeLogin.Text = employeeData.Login;
        EmployeePassword.Text = employeeData.Password;

        if (employeeData.Photo != null)
        {
            this.Height = 695;
            EmployeePhoto.Visibility = Visibility.Visible;

            BitmapImage bitmapImage = new BitmapImage();
            bitmapImage.BeginInit();
            bitmapImage.StreamSource = new
MemoryStream(employeeData.Photo);
            bitmapImage.EndInit();

            EmployeePhoto.Source = bitmapImage;
        }
    }
}

```

```

    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
    MessageBoxButton.OK, MessageBoxImage.Error);
}
}

private void LoadPhoto_Click(object sender, RoutedEventArgs e)
{
    try
    {
        OpenFileDialog openFileDialog = new OpenFileDialog();
        openFileDialog.Filter = "PNG (*.png)*.png|JPG
(*.jpg)*.jpg";

        if (openFileDialog.ShowDialog() == true)
        {
            employeePhotoPath = openFileDialog.FileName;

            this.Height = 695;
            EmployeePhoto.Visibility = Visibility.Visible;
            EmployeePhoto.Source = new BitmapImage(new
Uri(openFileDialog.FileName));
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void Save_Click(object sender, RoutedEventArgs e)
{
    try
    {
        string phonePattern = @"^\+7(\d{3})\d{3}-\d{2}-\d{2}$";
        string mailPattern = @"^[a-zA-Z0-9._%+-]+@mail.ru$";
        string gmailPattern = @"^[a-zA-Z0-9._%+-]+@gmail.com$";
        string yandexMailPattern = @"^[a-zA-Z0-9._%+-
]+@yandex.ru$";

        if (EmployeeSurname.Text == "" && EmployeeName.Text
== "" || EmployeePatronymic.Text == "" || EmployeeLoggin.Text == ""
|| EmployeePassword.Text == "" || EmployeeRole.SelectedIndex < 0)
        {
            MessageBox.Show("Данные не были заполнены",
            "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);
            return;
        }
    }
}

```

```

        if (!Regex.IsMatch(EmployeePhone.Text, phonePattern))
        {
            MessageBox.Show("Номер телефона заполняется в
формате +7(999)999-99-99", "Внимание", MessageBoxButton.OK,
            MessageBoxImage.Warning);
            return;
        }

        if (!Regex.IsMatch(EmployeeEmail.Text, mailPattern) &&
!Regex.IsMatch(EmployeeEmail.Text, gmailPattern) &&
!Regex.IsMatch(EmployeeEmail.Text, yandexMailPattern))
        {
            MessageBox.Show("Почта должна содержать доменный
адрес (@mail.ru / @gmail.com / @yandex.ru)", "Внимание",
            MessageBoxButton.OK, MessageBoxImage.Warning);
            return;
        }

        using (var dataBase = new CinemaEntities())
        {
            if (TransmittedData.idSelectedEmployee != -1)
            {
                var employeeData = dataBase.Employee.Where(w =>
w.IDEmployee ==
TransmittedData.idSelectedEmployee).FirstOrDefault();

                employeeData.Surname = EmployeeSurname.Text;
                employeeData.Name = EmployeeName.Text;
                employeeData.Patronymic = EmployeePatronymic.Text;
                employeeData.Phone = EmployeePhone.Text;
                employeeData.Email = EmployeeEmail.Text;

                var roleData = dataBase.Role.Where(w => w.Title ==
EmployeeRole.SelectedItem.ToString()).FirstOrDefault();

                employeeData.IDRole = roleData.IDRole;
                employeeData.Login = EmployeeLoggin.Text;
                employeeData.Password = EmployeePassword.Text;

                if (employeePhotoPath != null)
                {
                    employeeData.Photo =
File.ReadAllBytes(employeePhotoPath);
                }

                dataBase.SaveChanges();
            }
            else
            {
                var newEmployeeData = new Employee();

                newEmployeeData.Surname = EmployeeSurname.Text;
                newEmployeeData.Name = EmployeeName.Text;
            }
        }
    }
}

```

```

        newEmployeeData.Patronymic =
EmployeePatronymic.Text;
        newEmployeeData.Phone = EmployeePhone.Text;
        newEmployeeData.Email = EmployeeEmail.Text;

        var roleData = dataBase.Role.Where(w => w.Title ==
EmployeeRole.SelectedItem.ToString()).FirstOrDefault();

        newEmployeeData.IDRole = roleData.IDRole;
        newEmployeeData.Login = EmployeeLogin.Text;
        newEmployeeData.Password = EmployeePassword.Text;

        if (employeePhotoPath != null)
        {
            newEmployeeData.Photo =
File.ReadAllBytes(employeePhotoPath);
        }

        dataBase.Employee.Add(newEmployeeData);

        dataBase.SaveChanges();
    }

    MessageBox.Show("Данные были сохранены", "Готово",
MessageBoxButton.OK, MessageBoxImage.Information);
    this.Close();
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}
}
}
}

```

AddOrEditGenre.xaml

```

<Window x:Class="Cinema.AddOrEditGenre"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    xmlns:local="clr-namespace:Cinema"
    mc:Ignorable="d"
    Title="Жанр" Height="180" Width="500"
    ResizeMode="NoResize" Loaded="Window_Loaded"
    WindowStartupLocation="CenterScreen">
    <Grid Style="{DynamicResource MainColorStyle}">
        <StackPanel>
            <Grid>

```

```

                <Rectangle Style="{DynamicResource
SecondColorRectangleStyle}" />
                <TextBlock Text="ЖАНР" Style="{DynamicResource
TitleTextBlockStyle}" />
            </Grid>
            <Grid HorizontalAlignment="Center" Margin="10">
                <StackPanel>
                    <Grid Margin="0,0,0,10">
                        <Label Content="Название:" Style="{DynamicResource
MainLableStyle}" />
                        <TextBox x:Name="GenreTitle"
HorizontalAlignment="Left" Margin="100,0,0,0" Width="350"
Style="{DynamicResource MainTextBoxStyle}" />
                    </Grid>
                </StackPanel>
            </Grid>
            <Button x:Name="Save" Margin="0,0,0,0"
Content="Сохранить" Width="450" Style="{DynamicResource
MainButtonStyle}" Click="Save_Click" />
        </StackPanel>
    </Grid>
</Window>

```

AddOrEditGenre.xaml.cs

```

using System;
using System.Linq;
using System.Windows;
using static Cinema.Authorization;

namespace Cinema
{
    /// <summary>
    /// Логика взаимодействия для AddOrEditGenre.xaml
    /// </summary>
    public partial class AddOrEditGenre : Window
    {
        public AddOrEditGenre()
        {
            InitializeComponent();
        }

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            LoadData();
        }

        private void LoadData()
        {
            try
            {
                if (TransmittedData.idSelectedGenre != -1)
                {
                    using (var dataBase = new CinemaEntities())

```

```

    {
        var genreData = dataBase.Genre.Where(w => w.IDGenre
== TransmittedData.idSelectedGenre).FirstOrDefault();

        GenreTitle.Text = genreData.Title;
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}

private void Save_Click(object sender, RoutedEventArgs e)
{
    try
    {
        if (GenreTitle.Text == "")
        {
            MessageBox.Show("Данные не были заполнены",
"Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);
            return;
        }

        using (var dataBase = new CinemaEntities())
        {
            if (TransmittedData.idSelectedGenre != -1)
            {
                var genreData = dataBase.Genre.Where(w => w.IDGenre
== TransmittedData.idSelectedGenre).FirstOrDefault();

                genreData.Title = GenreTitle.Text;

                dataBase.SaveChanges();
            }
            else
            {
                var newGenreData = new Genre();

                newGenreData.Title = GenreTitle.Text;

                dataBase.Genre.Add(newGenreData);
                dataBase.SaveChanges();
            }

            this.Close();
        }
    }
    catch (Exception ex)
    {

```

```

        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
}

```

AddOrEditMovie.xaml

```

<Window x:Class="Cinema.AddOrEditMovie"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
xmlns:local="clr-namespace:Cinema"
mc:Ignorable="d"
Title="Фильм" Height="620" Width="1170"
Loaded="Window_Loaded" ResizeMode="NoResize"
WindowStartupLocation="CenterScreen">
    <Grid Style="{DynamicResource MainColorStyle}">
        <StackPanel>
            <Grid>
                <Rectangle Style="{DynamicResource
SecondColorRectangleStyle}" />
                <TextBlock Text="ФИЛЬМ" Style="{DynamicResource
TitleTextBlockStyle}" />
            </Grid>
            <Grid HorizontalAlignment="Center" Margin="10">
                <StackPanel>
                    <Grid Margin="0,0,0,10">
                        <Label Content="Название:" Style="{DynamicResource
MainLableStyle}" />
                        <TextBox x:Name="MovieTitle"
HorizontalAlignment="Left" Margin="200,0,0,0" Width="350"
Style="{DynamicResource MainTextBoxStyle}" />
                    </Grid>
                    <Grid Margin="0,0,0,10">
                        <Label Content="Страна производства:"
Style="{DynamicResource MainLableStyle}" />
                        <ComboBox x:Name="MovieCountry"
HorizontalAlignment="Left" Margin="200,0,0,0" Width="350"
Style="{DynamicResource MainComboBoxStyle}"
ItemContainerStyle="{DynamicResource
MainSelectedComboBoxStyle}" />
                    </Grid>
                    <Grid Margin="0,0,0,10">
                        <Label Content="Год публикации:"
Style="{DynamicResource MainLableStyle}" />
                        <TextBox x:Name="MovieYearOfPublication"
HorizontalAlignment="Left" Margin="200,0,0,0" Width="350"
Style="{DynamicResource MainTextBoxStyle}" />
                    </Grid>
                    <Grid Margin="0,0,0,10">

```

```

        <Label Content="Хронометраж (мин):"
Style="{DynamicResource MainLableStyle}"/>
        <TextBox x:Name="MovieTiming"
HorizontalAlignment="Left" Margin="200,0,0,0" Width="350"
Style="{DynamicResource MainTextBoxStyle}"/>
    </Grid>
    <Grid Margin="0,0,0,10">
        <Label Content="Возрастной рейтинг:"
Style="{DynamicResource MainLableStyle}"/>
        <TextBox x:Name="MovieAgeRating"
HorizontalAlignment="Left" Margin="200,0,0,0" Width="350"
Style="{DynamicResource MainTextBoxStyle}"/>
    </Grid>
    <Grid Margin="0,0,0,10">
        <Label Content="Описание:"
Style="{DynamicResource MainLableStyle}"/>
        <TextBox x:Name="MovieDescription"
HorizontalAlignment="Left" TextWrapping="Wrap"
Margin="200,0,0,0" Width="350" Height="200" MaxHeight="200"
Style="{DynamicResource MainTextBoxStyle}"/>
    </Grid>
    <Grid>
        <StackPanel>
            <Grid>
                <Label Content="Обложка:"
Style="{DynamicResource MainLableStyle}"/>
                <Grid Margin="270,0,0,5"
HorizontalAlignment="Left">
                    <Image x:Name="MovieCover" Width="200"
Height="200" HorizontalAlignment="Center" Visibility="Collapsed"/>
                </Grid>
            </Grid>
            <Button x:Name="LoadCover" Content="Загрузить"
Margin="270,0,0,0" Width="200" HorizontalAlignment="Left"
Style="{DynamicResource MainButtonStyle}"
Click="LoadCover_Click"/>
        </StackPanel>
    </Grid>
</StackPanel>
<Rectangle Margin="558,0,560,0" Fill="Black" Width="2"
Opacity="0.2"/>
<StackPanel Margin="580,0,0,0">
    <Grid Margin="0,0,0,10">
        <StackPanel>
            <Grid Margin="0,0,0,10">
                <Label Content="Жанры:"
Style="{DynamicResource MainLableStyle}"/>
                <WrapPanel>
                    <TextBox x:Name="MovieGenere"
HorizontalAlignment="Left" TextWrapping="Wrap" Margin="80,0,0,0"
Width="455" Style="{DynamicResource MainTextBoxStyle}"
IsReadOnly="True"/>

```

```

                    <Button x:Name="ClearGenre" Content="X"
Height="31.28" Style="{DynamicResource MainClearButtonStyle}"
Click="ClearGenre_Click"/>
                </WrapPanel>
            </Grid>
            <DataGrid x:Name="MovieGenreGrid" Height="120"
Margin="0,0,0,10" d:ItemsSource="{d:SampleData ItemCount=5}"
Style="{DynamicResource MainDataGridStyle}"
MouseDoubleClick="MovieGenreGrid_MouseDoubleClick"/>
        <WrapPanel HorizontalAlignment="Center">
            <Button x:Name="AddGenre" Margin="0,0,5,0"
Content="Добавить" Width="160" Style="{DynamicResource
MainButtonStyle}" Click="AddGenre_Click"/>
            <Button x:Name="EditGenre" Margin="0,0,5,0"
Content="Редактировать" Width="160" Style="{DynamicResource
MainButtonStyle}" Click="EditGenre_Click"/>
            <Button x:Name="RemoveGenre"
Content="Удалить" Width="160" Style="{DynamicResource
MainButtonStyle}" Click="RemoveGenre_Click"/>
        </WrapPanel>
    </StackPanel>
</Grid>
<Grid>
    <StackPanel>
        <Grid Margin="0,0,0,10">
            <Label Content="Актеры:"
Style="{DynamicResource MainLableStyle}"/>
            <WrapPanel>
                <TextBox x:Name="MovieActor"
HorizontalAlignment="Left" TextWrapping="Wrap" Margin="80,0,0,0"
Width="455" Style="{DynamicResource MainTextBoxStyle}"
IsReadOnly="True"/>
                <Button x:Name="ClearActor" Content="X"
Height="31.28" Style="{DynamicResource MainClearButtonStyle}"
Click="ClearActor_Click"/>
            </WrapPanel>
        </Grid>
        <DataGrid x:Name="MovieActorGrid" Height="120"
Margin="0,0,0,10" ColumnWidth="*" d:ItemsSource="{d:SampleData
ItemCount=5}" Background="White"
MouseDoubleClick="MovieActorGrid_MouseDoubleClick"/>
        <WrapPanel HorizontalAlignment="Center">
            <Button x:Name="AddActor" Margin="0,0,5,0"
Content="Добавить" Width="160" Style="{DynamicResource
MainButtonStyle}" Click="AddActor_Click"/>
            <Button x:Name="EditActor" Margin="0,0,5,0"
Content="Редактировать" Width="160" Style="{DynamicResource
MainButtonStyle}" Click="EditActor_Click"/>
            <Button x:Name="RemoveActor"
Content="Удалить" Width="160" Style="{DynamicResource
MainButtonStyle}" Click="RemoveActor_Click"/>
        </WrapPanel>
    </StackPanel>

```

```

        </Grid>
    </StackPanel>
</Grid>

<Button x:Name="Save" Margin="0,10,0,0"
Content="Сохранить" Width="250" Style="{DynamicResource
MainButtonStyle}" Click="Save_Click"/>
</StackPanel>
</Grid>
</Window>

```

AddOrEditMovie.xaml.cs

```

using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media.Imaging;
using static Cinema.Authorization;

namespace Cinema
{
    /// <summary>
    /// Логика взаимодействия для AddOrEditMovie.xaml
    /// </summary>
    public partial class AddOrEditMovie : Window
    {
        public AddOrEditMovie()
        {
            InitializeComponent();

            List<int> selectedGenre = new List<int>();
            List<int> selectedActor = new List<int>();
            public string coverPath;

            private void Window_Loaded(object sender, RoutedEventArgs e)
            {
                LoadGridData();
                LoadUserData();
            }

            private void LoadGridData()
            {
                try
                {
                    using (var dataBase = new CinemaEntities())
                    {
                        var genreAllData = dataBase.Genre.Select(s => new {
s.IDGenre, s.Title }).ToList();

```

```

var actorAllData = dataBase.Actor.Select(s => new {
s.IDActor, s.Surname, s.Name, s.Patronymic, s.Nickname }).ToList();

MovieGenreGrid.ItemsSource = genreAllData;
MovieActorGrid.ItemsSource = actorAllData;
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}

private void LoadUserData()
{
    try
    {
        using (var dataBase = new CinemaEntities())
        {
            var countryAllData = dataBase.Country.Select(s =>
s.Title).ToList();

            MovieCountry.ItemsSource = countryAllData;

            if (TransmittedData.idSelectedMovie != -1)
            {
                var movieData = (from movie in dataBase.Movie
                                join
                                country in dataBase.Country on
                                movie.IDCountry equals country.IDCountry into countryGroup
                                from country in
                                countryGroup.DefaultIfEmpty()
                                where (movie.IDMovie ==
TransmittedData.idSelectedMovie)
                                select new
                                {
                                    movie.IDMovie,
                                    movie.Cover,
                                    movieTitle = movie.Title,
                                    movie.YearOfPublication,
                                    movie.Timing,
                                    movie.AgeRating,
                                    movie.Description,
                                    movieCountry = country.Title,
                                }
                                ).FirstOrDefault();

                var genreData = (from genre in dataBase.Genre
                                join
                                movieGenre in dataBase.MovieGenre on
                                genre.IDGenre equals movieGenre.IDGenre into movieGenreGroup

```

```

        from movieGenre in
movieGenreGroup.DefaultIfEmpty()
        where (movieGenre.IDMovie ==
TransmittedData.idSelectedMovie)
        select new
        {
            genre.IDGenre,
            genre.Title,
        }).ToList();

foreach (var genreLine in genreData)
{
    selectedGenre.Add(genreLine.IDGenre);
}

var actorData = (from actor in dataBase.Actor
    join
        actorInMovie in dataBase.ActorsInMovies on
actor.IDActor equals actorInMovie.IDActor into actorInMovieGroup
    from actorInMovie in
actorInMovieGroup.DefaultIfEmpty()
    where (actorInMovie.IDMovie ==
TransmittedData.idSelectedMovie)
    select new
    {
        actor.IDActor,
        actor.Surname,
        actor.Name,
        actor.Patronymic,
        actor.Nickname,
    }).ToList();

foreach (var actorLine in actorData)
{
    selectedActor.Add(actorLine.IDActor);
}

MovieTitle.Text = movieData.movieTitle;
MovieCountry.SelectedItem = movieData.movieCountry;
MovieYearOfPublication.Text =
movieData.YearOfPublication.ToString();
MovieTiming.Text = movieData.Timing.ToString();
MovieAgeRating.Text =
movieData.AgeRating.ToString();
MovieDescription.Text = movieData.Description;

if (movieData.Cover != null)
{
    this.Height = 780;
    MovieCover.Visibility = Visibility.Visible;

    BitmapImage bitmapImage = new BitmapImage();
    bitmapImage.BeginInit();

```

```

        bitmapImage.StreamSource = new
MemoryStream(movieData.Cover);
        bitmapImage.EndInit();

        MovieCover.Source = bitmapImage;
    }

    foreach (var genreLine in genreData)
    {
        MovieGenre.Text += "|" + genreLine.Title + "|";
    }

    foreach (var actorLine in actorData)
    {
        MovieActor.Text += "|" + actorLine.Surname + " " +
actorLine.Name + " " + actorLine.Patronymic + " " +
actorLine.Nickname + "|";
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}

private void ClearGenre_Click(object sender, RoutedEventArgs e)
{
    selectedGenre.Clear();
    MovieGenre.Clear();
}

private void ClearActor_Click(object sender, RoutedEventArgs e)
{
    selectedActor.Clear();
    MovieActor.Clear();
}

private void LoadCover_Click(object sender, RoutedEventArgs e)
{
    try
    {
        OpenFileDialog openFileDialog = new OpenFileDialog();
        openFileDialog.Filter = "JPG (*.jpg)|*.jpg|PNG
(*.png)|*.png";

        if (openFileDialog.ShowDialog() == true)
        {
            coverPath = openFileDialog.FileName;

            this.Height = 780;

```

```

        MovieCover.Visibility = Visibility.Visible;
        MovieCover.Source = new BitmapImage(new
Uri(openFileDialog.FileName));
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}

private void Save_Click(object sender, RoutedEventArgs e)
{
    try
    {
        if (MovieCountry.SelectedIndex < 0 || MovieTitle.Text == "" ||
!int.TryParse(MovieYearOfPublication.Text, out _) ||
!double.TryParse(MovieTiming.Text, out _) ||
!int.TryParse(MovieAgeRating.Text, out _) || MovieDescription.Text ==
"" || MovieGenre.Text == "" || MovieActor.Text == "")
        {
            MessageBox.Show("Данные заполнены неправильно",
"Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);
            return;
        }

        using (var dataBase = new CinemaEntities())
        {
            if (TransmittedData.idSelectedMovie != -1)
            {
                var movieData = dataBase.Movie.Where(w =>
w.IDMovie == TransmittedData.idSelectedMovie).FirstOrDefault();

                movieData.Title = MovieTitle.Text;

                var selectedCountry = dataBase.Country.Where(w =>
w.Title == MovieCountry.SelectedItem.ToString()).FirstOrDefault();
                movieData.IDCountry = selectedCountry.IDCountry;

                movieData.YearOfPublication =
Convert.ToInt32(MovieYearOfPublication.Text);
                movieData.Timing =
Convert.ToInt32(MovieTiming.Text);
                movieData.AgeRating =
Convert.ToInt32(MovieAgeRating.Text);
                movieData.Description = MovieDescription.Text;

                if (coverPath != null)
                {
                    movieData.Cover = File.ReadAllBytes(coverPath);
                }
            }
        }
    }
}

```

```

        var removeGenre = dataBase.MovieGenre.Where(w =>
w.IDMovie == TransmittedData.idSelectedMovie).ToList();
        if (removeGenre != null)
            dataBase.MovieGenre.RemoveRange(removeGenre);

        foreach (var genreLine in selectedGenre)
        {
            var newMovieGenre = new MovieGenre();
            newMovieGenre.IDMovie = movieData.IDMovie;
            newMovieGenre.IDGenre = genreLine;

            dataBase.MovieGenre.Add(newMovieGenre);
        }

        var removeActorInMovie =
dataBase.ActorsInMovies.Where(w => w.IDMovie ==
TransmittedData.idSelectedMovie).ToList();
        if (removeActorInMovie != null)
            dataBase.ActorsInMovies.RemoveRange(removeActorInMovie);

        foreach (var actorLine in selectedActor)
        {
            var newMovieActor = new ActorsInMovies();
            newMovieActor.IDMovie = movieData.IDMovie;
            newMovieActor.IDActor = actorLine;

            dataBase.ActorsInMovies.Add(newMovieActor);
        }

        dataBase.SaveChanges();
    }
    else
    {
        var newMovieData = new Movie();

        newMovieData.Title = MovieTitle.Text;

        var selectedCountry = dataBase.Country.Where(w =>
w.Title == MovieCountry.SelectedItem.ToString()).FirstOrDefault();
        newMovieData.IDCountry = selectedCountry.IDCountry;

        newMovieData.YearOfPublication =
Convert.ToInt32(MovieYearOfPublication.Text);
        newMovieData.Timing =
Convert.ToInt32(MovieTiming.Text);
        newMovieData.AgeRating =
Convert.ToInt32(MovieAgeRating.Text);
        newMovieData.Description = MovieDescription.Text;

        if (coverPath != null)
        {

```



```

        newMovieData.Cover =
File.ReadAllBytes(coverPath);
    }

    dataBase.Movie.Add(newMovieData);
    dataBase.SaveChanges();

    foreach (var genreLine in selectedGenre)
    {
        var newMovieGenre = new MovieGenre();
        newMovieGenre.IDMovie = newMovieData.IDMovie;
        newMovieGenre.IDGenre = genreLine;

        dataBase.MovieGenre.Add(newMovieGenre);
    }

    foreach (var actorLine in selectedActor)
    {
        var newMovieActor = new ActorsInMovies();
        newMovieActor.IDMovie = newMovieData.IDMovie;
        newMovieActor.IDActor = actorLine;

        dataBase.ActorsInMovies.Add(newMovieActor);
    }

    dataBase.SaveChanges();
}

MessageBox.Show("Данные были сохранены", "Готово",
MessageBoxButton.OK, MessageBoxImage.Information);
this.Close();
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}
}

private void MovieGenreGrid_MouseDoubleClick(object sender,
MouseButtonEventArgs e)
{
    try
    {
        DataGridView row =
(DataGridView)MovieGenreGrid.ItemContainerGenerator.ContainerFrom
mIndex(MovieGenreGrid.SelectedIndex);

        DataGridViewCell cellId =
MovieGenreGrid.Columns[0].GetCellContent(row).Parent as
DataGridViewCell;

```

```

        DataGridViewCell cellTitle =
MovieGenreGrid.Columns[1].GetCellContent(row).Parent as
DataGridViewCell;

        int idGenre =
Convert.ToInt32(((TextBlock)cellId.Content).Text);

        string titleGenre = ((TextBlock)cellTitle.Content).Text;

        bool availabilityGenre = false;
        foreach (var genreLine in selectedGenre)
        {
            if (genreLine == idGenre)
            {
                availabilityGenre = true;
            }
        }

        if (!availabilityGenre)
        {
            selectedGenre.Add(idGenre);
            MovieGenre.Text += "|" + titleGenre + "|";
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void MovieActorGrid_MouseDoubleClick(object sender,
MouseButtonEventArgs e)
{
    try
    {
        DataGridView row =
(DataGridView)MovieActorGrid.ItemContainerGenerator.ContainerFrom
mIndex(MovieActorGrid.SelectedIndex);

        DataGridViewCell cellId =
MovieActorGrid.Columns[0].GetCellContent(row).Parent as
DataGridViewCell;

        DataGridViewCell cellSurname =
MovieActorGrid.Columns[1].GetCellContent(row).Parent as
DataGridViewCell;

        DataGridViewCell cellName =
MovieActorGrid.Columns[2].GetCellContent(row).Parent as
DataGridViewCell;

        DataGridViewCell cellPatronymic =
MovieActorGrid.Columns[3].GetCellContent(row).Parent as
DataGridViewCell;

        DataGridViewCell cellNickname =
MovieActorGrid.Columns[4].GetCellContent(row).Parent as
DataGridViewCell;

```

```

        int idActor =
Convert.ToInt32(((TextBlock)cellId.Content).Text);
        string surnameActor =
((TextBlock)cellSurname.Content).Text;
        string nameActor = ((TextBlock)cellName.Content).Text;
        string patronymicActor =
((TextBlock)cellPatronymic.Content).Text;
        string nicknameActor =
((TextBlock)cellNickname.Content).Text;

        bool availabilityActor = false;
        foreach (var actorLine in selectedActor)
        {
            if (actorLine == idActor)
            {
                availabilityActor = true;
            }
        }

        if (!availabilityActor)
        {
            selectedActor.Add(idActor);
            MovieActor.Text += "|" + surnameActor + " " + nameActor
+ " " + patronymicActor + " " + nicknameActor + "|";
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void AddGenre_Click(object sender, RoutedEventArgs e)
{
    try
    {
        TransmittedData.idSelectedGenre = -1;
        AddOrEditGenre addOrEditGenre = new AddOrEditGenre();
        addOrEditGenre.ShowDialog();
        LoadGridData();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void EditGenre_Click(object sender, RoutedEventArgs e)
{
    try
    {

```

```

        if (MovieGenreGrid.SelectedIndex >= 0)
        {
            DataGridRow row =
(DataGridRow)MovieGenreGrid.ItemContainerGenerator.ContainerFromIndex(MovieGenreGrid.SelectedIndex) as DataGridRow;
            DataGridCell cell =
MovieGenreGrid.Columns[0].GetCellContent(row).Parent as
DataGridCell;
            TransmittedData.idSelectedGenre =
Convert.ToInt32(((TextBlock)cell.Content).Text);

            AddOrEditGenre addOrEditGenre = new
AddOrEditGenre();
            addOrEditGenre.ShowDialog();
            LoadGridData();
        }
        else
        {
            MessageBox.Show("Выберите строку для
редактирования", "Внимание", MessageBoxButton.OK,
MessageBoxImage.Warning);
            return;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void RemoveGenre_Click(object sender, RoutedEventArgs
e)
{
    try
    {
        if (MovieGenreGrid.SelectedIndex >= 0)
        {
            if (MessageBox.Show("Вы действительно хотите удалить
выбранный жанр?", "Внимание", MessageBoxButton.YesNo,
MessageBoxImage.Warning) == MessageBoxResult.Yes)
            {
                using (var dataBase = new CinemaEntities())
                {
                    DataGridRow row =
(DataGridRow)MovieGenreGrid.ItemContainerGenerator.ContainerFromIndex(MovieGenreGrid.SelectedIndex) as DataGridRow;
                    DataGridCell cell =
MovieGenreGrid.Columns[0].GetCellContent(row).Parent as
DataGridCell;
                    int selectedGenre =
Convert.ToInt32(((TextBlock)cell.Content).Text);

```

```

        var removeGenre = dataBase.Genre.Where(w =>
w.IDGenre == selectedGenre).FirstOrDefault();

        var removeMovieGenre =
dataBase.MovieGenre.Where(w => w.IDGenre ==
selectedGenre).ToList();

        if (removeMovieGenre.Count != 0)
        {
            if (MessageBox.Show("Удаляемый жанр
используется в фильмах. Вы действительно хотите его удалить?",
"Внимание", MessageBoxButton.YesNo, MessageBoxImage.Warning)
== DialogResult.Yes)
            {

dataBase.MovieGenre.RemoveRange(removeMovieGenre);
                dataBase.Genre.Remove(removeGenre);

                dataBase.SaveChanges();

                MessageBox.Show("Жанр был удален",
"Готово", MessageBoxButton.OK, MessageBoxImage.Information);
                LoadGridData();
            }
            else
            {
                return;
            }
        }
        else
        {
            dataBase.Genre.Remove(removeGenre);

            dataBase.SaveChanges();

            MessageBox.Show("Жанр был удален", "Готово",
MessageBoxButton.OK, MessageBoxImage.Information);
            LoadGridData();
        }
    }
    else
    {
        MessageBox.Show("Выберите строку для
редактирования", "Внимание", MessageBoxButton.OK,
MessageBoxImage.Warning);
        return;
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}

```

```

    }
}

private void AddActor_Click(object sender, RoutedEventArgs e)
{
    try
    {
        TransmittedData.idSelectedActor = -1;
        AddOrEditActor addOrEditActor = new AddOrEditActor();
        addOrEditActor.ShowDialog();
        LoadGridData();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void EditActor_Click(object sender, RoutedEventArgs e)
{
    try
    {
        if (MovieActorGrid.SelectedIndex >= 0)
        {
            DataGridViewRow row =
(DataGridViewRow)MovieActorGrid.ItemContainerGenerator.ContainerFrom
Index(MovieActorGrid.SelectedIndex) as DataGridViewRow;
            DataGridViewCell cell =
MovieActorGrid.Columns[0].GetCellContent(row).Parent as
DataGridViewCell;

            TransmittedData.idSelectedActor =
Convert.ToInt32(((TextBlock)cell.Content).Text);

            AddOrEditActor addOrEditActor = new AddOrEditActor();
            addOrEditActor.ShowDialog();
            LoadGridData();
        }
        else
        {
            MessageBox.Show("Выберите строку для
редактирования", "Внимание", MessageBoxButton.OK,
MessageBoxImage.Warning);
            return;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

```

```

private void RemoveActor_Click(object sender, RoutedEventArgs
e)
{
    try
    {
        if (MovieActorGrid.SelectedIndex >= 0)
        {
            if (MessageBox.Show("Вы действительно хотите удалить
выбранный жанр?", "Внимание", MessageBoxButton.YesNo,
MessageBoxImage.Warning) == MessageBoxResult.Yes)
            {
                using (var dataBase = new CinemaEntities())
                {
                    DataGridRow row =
(DataGridRow)MovieActorGrid.ItemContainerGenerator.ContainerFro
mIndex(MovieActorGrid.SelectedIndex) as DataGridRow;
                    DataGridCell cell =
MovieActorGrid.Columns[0].GetCellContent(row).Parent as
DataGridCell;
                    int selectedActor =
Convert.ToInt32(((TextBlock)cell.Content).Text);
                    var removeActor = dataBase.Actor.Where(w =>
w.IDActor == selectedActor).FirstOrDefault();
                    var removeActorInMovie =
dataBase.ActorsInMovies.Where(w => w.IDActor ==
selectedActor).ToList();

                    if (removeActorInMovie.Count != 0)
                    {
                        if (MessageBox.Show("Удаляемый актер
присутствует в других фильмах. Вы действительно хотите его
удалить?", "Внимание", MessageBoxButton.YesNo,
MessageBoxImage.Warning) == MessageBoxResult.Yes)
                        {
                            dataBase.ActorsInMovies.RemoveRange(removeActorInMovie);
                            dataBase.Actor.Remove(removeActor);

                            dataBase.SaveChanges();

                            MessageBox.Show("Актер был удален",
"Готово", MessageBoxButton.OK, MessageBoxImage.Information);
                            LoadGridData();
                        }
                    }
                    else
                    {
                        return;
                    }
                }
            }
            else
            {
                dataBase.Actor.Remove(removeActor);
            }
        }
    }
}

```

```

dataBase.SaveChanges();

        MessageBox.Show("Актер был удален", "Готово",
MessageBoxButton.OK, MessageBoxImage.Information);
        LoadGridData();
    }
}
}
else
{
    MessageBox.Show("Выберите строку для
редактирования", "Внимание", MessageBoxButton.OK,
MessageBoxImage.Warning);
    return;
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}
}
}
}
}

```

AddOrEditSession.xaml

```

<Window x:Class="Cinema.AddOrEditSession"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
xmlns:local="clr-namespace:Cinema"
mc:Ignorable="d"
Title="Сеанс" Height="300" Width="520"
Loaded="Window_Loaded" ResizeMode="NoResize"
WindowStartupLocation="CenterScreen">
    <Grid Style="{DynamicResource MainColorStyle}">
        <StackPanel>
            <Grid>
                <Rectangle Style="{DynamicResource
SecondColorRectangleStyle}" />
                <TextBlock Text="СЕАНС" Style="{DynamicResource
TitleTextBlockStyle}" />
            </Grid>
            <Grid HorizontalAlignment="Center" Margin="10">
                <StackPanel>
                    <Grid Margin="0,0,0,10">
                        <Label Content="Фильм:" Style="{DynamicResource
MainLableStyle}" />
                        <ComboBox x:Name="SessionMovie"
HorizontalAlignment="Left" Margin="120,0,0,0" Width="350"

```

```

Style="{DynamicResource MainComboBoxStyle}"
ItemContainerStyle="{DynamicResource
MainSelectedComboBoxStyle}"/>
    </Grid>
    <Grid Margin="0,0,0,10">
        <Label Content="Дата сеанса:"
Style="{DynamicResource MainLabelStyle}"/>
        <DatePicker x:Name="SessionDate"
HorizontalAlignment="Left" Margin="120,0,0,0" Width="350"
Style="{DynamicResource MainDatePickerStyle}"/>
    </Grid>
    <Grid Margin="0,0,0,10">
        <Label Content="Время сеанса:"
Style="{DynamicResource MainLabelStyle}"/>
        <TextBox x:Name="SessionTime"
HorizontalAlignment="Left" Margin="120,0,0,0" Width="350"
Style="{DynamicResource MainTextBoxStyle}"/>
    </Grid>
    <Grid Margin="0,0,0,10">
        <Label Content="Цена билета:"
Style="{DynamicResource MainLabelStyle}"/>
        <TextBox x:Name="SessionTicketPrice"
HorizontalAlignment="Left" Margin="120,0,0,0" Width="350"
Style="{DynamicResource MainTextBoxStyle}"/>
    </Grid>
</StackPanel>
</Grid>
<Button x:Name="Save" Margin="0,0,0,0"
Content="Сохранить" Width="470" Style="{DynamicResource
MainButtonStyle}" Click="Save_Click"/>
</StackPanel>
</Grid>
</Window>

```

AddOrEditSession.xaml.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using System.Windows;
using static Cinema.Authorization;

namespace Cinema
{
    /// <summary>
    /// Логика взаимодействия для AddOrEditSession.xaml
    /// </summary>
    public partial class AddOrEditSession : Window
    {
        public AddOrEditSession()
        {
            InitializeComponent();
        }
    }
}

```

```

private void Window_Loaded(object sender, RoutedEventArgs e)
{
    LoadData();
}

private void LoadData()
{
    try
    {
        using (var dataBase = new CinemaEntities())
        {
            var movieData = dataBase.Movie.ToList();
            List<string> movieDataList = new List<string>();

            foreach (var movieLine in movieData)
            {
                movieDataList.Add(movieLine.IDMovie + "|" +
movieLine.Title);
            }

            SessionMovie.ItemsSource = movieDataList;

            if (TransmittedData.idSelectedSession != -1)
            {
                var sessionData = (from session in dataBase.Session
join
movie in dataBase.Movie on
session.IDMovie equals movie.IDMovie into movieGroup
from movie in movieGroup.DefaultIfEmpty()
where (session.IDSession ==
TransmittedData.idSelectedSession)
select new
{
    movie.IDMovie,
    movie.Title,
    session.DateAndTimeSession,
    session.TicketPrice
}).FirstOrDefault();

                SessionMovie.SelectedItem = sessionData.IDMovie + "|"
+ sessionData.Title;

                SessionDate.SelectedDate =
sessionData.DateAndTimeSession.Date;
                SessionTime.Text =
sessionData.DateAndTimeSession.TimeOfDay.ToString();
                SessionTicketPrice.Text =
sessionData.TicketPrice.ToString();
            }
        }
    }
    catch (Exception ex)
    {
    }
}

```

```

        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void Save_Click(object sender, RoutedEventArgs e)
{
    try
    {
        string timePattern = @"^(?:[01]\d|2[0-3]):[0-5]\d:[0-5]\d$";
        if (SessionMovie.SelectedIndex < 0 ||
        SessionDate.SelectedDate == null || SessionTicketPrice.Text == null ||
        !decimal.TryParse(SessionTicketPrice.Text, out _))
        {
            MessageBox.Show("Данные были заполнены неверно",
            "Внимание", MessageBoxButton.OK, MessageBoxIcon.Warning);
            return;
        }

        if (!Regex.IsMatch(SessionTime.Text, timePattern))
        {
            MessageBox.Show("Время указывается в формате
            00:00:00", "Внимание", MessageBoxButton.OK,
            MessageBoxIcon.Warning);
            return;
        }

        using (var dataBase = new CinemaEntities())
        {
            if (TransmittedData.idSelectedSession != -1)
            {
                var sessionData = dataBase.Session.Where(w =>
                w.IDSession == TransmittedData.idSelectedSession).FirstOrDefault();

                string[] sessionMovieTemp =
                SessionMovie.SelectedItem.ToString().Split('|');
                string[] sessionDateTemp =
                SessionDate.SelectedDate.Value.ToString().Split(' ');

                sessionData.IDMovie =
                Convert.ToInt32(sessionMovieTemp[0]);
                sessionData.DateAndTimeSession =
                Convert.ToDateTime(sessionDateTemp[0] + " " + SessionTime.Text);
                sessionData.TicketPrice =
                Convert.ToDecimal(SessionTicketPrice.Text);

                dataBase.SaveChanges();
            }
            else
            {
                var newSessionData = new Session();
            }
        }
    }
}

```

```

        string[] sessionMovieTemp =
SessionMovie.SelectedItem.ToString().Split("|");

        string[] sessionDateTemp =
SessionDate.SelectedDate.Value.ToString().Split(' ');

        newSessionData.IDMovie =
Convert.ToInt32(sessionMovieTemp[0]);
        newSessionData.DateAndTimeSession =
Convert.ToDateTime(sessionDateTemp[0] + " " + SessionTime.Text);
        newSessionData.TicketPrice =
Convert.ToDecimal(SessionTicketPrice.Text);

        dataBase.Session.Add(newSessionData);

        dataBase.SaveChanges();
    }

    MessageBox.Show("Данные были сохранены", "Готово",
MessageBoxButton.OK, MessageBoxImage.Information);
    this.Close();
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}
}
}
}

```

AdminPanel.xaml

```
<Window x:Class="Cinema.AdminPanel"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
xmlns:local="clr-namespace:Cinema"
mc:Ignorable="d"
Title="Панель администратора" Height="550" Width="1050"
MinHeight="550" MinWidth="1050" Loaded="Window_Loaded"
Closing="Window_Closing" WindowStartupLocation="CenterScreen">
<Grid Style="{DynamicResource MainColorStyle}">
<Grid VerticalAlignment="Top">
<WrapPanel Style="{DynamicResource
SecondColorWrapPanelStyle}">
<Button x:Name="EmployeePage" Content="Сотрудники"
Style="{DynamicResource TabOffButtonStyle}"
Click="SelectedPage_Click"/>
```

```

        <Button x:Name="SettingsPage" Content="Настройки зала"
Style="{DynamicResource TabOffButtonStyle}"
Click="SelectedPage_Click"/>
        <Button x:Name="BackUpPage" Content="Опции"
Style="{DynamicResource TabOffButtonStyle}"
Click="SelectedPage_Click"/>
    </WrapPanel>
    <WrapPanel HorizontalAlignment="Right">
        <Button x:Name="Help" Content="Справка" Width="100"
Style="{DynamicResource TabHelpButtonStyle}"
Click="Help_Click"/>
        <Button x:Name="Exit" Content="Выйти" Width="100"
Style="{DynamicResource TabExitButtonStyle}" Click="Exit_Click"/>
    </WrapPanel>
</Grid>
<Frame x:Name="PageManager" Content="MainFrame"
Margin="0,25,0,0" NavigationUITVisibility="Hidden"
Background="White"/>
</Grid>
</Window>

```

AdminPanel.xaml.cs

```

using Cinema.Pages;
using System;
using System.Diagnostics;
using System.IO;
using System.Reflection;
using System.Reflection.Emit;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Media.Media3D;
using static Cinema.Authorization;

namespace Cinema
{
    /// <summary>
    /// Логика взаимодействия для AdminPanel.xaml
    /// </summary>
    public partial class AdminPanel : Window
    {
        public AdminPanel()
        {
            InitializeComponent();
        }

        public bool exitMode;
        public Button currentSelectedButton;

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            try
            {

```

```

                PageManager.Navigate(new EmployeeControls());

                currentSelectedButton = EmployeePage;
                currentSelectedButton.Style =
                    (Style)Application.Current.Resources["TabOnButtonStyle"];
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Ошибка",
                    MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }

        private void SelectedPage_Click(object sender, RoutedEventArgs
e)
        {
            try
            {
                if (currentSelectedButton != null)
                {
                    currentSelectedButton.Style =
                        (Style)Application.Current.Resources["TabOffButtonStyle"];
                }

                currentSelectedButton = sender as Button;

                currentSelectedButton.Style =
                    (Style)Application.Current.Resources["TabOnButtonStyle"];

                switch (currentSelectedButton.Name)
                {
                    case "EmployeePage":
                        PageManager.Navigate(new EmployeeControls());
                        break;

                    case "SettingsPage":
                        PageManager.Navigate(new SettingsControls());
                        break;

                    case "BackUpPage":
                        PageManager.Navigate(new BackUpControls());
                        break;
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Ошибка",
                    MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }

        private void Exit_Click(object sender, RoutedEventArgs e)
        {

```

```

try
{
    exitMode = true;

    TransmittedData.idEmployee = 0;
    TransmittedData.idSelectedMovie = 0;
    TransmittedData.idSelectedGenre = 0;
    TransmittedData.idSelectedActor = 0;
    TransmittedData.idSelectedSession = 0;
    TransmittedData.idSelectedEmployee = 0;

    Authorization authorization = new Authorization();
    authorization.Show();

    this.Close();
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
    MessageBoxButton.OK, MessageBoxImage.Error);
}

private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
    try
    {
        if (!exitMode)
        {
            Application.Current.Shutdown();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void Help_Click(object sender, RoutedEventArgs e)
{
    try
    {
        Uri resourceUri = new
Uri("pack://application:;./Resource/HelpDocument.pdf",
UriKind.Absolute);

        Stream resourceStream =
Application.GetResourceStream(resourceUri)?.Stream;
        string tempFileName = Path.GetTempFileName() + ".pdf";
        using (FileStream fileStream = new
FileStream(tempFileName, FileMode.Create, FileAccess.Write))

```

```

{
    resourceStream.CopyTo(fileStream);
}

Process.Start(new ProcessStartInfo(tempFileName) {
UseShellExecute = true });
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
    MessageBoxButton.OK, MessageBoxImage.Error);
}
}
}

```

Authorization.xaml

```

<Window x:Class="Cinema.Authorization"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
xmlns:local="clr-namespace:Cinema"
mc:Ignorable="d"
Title="Авторизация" Height="240" Width="400"
ResizeMode="NoResize" Loaded="Window_Loaded"
Closing="Window_Closing" WindowStartupLocation="CenterScreen">
    <Grid Style="{DynamicResource MainColorStyle}">
        <StackPanel>
            <Grid Margin="0,0,0,10">
                <Rectangle Style="{DynamicResource
SecondColorRectangleStyle}" />
                <TextBlock Text="АВТОРИЗАЦИЯ"
Style="{DynamicResource TitleTextBlockStyle}" />
            </Grid>
            <StackPanel HorizontalAlignment="Center">
                <Grid>
                    <Label Content="Имя БД" HorizontalAlignment="Left"
Margin="0,0,0,0" VerticalAlignment="Center"
Style="{DynamicResource MainLableStyle}" />
                    <TextBox x:Name="DataBaseConnection"
HorizontalAlignment="Left" Margin="80,0,0,0"
VerticalAlignment="Center" Width="250" Style="{DynamicResource
MainTextBoxStyle}" KeyDown="DataBaseConnection_KeyDown" />
                </Grid>
                <Grid>
                    <Label Content="Логин" HorizontalAlignment="Left"
Margin="0,0,0,0" VerticalAlignment="Center"
Style="{DynamicResource MainLableStyle}" />
                    <TextBox x:Name="Login" HorizontalAlignment="Left"
Margin="80,0,0,0" VerticalAlignment="Center" Width="250"

```



```

Style="{DynamicResource MainTextBoxStyle}"
KeyDown="Login_KeyDown"/>
</Grid>
<Grid>
    <Label Content="Пароль" HorizontalAlignment="Left"
Margin="0,0,0,0" VerticalAlignment="Center"
Style="{DynamicResource MainLabelStyle}"/>
    <TextBox x:Name="Password"
HorizontalAlignment="Left" Margin="80,0,0,0"
VerticalAlignment="Center" Width="250" Style="{DynamicResource
MainTextBoxStyle}" KeyDown="Password_KeyDown"/>
</Grid>
<Button x:Name="Enter" Content="Войти"
Margin="0,10,0,0" Width="200" FontSize="16"
HorizontalAlignment="Center" Style="{DynamicResource
MainButtonStyle}" Click="Enter_Click"/>
</StackPanel>
</StackPanel>

<Grid VerticalAlignment="Bottom">
    <Label Content="v1.0.5" HorizontalAlignment="Left"/>
    <Label Content="Early access" Visibility="Collapsed"
HorizontalAlignment="Right"/>
</Grid>

<ProgressBar x:Name="LoadView" VerticalAlignment="Bottom"
Visibility="Collapsed" Style="{DynamicResource
MainProgressBarStyle}"/>
</Grid>
</Window>

```

Authorization.xaml.cs

```

using System;
using System.Configuration;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;

namespace Cinema
{
    /// <summary>
    /// Логика взаимодействия для Authorization.xaml
    /// </summary>
    public partial class Authorization : Window
    {
        public Authorization()
        {
            InitializeComponent();
        }
    }
}

```

```

// Передаваемые данные
public static class TransmittedData
{
    public static int idEmployee { get; set; }
    public static int idSelectedMovie { get; set; }
    public static int idSelectedGenre { get; set; }
    public static int idSelectedActor { get; set; }
    public static int idSelectedSession { get; set; }
    public static int idSelectedEmployee { get; set; }
    public static int idSelectedCashierMovie { get; set; }
    public static int idSelectedCashierSession { get; set; }
    public static int idSelectedMovieAnalysis { get; set; }
}

private void Window_Loaded(object sender, RoutedEventArgs e)
{
    try
    {
        if (File.Exists("ConnectionData.ini"))
        {
            DataBaseConnection.Text =
File.ReadAllText("ConnectionData.ini");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private async void AuthorizationUser()
{
    try
    {
        LoadView.Visibility = Visibility.Visible;

        string password = Password.Text;
        string login = Login.Text;
        string dataSource = DataBaseConnection.Text;

        var userData = await Task.Run(() =>
LoadDataBase(password, login, dataSource));

        LoadView.Visibility = Visibility.Collapsed;

        if (userData != null)
        {
            switch (userData)
            {
                case "Букер":
                    BookerPanel bookerPanel = new BookerPanel();
                    bookerPanel.Show();

```

```

        this.Hide();
        break;

    case "Кассир":
        CashierPanel cashierPanel = new CashierPanel();
        cashierPanel.Show();
        this.Hide();
        break;

    case "Директор":
        DirectorsPanel directorsPanel = new DirectorsPanel();
        directorsPanel.Show();
        this.Hide();
        break;

    case "Администратор":
        AdminPanel adminPanel = new AdminPanel();
        adminPanel.Show();
        this.Hide();
        break;
    }
}
else
{
    MessageBox.Show("Неверный логин или пароль",
"Внимание", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    return;
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxIcon.Error);
}
}

public async Task<string> LoadDataBase(string password, string
login, string newDataSource)
{
    try
    {
        ConnectionStringChanged(newDataSource);

        using (var dataBase = new CinemaEntities())
        {
            var userData = (from employee in dataBase.Employee
                join
                role in dataBase.Role on employee.IDRole equals
                role.IDRole
                where (employee.Login == login &&
employee.Password == password)
                select new
                {
                    employee.IDEmployee,
                    Role = role.Title
                }).FirstOrDefault();

            if (userData != null)
            {
                TransmittedData.idEmployee = userData.IDEmployee;

                return userData.Role;
            }
            else
            {
                return null;
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxIcon.Error);
        return null;
    }
}

private void ConnectionStringChanged(string newDataSource)
{
    try
    {
        string connectionStringName = "CinemaEntities";

        Configuration config =
ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.
None);
        ConnectionStringSettings settings =
config.ConnectionStrings.ConnectionStrings[connectionStringName];

        if (settings != null)
        {
            string currentConnectionString = settings.ConnectionString;

            string currentConnectionName = "";
            string[] tempData = currentConnectionString.Split(';');
            for (int i = 0; i < tempData.Length; i++)
            {
                if (tempData[i].Contains("data source="))
                {
                    currentConnectionName = tempData[i].Remove(0,
40);
                    break;
                }
            }
        }
    }
}

```

```

        string newConnectionString =
currentConnectionString.Replace(currentConnectionName,
newDataSource);

        settings.ConnectionString = newConnectionString;
        config.Save(ConfigurationSaveMode.Modified);

ConfigurationManager.RefreshSection("connectionStrings");

        File.WriteAllText("ConnectionData.ini", newDataSource);
    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
}

private void Enter_Click(object sender, RoutedEventArgs e)
{
    AuthorizationUser();
}

private void DataBaseConnection_KeyDown(object sender,
KeyEventArgs e)
{
    try
    {
        if (e.Key == Key.Enter)
        {
            Login.Focus();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void Login_KeyDown(object sender, KeyEventArgs e)
{
    try
    {
        if (e.Key == Key.Enter)
        {
            Password.Focus();
        }
    }
    catch (Exception ex)
    {

```

```

        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void Password_KeyDown(object sender, KeyEventArgs e)
{
    try
    {
        if (e.Key == Key.Enter)
        {
            AuthorizationUser();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
    try
    {
        Application.Current.Shutdown();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
}

```

BookerPanel.xaml

```

<Window x:Class="Cinema.BookerPanel"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
xmlns:local="clr-namespace:Cinema"
mc:Ignorable="d"
Title="Панель менеджера" Height="550" Width="1050"
MinHeight="550" MinWidth="1050" Loaded="Window_Loaded"
Closing="Window_Closing" WindowStartupLocation="CenterScreen">

<Grid Style="{DynamicResource MainColorStyle}">
    <Grid VerticalAlignment="Top">

```

```

        <WrapPanel Style="{DynamicResource
SecondColorWrapPanelStyle}">
            <Button x:Name="MoviePage" Content="Фильмы"
Style="{DynamicResource TabOffButtonStyle}"
Click="SelectedPage_Click"/>
            <Button x:Name="SessionPage" Content="Сеансы"
Style="{DynamicResource TabOffButtonStyle}"
Click="SelectedPage_Click"/>
            <Button x:Name="TicketPage" Content="Продажи"
Style="{DynamicResource TabOffButtonStyle}"
Click="SelectedPage_Click"/>
        </WrapPanel>
        <WrapPanel HorizontalAlignment="Right">
            <Button x:Name="Help" Content="Справка" Width="100"
Style="{DynamicResource TabHelpButtonStyle}"
Click="Help_Click"/>
            <Button x:Name="Exit" Content="Выйти" Width="100"
Style="{DynamicResource TabExitButtonStyle}" Click="Exit_Click"/>
        </WrapPanel>
    </Grid>
    <Frame x:Name="PageManager" Content="MainFrame"
Margin="0,25,0,0" NavigationUIVisibility="Hidden"
Background="White"/>
</Grid>
</Window>

```

BookerPanel.xaml.cs

```

using System;
using System.Diagnostics;
using System.IO;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using static Cinema.Authorization;

namespace Cinema
{
    /// <summary>
    /// Логика взаимодействия для BookerPanel.xaml
    /// </summary>
    public partial class BookerPanel : Window
    {
        public BookerPanel()
        {
            InitializeComponent();
        }

        public bool exitMode;
        public Button currentSelectedButton;

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            try

```

```

        {
            PageManager.Navigate(new MovieControls());

            currentSelectedButton = MoviePage;
            currentSelectedButton.Style =
(Style)Application.Current.Resources["TabOnButtonStyle"];
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }

    private void SelectedPage_Click(object sender, RoutedEventArgs
e)
    {
        try
        {
            if (currentSelectedButton != null)
            {
                currentSelectedButton.Style =
(Style)Application.Current.Resources["TabOffButtonStyle"];
            }

            currentSelectedButton = sender as Button;

            currentSelectedButton.Style =
(Style)Application.Current.Resources["TabOnButtonStyle"];

            switch (currentSelectedButton.Name)
            {
                case "MoviePage":
                    PageManager.Navigate(new MovieControls());
                    break;

                case "SessionPage":
                    PageManager.Navigate(new SessionControls());
                    break;

                case "TicketPage":
                    PageManager.Navigate(new TicketAnalysis());
                    break;
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }

    private void Exit_Click(object sender, RoutedEventArgs e)

```

```

{
    try
    {
        exitMode = true;

        TransmittedData.idEmployee = 0;
        TransmittedData.idSelectedMovie = 0;
        TransmittedData.idSelectedGenre = 0;
        TransmittedData.idSelectedActor = 0;
        TransmittedData.idSelectedSession = 0;
        TransmittedData.idSelectedEmployee = 0;

        Authorization authorization = new Authorization();
        authorization.Show();

        this.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

public void TicketPage_Click(object sender, RoutedEventArgs e)
{
    TicketAnalysisOpen();
}

public void TicketAnalysisOpen()
{
    try
    {
        PageManager.Navigate(new TicketAnalysis());
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

public void MovieAnalysisOpen()
{
    try
    {
        PageManager.Navigate(new MovieAnalysis());
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
}

private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
    try
    {
        if (!exitMode)
        {
            Application.Current.Shutdown();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void Help_Click(object sender, RoutedEventArgs e)
{
    try
    {
        Uri resourceUri = new
        Uri("pack://application:,,,/Resource/HelpDocument.pdf",
        UriKind.Absolute);

        Stream resourceStream =
        Application.GetResourceStream(resourceUri)?.Stream;
        string tempFileName = Path.GetTempFileName() + ".pdf";
        using (FileStream fileStream = new
        FileStream(tempFileName, FileMode.Create, FileAccess.Write))
        {
            resourceStream.CopyTo(fileStream);
        }

        Process.Start(new ProcessStartInfo(tempFileName) {
        UseShellExecute = true });
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
}

```

CashierPanel.xaml

```

<Window x:Class="Cinema.CashierPanel"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

```

```

xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
    xmlns:local="clr-namespace:Cinema"
    mc:Ignorable="d"
    Title="Панель кассира" Height="550" Width="1050"
MinHeight="550" MinWidth="1050" Loaded="Window_Loaded"
Closing="Window_Closing" WindowStartupLocation="CenterScreen">

    <Grid Style="{DynamicResource MainColorStyle}">
        <Grid VerticalAlignment="Top">
            <WrapPanel Style="{DynamicResource
SecondColorWrapPanelStyle}">
                <Button x:Name="MainPage" Content="Главная страница"
Style="{DynamicResource TabOffButtonStyle}"
Click="SelectedPage_Click"/>
                <Button x:Name="CurrentMoviesPage" Content="Фильмы
сегодня" Style="{DynamicResource TabOffButtonStyle}"
Click="SelectedPage_Click"/>
                <Button x:Name="GoBack" Content="Назад"
Style="{DynamicResource TabOffButtonStyle}"
Click="SelectedBackPage_Click"/>
            </WrapPanel>
            <WrapPanel HorizontalAlignment="Right">
                <Button x:Name="Help" Content="Справка" Width="100"
Style="{DynamicResource TabHelpButtonStyle}"
Click="Help_Click"/>
                <Button x:Name="Exit" Content="Выйти" Width="100"
Style="{DynamicResource TabExitButtonStyle}" Click="Exit_Click"/>
            </WrapPanel>
        </Grid>
        <Frame x:Name="PageManager" Content="MainFrame"
Margin="0,25,0,0" NavigationUIVisibility="Hidden"
Background="White"/>
    </Grid>
</Window>

```

CashierPanel.xaml.cs

```

using Cinema.Pages;
using System;
using System.Diagnostics;
using System.IO;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using static Cinema.Authorization;

namespace Cinema
{
    /// <summary>
    /// Логика взаимодействия для CashierPanel.xaml
    /// </summary>
    public partial class CashierPanel : Window
    {

```

```

        public CashierPanel()
        {
            InitializeComponent();
        }

        public bool exitMode;
        public Button currentSelectedButton;
        public string currentSelectedPage;

        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            try
            {
                MoviePageOpen();

                currentSelectedButton = MainPage;
                currentSelectedButton.Style =
                (Style)Application.Current.Resources["TabOnButtonStyle"];
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Ошибка",
                MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }

        private void SelectedPage_Click(object sender, RoutedEventArgs
e)
        {
            try
            {
                if (currentSelectedButton != null)
                {
                    currentSelectedButton.Style =
                    (Style)Application.Current.Resources["TabOffButtonStyle"];
                }

                currentSelectedButton = sender as Button;

                currentSelectedButton.Style =
                (Style)Application.Current.Resources["TabOnButtonStyle"];

                switch (currentSelectedButton.Name)
                {
                    case "CurrentMoviesPage":
                        CurrentMoviesPageOpen();
                        break;

                    case "MainPage":
                        MoviePageOpen();
                        break;
                }
            }
        }
    }
}

```

```

        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }

    private void SelectedBackPage_Click(object sender,
RoutedEventArgs e)
    {
        try
        {
            Page currentPage = PageManager.Content as Page;

            if (currentPage.GetType().Name == "PlacesCashierControls")
            {
                SessionPageOpen();
            }
            else
            if (currentPage.GetType().Name ==
"SessionCashierControls")
            {
                MoviePageOpen();
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }

    public void CurrentMoviesPageOpen()
    {
        try
        {
            PageManager.Navigate(new CurrentMovieCashierControls());

            GoBack.Style =
(Style)Application.Current.Resources["TabOffButtonStyle"];
            CurrentMoviesPage.Style =
(Style)Application.Current.Resources["TabOnButtonStyle"];
            currentSelectedButton = CurrentMoviesPage;
            GoBack.Content = "";
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }

    public void MoviePageOpen()
    {
        try
        {
            PageManager.Navigate(new MovieCashierControls());

            GoBack.Style =
(Style)Application.Current.Resources["TabOffButtonStyle"];
            MainPage.Style =
(Style)Application.Current.Resources["TabOnButtonStyle"];
            currentSelectedButton = MainPage;
            GoBack.Content = "";
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }

    public void SessionPageOpen()
    {
        try
        {
            PageManager.Navigate(new SessionCashierControls());

            MainPage.Style =
(Style)Application.Current.Resources["TabOffButtonStyle"];
            CurrentMoviesPage.Style =
(Style)Application.Current.Resources["TabOffButtonStyle"];
            GoBack.Style =
(Style)Application.Current.Resources["TabOnButtonStyle"];
            currentSelectedButton = GoBack;
            GoBack.Content = "Фильмы";
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }

    public void PlacesPageOpen()
    {
        try
        {
            PageManager.Navigate(new PlacesCashierControls());
            GoBack.Content = "Сеансы";
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }

```

```

    }

    private void Exit_Click(object sender, RoutedEventArgs e)
    {
        try
        {
            exitMode = true;

            TransmittedData.idSelectedCashierMovie = 0;
            TransmittedData.idSelectedCashierSession = 0;

            Authorization authorization = new Authorization();
            authorization.Show();

            this.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message, "Ошибка",
            MessageBoxButton.OK, MessageBoxImage.Error);
        }

        private void Window_Closing(object sender,
        System.ComponentModel.CancelEventArgs e)
        {
            try
            {
                if (!exitMode)
                {
                    Application.Current.Shutdown();
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Ошибка",
                MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }

        private void Help_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                Uri resourceUri = new
                Uri("pack://application:,,,/Resource/HelpDocument.pdf",
                UriKind.Absolute);

                Stream resourceStream =
                Application.GetResourceStream(resourceUri)?.Stream;
                string tempFileName = Path.GetTempFileName() + ".pdf";
                using (FileStream fileStream = new
                FileStream(tempFileName, FileMode.Create, FileAccess.Write))

```

```

        {
            resourceStream.CopyTo(fileStream);
        }

        Process.Start(new ProcessStartInfo(tempFileName) {
        UseShellExecute = true });
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

```

DirectorsPanel.xaml

```

<Window x:Class="Cinema.DirectorsPanel"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
xmlns:local="clr-namespace:Cinema"
mc:Ignorable="d"

    Title="Панель директора" Height="550" Width="1050"
    MinHeight="550" MinWidth="1050" Loaded="Window_Loaded"
    Closing="Window_Closing" WindowStartupLocation="CenterScreen">

    <Grid Style="{DynamicResource MainColorStyle}">
        <Grid VerticalAlignment="Top">
            <WrapPanel Style="{DynamicResource
            SecondColorWrapPanelStyle}">
                <Button x:Name="EmployeePage" Content="Сотрудники"
                Style="{DynamicResource TabOffButtonStyle}"
                Click="SelectedPage_Click"/>
                <Button x:Name="TicketPage" Content="Продажи"
                Style="{DynamicResource TabOffButtonStyle}"
                Click="SelectedPage_Click"/>
            </WrapPanel>
            <WrapPanel HorizontalAlignment="Right">
                <Button x:Name="Help" Content="Справка" Width="100"
                Style="{DynamicResource TabHelpButtonStyle}"
                Click="Help_Click"/>
                <Button x:Name="Exit" Content="Выйти" Width="100"
                Style="{DynamicResource TabExitButtonStyle}" Click="Exit_Click"/>
            </WrapPanel>
        </Grid>

        <Frame x:Name="PageManager" Content="MainFrame"
        Margin="0,25,0,0" NavigationUIVisibility="Hidden"
        Background="White"/>

```



```
</Grid>
</Window>
```

DirectorsPanel.xaml.cs

```
using System;
using System.Diagnostics;
using System.IO;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using static Cinema.Authorization;

namespace Cinema
{
    /// <summary>
    /// Логика взаимодействия для DirectorsPanel.xaml
    /// </summary>
    public partial class DirectorsPanel : Window
    {
        public DirectorsPanel()
        {
            InitializeComponent();

            public bool exitMode;
            public Button currentSelectedButton;

            private void Window_Loaded(object sender, RoutedEventArgs e)
            {
                try
                {
                    PageManager.Navigate(new EmployeeControls());

                    currentSelectedButton = EmployeePage;
                    currentSelectedButton.Style =
                        (Style)Application.Current.Resources["TabOnButtonStyle"];
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message, "Ошибка",
                        MessageBoxButton.OK, MessageBoxImage.Error);
                }
            }

            private void SelectedPage_Click(object sender, RoutedEventArgs
e)
            {
                try
                {
                    if (currentSelectedButton != null)
                    {
                        currentSelectedButton.Style =
                            (Style)Application.Current.Resources["TabOffButtonStyle"];

```

```

                }

                currentSelectedButton = sender as Button;

                currentSelectedButton.Style =
                    (Style)Application.Current.Resources["TabOnButtonStyle"];

                switch (currentSelectedButton.Name)
                {
                    case "EmployeePage":
                        PageManager.Navigate(new EmployeeControls());
                        break;

                    case "TicketPage":
                        TicketAnalysisOpen();
                        break;
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Ошибка",
                    MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }

        private void Exit_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                exitMode = true;

                TransmittedData.idEmployee = 0;
                TransmittedData.idSelectedMovie = 0;
                TransmittedData.idSelectedGenre = 0;
                TransmittedData.idSelectedActor = 0;
                TransmittedData.idSelectedSession = 0;
                TransmittedData.idSelectedEmployee = 0;

                Authorization authorization = new Authorization();
                authorization.Show();

                this.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Ошибка",
                    MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }

        public void TicketAnalysisOpen()
        {
            try

```

```

    {
        PageManager.Navigate(new TicketAnalysis());
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

public void MovieAnalysisOpen()
{
    try
    {
        PageManager.Navigate(new MovieAnalysis());
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void Window_Closing(object sender,
System.ComponentModel.CancelEventArgs e)
{
    try
    {
        if (!exitMode)
        {
            Application.Current.Shutdown();
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

private void Help_Click(object sender, RoutedEventArgs e)
{
    try
    {
        Uri resourceUri = new
        Uri("pack://application:;./Resource/HelpDocument.pdf",
        UriKind.Absolute);

        Stream resourceStream =
        Application.GetResourceStream(resourceUri)?.Stream;
        string tempFileName = Path.GetTempFileName() + ".pdf";
        using (FileStream fileStream = new
        FileStream(tempFileName, FileMode.Create, FileAccess.Write))

```

```

    {
        resourceStream.CopyTo(fileStream);
    }

    Process.Start(new ProcessStartInfo(tempFileName) {
    UseShellExecute = true });
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка",
        MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
}
}

```

Light.xaml

```

<ResourceDictionary
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

    <Style x:Key="MainColorStyle" TargetType="Grid">
        <Setter Property="Background" Value="White"/>
    </Style>
    <Style x:Key="SecondColorStyle" TargetType="Grid">
        <Setter Property="Background" Value="Aqua"/>
        <Setter Property="Height" Value="35"/>
    </Style>
    <Style x:Key="SecondColorRectangleStyle"
    TargetType="Rectangle">
        <Setter Property="Fill" Value="Aqua"/>
        <Setter Property="Height" Value="35"/>
    </Style>
    <Style x:Key="SecondColorRectanglePlaceStyle"
    TargetType="Rectangle">
        <Setter Property="Fill" Value="Aqua"/>
        <Setter Property="Opacity" Value="0.5"/>
    </Style>
    <Style x:Key="SecondColorSeparatorsRectangleStyle"
    TargetType="Rectangle">
        <Setter Property="Fill" Value="Black"/>
        <Setter Property="Height" Value="25"/>
        <Setter Property="Width" Value="2"/>
        <Setter Property="Opacity" Value="0.2"/>
    </Style>
    <Style x:Key="SecondColorWrapPanelStyle"
    TargetType="WrapPanel">
        <Setter Property="Background" Value="#00dbdb"/>
        <Setter Property="Height" Value="25"/>
    </Style>

    <Style x:Key="TabOnButtonStyle" TargetType="Button">
        <Setter Property="Focusable" Value="False"/>

```

```

<Setter Property="IsTabStop" Value="False"/>
<Setter Property="Width" Value="150"/>
<Setter Property="Height" Value="25"/>
<Setter Property="FontSize" Value="16"/>
<Setter Property="Foreground" Value="Black"/>
<Setter Property="Template">
    <Setter.Value>
        <ControlTemplate TargetType="{x:Type Button}">
            <Border Background="Aqua" CornerRadius="5,5,0,0">
                <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
            </Border>
        </ControlTemplate>
    </Setter.Value>
</Setter>
</Style>
<Style x:Key="TabOffButtonStyle" TargetType="Button">
    <Setter Property="Focusable" Value="False"/>
    <Setter Property="IsTabStop" Value="False"/>
    <Setter Property="Width" Value="150"/>
    <Setter Property="Height" Value="25"/>
    <Setter Property="FontSize" Value="16"/>
    <Setter Property="Foreground" Value="Black"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type Button}">
                <Border Background="Transparent"
CornerRadius="5,5,0,0" Opacity="0.6">
                    <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>
<Style x:Key="TabHelpButtonStyle" TargetType="Button">
    <Setter Property="Focusable" Value="False"/>
    <Setter Property="IsTabStop" Value="False"/>
    <Setter Property="Width" Value="150"/>
    <Setter Property="Height" Value="25"/>
    <Setter Property="FontSize" Value="16"/>
    <Setter Property="Foreground" Value="Black"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type Button}">
                <Border Background="Aqua" CornerRadius="5,0,0,0"
BorderThickness="0,0,1,0" BorderBrush="#00dbdb">
                    <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>

```

```

</Style>
<Style x:Key="TabExitButtonStyle" TargetType="Button">
    <Setter Property="Focusable" Value="False"/>
    <Setter Property="IsTabStop" Value="False"/>
    <Setter Property="Width" Value="150"/>
    <Setter Property="Height" Value="25"/>
    <Setter Property="FontSize" Value="16"/>
    <Setter Property="Foreground" Value="Black"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type Button}">
                <Border Background="Aqua" CornerRadius="0,5,0,0"
BorderThickness="1,0,0,0" BorderBrush="#00dbdb">
                    <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>
<Style x:Key="TabControlButtonStyle" TargetType="Button">
    <Setter Property="FontSize" Value="16"/>
    <Setter Property="Foreground" Value="Black"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type Button}">
                <Border Background="White" CornerRadius="2"
BorderThickness="0" BorderBrush="Black">
                    <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>
<Style x:Key="MainButtonStyle" TargetType="Button">
    <Setter Property="FontSize" Value="16"/>
    <Setter Property="Foreground" Value="Black"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type Button}">
                <Border Background="White" CornerRadius="2"
BorderThickness="1" BorderBrush="Black">
                    <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
                </Border>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>
<Style x:Key="MainClearButtonStyle" TargetType="Button"
BasedOn="{StaticResource MainButtonStyle}">
    <Setter Property="Template">

```

```

        <Setter.Value>
        <ControlTemplate TargetType="{x:Type Button}">
            <Border Background="White">
                <ContentPresenter HorizontalAlignment="Center"
VerticalAlignment="Center"/>
            </Border>
        </ControlTemplate>
    </Setter.Value>
</Setter>
</Style>

<Style x:Key="TitleTextBlockStyle" TargetType="TextBlock">
    <Setter Property="VerticalAlignment" Value="Center"/>
    <Setter Property="HorizontalAlignment" Value="Center"/>
    <Setter Property="FontSize" Value="20"/>
    <Setter Property="FontWeight" Value="Bold"/>
    <Setter Property="Foreground" Value="Black"/>
    <Setter Property="Background" Value="Transparent"/>
    <Setter Property="TextAlignment" Value="Center"/>
</Style>

<Style x:Key="MainTextBlockStyle" TargetType="TextBlock">
    <Setter Property="FontSize" Value="16"/>
    <Setter Property="Foreground" Value="Black"/>
    <Setter Property="Background" Value="Transparent"/>
</Style>

<Style x:Key="MainLableStyle" TargetType="Label">
    <Setter Property="FontSize" Value="16"/>
    <Setter Property="Foreground" Value="Black"/>
</Style>

<Style x:Key="MainTextBoxStyle" TargetType="TextBox">
    <Setter Property="FontSize" Value="16"/>
    <Setter Property="Foreground" Value="Black"/>
    <Setter Property="Background" Value="Transparent"/>
    <Setter Property="BorderThickness" Value="0,0,0,1"/>
    <Setter Property="BorderBrush" Value="Black"/>
</Style>

<Style x:Key="MainComboBoxStyle" TargetType="ComboBox">
    <Setter Property="FontSize" Value="16"/>
    <Setter Property="Foreground" Value="Black"/>
    <Setter Property="SelectedIndex" Value="0"/>
    <Setter Property="BorderThickness" Value="0,0,0,1"/>
    <Setter Property="BorderBrush" Value="Black"/>
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="ComboBox">
                <Grid>
                    <ToggleButton Grid.Column="2" Focusable="false"
IsChecked="{Binding
Path=IsDropDownOpen,Mode=TwoWay,RelativeSource={RelativeSou
rce TemplatedParent}}">
                        <ToggleButton.Template>
                            <ControlTemplate>
                                <Grid>
                                    <Grid.ColumnDefinitions>
                                        <ColumnDefinition Width="5*" />
                                        <ColumnDefinition Width="*" />
                                    </Grid.ColumnDefinitions>
                                    <Border x:Name="Border"
Grid.ColumnSpan="2" CornerRadius="0" Background="Transparent"
BorderBrush="Black" BorderThickness="0,0,0,1" />
                                    <Path x:Name="Arrow" Grid.Column="1"
Fill="Black" HorizontalAlignment="Center"
VerticalAlignment="Center" Data="M 0 0 L 4 4 L 8 0 Z"/>
                                </Grid>
                                <ControlTemplate.Triggers>
                                    <Trigger
Property="ToggleButton.IsMouseOver" Value="true">
                                        <Setter TargetName="Border"
Property="Background" Value="Aqua" />
                                    </Trigger>
                                    <Trigger Property="ToggleButton.IsChecked"
Value="true">
                                        <Setter TargetName="Border"
Property="Background" Value="Transparent" />
                                    </Trigger>
                                </ControlTemplate.Triggers>
                            </ControlTemplate>
                        </ToggleButton.Template>
                    </ToggleButton>
                    <ContentPresenter Name="ContentSite"
IsHitTestVisible="False" Content="{TemplateBinding
SelectionBoxItem}" ContentTemplate="{TemplateBinding
SelectionBoxItemTemplate}"
ContentTemplateSelector="{TemplateBinding ItemTemplateSelector}"
Margin="3" />
                    <TextBox x:Name="PART_EditableTextBox"
Visibility="Hidden" IsReadOnly="{TemplateBinding IsReadOnly}"/>
                    <Popup Name="Popup" Placement="Bottom"
IsOpen="{TemplateBinding IsDropDownOpen}"
AllowsTransparency="True" Focusable="False"
PopupAnimation="Slide">
                        <Grid Name="DropDown"
SnapsToDevicePixels="True" MinWidth="{TemplateBinding
ActualWidth}" MaxHeight="{TemplateBinding
MaxDropDownHeight}">
                            <Border x:Name="DropDownBorder"
Background="White" BorderBrush="Black" BorderThickness="1"/>
                            <ScrollViewer Margin="1"
SnapsToDevicePixels="True">
                                <StackPanel IsItemsHost="True" />
                            </ScrollViewer>
                        </Grid>
                    </Popup>
                </Grid>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>

```

```

        </ControlTemplate>
    </Setter.Value>
</Setter>
</Style>
<Style x:Key="MainSelectedComboBoxStyle" TargetType="{x:Type
ComboBoxItem}">
    <Setter Property="Template">
        <Setter.Value>
            <ControlTemplate TargetType="{x:Type ComboBoxItem}">
                <Border x:Name="myBorder">
                    <ContentPresenter />
                </Border>
                <ControlTemplate.Triggers>
                    <Trigger Property="IsSelected" Value="True">
                        <Setter TargetName="myBorder"
Property="Background" Value="Aqua" />
                    </Trigger>
                    <Trigger Property="IsMouseOver" Value="True">
                        <Setter TargetName="myBorder"
Property="Background" Value="#adffff" />
                    </Trigger>
                </ControlTemplate.Triggers>
            </ControlTemplate>
        </Setter.Value>
    </Setter>
</Style>

<Style x:Key="MainDataGridStyle" TargetType="DataGrid">
    <Setter Property="FontSize" Value="12"/>
    <Setter Property="Foreground" Value="Black"/>
    <Setter Property="Background" Value="Transparent"/>
    <Setter Property="BorderBrush" Value="Transparent"/>
    <Setter Property="BorderThickness" Value="0"/>
    <Setter Property="ColumnWidth" Value="*/>
</Style>

<Style x:Key="MainDatePickerStyle" TargetType="DatePicker">
    <Setter Property="FontSize" Value="16"/>
    <Setter Property="Foreground" Value="Black"/>
    <Setter Property="Background" Value="Transparent"/>
    <Setter Property="BorderThickness" Value="0,0,0,1"/>
    <Setter Property="BorderBrush" Value="Black"/>
</Style>

<Style x:Key="SecondDatePickerStyle" TargetType="DatePicker">
    <Setter Property="FontSize" Value="12"/>
    <Setter Property="Foreground" Value="Black"/>
    <Setter Property="Background" Value="Transparent"/>
    <Setter Property="BorderBrush" Value="Transparent"/>
    <Setter Property="IsEnabled" Value="False"/>
</Style>

<Style x:Key="MainProgressBarStyle" TargetType="ProgressBar">
    <Setter Property="Height" Value="5"/>

```

```

        <Setter Property="IsIndeterminate" Value="True"/>
    <Setter Property="Foreground" Value="Aqua"/>
    <Setter Property="BorderBrush" Value="Transparent"/>
</Style>

<Style x:Key="MainCheckBoxStyle" TargetType="CheckBox">
    </Style>
</ResourceDictionary>

```

App.xaml

```

<Application x:Class="Cinema.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="clr-namespace:Cinema"
    StartupUri="Windows/Authorization.xaml">
    <Application.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary Source="Style/Light.xaml"/>
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </Application.Resources>
</Application>

```

РУКОВОДСТВО ОПЕРАТОРА

Функциональным назначением программы является автоматизация продажи билетов в кинотеатре.

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

- Управление сотрудниками;
- Управление фильмами;
- Управление сеансами;
- Настройка конфигурации зала;
- Резервное копирование базы данных;
- Продажа билетов на выбранный сеанс фильма;
- Расчет количества проданных билетов на фильмы;
- Расчет количества проданных билетов на сеансы выбранного фильма;
- Сохранение отчета по количеству проданных билетов на фильмы в файл формата pdf;
- Сохранение отчета по количеству проданных билетов на сеансы выбранного фильма в файл формата pdf;
- Открытие pdf файла проданного билета для его печати или сохранения;

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

В состав технических средств должен входить IBM-совместимый персональный компьютер (ПЭВМ), включающий себя:

- процессор с тактовой частотой, 1,5 ГГц, не менее;

- оперативную память объемом 1 гб, не менее;
- жесткий диск со свободным местом 500 Мб, не менее;
- монитор, с разрешением экрана 1280*720, не менее;
- компьютерная мышь;
- клавиатура;

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционной системы Windows 7/8/10/11.

Все пользователи должны обладать навыками работы с графическим пользовательским интерфейсом операционной системы.

Для запуска программного продукта запустить исполняемый файл Cinema.exe.

После запуска программа откроется окно авторизации (Рисунок 1). Оно представлено именем базы данных (именем сервера), логином и паролем.

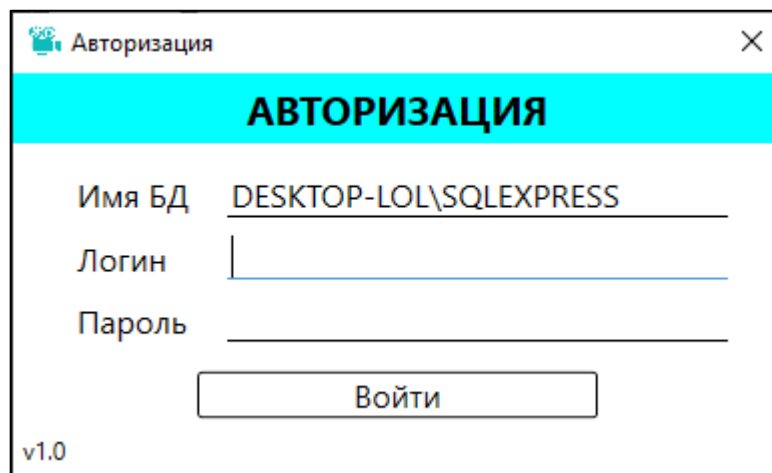


Рисунок 1 – Окно авторизации

Пользователь может авторизоваться по четырем ролям: администратор, кассир, букер(менеджер), директор.

После авторизации под ролью администратора мы видим панель администратора (Рисунок 2). Она открывается на окне “Сотрудники” здесь мы можем найти и просмотреть информацию о сотрудниках, а также добавить удалить или редактировать сотрудников.

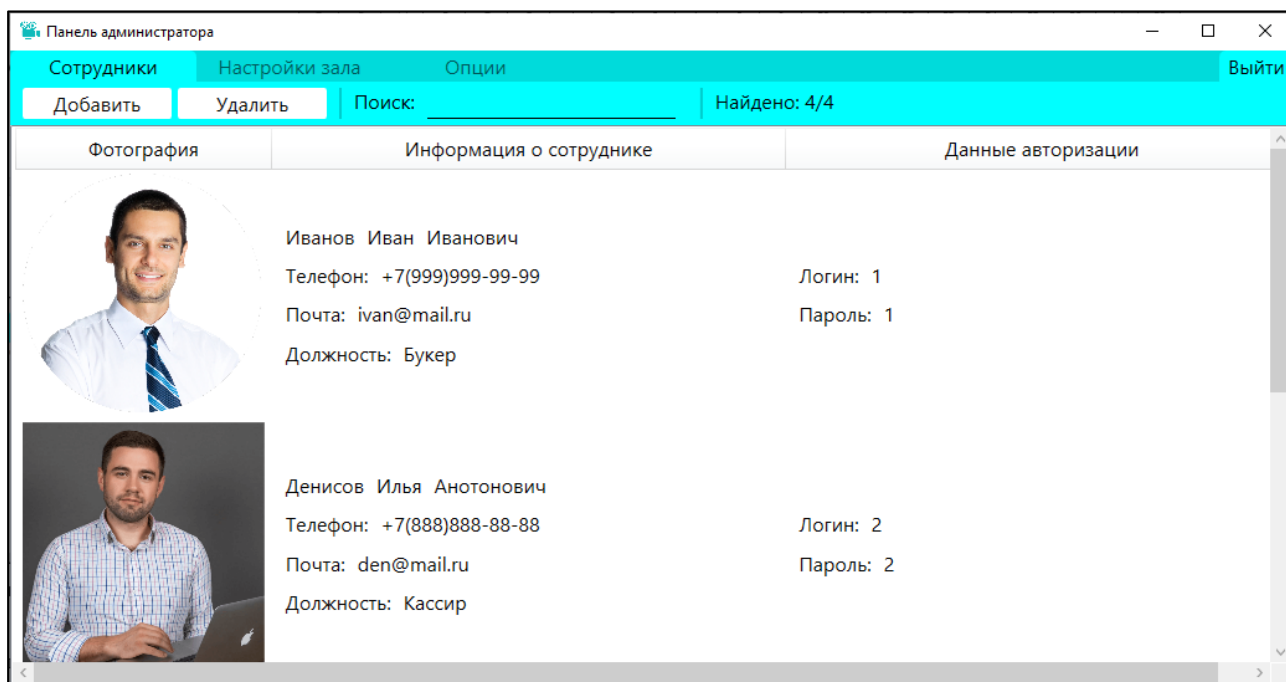


Рисунок 2 – Панель администратора с открытым окном сотрудники

Для того чтобы добавить сотрудника необходимо нажать на кнопку “Добавить” после чего откроется окно добавления сотрудник (Рисунок 3).

Сотрудник

СОТРУДНИК

Фамилия:

Имя:

Отчество:

Телефон:

Эл. Почта:

Должность: Букер

Логин:

Пароль:

Фотография:

Рисунок 3 – Окно добавления сотрудника

После добавления данных о сотруднике приложение отобразит сообщение о успешном сохранение (Рисунок 4).

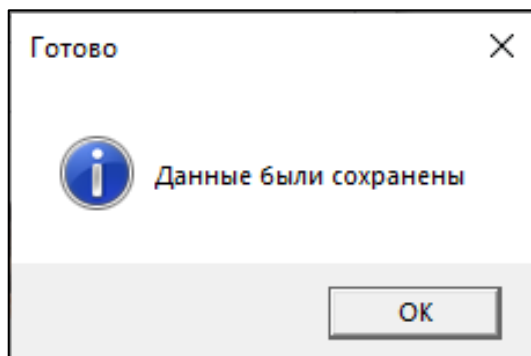


Рисунок 4 – Сообщение о успешном сохранение данных

Для редактирования данных сотрудника необходимо дважды нажать на редактируемого сотрудника, после чего откроется окно редактирования сотрудника (Рисунок 3).

Для удаления сотрудника необходимо выбрать сотрудника и нажать на кнопку удалить после чего отойться сообщение-подтверждение удаления (Рисунок 4).

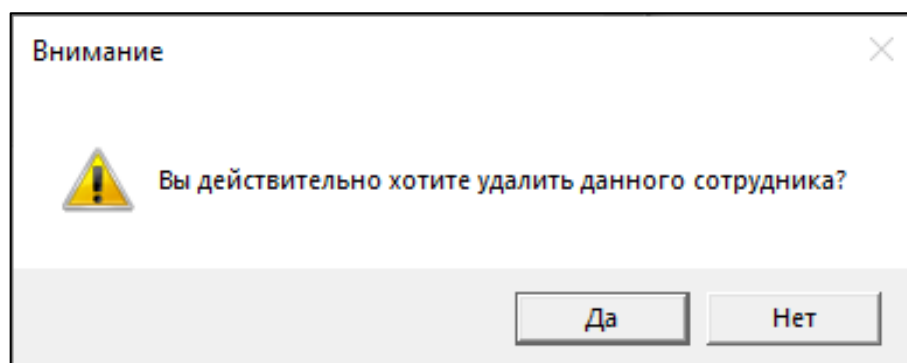


Рисунок 4 – Сообщение подтверждение удаления сотрудника

Также панель администратора обладает настройкой конфигурацией зала (Рисунок 5). В данном окне пользователь может настроить размерность зала выбрав количество рядов и мест, а также убрать лишние места в зале. После настройки зала следует сохранить конфигурацию при помощи кнопки “Сохранить”.

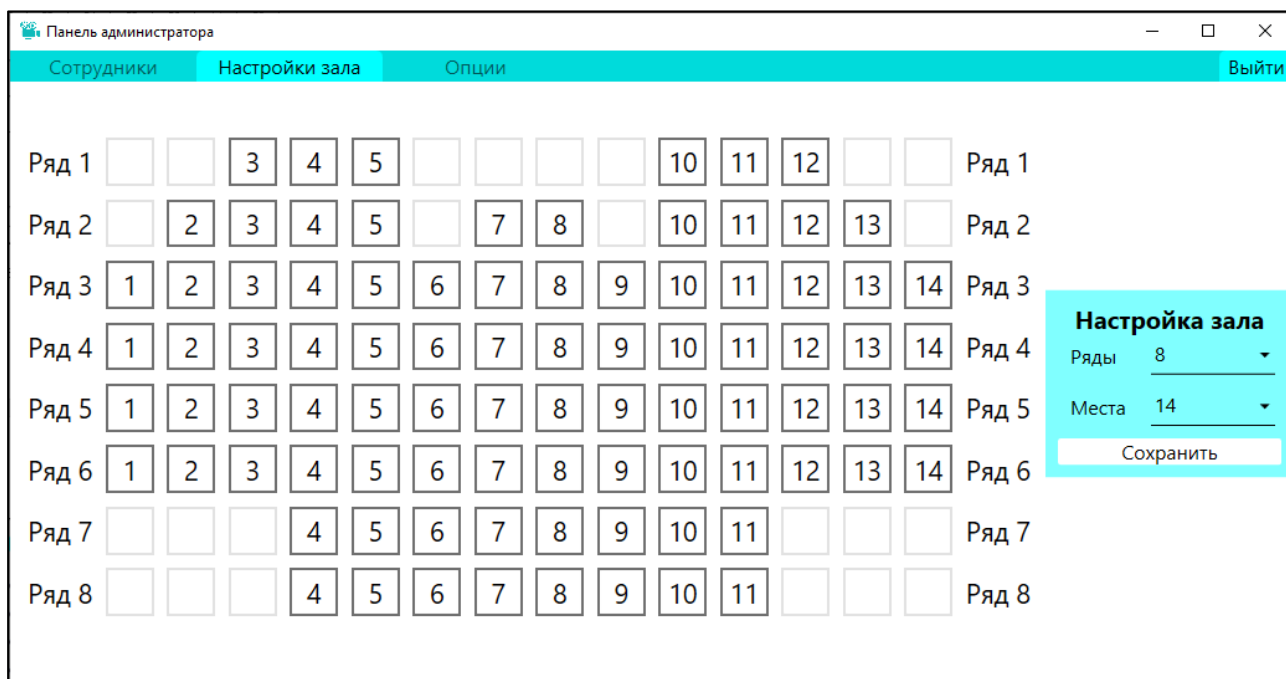


Рисунок 5 – Панель администратора с открытым окном настройки зала

После сохранения конфигурации откроется сообщение о успешном сохранение (Рисунок 6).

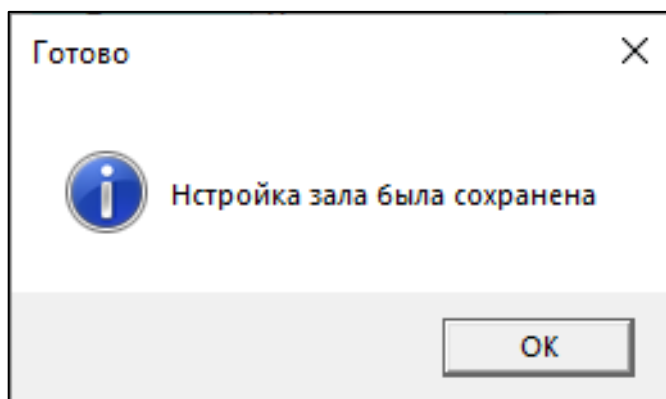


Рисунок 6 – Сообщение о успешном сохранение конфигурации

Панель администратора обладает возможностью резервного копирования, которая находится во вкладке “Опции” (Рисунок 7). В данном окне есть 2 панели для создания полной резервной копии и восстановление из файла резервной копии.

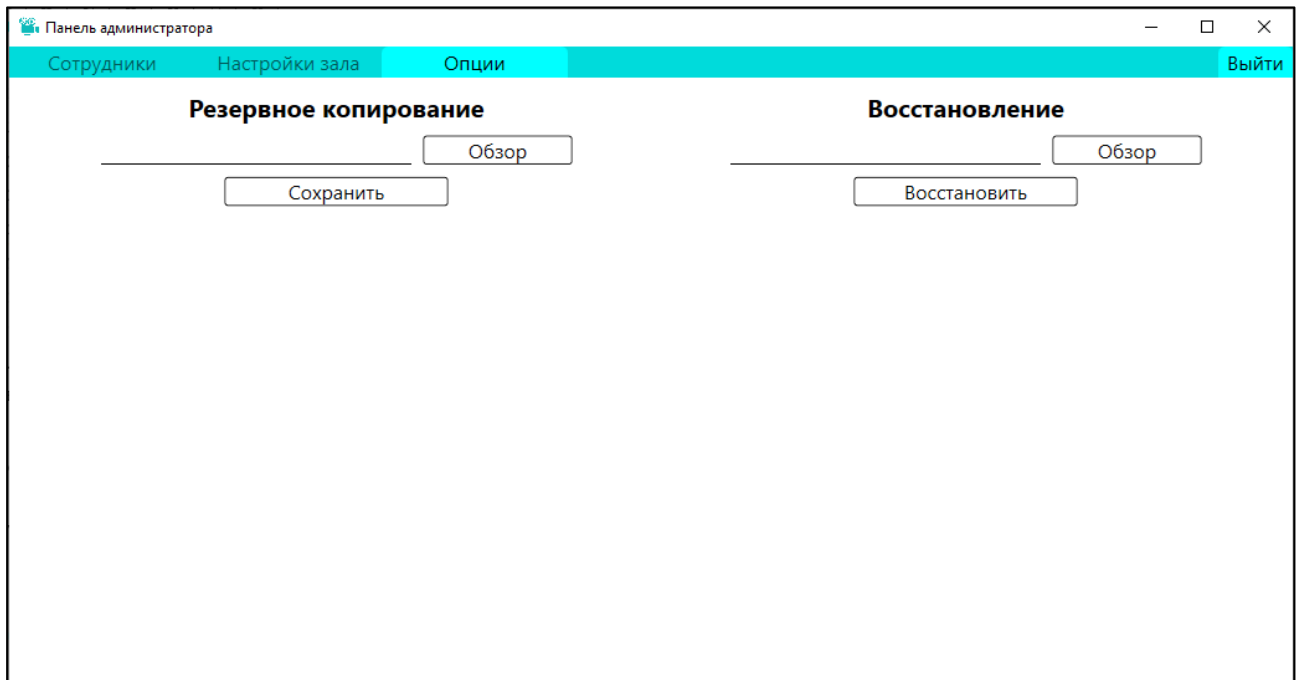


Рисунок 7 – Панель администратора с открытым окном опции

После авторизации под ролью букера (менеджера) мы видим панель менеджера (Рисунок 8). Она открывается на окне “Фильмы” здесь мы можем найти и просмотреть информацию о фильмах, а также добавить удалить или редактировать фильмы.

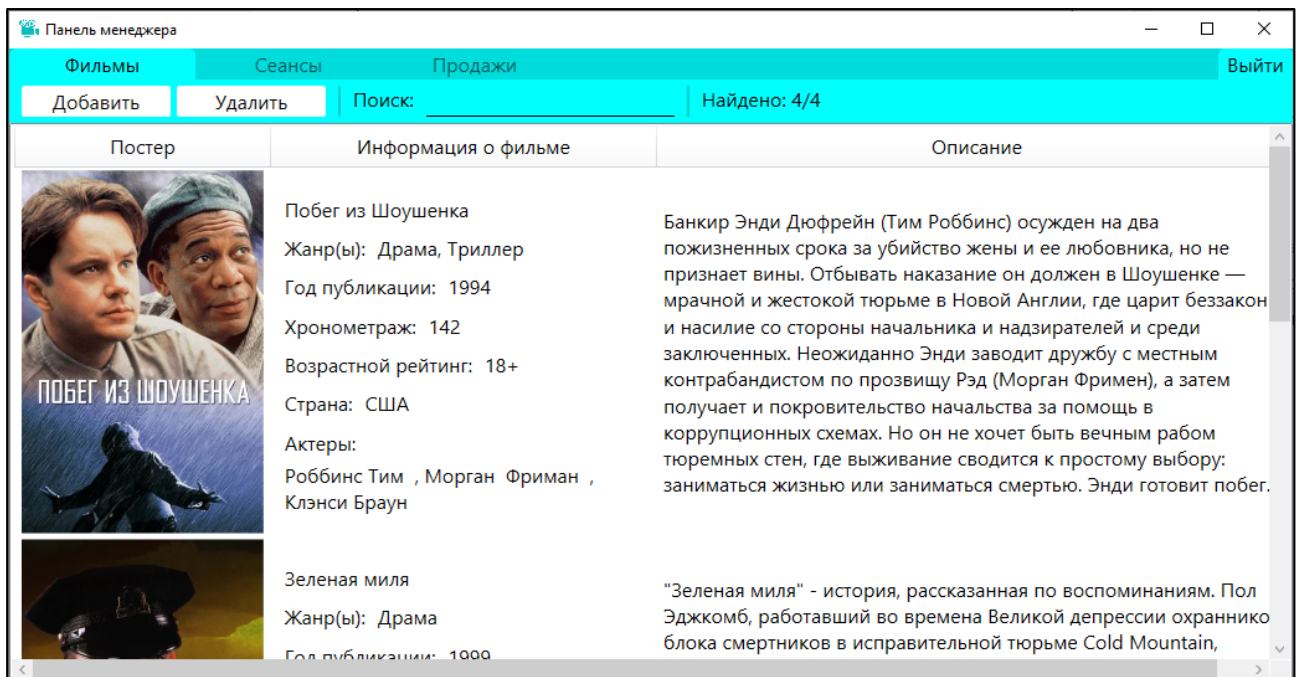


Рисунок 8 – Панель менеджера с открытым окном фильма

Окно добавления фильма поддерживает добавление, изменение и удаление жанров для этого нажать соответствующую кнопку, а в случае редактирования и удаления необходимо предварительно выбрать жанр из списка (Рисунок 9). Для выбора жанра необходимо дважды нажать на нужный жанр.




Рисунок 9 – Окно добавления жанра

Окно добавления фильма также поддерживает добавление, изменение и удаление актеров для этого нажать соответствующую кнопку, а в случае редактирования и удаления необходимо предварительно выбрать актера из списка (Рисунок 10).



Рисунок 10 – Окно добавления жанра

Для того чтобы добавить фильм необходимо нажать на кнопку “Добавить” после чего откроется окно добавления фильма (Рисунок 11).

Фильм

Название: _____

Страна производства: Австралия ▾

Год публикации: _____

Хронометраж (мин): _____

Возрастной рейтинг: _____

Описание: _____

Обложка: _____

Жанры: _____ X

IDGenre	Title
1	Драма
2	Триллер
3	Психологический фильм ужасов
6	Научная фантастика

Актеры: _____ X

IDActor	Surname	Name	Patronymic	Nickname
5	Пеппер	Барри		
6	Морис	Дэвид		
7	Митчелл	Рада		
8	Ферланд	Джодель		

Рисунок 11 – Окно добавления фильма

После добавления данных о фильме приложение отобразит сообщение о успешном сохранение (Рисунок 12).

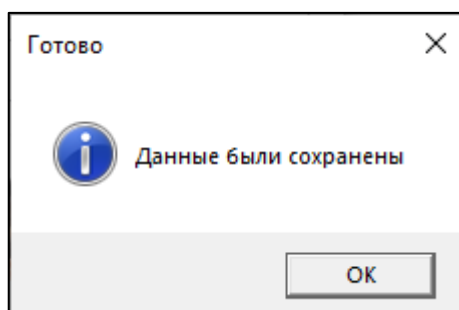


Рисунок 12 – Сообщение о успешном сохранение данных

Для редактирования данных фильма необходимо дважды нажать на редактируемый фильм, после чего откроется окно редактирования фильма (Рисунок 11).

Для удаления фильма необходимо выбрать фильм и нажать на кнопку удалить после чего отойти сообщение-подтверждение удаления (Рисунок 13).

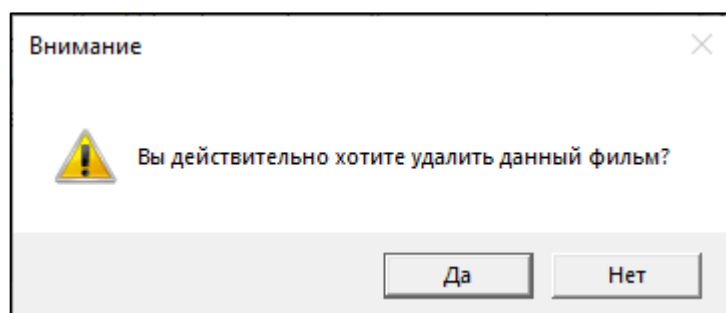


Рисунок 13 – Сообщение подтверждение удаления фильма

Также панель менеджера содержит окно сеансов (Рисунок 14). Здесь мы можем найти и просмотреть информацию о сеансах, а также добавить удалить или редактировать сеанс.

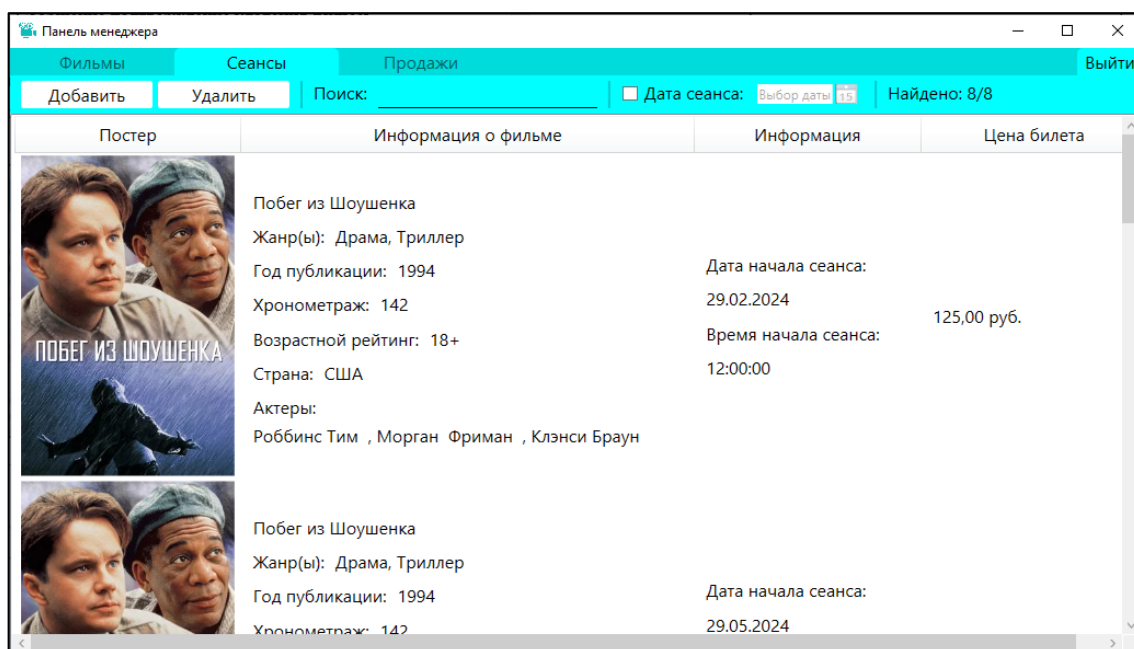
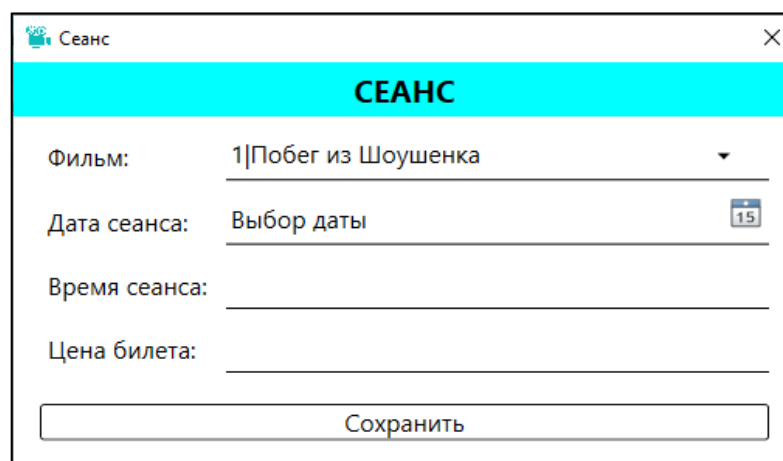


Рисунок 14 – Панель менеджера с открытым окном сеансы

Для того чтобы добавить сеанс необходимо нажать на кнопку “Добавить” после чего откроется окно добавления сеанса (Рисунок 15).



The screenshot shows a window titled 'Сеанс' (Session) with a cyan header bar containing the word 'СЕАНС'. Below the header, there are four input fields: 'Фильм:' (Movie) with a dropdown menu showing '1|Побег из Шоушенка' (The Shawshank Redemption), 'Дата сеанса:' (Session date) with a date picker showing '15', 'Время сеанса:' (Session time), and 'Цена билета:' (Ticket price). At the bottom of the window is a button labeled 'Сохранить' (Save).

Рисунок 15 – Окно добавления сеанса

После добавления данных о сеансе приложение отобразит сообщение о успешном сохранение (Рисунок 16).

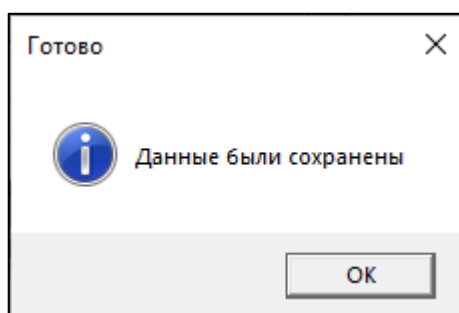


Рисунок 16 – Сообщение о успешном сохранение данных

Для редактирования данных сеанса необходимо дважды нажать на редактируемый сеанс, после чего откроется окно редактирования сеанса (Рисунок 15).

Для удаления сеанса необходимо выбрать сеанс и нажать на кнопку удалить после чего отроиться сообщение-подтверждение удаления (Рисунок 17).

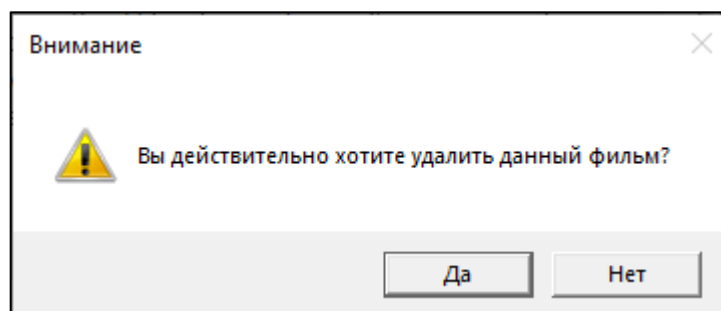


Рисунок 17 – Сообщение подтверждение удаления фильма

Также панель менеджера содержит окно продаж для просмотра и анализа данных по проданным билетам (Рисунок 18). На данном окне можно найти и просмотреть информацию о проданных билетах на фильмы. А также пользователь может сохранить информацию в файл формата PDF.

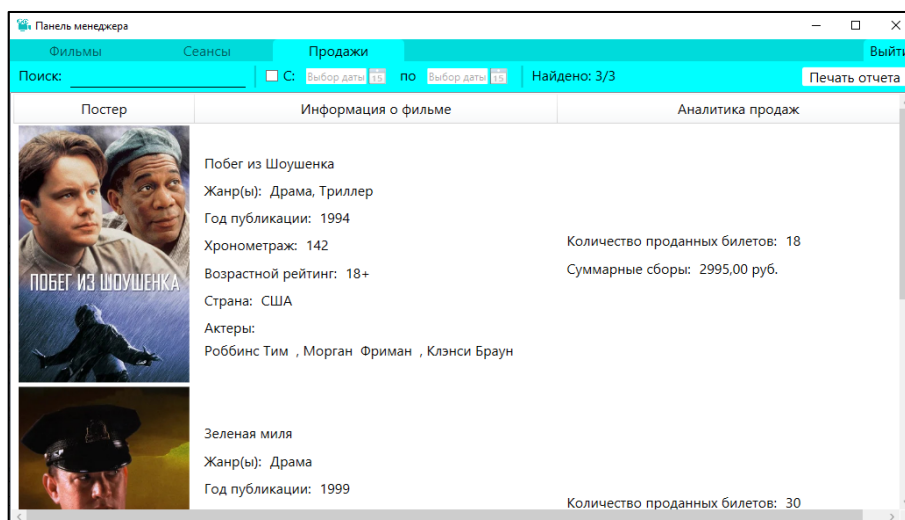


Рисунок 18 – Панель менеджера с открытым окном продаж

Для более подробного просмотра проданных билетов пользователь может нажать на интересующий его фильм и перед ним откроется таблица продаж билетов по сеансам (Рисунок 19). На этом окне также можно найти интересующий вас сеанс, а также сохранить информацию в файл формата PDF.

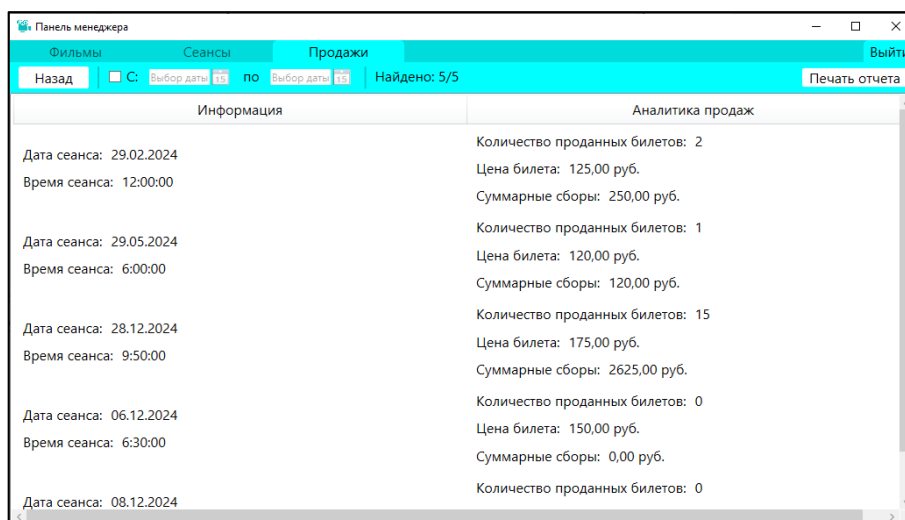


Рисунок 19 – Панель менеджера с открытым окном подробного просмотра продажи билетов

После авторизации под ролью кассира мы видим панель кассира (Рисунок 20). Она открывается на окне “Главная страница” здесь мы можем найти и просмотреть информацию о фильмах.

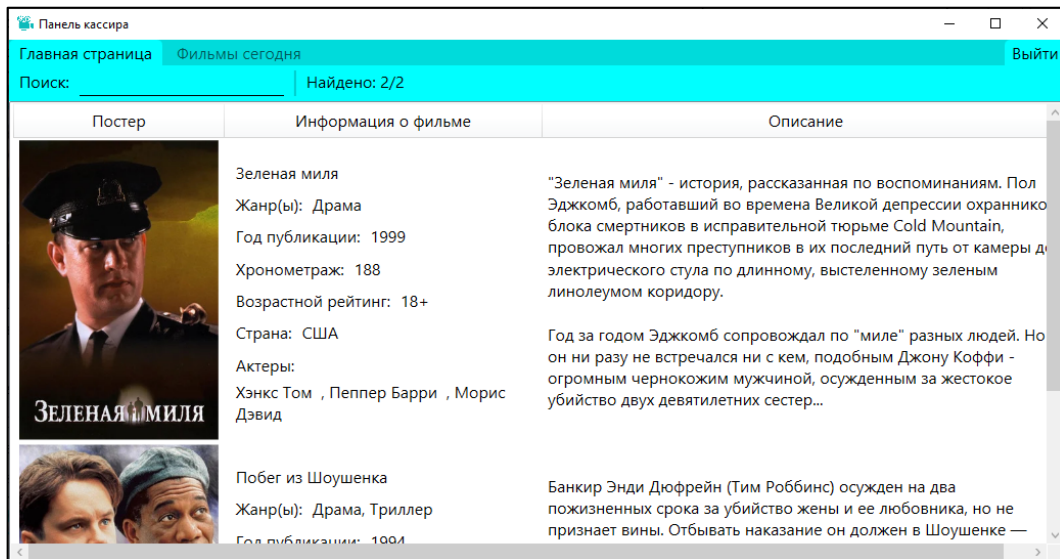


Рисунок 20 – Панель кассира с открытым окном главной страницы

На данном окне пользователь может выбрать фильм, на который нужно забронировать билет. После выбора фильма откроется окно выбора сеансов на выбранный фильм (Рисунок 21). На открытом окне пользователь может выбрать актуальные сеансы.

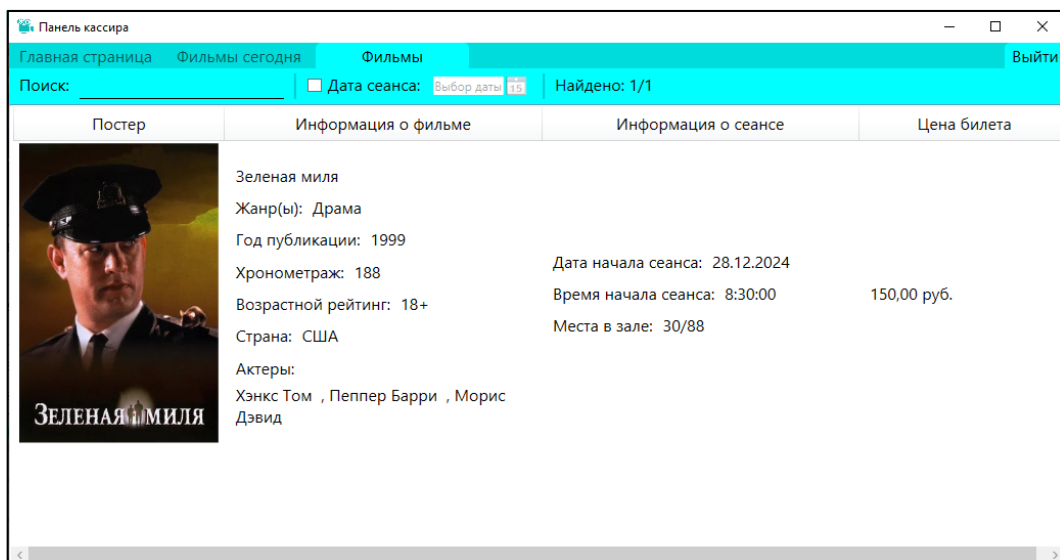


Рисунок 21 – Панель кассира с открытым окном сеансов

После выбора сеанса откроется окно для выбора места в зале (Рисунок 22).
На данном окне пользователь может выбрать место в зале.

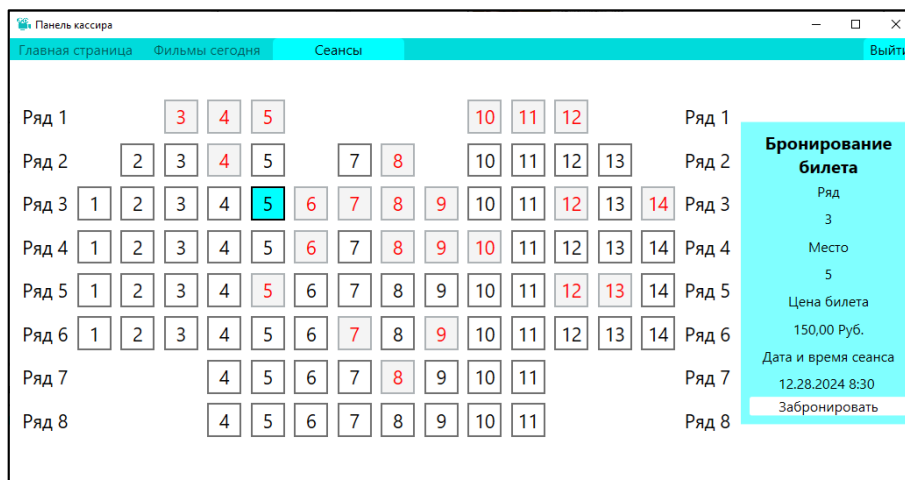


Рисунок 22 – Панель кассира с открытым окном бронирования места в зале

После подтверждения сообщения о бронирование билета (Рисунок 23), он автоматически откроется в PDF редакторе (Рисунок 24).

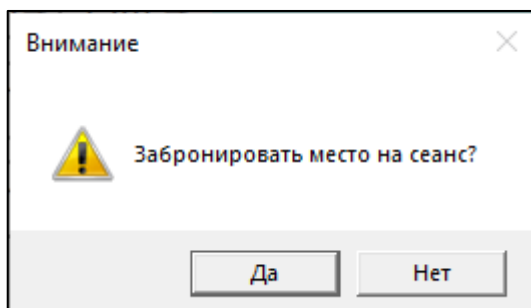


Рисунок 23 – Окно подтверждения бронирования билета

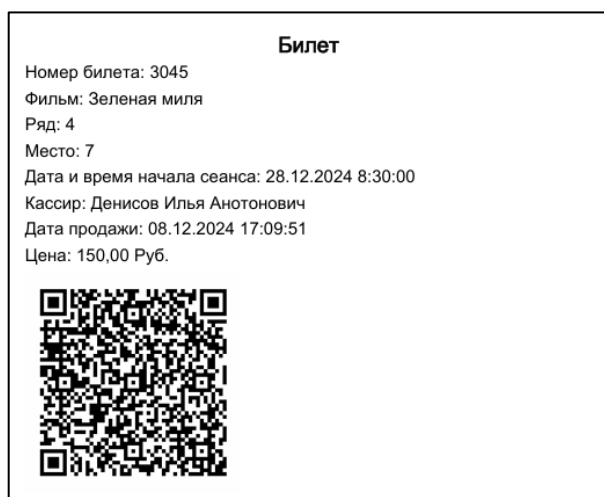


Рисунок 24 – Билет на бронируемый фильм

Для более удобной работы панель кассира обладает возможностью просмотра фильмов, которые буду проходить сегодня (Рисунок 25).

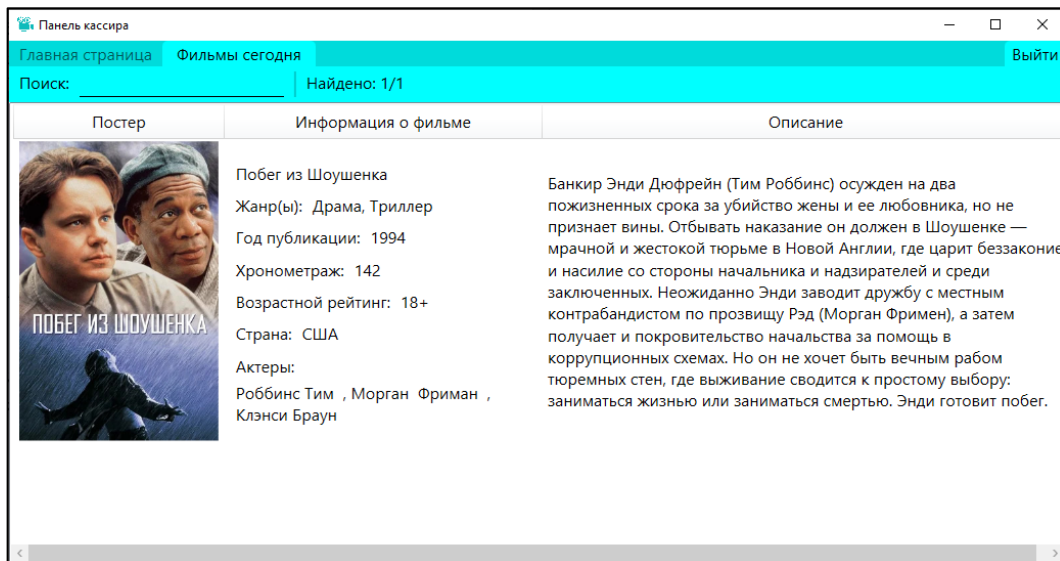


Рисунок 25 – Панель кассира с открытым окном фильма сегодня

После авторизации под ролью директора мы видим панель директора (Рисунок 26). Она открывается на окне “Сотрудники” здесь мы можем найти и просмотреть информацию о сотрудниках, а также добавить удалить или редактировать сотрудников.

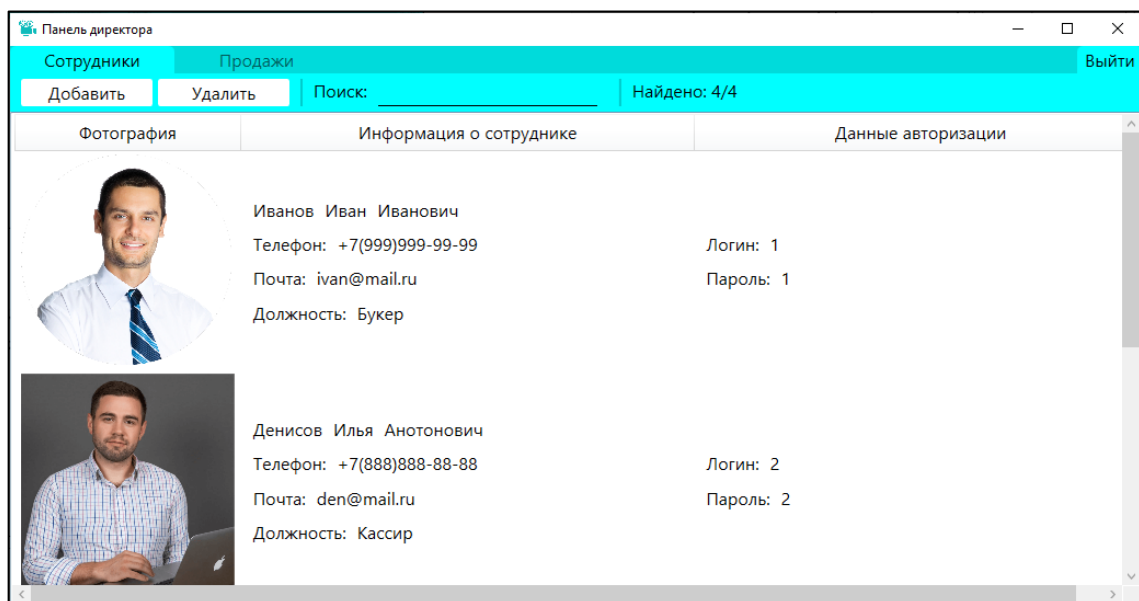


Рисунок 26 – Панель директора с открытым окном сотрудники

Для того чтобы добавить сотрудника необходимо нажать на кнопку “Добавить” после чего откроется окно добавления сотрудник (Рисунок 27).

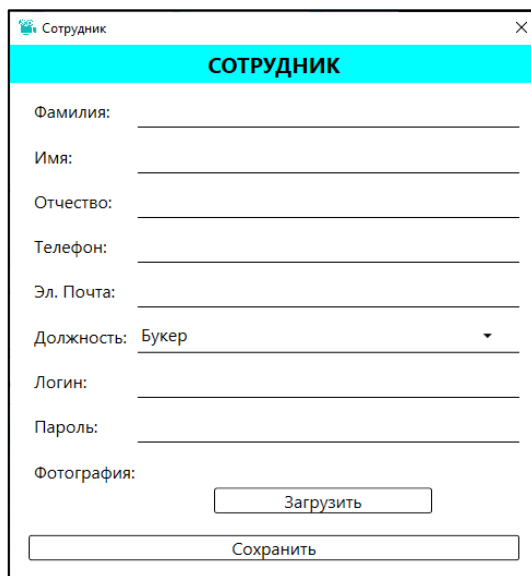


Рисунок 27 – Окно добавления сотрудника

После добавления данных о сотруднике приложение отобразит сообщение о успешном сохранение (Рисунок 28).

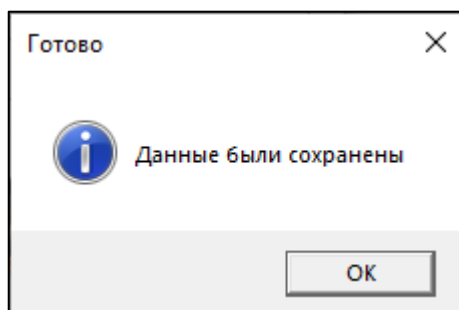


Рисунок 28 – Сообщение о успешном сохранение данных

Для редактирования данных сотрудника необходимо дважды нажать на редактируемого сотрудника, после чего откроется окно редактирования сотрудника (Рисунок 27).

Для удаления сотрудника необходимо выбрать сотрудника и нажать на кнопку удалить после чего отроиться сообщение-подтверждение удаления (Рисунок 29).

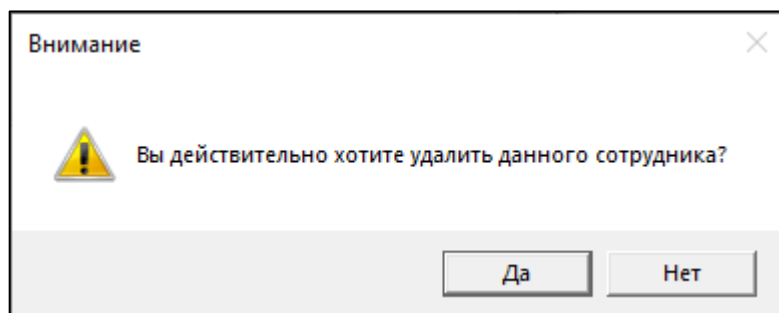


Рисунок 29 – Сообщение подтверждение удаления сотрудника

Также панель директора содержит окно продаж для просмотра и анализа данных по проданным билетам (Рисунок 30). На данном окне можно найти и просмотреть информацию о проданных билетах на фильмы. А также пользователь может сохранить информацию в файл формата PDF.

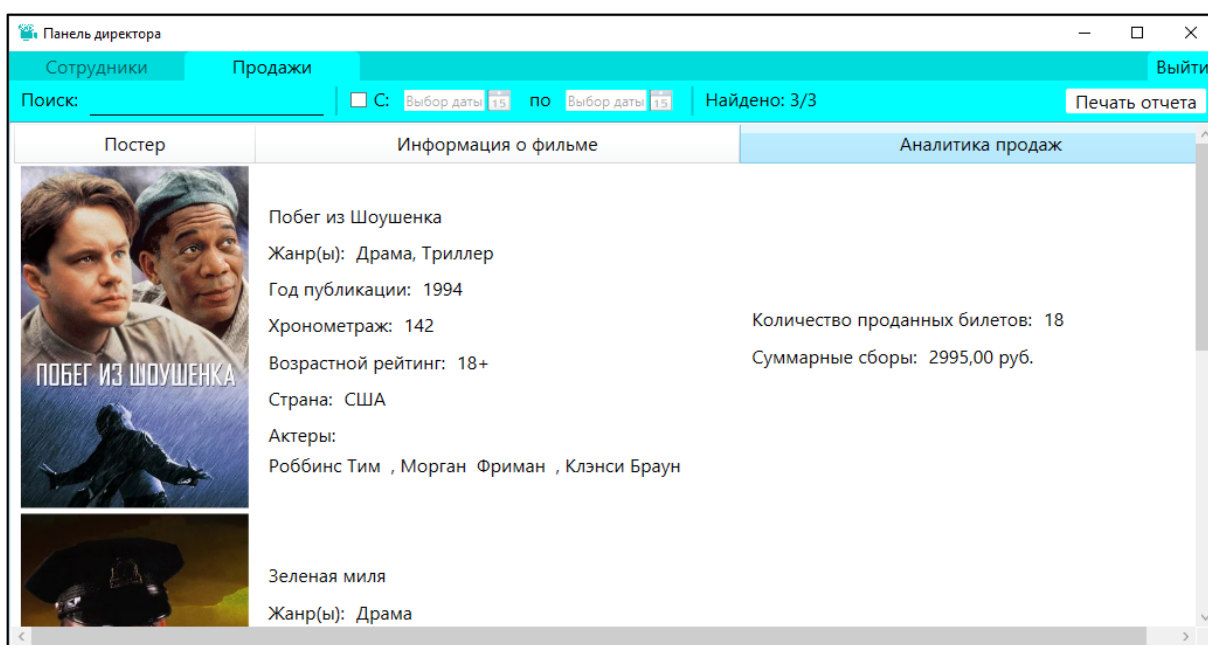


Рисунок 30 – Панель директора с открытым окном продаж

Для более подробного просмотра проданных билетов пользователь может нажать на интересующий его фильм и перед ним откроется таблица продажи билетов по сеансам (Рисунок 31). На этом окне также можно найти интересующий вас сеанс, а также сохранить информацию в файл формата PDF.

Информация	Аналитика продаж
Дата сеанса: 29.02.2024 Время сеанса: 12:00:00	Количество проданных билетов: 2 Цена билета: 125,00 руб. Суммарные сборы: 250,00 руб.
Дата сеанса: 29.05.2024 Время сеанса: 6:00:00	Количество проданных билетов: 1 Цена билета: 120,00 руб. Суммарные сборы: 120,00 руб.
Дата сеанса: 28.12.2024 Время сеанса: 9:50:00	Количество проданных билетов: 15 Цена билета: 175,00 руб. Суммарные сборы: 2625,00 руб.
Дата сеанса: 06.12.2024 Время сеанса: 6:30:00	Количество проданных билетов: 0 Цена билета: 150,00 руб. Суммарные сборы: 0,00 руб.

Рисунок 31 – Панель менеджера с открытым окном подробного просмотра продажи билетов

Приложение 3

Программный продукт поставляется на USB флеш накопителе.

На USB накопителя находиться:

- Установочник программы
- Проект программного продукта
- Скрипт базы данных без тестовых данных
- Скрипт базы данных с тестовыми данными
- Файл курсового проекта в формате Microsoft Word (docx)
- Файл с презентацией курсового проекта

Также программный продукт можно получить, перейдя по ссылке:

- Облако Mail:
- GitHub: <https://github.com/ALTERRA4546/Cinema>